REPÚBLICA DEL ECUADOR
Escuela Politécnica Nacional
"E SCIENTIA HOMINIS SALUS"

La versión digital de esta tesis está protegida por la Ley de Derechos de Autor del Ecuador.

Los derechos de autor han sido entregados a la "ESCUELA POLITÉCNICA NACIONAL" bajo el libre consentimiento del (los) autor(es).

Al consultar esta tesis deberá acatar con las disposiciones de la Ley y las siguientes condiciones de uso:

- Cualquier uso que haga de estos documentos o imágenes deben ser sólo para efectos de investigación o estudio académico, y usted no puede ponerlos a disposición de otra persona.

- Usted deberá reconocer el derecho del autor a ser identificado y citado como el autor de esta tesis.

- No se podrá obtener ningún beneficio comercial y las obras derivadas tienen que estar bajo los mismos términos de licencia que el trabajo original.

El Libre Acceso a la información, promueve el reconocimiento de la originalidad de las ideas de los demás, respetando las normas de presentación y de citación de autores con el fin de no incurrir en actos ilegítimos de copiar y hacer pasar como propias las creaciones de terceras personas.

Respeto hacia sí mismo y hacia los demás.

ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

ANÁLISIS, SIMULACIÓN E IMPLEMENTACIÓN DE UNA RED DE CONMUTACIÓN TIPO BANYAN EMPLEANDO UN FPGA (FIELD PROGRAMMABLE GATE ARRAY)

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

NELLY ALEXANDRA BRAVO LÓPEZ
nellyabl@hotmail.com

DIRECTOR: PhD. IVÁN MARCELO BERNAL CARRILLO
Ivan.bernal@epn.edu.ec

Quito, Mayo 2014

DECLARACIÓN

Yo, Nelly Alexandra Bravo López, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

_____

Nelly Alexandra Bravo López

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Nelly Alexandra Bravo López, bajo mi supervisión.

_____

Ph.D. Iván Bernal
DIRECTOR DEL PROYECTO

AGRADECIMIENTOS

A mi Dios, quien fue mi principal soporte y ayuda durante la realización de este proyecto.

A mis padres por su amor y apoyo incondicional, sin los cuales no hubiese sido posible alcanzar esta meta tan anhelada.

A mi hijo, quien con su alegría e inocencia me infundió el valor y la motivación necesaria para seguir adelante y alcanzar mis metas.

A mis amigos con los que compartimos inolvidables momentos, gracias por su amistad y apoyo.

Al Dr. Iván Bernal, por su confianza, su apoyo y por tantos conocimientos impartidos tanto en las aulas de clase como durante el desarrollo de este proyecto.

DEDICATORIA

Al único y sabio Dios sea la gloria,
porque de Él, por Él y para Él son todas las cosas.

CONTENIDO

CONTENIDO ..... **i**
ÍNDICE DE FIGURAS ..... **vi**
ÍNDICE DE TABLAS ..... **ix**
RESUMEN ..... **xi**
PRESENTACIÓN ..... **xii**

CAPÍTULO 1

INTRODUCCIÓN A REDES DE INTERCONEXIÓN Y REDES BANYAN **1**

1.1 REDES DE INTERCONEXIÓN 1
1.1.1 CONCEPTOS BÁSICOS 1
1.1.2 CARACTERÍSTICAS 2
1.1.3 CLASIFICACIÓN 3
1.1.3.1 En base a su topología 3
1.1.3.2 En base al tipo de enrutamiento 6
1.1.3.3 En base a la característica de bloqueo 7
1.2 REDES DE INTERCONEXIÓN MULTI-ETAPA (MIN) 9
1.2.1 DEFINICIÓN 9
1.2.2 ELEMENTOS DE CONMUTACIÓN 10
1.2.3 PERMUTACIONES 11
1.3 REDES BANYAN 18
1.3.1 DEFINICIÓN 18
1.3.2 CARACTERÍSTICAS 20
1.3.3 ALGORITMO DE AUTO-ENRUTAMIENTO 21
1.3.4 PRINCIPALES TOPOLOGÍAS DE LAS REDES BANYAN 23
1.3.4.1 Red Omega (Ω) 24
1.3.4.2 Red *Baseline* (Φ) 24
1.3.4.3 Red *Butterfly* (Σ) 26
1.3.4.4 Red n-Cúbica (Γ) 27

1.3.5 EQUIVALENCIA ENTRE REDES BANYAN 28
1.3.6 DESEMPEÑO 30
1.3.6.1 Almacenamiento (*Buffering)* 30
1.3.6.2 Velocidad interna 34
1.3.6.3 Redes Banyan dilatadas y replicadas 34
1.3.6.4 Red de ordenamiento 35
1.3.7 APLICACIONES DE LAS REDES BANYAN 38
1.3.7.1 Conmutador Batcher-Banyan 38
1.3.7.2 Conmutador Starlite 43
1.3.7.3 Conmutador Sunshine 45
1.3.7.4 Conmutador Tandem Banyan 47

REFERENCIAS 51

CAPÍTULO 2

ANÁLISIS MATEMÁTICO DE UNA RED DE CONMUTACIÓN TIPO BANYAN 55

2.1 DESCRIPCIÓN DE LA RED A UTILIZAR 55
2.1.1 CARACTERÍSTICAS 55
2.1.2 ENRUTAMIENTO 57
2.2 DESCRIPCIÓN DE LOS ELEMENTOS DE CONMUTACIÓN 57
2.2.1 ARQUITECTURA DEL ELEMENTO DE CONMUTACIÓN BANYAN DUAL 58
2.2.2 FUNCIONAMIENTO 59
2.3 ANÁLISIS MATEMÁTICO 61
2.3.1 CONDICIONES DEL SISTEMA 61
2.3.2 DESARROLLO MATEMÁTICO 62
2.3.2.1 Diagrama de Transición de Estados 63
2.3.2.2 Probabilidades de Transición de Estado 64
2.3.2.3 Ecuaciones de Corte 66
2.3.2.4 Parámetros de interés 70
2.3.2.5 Generalización para toda la red 72

2.4 CÁLCULOS NUMÉRICOS 74
2.4.1 CONDICIONES DE TRÁFICO 75
2.4.2 ECUACIÓN A RESOLVER 76
2.5 RESULTADOS OBTENIDOS 76

REFERENCIAS 79

CAPÍTULO 3

DISEÑO Y SIMULACIÓN DE LA RED DE CONMUTACIÓN BANYAN 80

3.1 GENERALIDADES 80
3.2 DISEÑO DE LA RED DE CONMUTACIÓN BANYAN 81
3.2.1 CONTROLADOR DE ENTRADA 83
3.2.2 BUFFER INTERNO 84
3.2.2.1 Funcionamiento 84
3.2.2.2 Entidad VHDL 85
3.2.2.3 Diseño del buffer interno 86
3.2.3 CONTROLADOR DE SALIDA 90
3.2.4 ELEMENTO DE CONMUTACIÓN BANYAN DUAL 92
3.2.5 RED DE CONMUTACIÓN 95
3.3 SIMULACIÓN DE LA RED DE CONMUTACIÓN 99
3.3.1 MÓDULO PARA CÁLCULO DE RETARDO PROMEDIO 99
3.3.1.1 Funcionamiento 99
3.3.1.2 Entidad VHDL 101
3.3.2 EQUILIBRIO ESTADÍSTICO 101
3.3.3 SIMULACIÓN DE LA RED 103
3.3.3.1 Parámetros de interés 104
3.4 COMPARACIÓN DE RESULTADOS TEÓRICOS Y SIMULADOS 106
3.4.1 JUSTIFICACIÓN DE DIFERENCIAS 107

REFERENCIAS 110

CAPÍTULO 4

IMPLEMENTACIÓN DE LA RED DE CONMUTACIÓN Y PRUEBAS DE FUNCIONAMIENTO.....111

4.1 GENERALIDADES..... 111
4.2 INTERFAZ GRÁFICA DE CONTROL..... 114
4.2.1 COMPROBACIÓN DE CONECTIVIDAD CON FPGA..... 115
4.2.2 EDICIÓN DE PARÁMETROS SUSCEPTIBLES A MODIFICACIÓN..... 115
4.2.3 GENERACIÓN DE DATOS..... 116
4.2.4 ENVÍO DE DATOS ORIGINALES A EXCEL..... 119
4.2.5 TRANSMISIÓN SERIAL DE DATOS GENERADOS..... 119
4.2.6 RECEPCIÓN DE DATOS..... 120
4.2.7 ENVÍO A EXCEL DE DATOS RECIBIDOS..... 121
4.2.8 VISUALIZACIÓN DE RESULTADOS..... 122
4.3 MÓDULOS VHDL DISEÑADOS..... 123
4.3.1 MÓDULO DE COMUNICACIÓN SERIAL UART..... 124
4.3.1.1 Subsistema de Recepción..... 124
4.3.1.2 Subsistema de Transmisión..... 132
4.3.2 DEMUX 1X8..... 135
4.3.3 FIFOS DE ENTRADA..... 137
4.3.4 RED DE CONMUTACIÓN..... 138
4.3.5 FIFOS DE SALIDA..... 141
4.3.6 MUX 8X1..... 142
4.3.7 MÓDULO DE CONTROL..... 144
4.4 ENTIDAD VHDL DE MÁS ALTO NIVEL..... 150
4.5 PROCESO DE IMPLEMENTACIÓN DEL SISTEMA..... 152
4.5.1 ARCHIVO DE RESTRICCIONES DE USUARIO..... 153
4.5.2 RECURSOS UTILIZADOS..... 154
4.5.3 FRECUENCIA MÁXIMA..... 156
4.5.4 PROGRAMACIÓN DEL FPGA..... 157
4.6 PRUEBAS DE FUNCIONAMIENTO..... 158

REFERENCIAS.....162

CAPÍTULO 5

CONCLUSIONES Y RECOMENDACIONES 163

CONCLUSIONES 163
RECOMENDACIONES 169

REFERENCIAS BIBLIOGRÁFICAS.....169

ANEXOS

ANEXO 1 178
ANEXO 2 184
ANEXO 3 188
ANEXO 4 194
ANEXO 5 198
ANEXO 6 205
ANEXO 7 206

ÍNDICE DE FIGURAS

CAPÍTULO 1

Figura 1-1. Diagrama básico de una red de interconexión. 1
Figura 1-2. Ejemplos de redes de interconexión directas. 4
Figura 1-3. Ejemplo de redes indirectas: (a) Red *Crossbar* (b) Red Banyan. 5
Figura 1-4. Esquema básico de una MIN. 9
Figura 1-5. Posibles configuraciones de un elemento de conmutación 2X2. 10
Figura 1-6. Procedimiento para realizar la permutación *Perfect Shuffle*. 13
Figura 1-7. Permutación *Perfect Shuffle* con 8 elementos. 14
Figura 1-8. Permutación *Perfect Shuffle* Inversa con 8 elementos. 15
Figura 1-9. Permutaciones *Butterfly* de 8 elementos de: a) orden 1 y b) orden 2. 16
Figura 1-10. Permutaciones a) *Bit switch* y b) *Bit reversal* para N=16. 17
Figura 1-11. Árbol Banyan. 19
Figura 1-12. Red Banyan 8x8. 19
Figura 1-13. Red Omega 8x8. 24
Figura 1-14. Construcción Recursiva de Red *Baseline* 16x16. 25
Figura 1-15. Red *Butterfly* 2^4x2^4. 26
Figura 1-16. Red 4-cúbica. 27
Figura 1-17. Equivalencia entre redes Omega y *Baseline*. 28
Figura 1-18. *Head-of-line blocking* 31
Figura 1-19. Elemento de conmutación con buffers a la entrada o salida. 32
Figura 1-20. *Virtual Output Queuing*. 33
Figura 1-21. a) Red Banyan 2-dilatada y b) Red Banyan 4-replicada. 35
Figura 1-22. Red de Batcher para N=16. 37
Figura 1-23. Arquitectura de un conmutador Batcher-Banyan. 39
Figura 1-24. Flujo de paquetes en el algoritmo de tres fases. 41
Figura 1-25. Ejemplo de conmutación usando el algoritmo de tres fases. 42
Figura 1-26. Estructura del conmutador Starlite. 44
Figura 1-27. Arquitectura del conmutador Sunshine. 46
Figura 1-28. Arquitectura del conmutador Tandem Banyan. 48
Figura 1-29. Arquitectura del conmutador Tandem Banyan Dilatado. 49

CAPÍTULO 2

Figura 2-1. Red de Conmutación Omega de tamaño *16 x 16*. 56
Figura 2-2. Arquitectura de un elemento de conmutación Banyan dual. 59
Figura 2-3. Diagrama de Transición de Estados del elemento de conmutación Banyan Dual. 64

Figura 2-4. Promedio de paquetes perdidos en la red en función del tamaño de los buffers. 78
Figura 2-5. Retardo promedio en atravesar la red en función del tamaño de los buffers. 78

CAPÍTULO 3

Figura 3-1. Diagrama de bloques del sistema a implementar..... 81
Figura 3-2. Estructura interna de la red Banyana diseñar. 82
Figura 3-3. Paquete de datos a conmutar en la red. 82
Figura 3-4. Entidad del controlador de entrada. 83
Figura 3-5. Diagrama básico de un FIFO. 85
Figura 3-6. Entidad correspondiente a un buffer. 86
Figura 3-7. Buffer circular con punteros de lectura y escritura. 87
Figura 3-8. Funcionamiento del buffer circular. 88
Figura 3-9. Entidad correspondiente al controlador de salida..... 92
Figura 3-10. Elemento de Conmutación Banyan Dual..... 93
Figura 3-11. Entidad VHDL del elemento de conmutación diseñado. 94
Figura 3-12. Red Omega de tamaño 8x8. 95
Figura 3-13. Entidad correspondiente a la red de conmutación. 96
Figura 3-14. Entidad VHDL del módulo para medición de retardo. 101
Figura 3-15. Simulación de la red de conmutación con B=4. 104
Figura 3-16. Promedio de paquetes perdidos en la red (Simulación)..... 105
Figura 3-17. Retardo promedio de un paquete en atravesar la red (Simulación) 106
Figura 3-18. Comparación del valor promedio de paquetes perdidos en la red. 107
Figura 3-19. Comparación del retardo promedio de un paquete en atravesar la red. 107

CAPÍTULO 4

Figura 4-1. Diagrama de bloques del sistema a implementar..... 112
Figura 4-2. Interfaz gráfica de control. 115
Figura 4-3. Conexión establecida con el FPGA. 116
Figura 4-4. Ventana para edición de parámetros del sistema 115
Figura 4-5. Generación de datos en interfaz gráfica de control. 118
Figura 4-6. Ejemplo de datos originales en Excel. 119
Figura 4-7. Ejemplo de datos recibidos en Excel. 121
Figura 4-8. Visualización de resultados obtenidos. 123
Figura 4-9. Subsistema UART de recepción. 125
Figura 4-10. Entidad del generador de tasa de muestreo. 126
Figura 4-11. Diagrama ASM del subsistema de recepción. 129
Figura 4-12. Entidad correspondiente al receptor de datos. 130
Figura 4-13. Entidad correspondiente al buffer de almacenamiento. 131
Figura 4-14. Sistema UART completo. 132
Figura 4-15. Diagrama ASM correspondiente al transmisor. 133

Figura 4-16. Entidad correspondiente al transmisor. 134
Figura 4-17. Entidad correspondiente al demux 1x8. 135
Figura 4-18. Entidad correspondiente a un fifo de entrada. 137
Figura 4-19. Entidad correspondiente a la red de conmutación a implementar. 140
Figura 4-20. Entidad correspondiente a un FIFO de salida. 141
Figura 4-21. Entidad correspondiente al mux 8x1. 142
Figura 4-22. Entidad correspondiente al módulo de control. 145
Figura 4-23. Entidad de más alto nivel. 151
Figura 4-24. Archivo de restricciones de usuario para el sistema diseñado. 153
Figura 4-25. Recursos utilizados en el FPGA para B=2. 155
Figura 4-26. Recursos utilizados en el FPGA para B=4. 155
Figura 4-27. Frecuencia máxima de operación del circuito con B=2. 156
Figura 4-28. Frecuencia máxima de operación del circuito con B=4. 156
Figura 4-29. Interfaz gráfica de la herramienta de programación Impact. 158
Figura 4-30. Equipo utilizado para realizar las pruebas de funcionamiento. 159
Figura 4-31. Interfaz gráfica mostrando los resultados obtenidos para B=2..... 160
Figura 4-32. Interfaz gráfica mostrando los resultados obtenidos para B=4..... 160

ANEXO 1

Figura A1-1. a) Red Omega, b) Grafo correspondiente a la red Omega. 178
Figura A1-2. Ejemplo de isomorfismo entre grafos. 179
Figura A1-3. Ejemplo de isomorfismo entre redes. 180
Figura A1-4. Ejemplo de redes topológicamente equivalentes pero no isomorfas. 181
Figura A1-5. Ejemplo de redes funcionalmente equivalentes. 182
Figura A1-6. Ejemplo de redes isomorfas pero no funcionalmente equivalentes. 183

ANEXO 3

Figura A3-1. Simulación correspondiente al controlador de entrada. 188
Figura A3-2. Simulación correspondiente al controlador de salida..... 189
Figura A3-3. Simulación correspondiente al FIFO interno. 190
Figura A3-4. Simulación correspondiente al elemento de conmutación Banyan dual. 190
Figura A3-5. Simulación correspondiente a la red de conmutación Omega..... 191
Figura A3-6. Simulación correspondiente al demux 8x1. 192
Figura A3-7. Simulación correspondiente al mux 1x8. 193

ANEXO 4

Figura A4-1. Estado de equilibrio para B=2. 195
Figura A4-2. Estado de equilibrio para B=4. 195
Figura A4-3. Estado de equilibrio para B=8. 196
Figura A4-4. Estado de equilibrio para B=16. 196
Figura A4-5. Estado de equilibrio para B=32. 197

ÍNDICE DE TABLAS

CAPÍTULO 1

Tabla 1-1. Reglas de enrutamiento en redes Banyan. 23
Tabla 1-2. Equivalencia entre Redes Banyan. 29

CAPÍTULO 2

Tabla 2-1. Parámetros de interés para diferentes tamaños de buffer de los SE.. 77

CAPÍTULO 3

Tabla 3-1. Puertos de entrada y salida del controlador de entrada..... 83
Tabla 3-2. Puertos de entrada y salida del buffer interno. 86
Tabla 3-3. Puertos de entrada y salida del controlador de salida. 92
Tabla 3-4. Puertos de entrada y salida del elemento de conmutación..... 94
Tabla 3-5. Puertos de entrada y salida de la red de conmutación. 96
Tabla 3-6. Puertos de entrada y salida del módulo de medición de retardo. 101
Tabla 3-7. Resultados obtenidos de la simulación de la red de conmutación.... 105
Tabla 3-8. Resultados comparativos de los parámetros de interés en la red de conmutación Banyan. 106

CAPÍTULO 4

Tabla 4-1. Puertos de entrada y salida del generador de tasa de muestreo..... 127
Tabla 4-2. Puertos de entrada y salida del receptor de datos..... 130
Tabla 4-3. Puertos de entrada y salida del buffer de almacenamiento. 131
Tabla 4-4. Puertos de entrada y salida del transmisor. 135
Tabla 4-5. Puertos de entrada y salida del demux 1x8. 136
Tabla 4-6. Puertos de entrada y salida de un FIFO de entrada. 138
Tabla 4-7. Puertos de entrada y salida de la red de conmutación. 140
Tabla 4-8. Puertos de entrada y salida de un FIFO _out. 142
Tabla 4-9. Puertos de entrada y salida del mux 8x1. 143
Tabla 4-10. Puertos de entrada y salida del módulo de control. 145
Tabla 4-11. Puertos de entrada y salida de la entidad de más alto nivel. 151
Tabla 4-12. Resultados comparativos entre teoría, simulación eimplementación. ……..159

ANEXO 2

Tabla A2-1. Resultados Teóricos para B=2. 185
Tabla A2-2. Resultados Teóricos para B=4. 186
Tabla A2-3. Resultados Teóricos para B=8. 186
Tabla A2-4. Resultados Teóricos para B=16. 186
Tabla A2-5. Resultados Teóricos para B=32. 187

RESUMEN

Las redes Banyan son un tipo de red de interconexión multietapa (*Multi-stage Interconnection Network*, MIN) usadas tanto en el área de procesamiento paralelo en computación, así como dentro de la arquitectura de conmutadores y enrutadores de alta velocidad. La característica distintiva que se ha empleado en este caso es el uso de un tipo especial de elemento de conmutación denominado Banyan dual, este elemento hace uso de una técnica de almacenamiento denominada *Virtual Output Queueing* (VOQ) con el fin de mejorar el desempeño de la red.

El objetivo del presente proyecto es el de medir el nivel de desempeño global de la red a medida que se incrementa el tamaño de los buffers internos; los parámetros a considerar son el promedio de paquetes perdidos en la red en cada ranura de tiempo y el retardo promedio de los paquetes en atravesar la red. Con el fin de obtener las expresiones correspondientes a estos parámetros, así como sus valores numéricos, se realiza el análisis matemático de la red. Además, se hace también la descripción de la red en lenguaje VHDL y su respectiva simulación para comparar los resultados teóricos con los simulados. Finalmente, se explican las consideraciones que se ha tomado en cuenta para la implementación práctica de la red en el FPGA.

PRESENTACIÓN

El presente proyecto expone el análisis matemático, simulación e implementación realizadas a una red de conmutación tipo Banyan. La descripción de la red se la ha realizado en lenguaje VHDL y se ha empleado el ambiente de desarrollo *ISE Design Suite* para su simulación, compilación y posterior implementación el en FPGA. En cuanto a la implementación práctica de la red se dispone del módulo de desarrollo *Spartan-3ANStarter Kit Board*, mismo que contiene un FPGA XC3S700A.

El desarrollo del proyecto se lo ha dividido en cinco capítulos:

Capítulo 1:Introducción a redes de interconexión y redes Banyan. En este capítulo se abordan los conceptos básicos relacionados a las redes de interconexión y, más específicamente, a las redes Banyan. Se da a conocer las características principales de estas redes, su clasificación, funcionamiento y aplicaciones.

Capítulo 2:Análisis matemático de una red de conmutación tipo Banyan. En la primera parte de este capítulo se muestra la red específica que se usará en el resto del proyecto, se detallan sus características, y se enfoca la atención en los elementos de conmutación usados, mismos que emplean una técnica especial de almacenamiento. A continuación, se establecen las condiciones bajo las cuales se realizará el análisis matemático de la red. Luego, se muestra en forma detallada el desarrollo matemático y las expresiones obtenidas para los parámetros de interés. Finalmente, se muestran los resultados numéricos obtenidos para el caso de la red específica de este proyecto.

Capítulo 3: Diseño y simulación de la red de conmutación Banyan. En este capítulo, la atención se enfoca en la descripción en VHDL de cada uno de los componentes de la red Banyan, para luego realizar su interconexión y obtener asíla descripción completa de la red. Con ello, se realiza la simulación de la red

tomando en cuenta las mismas consideraciones que para la parte teórica. Finalmente, se realiza una comparación entre los resultados obtenidos del análisis matemático y los obtenidos de la simulación de la red.

Capítulo 4:Implementación de la red y pruebas de funcionamiento. En este capítulo se describe el sistema completo a implementar para realizar las pruebas de funcionamiento de la red en el FPGA. Se realiza la descripción en VHDL de los módulos adicionales necesarios para la adecuación de los datos y la comunicación serial entre el módulo de desarrollo y la computadora, la cual será la encargada de la generación aleatoria de los datos, su transmisión serial, la recepción de los datos conmutados y la visualización de los resultados.

Capítulo 5: Conclusiones y recomendaciones. En este capítulo se exponen las conclusiones y recomendaciones que se obtuvieron durante la elaboración del presente proyecto.

Por último, se muestran los anexos con información adicional relacionada a los temas expuestos en este proyecto, así mismo se incluye tanto el código VHDL de todos los módulos diseñados como el código fuente de la interfaz gráfica elaborada usando el software *Matlab*.

CAPÍTULO 1

INTRODUCCIÓN A REDES DE INTERCONEXIÓN Y REDES BANYAN

1.1 REDES DE INTERCONEXIÓN

1.1.1 CONCEPTOS BÁSICOS

De forma general, se puede definir a una red de interconexión como el medio físico a través del cual se conectan N entradas con M salidas a fin de intercambiar información en forma de paquetes, como se aprecia en la Figura 1-1.

Dependiendo de la estructura interna de la red, la conexión de una entrada con una salida se la puede realizar ya sea a través de enlaces directos (punto a punto) o haciendo uso de elementos intermedios de conmutación ubicados en el interior de la red. En la Sección 1.1.3 se abordará con más detalle la clasificación de las redes de interconexión tomando en cuenta estas características.

Figura 1-1. Diagrama básico de una red de interconexión.

Inicialmente, el interés en estudiar las redes de interconexión fue motivado por el deseo de satisfacer las necesidades de comunicación, principalmente en el área

de la conmutación telefónica. Luego, con el gran crecimiento de la industria de la computación, se empezaron a ver aplicaciones de estas redes dentro de la arquitectura de los computadores, principalmente, en el campo de procesamiento paralelo, por ejemplo, en la comunicación procesador-memoria y entre procesadores.

Posteriormente, con el auge de la conmutación de paquetes de alta velocidad se concluyó que muchas de las redes inicialmente propuestas para procesamiento paralelo también se podían emplear en el diseño de conmutadores de paquetes que operan a alta velocidad, como por ejemplo: conmutadores gigabit Ethernet, enrutadores Terabit, conmutadores ATM, entre otros.[1]

1.1.2 CARACTERÍSTICAS

Antes de pasar a mencionar las características de las redes de interconexión, es necesario aclarar que se emplea el término*nodo*para referirse a cualquier dispositivo que desea conectarse a la red, tal como elementos de comunicación, módulos de memoria, procesadores, etc., así mismo se empleará el término *puerto* para referirse a cada una de las interfaces de entrada o salida, ya sea de la red en general o de cada uno de los elementos de conmutación que la conforman.

Entre las principales características de una red de interconexión se encuentran:

- **Topología:** Estructura de la interconexión física de la red.

- **Enrutamiento:** Método que determina la ruta que sigue un paquete desde un nodo origen a un nodo destino.

- **Control de flujo:**Método que determina en qué momento el paquete se mueve a lo largo de la ruta establecida. Sirve para evitar que los paquetes se mezclen y pierdan, asegurando un movimiento ordenado de los paquetes a través de la red. Si dos paquetes quieren usar la misma ruta al

mismo tiempo, el control de flujo es el encargado de decidir quién usa esa rutay qué pasa con el otro paquete (estrategia de espera).

- **Bloqueo:**Se refiere a que la red tenga o no la capacidad de siempre conectar una entrada con una salida libre, sin importar las conexiones previamente establecidas en la red.

1.1.3 CLASIFICACIÓN

Las redes de interconexión pueden clasificarse de varias maneras tomando en cuenta diferentes características de las mismas. En esta sección se realiza una clasificación general de las redes de interconexión en base a algunos de los parámetros antes expuestos.

1.1.3.1 En base a su topología

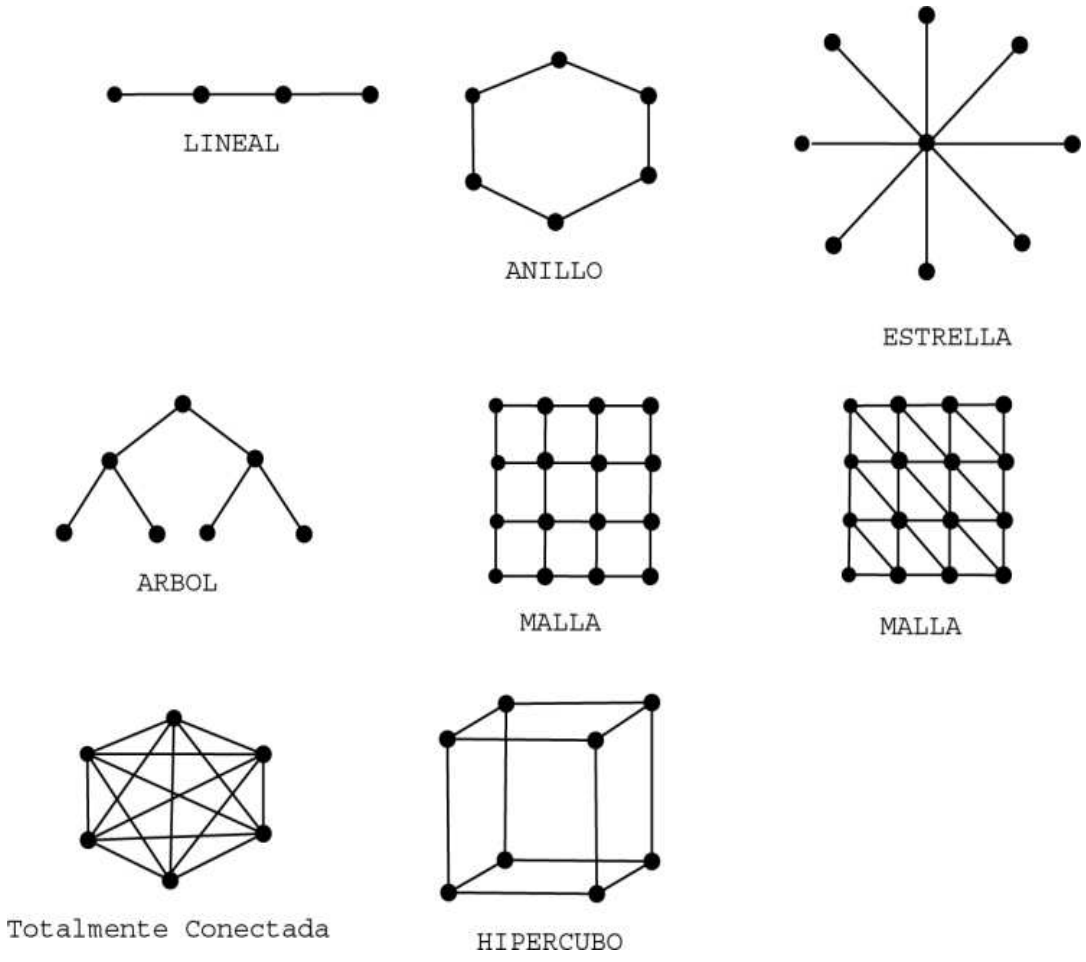
Tomando en cuenta la estructura física de la red, las redes de interconexión se clasifican en: redes estáticas y redes dinámicas.

- *Redes estáticas o directas*

Como su nombre lo sugiere, estas redes se denominan estáticas ya que las conexiones entre nodos no cambian a lo largo del tiempo y son fijas. La principal característica de estas redes es que en ellas existen enlaces punto a punto que conectan directamente cada nodo con un conjunto de otros nodos de la red. En este caso para la comunicación entre nodos no vecinos requiere que la información se envíe a través de nodos intermedios [2]. En la Figura 1-2 se observan ejemplos de este tipo de redes, en esta figura se observan los nodos representados por puntos y las líneas representan los enlaces directos que existen entre nodos adyacentes.

En estas redes, el retardo que experimenta un paquete a lo largo de la red depende de la distancia entre los nodos a comunicarse. Además, estas estructuras estáticas, por lo general, son bastante difíciles de expandir a redes grandes manteniendo la regularidad de la estructura.

En general, el uso de estas redes se ha restringido al área de la computación paralela, específicamente ensistemas multiprocesador con memoria distribuida o también llamados procesadores débilmente acoplados. Estos sistemas constan de un conjunto de procesadores interconectados, cada uno de los cuales tiene su propio módulo de memoria y sus propios canales de entrada/salida[1, 3].



Fuente [6]

Figura 1-2. Ejemplos de redes de interconexión directas.

- *Redes dinámicas o indirectas*

Estas redes tienen la característica de conectar los nodos haciendo uso de un arreglo de conmutadores reconfigurables dinámicamente, ya sea por un procesador central o por un algoritmo distribuido. Esta característica facilita mucho la escalabilidad de estas redes.

Los conmutadores que forman la red, se conectan entre sí por enlaces directos. La comunicación entre cualquier par de nodos requiere que la información pase por uno o más conmutadores, lo que introduce el concepto de redes mono-etapa y redes multi-etapa.

En la Figura 1-3 se aprecian ejemplos de estas redes;en (a) se presenta un ejemplo de red mono-etapa, la red *Crossbar*; mientras que en (b) se tiene un ejemplo de red multi-etapa llamada red Banyan.

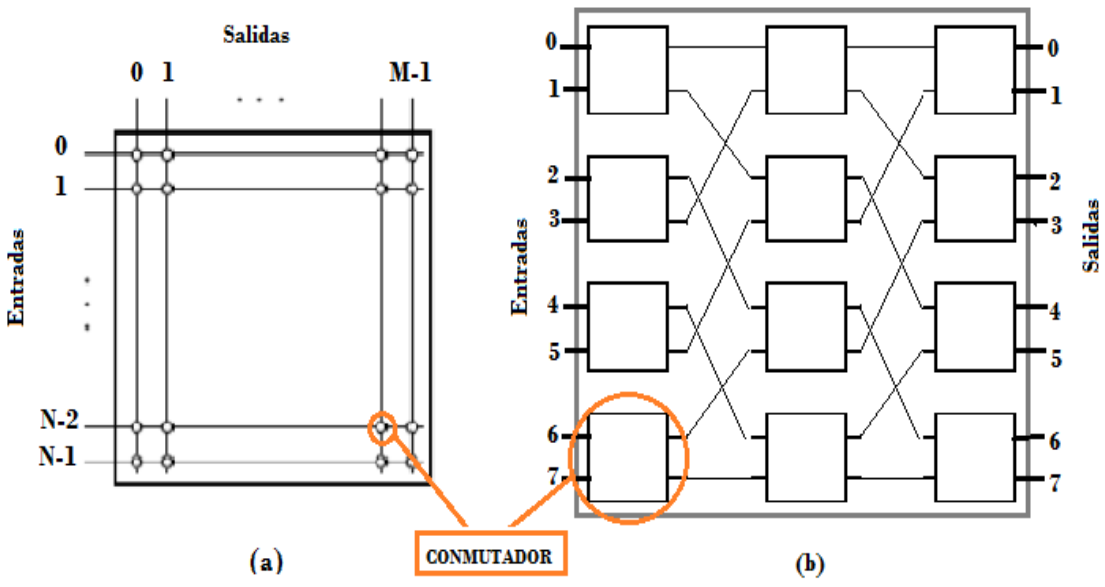


Figura 1-3. Ejemplo de redes indirectas: (a) Red *Crossbar* (b) Red Banyan.

Al hablar de redes indirectas, es necesario empezar por conocer la red *Crossbar*, cuya estructura básica se aprecia en la Figura 1-3 (a). Se trata de una red de interconexión mono-etapa, es decir, que para comunicar cualquier entrada con

cualquier salida, la información solamente debe atravesar por un conmutador en la red. Los conmutadores de esta red solamente tienen dos posibles estados: abierto o cerrado. El uso de un control centralizado permite que la red efectúe todas las conexiones punto-punto (*unicast*) y punto-multipunto (*multicast*). Se debe destacar que se trata de una red no bloqueante, lo cual implica que siempre será posible conectar una entrada con una salida disponible, sin importar las conexiones que estén previamente establecidas en la red.

En una red *Crossbar* con N entradas y M salidas, se requieren$N \ x \ M$puntos de cruce (conmutadores) y, por ende, el hardware requerido para implementar este tipo de red se incrementa rápidamente al aumentar el tamaño de la red, ésta es su mayor desventaja [4].

En cuanto a las redes multi-etapa, las mismas se las tratará con mayor detalle en la Sección 1.2.

1.1.3.2 En base al tipo de enrutamiento

Las redes de interconexión también se las puede clasificar de acuerdo al tipo de enrutamiento que emplean. Si el algoritmo se lleva a cabo en un procesador central, el estado actual de todas las conexiones puede conocerse a fin de tomar las decisiones de enrutamiento. Este es el mecanismo usado por la gran mayoría de conmutadores telefónicos.

Para aplicaciones de conmutación de paquetes a alta velocidad, se prefiere que el algoritmo de enrutamiento sea distribuido a lo largo de toda la red de conmutación y debe ser capaz de operar sin necesidad de conocer el estado actual de toda la red.

Existen principalmente tres clases de algoritmos distribuidos que son importantes: enrutamiento en fuente, auto-enrutamiento y enrutamiento regular.

- **Enrutamiento en fuente:**Requiere que se inserte una etiqueta a cada paquete que ingresa a la red, en la cual se especifican todas o algunas de las decisiones de enrutamiento que se tomarán al interior de la red [5].

- **Auto-enrutamiento:**Se aplica a redes multi-etapa dinámicas. En este caso, también se inserta una etiqueta en la cabecera del paquete indicando el número del puerto de destino requerido. Se lo puede implementar de modo que cada elemento de conmutación de la red tome una simple decisión de enrutamiento en base al bit o bits correspondientes de la etiqueta, independientemente de la posición del elemento de conmutación al interior de la red. Este tipo de algoritmo produce un mínimo retardo de conmutación y se lo puede ejecutar usando hardware bastante sencillo; es por esto que resulta de interés en aplicaciones de conmutación de paquetes a alta velocidad.

- **Enrutamiento regular:**Se aplica a redes estáticas. Aquí cada nodo de la red toma la decisión de enrutamiento basado no solo en la etiqueta con el nodo destino, sino también en la posición del nodo en la red. Por ende, la decisión requiere algo de cálculos e implica retardo [1].

1.1.3.3 En base a la característica de bloqueo

Este es un aspecto de suma importancia en las redes de interconexión, ya que afecta directamente al desempeño dela red ante un tráfico con una distribución aleatoria de destinos para los paquetes.

Tomando en cuenta esta característica, se tiene la siguiente clasificación para las redes de interconexión.

- **No bloqueante:** Se trata de una red que siempre está en capacidad de conectar una entrada con una salida libre, sin importar las conexiones que

ya estén establecidas en la red. Ejemplos de este tipo de redes son: la red *Crossbar* o la red Clos [1].

- **No bloqueante re-ordenable:** En este tipo de red, es posible conectar una entrada con una salida disponible, pero para ello puede ser necesario un reacomodo de las conexiones que estén ya establecidas en la red. Un ejemplo de esta clase de red es la llamada red Benes [6].

- **Bloqueante:** Una red se dice que es bloqueante cuando, para una entrada cualquiera que desea transmitir, puede no ser posible establecer la conexión con el destino deseado, a pesar de que éste se encuentre libre, debido a que las conexiones establecidas previamente en el interior de la red impiden que se establezca una ruta para comunicar estos puertos. Un ejemplo de esto son las redes Banyan.

Las ventajas que presentan las redes no bloqueantes conllevan un incremento en la complejidad de las mismas, así como también un costo elevado, ya que se requieren más elementos de conmutación y más interconexiones en comparación con una red no bloqueante re-ordenable, la misma que a su vez es más compleja y costosa que una red bloqueante.

Finalmente, cabe mencionar que una red no bloqueante re-ordenable solo provee la característica de ser libre de bloqueo si existe un algoritmo centralizado de control para que realice el reordenamiento de las conexiones[1].

En el presente documento, la atención se enfocará en las redes de interconexión multi-etapa (MIN, *Multi-stage Interconnection Network*), y dentro de ellas, en particular, las redes Banyan, las cuales son de tipo bloqueante.

1.2 REDES DE INTERCONEXIÓN MULTI-ETAPA (MIN)

1.2.1 DEFINICIÓN

Una red de interconexión multi-etapa, como lo sugiere su nombre, es una red que se usa para interconectar N entradas con M salidas usando varias etapas intermedias, cada una de las cuales está constituida por un conjunto de elementos de conmutación simples (SE, *Switching Element*). Los elementos de conmutación de etapas adyacentes se interconectan entre sí con enlaces, en una determinada topología [7], como se puede apreciar en la Figura 1-4.

Lo que se puede conectar a los extremos de la red, por lo general, son puertos de entrada/salida de datos en un conmutador o enrutador, así como también pueden ser procesadores en un extremo y un conjunto de módulos de memoria en el otro.

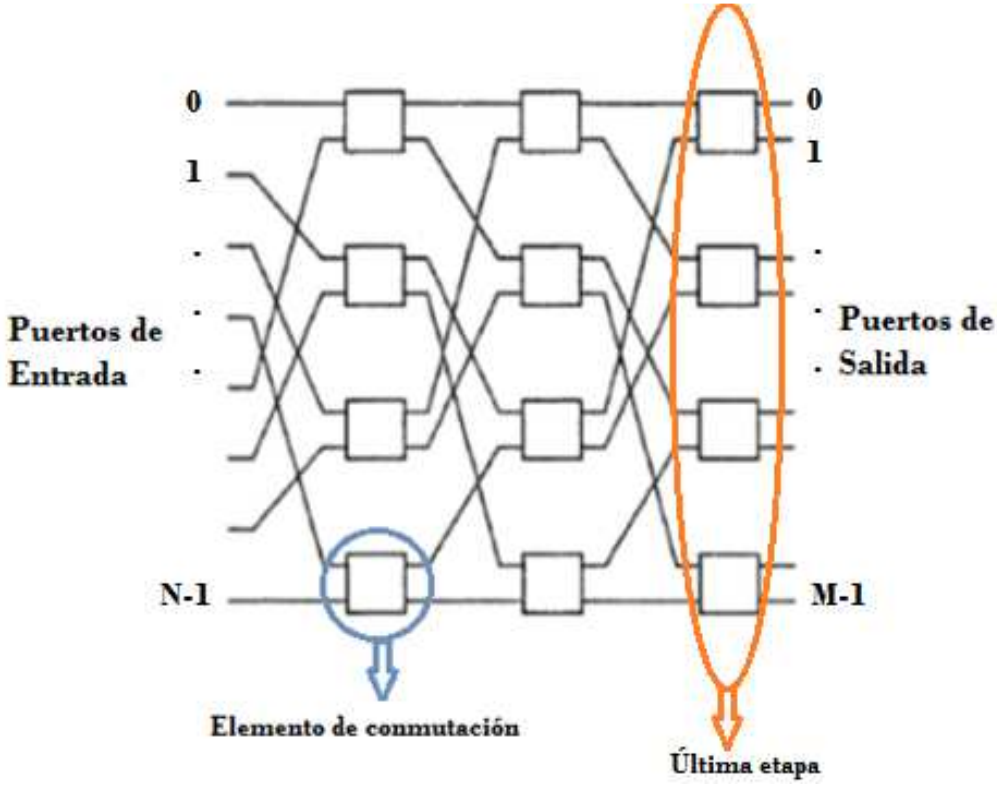


Figura 1-4. Esquema básico de una MIN.

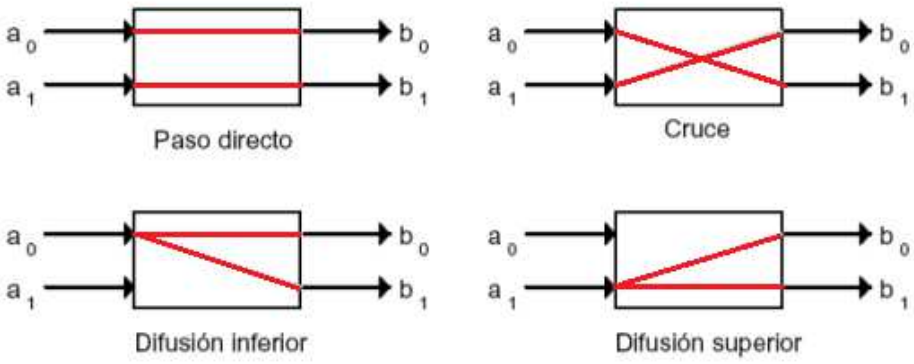
En estas redes, los elementos de conmutación pueden cambiar de estado de forma dinámica a fin de establecer las conexiones requeridas en cada momento.

La principal ventaja de este tipo de redes sobre las redes *Crossbar* es que se requiere menor número de elementos de conmutación para su implementación. Además, estas redes también tienen las ventajas de ser modulares, escalables y tener un costo bajo respecto a otro tipo de redes.

La principal desventaja es que muchas de estas redes son bloqueantes, lo cual disminuye su desempeño, además, también se puede decir que se introduce retardo a los paquetes al atravesar toda la red.

1.2.2 ELEMENTOS DE CONMUTACIÓN

Un elemento de conmutación (SE)de tamaño $p \times q$, es un circuito con *p* entradas y *q* salidas que puede conectar cualquiera de sus entradas con una o varias de sus salidas. En la Figura 1-5 se observan las posibles configuraciones para un conmutador de dos entradas y dos salidas.



Fuente [20]

Figura 1-5. Posibles configuraciones de un elemento de conmutación 2X2.

Debido a que, actualmente la gran mayoría de información se transmite en formato digital, específicamente usando valores binarios; generalmente, en estos conmutadores se tiene el mismo número de puertos de entrada y salida y los

mismos están en un número que es potencia de 2 ($p=q=2^r$, $r>0$). Además, en ningún caso se permite la conexión simultánea de una salida con varias entradas. En el caso del elemento de conmutacióntipo *Crossbar*, que es el más usado, solamente se permite la conexión de cada entrada con una sola salida [4].

Las redes de interconexión multi-etapa, como ya se mencionó, se implementan con elementos de conmutación dispuestos en etapas, los elementos de conmutación de una etapa cualquiera se interconectan con los de las etapas adyacentes. Estos elementos de conmutación generalmente se los construye en base a conmutadores de tipo *Crossbar* de bajo grado (pocos puertos de entrada/salida).

Si bien la estructura mostrada en la Figura 1-5 es el esquema básico de un elemento de conmutación, en la Sección 1.3.6.1 se verá que también es posible implementar elementos de conmutación con buffers tanto a la entrada como internos;además, en el Capítulo 2 de este proyecto se analizará en detalle, tanto la estructura como el funcionamiento de un tipo especial de elemento de conmutación llamado elemento de conmutación Banyan dual.

1.2.3 PERMUTACIONES

Como se conoce, en sentido general, una permutación es un arreglo de elementos en donde lo que interesa es la posición que ocupa cada uno de los elementos que constituyen dicho arreglo.

Para el contexto de las redes de interconexión multi-etapa, las permutaciones constituyen las formas cómo se interconectan los puertos de salida de los conmutadores de la etapa i con los puertos de entrada de los conmutadores de la etapa i+1. En otras palabras, las permutaciones son los patrones de interconexión entre etapas, aunque cabe mencionar que estas permutaciones pueden también estar presentes a la entrada y/o salida de la red.A continuación, se discutirán brevementealgunas de las permutaciones que se usan más comúnmente en las redes de interconexión multi-etapa.

- **Permutación *Perfect Shuffle* (σ)**

Esta es quizá la permutación más usada en las redes de interconexión. Su nombre se deriva de la acción de barajar un mazo de cartas, en la cual se divide al mazo en dos mitades y se va alternadamente tomando una carta de cada mitad, consiguiéndose así que cartas inicialmente adyacentes ahora estén separadas.

Ahora, se aplicará esta misma idea para realizar las conexiones entre los puertos de salida de los SE de la etapa genérica i con los puertos de entrada de los SE de la etapa siguiente, i+1. Los pasos a seguir para realizar esta permutación se los enumera a continuación y también se los ilustra en la Figura 1-6:

1. Se toman los puertos de salida de los SE de la etapa i y se los divide en dos grupos iguales.

2. Se numeranestos puertos de salida de la etapa i en cada mitad generada, empezando desde los puertos superiores.

3. Se numeran también los puertos de entrada de los SE de la etapa i+1, empezando desde los puertos superiores (Figura 1-6 (a)).

4. Se realizan las conexiones de la siguiente manera: el primer puerto de salida de la mitad superior con el primer puerto de entrada de la siguiente etapa; el primer puerto de salida de la mitad inferior con el segundo puerto de entrada (Figura 1-6 (b)); el segundo puerto de salida de la mitad superior con el tercer puerto de entrada, y así sucesivamente hasta tener todos los puertos conectados, como se observa en la Figura 1-6 (c) y (d).

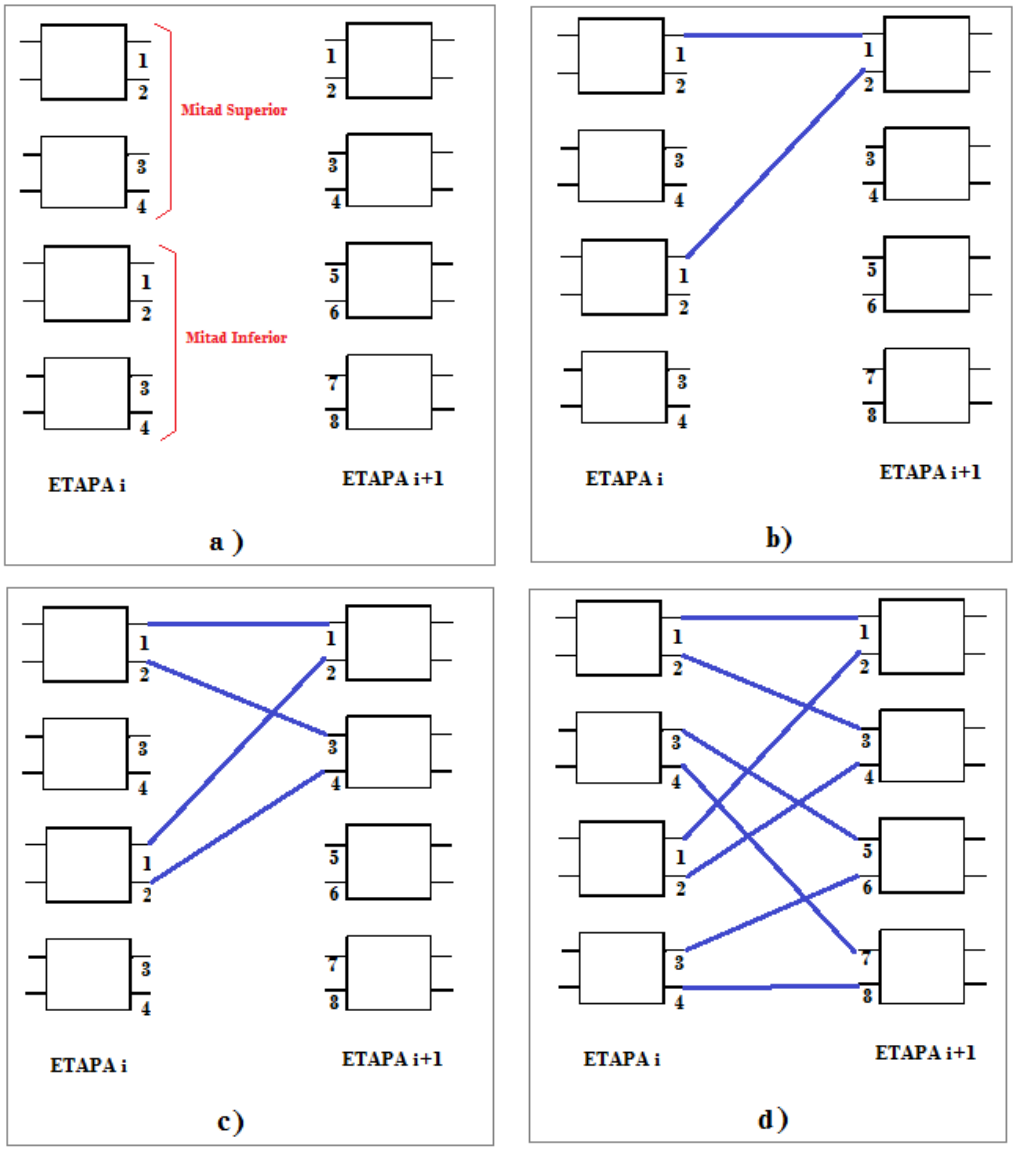


Figura 1-6. Procedimiento para realizar la permutación *Perfect Shuffle*.

Otro procedimiento mediante el cual es posible realizar esta permutación se detalla a continuación y se puede observar en la Figura 1-7:

1. Se numeran, tanto los puertos de salida de la etapa i, así como los puertos de entrada de la etapa i+1 en binario.

2. Se rotan los bits correspondientes al puerto de salida hacia la izquierda de forma circular, de modo que el bit más significativo pasa a ocupar la posición del bit menos significativo.

3. El resultado obtenido de esta rotación, corresponde al valor binariodel puerto de entrada de la siguiente etapa con el cual se debe conectar el puerto de salida en cuestión [4].

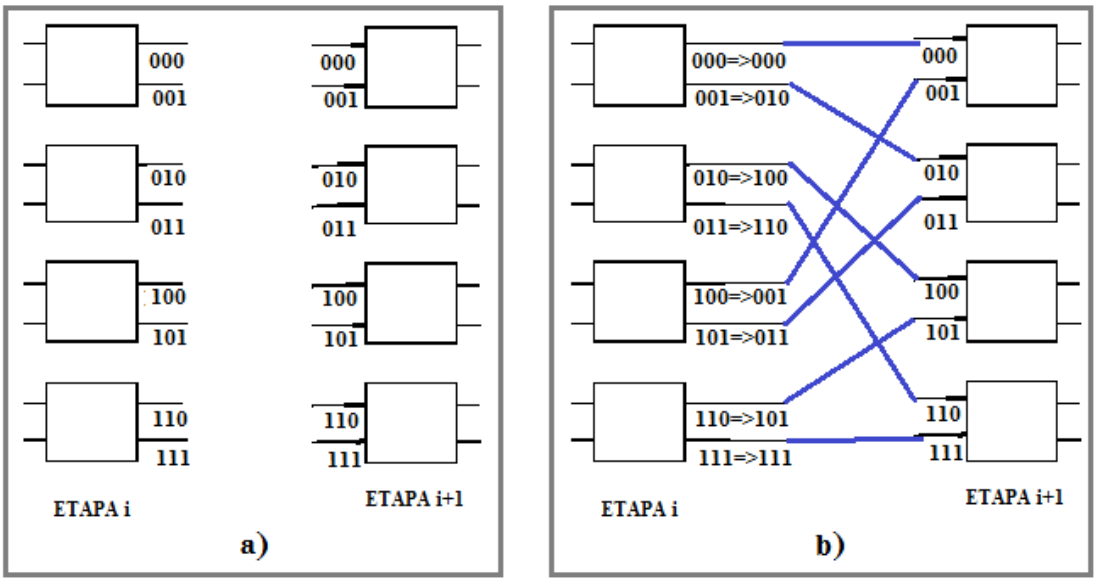


Figura 1-7. Permutación *Perfect Shuffle* con 8 elementos.

- **Permutación *Perfect Shuffle* Inversa(σ^{-1})**

Esta es otra permutación bastante común en las redes de interconexión multi-etapa. Como es de esperarse, esta permutación se la realiza rotando hacia la derecha los bits correspondientes al puerto de salida, y de esta forma, se obtiene el número del respectivo puerto de entrada de la etapa siguiente con el cual se debe realizar la conexión. Este procedimiento, para el caso de 8 elementos, se encuentra ilustrado en la Figura 1-8.

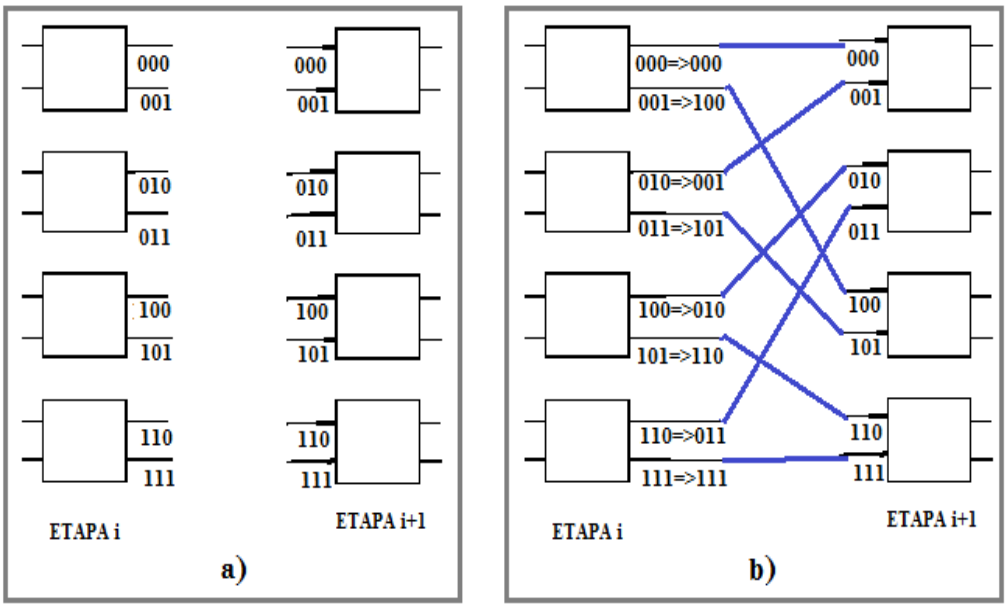


Figura 1-8. Permutación *Perfect Shuffle* Inversa con 8 elementos.

- **Permutación *Butterfly(β)***

Es otro tipo de permutación bastante utilizada para redes de interconexión multi-etapa. En el caso de esta permutación, para 2^n elementos, se pueden realizar n-1 diferentes configuraciones de esta permutación, denominándose cada una de ellas, permutación *Butterfly* de orden *k* (*0<k<n*).

La permutación *Butterfly* de 2^n elementos, de orden *k* (*0<k<n*)se la realiza de la siguiente manera:

1. Se numeran tanto los puertos de salida de la etapa i como los puertos de entrada de la etapa i+1 en binario.

2. Se intercambian los bits 0 con los bits *k*, empezando desde el menos significativo, del número de puerto de salida y así se obtiene el número correspondiente al puerto de entrada de la etapa siguiente con que se debe conectar.

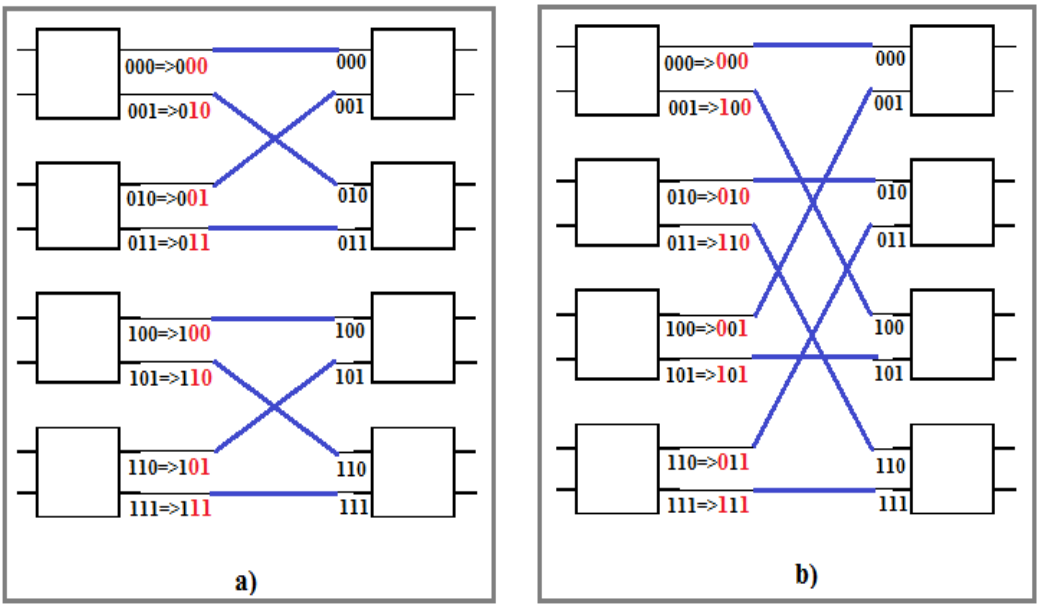


Figura 1-9. Permutaciones *Butterfly* de 8 elementos de: a) orden 1 y b) orden 2.

En la Figura 1-9 se puede apreciar la realización de las permutaciones *Butterfly* de 8 elementos de orden 1 y 2, en esta figura se han resaltado los bits intercambiados en cada caso.

Cabe mencionar que esta permutación tiene la propiedad de ser simétrica, es decir que es igual a su inversa, entendiéndose por permutación inversa, a la permutación que resulta como reflejo de la permutación original [4].

- **Permutación *Bit Switch(δ)***

Esta permutación, junto con la permutación *Bit Reversal,* son permutaciones usadas con menor frecuencia; su principal aplicación es en la obtención de redes equivalentes, como se verá en la Sección 1.3.5.

En esta permutación, el número de puerto de entrada que se debe conectar al puerto de salida a viene definido por la siguiente función:

$$\delta(a_{n-1} \dots a_0) = a_1 a_2 \dots a_{n-1} a_0 \quad (Bit\ switch)$$

Representando como N al número de puertos, se tiene $a = a_{n-1} \dots a_0$ $(n = log_2 N)$,que representa el número del puerto de salida expresado en base 2 y a_{n-1} es el bit más significativo [8]. En la Figura 1-10 (a) se puede apreciar la realización de esta permutación para N=16.

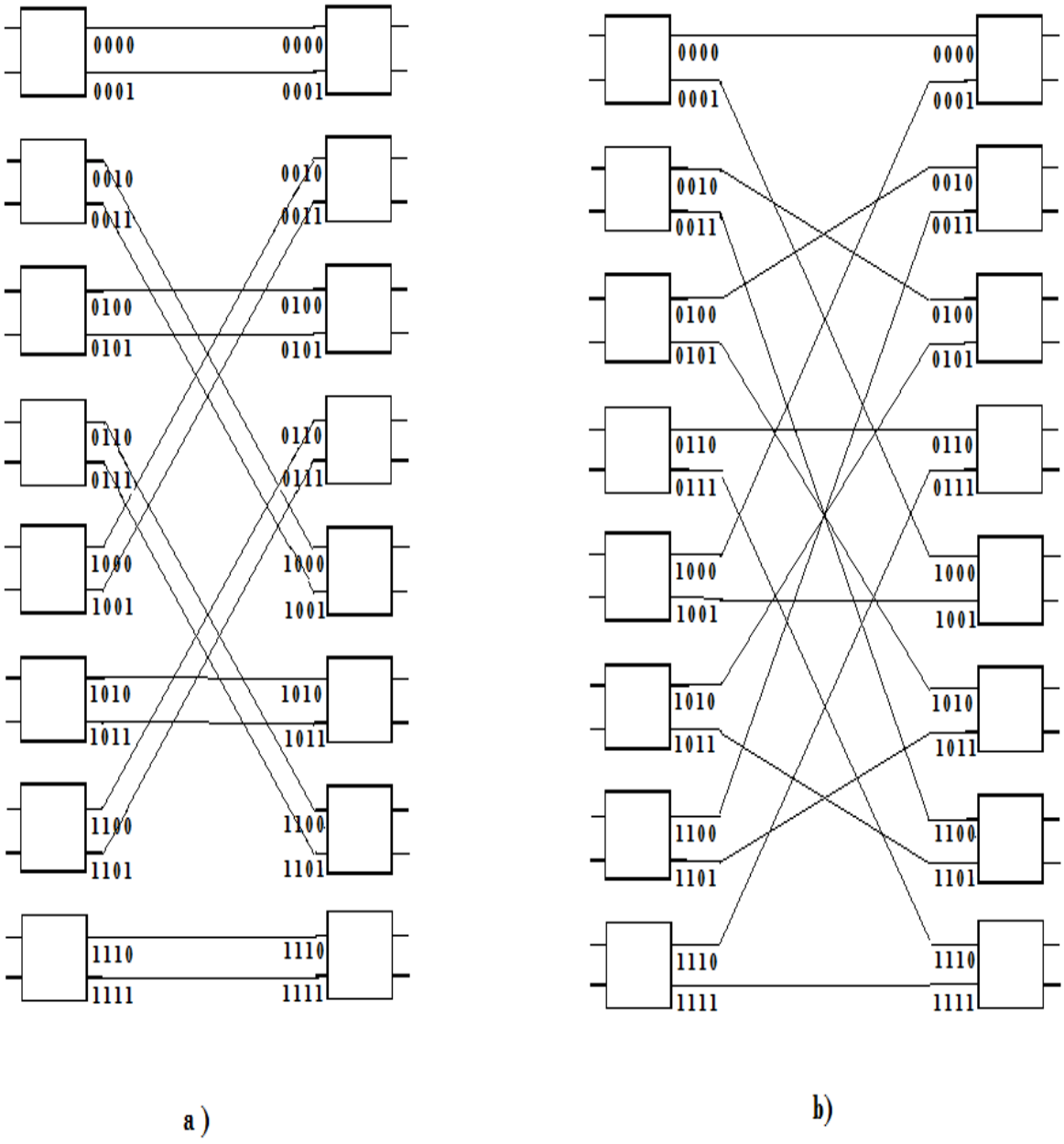


Figura 1-10. Permutaciones a) *Bit switch* y b) *Bit reversal* para N=16.

- **Permutación *Bit Reversal (ρ)***

Al igual que la permutación anterior, el campo de aplicación de esta permutación se limita a la obtención de redes equivalentes. Tomando en cuenta las mismas consideraciones que para la permutación anterior, la función que define el número de puerto de entrada que se debe conectar al puerto de salida a, en este caso es:

$$\rho(a_{n-1} \dots a_0) = a_0 a_1 \dots a_{n-2} a_{n-1} \quad (Bit\ reversal)$$

El resultado de aplicar esta función para N=16, se observa en la Figura 1-10 (b).

Una vez expuestos estos conceptos básicos sobre redes de interconexión, se enfocará la atención en una clase específica de MIN, las redes Banyan.

1.3 REDES BANYAN

1.3.1 DEFINICIÓN

Son un tipo de red de interconexión multi-etapa que consta de conmutadores *Crossbar*simples como elementos de conmutación, los cuales se interconectan entre sí conforme a un determinado patrón o permutación entre cada etapa de la red. Su nombre lo obtienen de una especie de árbol de higo(Figura 1-11), en el cual sus ramas se entrelazan formando patrones muy particulares y complejos. Las interconexiones entre las etapas de las redes Banyan se supone que se asemejan a esos patrones, de ahí el nombre.

Las redes Banyan se definen básicamente como redes de interconexión que proporcionan una y solo una ruta posible entre un puerto de entrada y uno de salida [9],de modo que, con esta definición, se abarca un gran grupo de posibles estructuras de redes; sin embargo muchas de ellas son equivalentes entre sí, esto se explicará en la Sección 1.3.5.

Fuente [24]

Figura 1-11. Árbol Banyan.

Generalmente, tanto el número de puertos de entrada y salida de la red, como el tamaño de los elementos de conmutación es potencia de 2. Un ejemplo de una red Banyan 8x8 se presenta en la Figura 1-12, en la cual se observan los puertos de entrada y salida, así como los elementos de conmutación de tamaño 2x2.

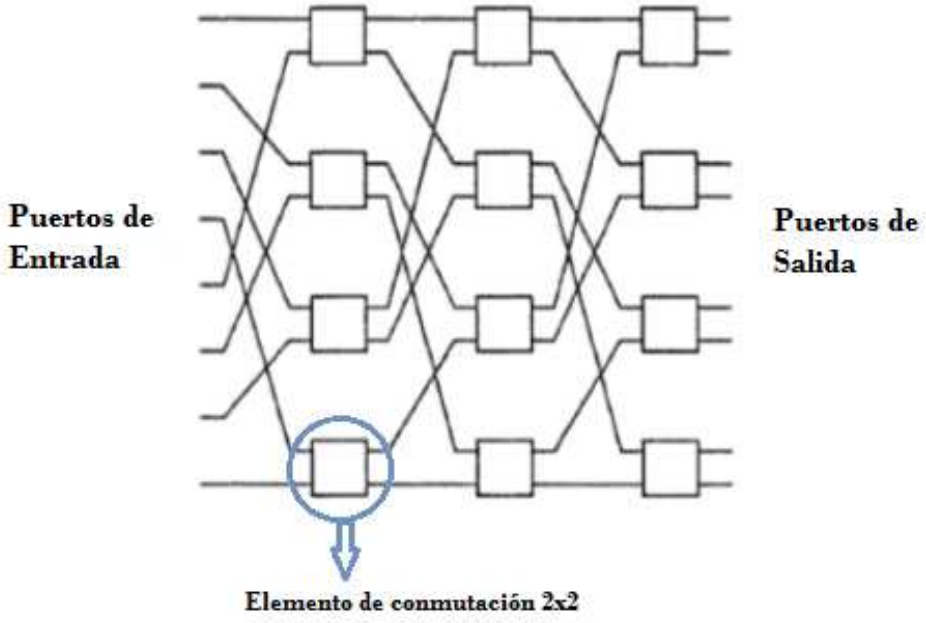


Figura 1-12. Red Banyan 8x8.

1.3.2 CARACTERÍSTICAS

Las redes Banyan tienen muchas ventajas que las hacen atractivas para aplicaciones en procesamiento paralelo y diseño de conmutadores de paquetes de alta velocidad. A continuación se mencionan las principales características, tanto positivas como negativas, de estas redes.

- **Sencillez:** En una red de tamaño $N \times N$, implementada con elementos de conmutación de tamaño $k \times k$, se requieren solamente $\log_k N$etapas, con $\frac{N}{k}$ elementos de conmutaciónen cada una, en contraste con los conmutadores *Crossbar*, los cuales requieren N^2 conmutadores.

- **Auto-enrutamiento:** Esto quiere decir que cada elemento de conmutación que recibe un paquete en cualquiera de sus puertos de entrada, puede decidir hacia cuál de sus puertos de salida enviar dicho paquete basándose solamente en la dirección de destino del paquete en cuestión. Esta propiedad de auto-enrutamiento es muy importante ya que implica que no es necesaria la implementación de un complejo algoritmo de enrutamiento centralizado [7, 9].

- **Bi-direccionalidad:** Esto significa que las redes Banyan mantienen la propiedad de auto-enrutamiento en ambas direcciones, es decir, el camino desde un puerto destino, de regreso hacia un puerto origen se puede obtener basado en la dirección del puerto de origen al que se desea llegar. Esto también constituye una ventaja, dado que en muchas aplicaciones, el tráfico que atraviesa la red es bidireccional, ya que los puertos de origen envían los paquetes pero los puertos destino envían confirmaciones o información de regreso [10].

- **Regularidad y Escalabilidad:** Al decir que se trata de redes regulares, se quiere decir que todos los elementos de conmutación son idénticos. Este hecho, junto a sus patrones de interconexión, son características muy

beneficiosas al momento de implementar estas redes en VLSI (*Very Large Scale Integration*) ya que se trata de redes totalmente escalables.

- **Retardo:** Se puede mencionar como otra ventaja de estas redes, el hecho de que se tiene igual retardo entre cualquier par de puertos de entrada y salida. Aunque, por tratarse de redes de interconexión multi-etapa, se puede alegar que el retardo se incrementa al aumentar el tamaño de la red (número de etapas).

- **Bloqueo:** Como se mencionó anteriormente, esto quiere decir, que una salida libre puede no estar disponible para un puerto de entrada cualquiera, debido a las conexiones previamente establecidas en el interior de la red. Esto implica que si dos o más paquetes entran en contención por el mismo enlace o puerto en la red, solamente uno de ellos será transmitido. Los otros paquetes deberán ser almacenados y reintentar la transmisión en el siguiente ciclo (asumiendo que la red no descarta los paquetes, siempre y cuando haya espacio en los buffers) [11, 12].

- **Sensibilidad a fallos:** Debido a que, en las redes Banyan, existe una sola ruta posible entre cada par de puertos de entrada/salida, si una de las conexiones se corta, no será posible disponer de muchas de las rutas.

1.3.3 ALGORITMO DE AUTO-ENRUTAMIENTO

Sea N el número de puertos de entrada y salida de una red Banyan, formada con conmutadores de tamaño $k \times k$, entonces se tiene $n = \log_k N$ etapas, cada una formada con $\frac{N}{k}$ conmutadores.

Para empezar,lo que se hace es expresar la dirección del puerto destino en base k, con lo cual se requerirán$n = \log_k N$ dígitos para representarlo, un dígito por cada etapa de la red. Este número se añade como una etiqueta en la cabecera de cada paquete.

Dado que, en la actualidad, casi la totalidad de la información se halla en formato digital, específicamente, en binario, de aquí en adelante se asumirá que $k = 2$, con lo cual se tienenelementos de conmutación de tamaño 2x2, y la dirección de destino se la expresa en binario, $d = d_{n-1},\ d_{n-2}, \dots, d_0$, siendo d_{n-1} el bit más significativo.Además, sea *h* el número de etapa actual, esto es $h = 1, 2, \dots, n$.

Entonces, cada conmutador de la etapa *h,* puede decidir el puerto por el cual enviar el paquete solamente basado en un bit de la dirección destino. Si este bit es 0, envía el paquete por el puerto 'superior' y si el bit es 1, el paquete es enviado por el puerto 'inferior' [7, 9].

En la mayoría de redes Banyan, el bit de enrutamiento a tomar en cuenta en cada etapa viene dado por: d_{n-h}, con lo cual resulta que se empieza a tomar desde el bit más significativo de la etiqueta para la primera etapa, el siguiente bit para el segunda etapa, y así sucesivamente hasta que el bit menos significativo se lo emplea en la última etapa.

Cuando cada etapa usa el bit correspondiente, dicho bit se lo puede retirar de la cabecera o se puede rotar toda esa etiqueta a fin de que siempre el primer bit sea el que la siguiente etapa de conmutación vaya a usar.

La excepción a esta regla es la red *Butterfly,* llamada también SW-Banyan, la cual se analizará en la Sección 1.3.4.3, en esta red, el bit a usar en la etapa *h,* viene dado por: d_{hmodn}.

El patrón de interconexión existente entre etapas adyacentes en las redes Banyan está arreglado de tal forma que el paquete saldrá por el puerto de destino correcto sin importar el puerto de entrada en el cual se originó, siempre y cuando no sea bloqueado en algún lugar al interior de la red. En otras palabras, son justamente las permutaciones que existen en las redes Banyan, las que dan origen a la propiedad de auto-enrutamiento[4, 10].

En la Tabla 1-1 se resumen las reglas de enrutamiento para las cuatro topologías de redes Banyan más comunes, las mismas que se analizarán en la siguiente sección, así como también para sus inversas,entendiéndose como topología inversa de una red, la topología obtenida como el reflejo de la misma red[8].

Cabe mencionar que, en la Tabla 1-1 se muestran las reglas de enrutamiento tanto para ir de un puerto de entrada hacia un puerto de salida (conexión $\mathrm{I} \rightarrow \mathrm{O}$), así como también para ir de un puerto de salida, de regreso hacia un puerto de entrada (conexión $\mathrm{O} \rightarrow \mathrm{I}$).

Tabla 1-1. Reglas de enrutamiento en redes Banyan.

Topología	**Bit de auto-enrutamiento (I→O)**	**Bit de auto-enrutamiento (O→I)**
Omega	d_{n-h}	d_{n-h}
Omega^{-1}	d_{h-1}	d_{h-1}
SW-Banyan	$d_{h \bmod n}$	d_{h-1}
SW-Banyan^{-1}	d_{n-h}	$d_{(n-h+1) \bmod n}$
n-cúbica	d_{n-h}	d_{n-h}
n-cúbica^{-1}	d_{h-1}	d_{h-1}
Baseline	d_{n-h}	d_{h-1}
Baseline^{-1}	d_{n-h}	d_{h-1}

Fuente[8]

1.3.4 PRINCIPALES TOPOLOGÍAS DE LAS REDES BANYAN

En la literatura se han descrito varias topologías de redes Banyan, las cuales difieren unas de otras en las permutaciones existentes entre SE de etapas adyacentes, así como las permutaciones que pueden existir a la entrada y/o salida de la red.

A continuación se analizan cuatro topologías bastante comunes en este tipo de redes. Aunque se han definido otras topologías, no se las estudiará ya que son

equivalentes a las mencionadas a continuación, en el sentido que, al aplicar la permutación adecuada a la entrada y/o salida de la red, una topología puede obtenerse a partir de otra, como se verá en la Sección 1.3.5.

1.3.4.1 Red Omega(Ω)

Poco tiempo después que se definieron las redes Banyan, también se introdujo la red Omega, que fue la primera red de interconexión multi-etapa en demostrar la propiedad de auto-enrutamiento[1]. La característica que la identifica es el patrón de interconexión presente tanto a la entrada de la red, así como también entre todas las etapas adyacentes, el cual es el *Perfect Shuffle*[4, 13]. A la salida de la última etapa ya no se aplica este patrón, como se observa en la Figura 1-13.

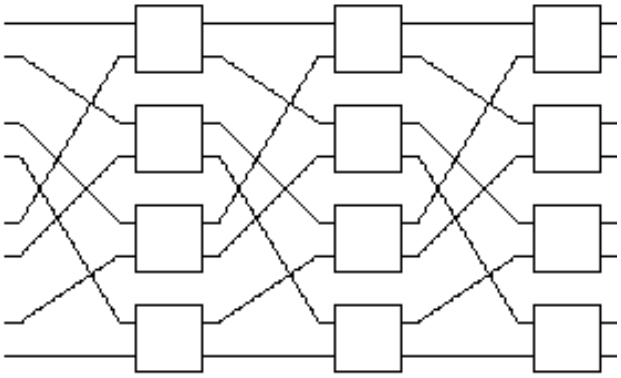


Figura 1-13. Red Omega 8x8.

1.3.4.2 Red *Baseline*(Φ)

Esta red tiene la característica de ser modular, es decir que se la puede generar de forma recursiva como se observa en la Figura 1-14 (para el caso de una red de tamaño 16 x 16). El procedimiento a seguir para generar esta red es el siguiente:

1. De acuerdo al tamaño de la red a implementar, se determina el número de etapas a usar y el correspondiente número de elementos de conmutación en cada etapa.

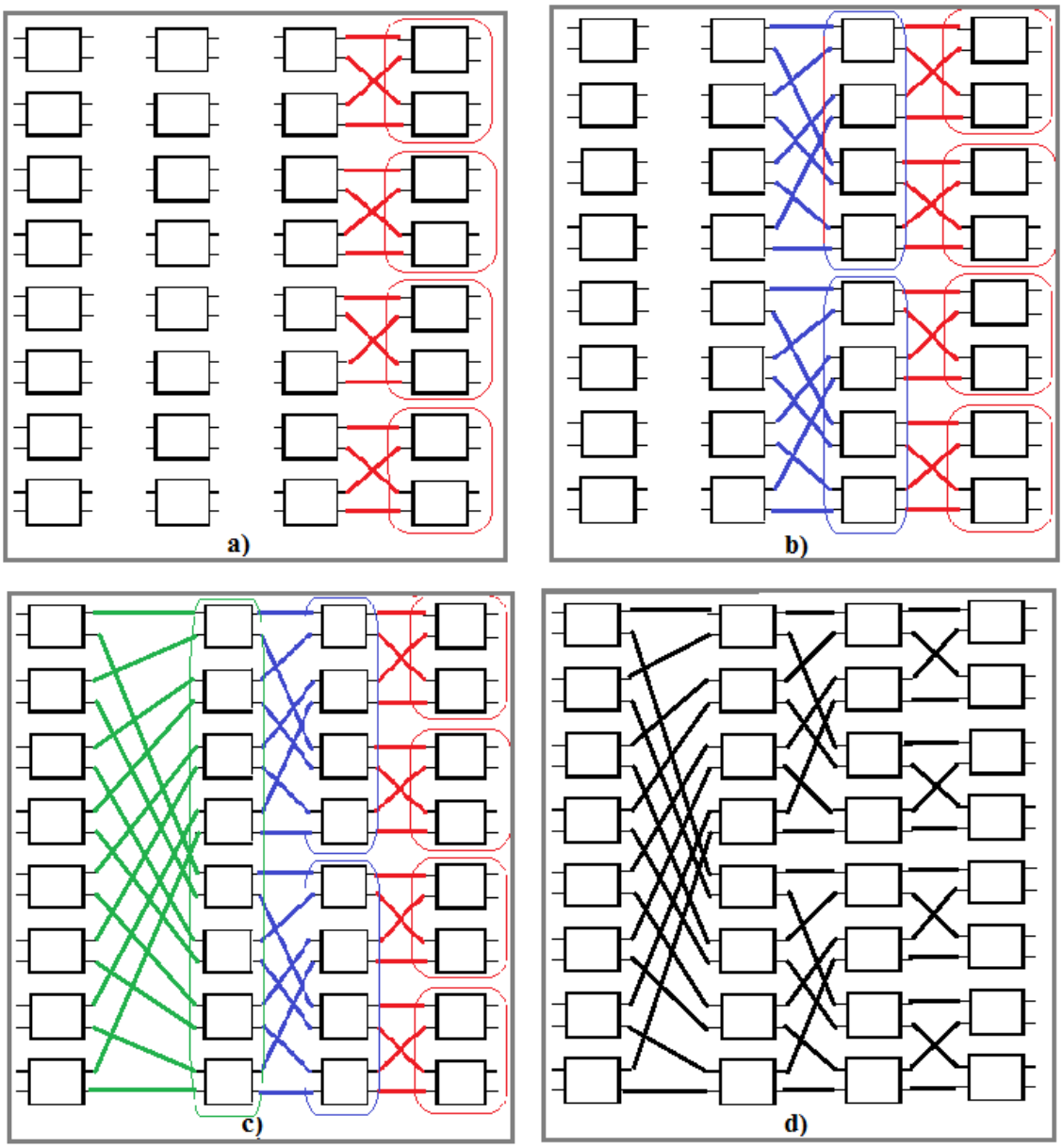


Figura 1-14. Construcción Recursiva de Red *Baseline* 16x16.

2. Se unen las entradas de cada par de elementos de conmutación de la última etapa mediante un *Perfect Shuffle*Inverso con las salidas de los conmutadores de la etapa anterior (Figura 1-14(a)).

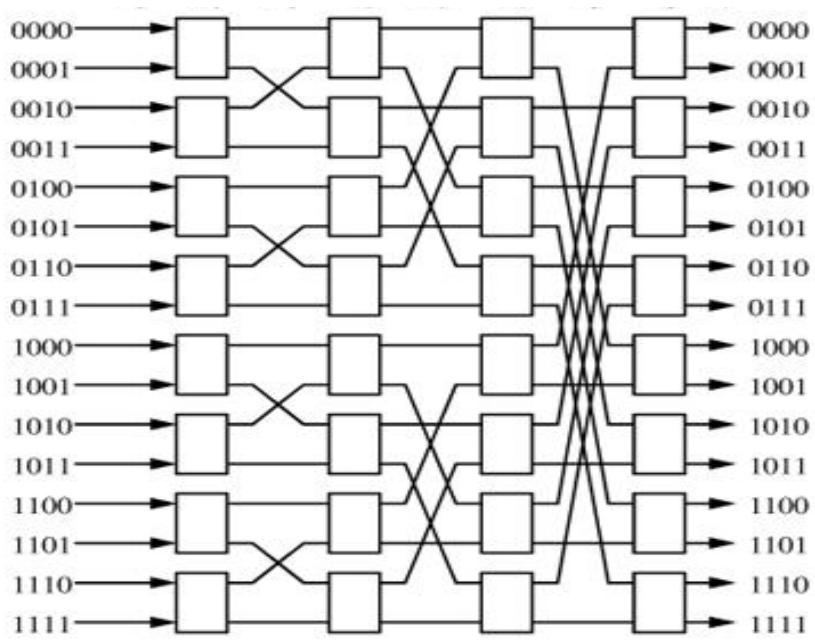
3. A continuación, se agrupan las entradas del doble número de conmutadores que en la etapa anterior y se vuelve a realizar un *Perfect*

Shuffle Inverso hacia las salidas de los conmutadores de la etapa anterior (Figura 1-14(b)).

4. Este proceso se repite hasta tener finalmente un solo *Perfect Shuffle* Inverso que una las salidas de todos los conmutadores de la que será la primera etapa(Figura 1-14 (c, d)) [4].

1.3.4.3 Red *Butterfly*(Σ)

Una red *Butterfly* de tamaño $2^n \times 2^n$ está formada por *n* etapas, con 2^{n-1}conmutadores de tamaño2×2 en cada etapa.Estas etapas se interconectan usando *n-1* permutaciones *Butterfly* de base 2 y orden desde 1 hasta *n-1*. En la Figura 1-15 se puede apreciar una red *Butterfly* de tamaño $2^4 \times 2^4$.



Fuente [26]

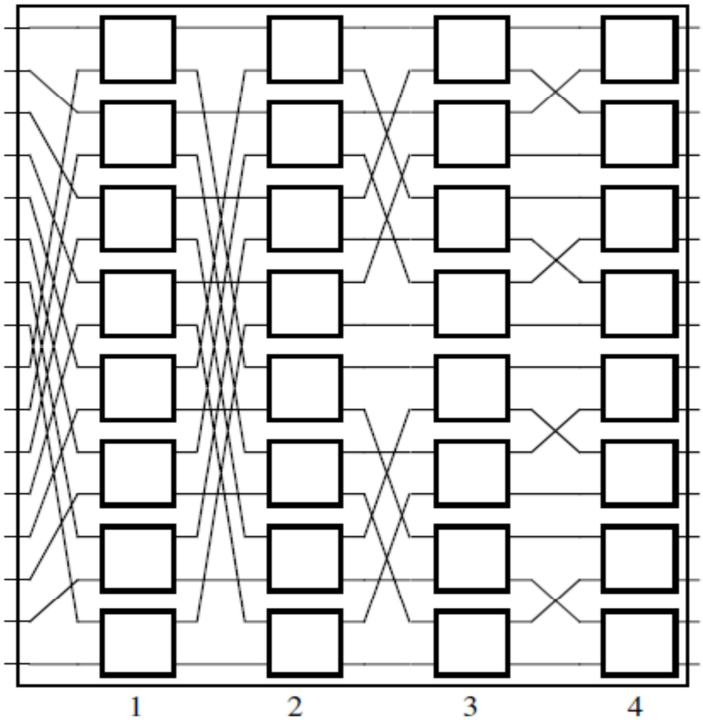
Figura 1-15. Red *Butterfly* 2^4x2^4.

Esta red toma su nombre debido a los patrones de interconexión entre etapas, los mismos que se supone son semejantes a las alas de una mariposa.

Cabe mencionar que esta topología también es modular, ya que una red *Butterfly* de tamaño$2^n \times 2^n$ se puede construir en base a dos redes *Butterfly* de tamaño $2^{n-1} \times 2^{n-1}$, añadiendo una etapa final de2^{n-1} conmutadores la cual sirve para unir las dos redes de menor tamaño. Estas redes se unen a la última etapa añadida mediante una permutación *Butterfly* de base 2 y orden *n-1*[4].

1.3.4.4 Red n-Cúbica(Γ)

En esta red, el parámetro *n* indica el número de etapas que constituyen la red. Su estructura se puede ver en la Figura 1-16, en la cual se aprecian los patrones de interconexión que la caracterizan, los cuales son: a la entrada de la red un *Perfect Shuffle*, mientras que entre las etapas adyacentes se aplica las permutaciones *Butterfly*, pero de orden decreciente en cada etapa, empezando con una permutación de orden (n-1) luego de la primera etapa, hasta terminar en orden 1.

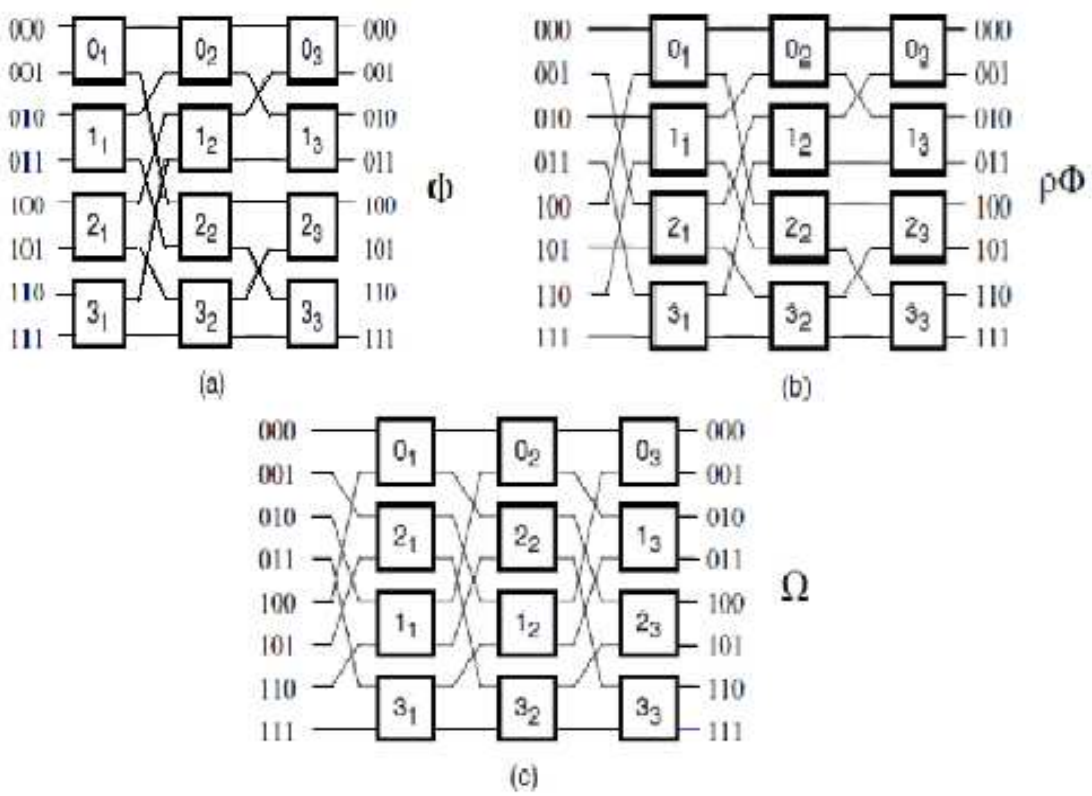


Fuente [26]

Figura 1-16. Red 4-cúbica.

1.3.5 EQUIVALENCIA ENTRE REDES BANYAN

En la literatura, se han definido varias relaciones de equivalencia entre redes haciendo uso de la teoría de grafos[1]. De lo cual se puede encontrar mayor detalle en el Anexo 1.En el presente documento, la atención está enfocada en las redes Banyan y, específicamente, en las cuatro topologías descritas en la sección anterior; por ello, solamente se dirá que es posible demostrar que estas cuatro topologías de red son equivalentes entre sí, en el sentido que una red Banyan B puede obtenerse en base a otra red Banyan A de las mencionadas, simplemente moviendo los SE de A de tal manera que la red obtenida tenga los mismos patrones de interconexión entre etapas que la red B [8].



Fuente [8]

Figura 1-17. Equivalencia entre redes Omega y *Baseline*.

Por ejemplo, si la red A es una red *Baseline* (Φ) y la red B a obtenerse es la red Omega (Ω), ambas de tamaño 8x8, la Figura 1-17 muestra el proceso como

[1]**Teoría de Grafos:** Campo de estudio de las matemáticas y las ciencias de la computación, que estudia las propiedades de los grafos.

obtener dicha red. Como se puede ver, es necesario a la entrada de la red *Baseline* añadir primero la permutación ρ (Bit reversal) (Figura 1-17 b) y luego mover los SE de las etapas 1 y 2 a fin de obtener la topología de la red Omega (Figura 1-17 c).

En la Tabla 1-2 se muestranlas permutaciones que se deben añadir a la entrada y/o salida de cada una de las cuatrotopologías de red Banyan analizadas anteriormente, para que estas redessean equivalentes entre sí. La representación usada es la siguiente: Omega (Ω), SW-Banyan (Σ), n-cúbica(Γ), *Baseline*(Φ).

De acuerdo a la Tabla 1-2, se tiene que, en realidad, la red Omega y la red n-cúbica son equivalentes sin necesidad de añadir permutación alguna. De hecho es fácilmente verificable que la una topología se puede obtener a partir de la otra simplemente moviendo adecuadamente los SE de las etapas intermedias.

Tabla 1-2. Equivalencia entre Redes Banyan.

	Ω	Σ	Γ	Φ
Omega	Ω	ρΣδ	Γ	ρΦ
Omega^{-1}	ρΩρ	Σδρ	ρΓρ	Φρ
SW-banyan	ρΩδ	Σ	ρΓδ	Φδ
SW-banyan^{-1}	δρΩ	δΣδ	δρΓ	δΦ
n-cube	Ω	ρΣδ	Γ	ρΦ
n-cube^{-1}	ρΩρ	Σδρ	ρΓρ	Φρ
Baseline	ρΩ	Σδ	ρΓ	Φ
Baseline^{-1}	ρΩ	Σδ	ρΓ	Φ

Fuente [8]

1.3.6 DESEMPEÑO

En forma general, en lo referente a conmutadores de paquetes de alta velocidad se tiene que hay dos componentes fundamentales que influyen de forma directa en su implementación y desempeño: la conmutación y el almacenamiento (*buffering*).

Hablando específicamente de las redes Banyan, se tiene que, su desempeño es independiente de la topología específica, siempre y cuando se tenga un tráfico igualmente distribuido entre los puertos de entrada y salida, es decir, que un paquete que llega a cualquier puerto de entrada de la red, tiene la misma probabilidad de ir dirigido a cualquiera de las puertos de salida[18].

En cuanto a la conmutación en estas redes, como se mencionó en la Sección 1.3.2, debido a la propiedad de auto-enrutamiento, se tiene un retardo por conmutación mínimo, por tratarse de un algoritmo de enrutamiento distribuido.

En cuanto al almacenamiento, la implementación de buffers es una de las estrategias que se puede usar a fin de reducir la probabilidad de bloqueo en estas redes, aunque esta solución suele ser un tanto costosa [14]. A continuación se analizan varias de las técnicas usadas con el fin de mejorar el desempeño de las redes Banyan.

1.3.6.1 Almacenamiento (*Buffering)*

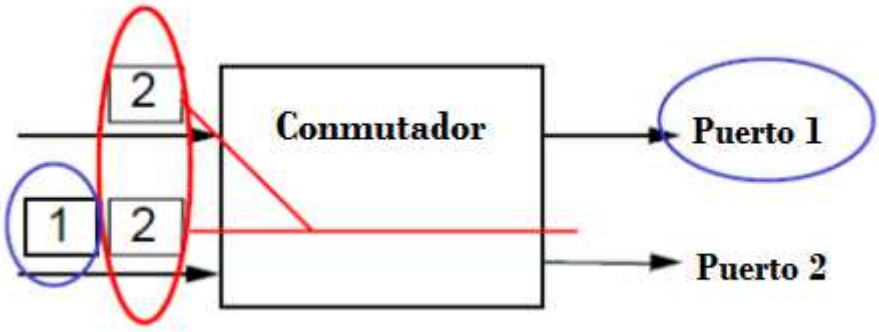
Si los buffers se hallan ubicados fuera de la red de conmutación, se dice que la red tiene buffers externos. Así mismo, si los buffers se implementan en el interior de la red de conmutación, es decir, en cada SE, se dice que la red tiene buffers internos.

Otra cuestión a considerar es no solo la ubicación de los buffers, sino también su tamaño, es decir, la cantidad de paquetes que pueden almacenar. Ambos aspectos afectan de forma muy notoria el desempeño de la red de conmutación.

- *Buffers Externos*

Dentro de las redes con buffers externos existe la posibilidad de colocar buffers a la entrada de la red o colocarlos a la salida de la misma.

Un diseño que usa buffers a la entrada de la red de conmutación es más sencillode implementar que un diseño similar pero con buffers a la salida, esto debido a que el uso de buffers a la salida implica también un aumento de la velocidad interna de operación de la red respecto a la velocidad a la que operan los puertos de entrada y salida, como se verá más adelante[16, 17].A pesar de tener la ventaja de la simplicidad, el uso de buffers a la entrada de la red tiene un gran inconveniente: su performance es aproximadamente de la mitad comparado con un diseño ideal usando una red no bloqueante con buffers a la salida[9, 14, 15]. Esto se debe al efecto conocido como *head-of-line blocking*, que consiste en que el paquete que encabeza la fila del buffer no puede ser transmitido porque existe contención por el puerto de salida y este paquete puede impedir que los paquetes que se encuentran después de él sean transmitidos, aun cuando estos pudieran llegar a su destino sin problema como se aprecia en la Figura 1-18.



Fuente [14]

Figura 1-18.*Head-of-line blocking*.

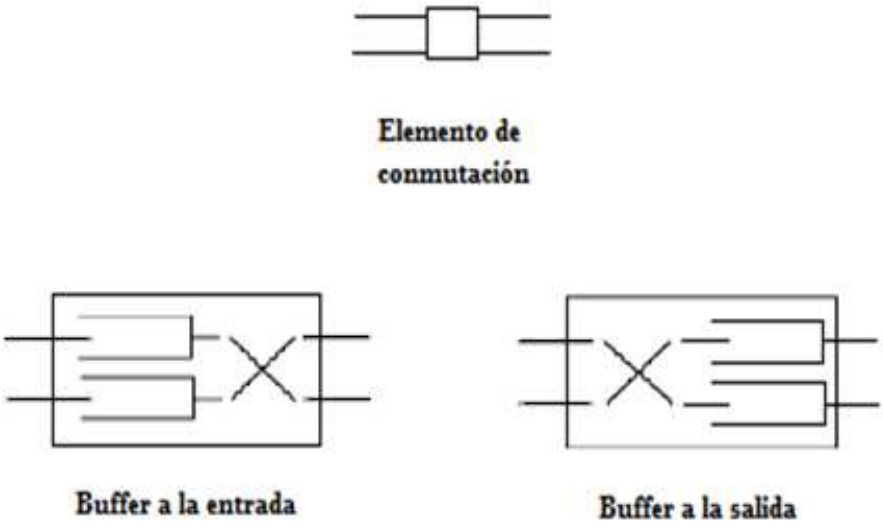
En lo referente a redes con buffers a la salida, este tipo de diseño tiene sentido emplearlo en implementaciones en las cuales la velocidad de los enlaces internos de la red es mayor que la velocidad a la que operan los puertos de entrada y

salida, ya que, en este caso, podrían llegar simultáneamente dos o más paquetes a un mismo puerto de salida[9, 18], esto se explicará con mayor detalle en la Sección 1.3.6.2.

- *Buffers Internos*

En el caso de tener un diseño con buffers en el interior de la red de conmutación, en cada elemento de conmutación se colocan buffers sea a la entrada o salida del mismo como se puede ver en la Figura 1-19.

El performance de una red implementada con elementos de conmutación que tengan buffers a la entrada se asemeja al de unared con buffers externos ubicados a la entrada de la red, pero considerando una red no bloqueante. De manera similar, el performance de una redimplementada con elementos de conmutación que tengan buffers a la salida es similar al de unared con buffers externos ubicados luego de la red, pero implementada con elementos de conmutación de grado elevado [15].

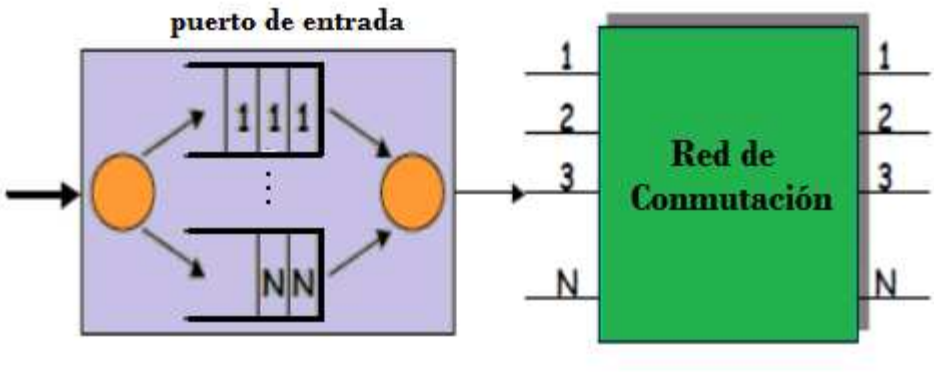


Fuente [14]

Figura 1-19. Elemento de conmutación con buffers a la entrada o salida.

- *Virtual Output Queueing*

Se trata de una técnica para mejorar el desempeño de una red con buffers a la entrada y tratar de mitigar el efecto del *head-of-line blocking*. La idea es que cada puerto de entrada mantenga una cola separada para cada puerto de salida. Así, cada paquete que llega se dirige a un controlador a la entrada del puerto, mismo que lo clasifica en base al destino requerido y loenvía a la cola correspondiente.También existe un controlador a la salida del puerto, el cual es el encargado de seleccionar, en cada ciclo, el paquete que se enviará hacia la red para ser conmutado de entre los N paquetes que pueden estar encabezando cada una de las colas.En la Figura 1-20 se observa la estructura básica de uno de los puertos de entrada a la red empleando esta técnica.



Fuente [9]

Figura 1-20.*Virtual Output Queuing*.

Al implementar esta estrategia de almacenamiento, se logra que todos los puertos de entrada sean capaces de enviar los paquetes a cualquier puerto de salida disponible. Se ha demostrado que al usar esta técnica y bajo la premisa de tener un tráfico uniforme, el *throughput* puede llegar a ser muy cercanoal 100% [9].

El factor limitante en la implementación práctica de esta técnica es la cantidad disponible de memoria para la implementación de los buffers, ya que, para tamaños de redes grandes, el número de buffers requeridos se incrementa en gran manera.

En el Capítulo 2 de este proyecto se analiza a profundidad un tipo de elemento de conmutación que emplea esta estrategia de almacenamiento.

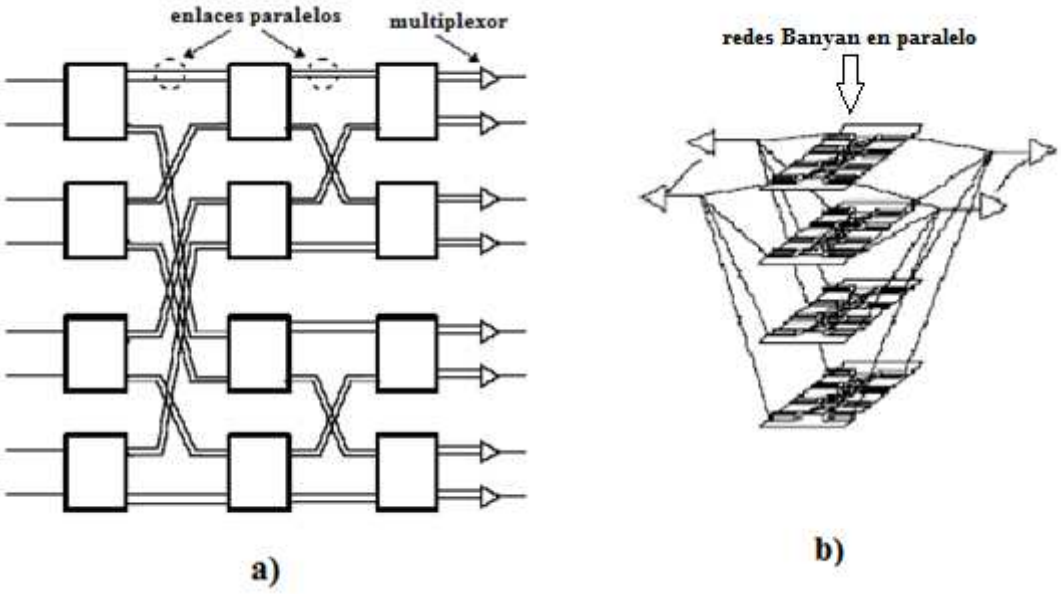
1.3.6.2 Velocidad interna

Otra opción usada para mejorar el desempeño de las redes de conmutación, es incrementar la velocidad de los enlaces internos en la red. En la discusión anterior referente a los buffers se asumió que la máxima velocidad de los puertos de entrada era igual a la velocidad a la que trabajaban los enlaces internos de la red. Esto implica que, en cada ciclo de reloj, se puede entregar solo un paquete a cada puerto de salida.

En cambio, si los enlaces internos son *r* veces más rápidos que los puertos de entrada y salida, entonces se tiene que como máximo *r* paquetes pueden entregarse a un puerto de salida en un ciclo. En el caso extremo, si se tuviera *r=N*, entonces se podría conseguir una red internamente no bloqueante. Pero en casos prácticos *r* debe ser pequeño. Se dispone de estudios en los cuales para *r=2* y *r=3* el máximo *throughput* conseguido puede ser de 0.8845 y 0.9755, respectivamente[18]. Para este caso, cuando más de un paquete puede llegar al mismo puerto de salida en un solo ciclo, se requiere necesariamente que existan buffers en los puertos de salida [9].

1.3.6.3 Redes Banyan dilatadas y replicadas

Además de las opciones ya mencionadas para incrementar el desempeño de las redes Banyan, existe lo que se conoce como redes dilatadas y redes replicadas; ejemplos de estas redes se pueden apreciar en la Figura 1-21.



Fuente (a) [19], (b) [27]

Figura 1-21. a) Red Banyan 2-dilatada y b) Red Banyan 4-replicada.

Una red con un factor de dilatación *d,* o llamada red *d*-dilatada, tiene la característica que cada elemento de conmutación se conecta hacia sus sucesores en la siguiente etapa con *d* enlaces. De este modo, se consigue que *d* paquetes contendiendo por un enlace puedan ser transportados de una etapa a la siguiente.

Las redes *d*-replicadas, en cambio, consisten en*d* redes Banyan paralelas, de modo que, para cada paquete que ingresa a la red, se debe elegir cuál de las redes usar. Un algoritmo simple para tomar esta decisión es escoger aleatoriamente. Tanto el método de redes replicadas como dilatadas son semejantes al método de incrementar la velocidad de los enlaces internos de la red [18,19].

1.3.6.4 Red de ordenamiento

Todos los métodos mencionados anteriormente son básicamente extensiones de las redes Banyan simples, esto quiere decir que todas ellas son internamente bloqueantes, es decir, la probabilidad de que dos paquetes con puertos de destino diferentes colisionen en alguno de los enlaces internos de la red, no es cero.

Existe una premisa bajo la cual es posible que una red Banyan sea internamente no bloqueante, y es que los puertos de entrada activos, es decir, aquellos que tienen un paquete para transmitir $x_{1,}x_2, \dots, x_m$ $(x_j > x_i,\ si\ j > i)$ y sus correspondientes puertos destinos $y_{1,}y_2, \dots, y_m$cumplan las siguientes condiciones:

1. Puertos de salida distintos y monotónicos[2]: $y_1 < y_2 < \cdots < y_m\ \ o\ \ y_1 > y_2 > \cdots > y_m$.

2. Puertos de entrada concentrados: Cualquier entrada entre dos entradas activas, también está activa. Esto es, $x_i < w < x_j$implica que la entrada *w* también está activa.

La demostración correspondiente al teorema anterior se la puede encontrar en la referencia [11].

La forma práctica de aplicar este teorema, y poder cumplir con las condiciones arriba mencionadas, es anteponer una red de ordenamiento a la red Banyan. La función de esta red de ordenamiento es la de organizar los paquetes que ingresan a la red, sea en forma ascendente o descendente, en base al número de puerto destino requerido.

Al anteponer una red de ordenamiento a la red Banyan, ya no se produce contención al interior de la red Banyan y por ende, ya no es necesario usar buffers en los elementos de conmutación. Pero, en este caso, se deben usar buffers en los puertos de entrada, ya que si dos o más paquetes tienen el mismo puerto de destino, solo uno será transportado mientras que los demás deberán ser almacenados en los buffers de entrada hasta que puedan ser finalmente enviados hacia su destino.

Una red muy popular de éste tipo es la red de ordenamiento de Batcher, la cual se analizará a continuación. Pero el inconveniente mayor de estas redes es su gran

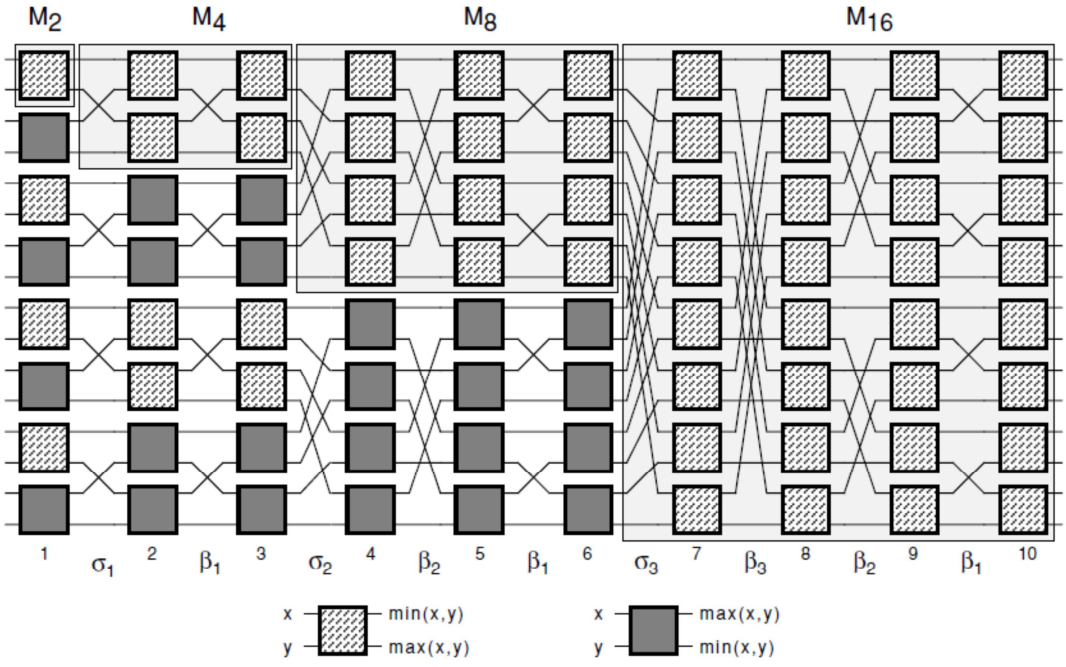
[2]**Función monotónica, f(x)**:Función que tiende a moverse solamente en una dirección mientras el parámetro *x* se incrementa.

tamaño, es decir, que se incrementa mucho el número de etapas que se debe atravesar para conectar una entrada con una salida.

- *Red de Ordenamiento de Batcher*

Una red de Batcher sirve para ordenar cualquier secuencia arbitraria de números en orden sea ascendente o descendente. Se construye en base a clasificadores, cada uno de los cuales constituye en sí mismo una red de interconexión multi-etapa basada en elementos de comparación 2x2, los cuales reciben de manera sincrónica dos números en sus puertos de entrada y su función es sacar el mayor de estos números en uno de sus puertos de salida y el menor en el otro[1].

En la Figura 1-22 se aprecia la estructura interna de una red deordenamiento ascendente de tamaño 16, en la cual se puede constatar que cada clasificadorinterno, tiene la topología de una red multi-etapa.



Fuente[8]

Figura 1-22. Red de Batcher para N=16.

En el ámbito de las redes de conmutación, se usa estas redes de ordenamiento para organizar los paquetes que ingresan a la red de acuerdo a la etiqueta que se

coloca en la cabecera de cada uno, la cual contiene la dirección del puerto de destino.

La red Batcher es de interés en aplicaciones de conmutación rápida de paquetes porque se puede formar una red no bloqueante anteponiendo una red de ordenamiento de Batcher a una red de enrutamiento de Banyan, dando lugar a lo que se conoce como conmutador Batcher-Banyan. De este conmutador se hablará algo más en la Sección1.3.7.1.

1.3.7 APLICACIONES DE LAS REDES BANYAN

Como se explicó anteriormente, las principales aplicaciones de las redes Banyan son para procesamiento paralelo en computación, y en el diseño de conmutadores y enrutadores de alta velocidad. A continuación, se exponen algunas de las implementaciones más comunes de redes de interconexión basadas en redes Banyan.

Para iniciar se explicará detalladamente el funcionamiento del conmutador Batcher-Banyan pues este constituye el hito sobre el cual se han desarrollado muchas otras arquitecturas de redes de interconexión.

1.3.7.1 Conmutador Batcher-Banyan

Se trata de un conmutador multi-etapa ampliamente usado en la interconexión de multiprocesadores con memoria compartida[3], así como en la implementación de conmutadores ATM.

La red de Batcher ordena los paquetes que le van llegando según su puerto de destino. En esta red, los clasificadores de cada etapa toman en cuenta toda la

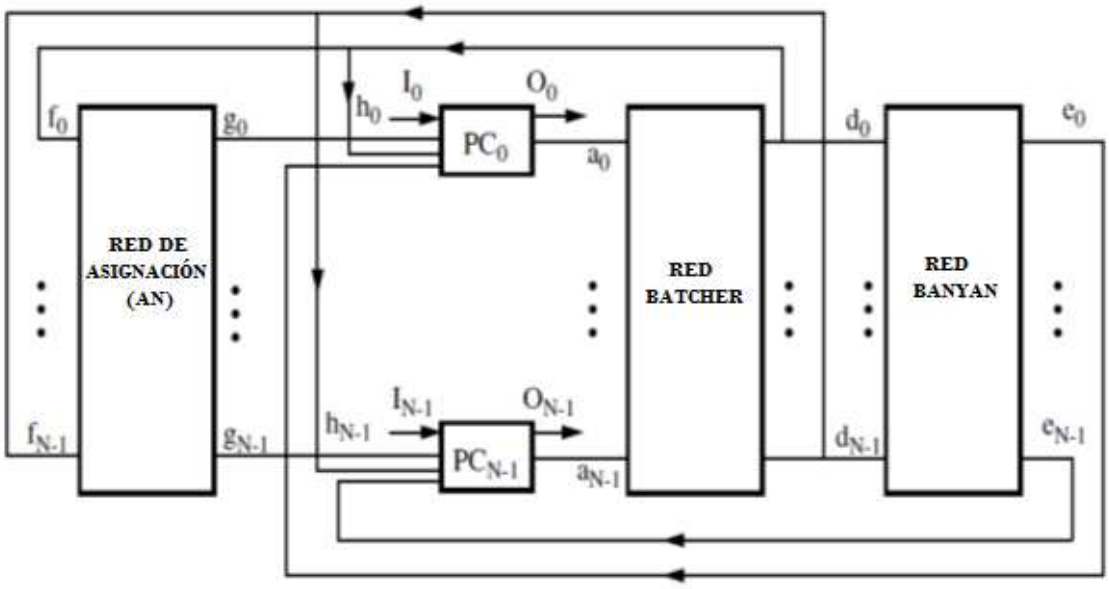
[3] **Multiprocesadores con Memoria Compartida:**También llamados multiprocesadores estrechamente acoplados. Son sistemas con múltiples procesadores que comparten un único espacio de direcciones de memoria, es decir,cualquier procesador puede acceder a los mismos datos.

etiqueta con el número de puerto de destino para realizar el ordenamiento de los paquetes.

Cabe mencionar que en este tipo de redes aún existe la posibilidad de que se presente contención por dos paquetes que tengan el mismo destino, ya que, a la salida de la red de ordenamiento de Batcher se tienen los paquetes con igual destino en puertos adyacentes[14].

Para resolver la contención por los puertos de salida, generalmente, se implementa un algoritmo de tres fases, a través del cual se detecta cuando hay paquetes contendiendo por un puerto de salida y se hace que las solicitudes que no puedan ser atendidas en ese intervalo de tiempo sean almacenadas a la entrada de la red para ser re-enviadas en el siguiente intervalo, recordando el efecto del *head-of-line blocking* ya mencionado[15].

La estructura de este tipo de conmutador se ilustra en la Figura 1-23.



Fuente [8]

Figura 1-23. Arquitectura de un conmutador Batcher-Banyan.

En la Figura 1-23 cabe aclarar que las siglas PC significan controlador de puerto y constituye el interfaz de la red con el medio externo, esto es, con los puertos de

entrada (I_0,...,I_{N-1}) junto con sus respectivos buffers, así como también con los puertos de salida (O_0,...,O_{N-1}). A continuación, se explicará el algoritmo de tres fases con el cual opera este conmutador y se verá las funciones de cada elemento mostrado.

- *Algoritmo de Tres Fases*

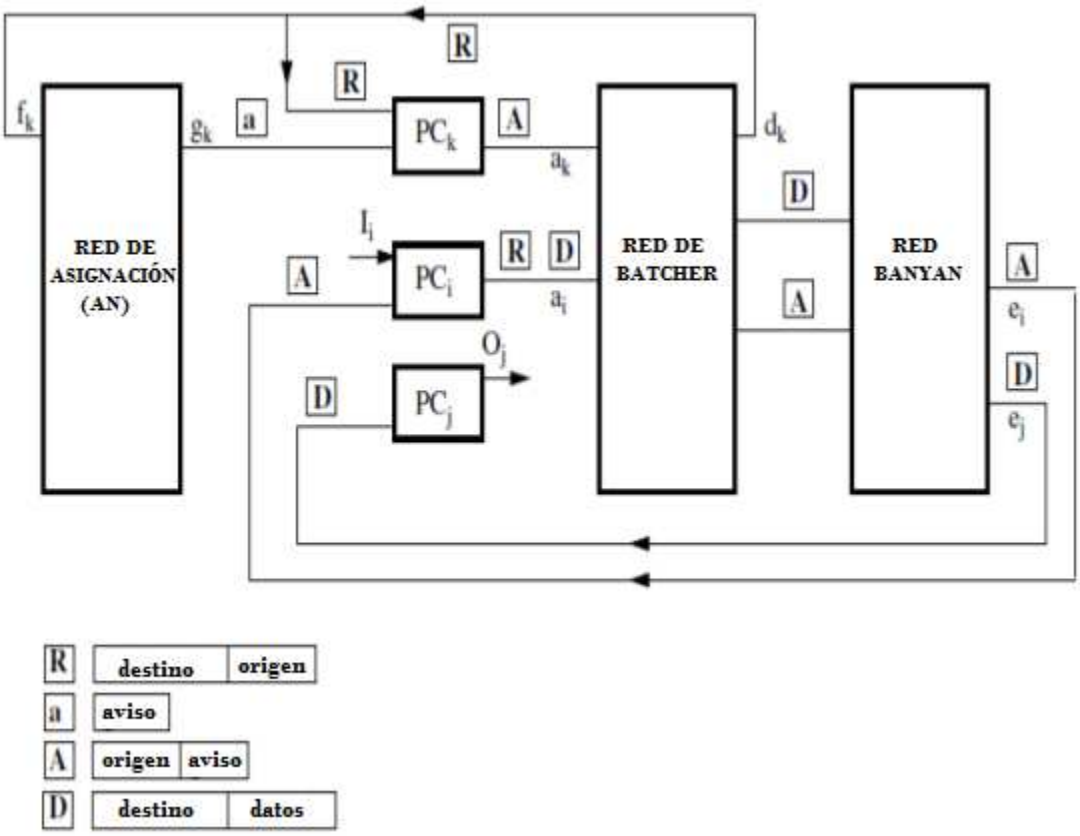
A fin de facilitar su comprensión, en la Figura 1-24 se aprecia el flujo de los paquetes entre los componentes de este conmutador.Las tres fases del algoritmo son:

1. **Arbitraje:**En esta fase, cada controlador de puerto (PC) que tiene un paquete en su buffer para transmitir envía un paquete piloto (R), que consiste del número de puerto de salida requerido, seguido del número de puerto de origen. Estos paquetes piloto atraviesan la red de Batcher y son ordenados en forma no decreciente tomando en cuenta tanto el puerto de salida requeridocomoel puerto origen. Luego, estos paquetes R ordenados entran a la red de asignación (AN), misma que genera un bit de aviso para marcar un solo paquete como ganador $a = 0$ por cada puerto destino requerido y el resto los marca con $a = 1$. Este bit de aviso es nuevamente enviado a los PC junto con el paquete R.En esta fase, los puertos de entrada aún no saben el resultado del arbitraje, por ello se hace necesaria la segunda fase.

2. **Reconocimiento:**En la segunda fase, los PC generan los paquetes de confirmación (A) que consisten en los paquetes recibidos desde la AN, pero sin incluir la dirección destino, es decir, los paquetes A solamente incluyen la dirección del puerto origen y el bit de aviso$a$, estos paquetes son enrutados a través de la propia red Batcher-Banyan hacia los PC correspondientes.

3. **Envío:** Aquí, los PC que recibieron el paquete Acon el bit $a = 0$pueden transmitir sus paquetes de datos (D) a través de la red sin que haya

conflicto alguno. Pero aquellos que reciben el paquete A con el bit$a = 1$ deben almacenar su paquete al menos hasta el siguiente ciclo donde deberán volver a intentar la transmisión[8,15, 17].



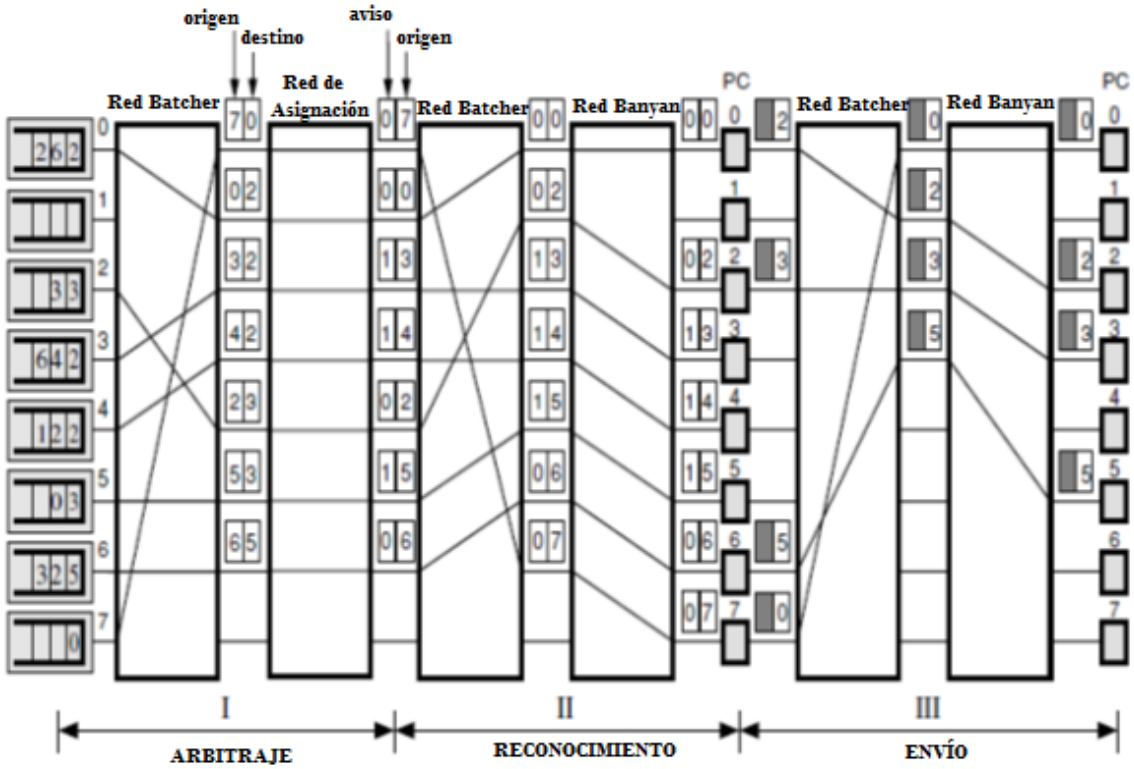
Fuente [8]

Figura 1-24. Flujo de paquetes en el algoritmo de tres fases.

En la Figura 1-25 se presenta un ejemplo de conmutación usando este algoritmo, a fin de aclarar su modo de operación. En esta figura se puede observar que, inicialmente, en las colas de los puertos de entrada existen tres paquetes dirigidos hacia el puerto de salida número dos, se enfocará la atención en estos paquetes para entender la manera cómo se resuelve la contención entre ellos.

Primero, estos tres paquetes con el mismo destino son ordenados al atravesar la red de Batcher en función de su puerto destino y su puerto origen. A continuación, estos paquetes ingresan a la red de asignación, la cual marca uno solo de ellos como ganador; ahora, estos paquetes son entregados nuevamente al controlador de puerto, el cual retira la dirección destino y envía estos paquetes nuevamente a

través de la red de Batcher para ser ordenados en base a la dirección de su puerto origen y proceder a ser enrutados en la red Banyan sin que exista contención. Finalmente, las salidas de la red Banyan están conectadas a los respectivos controladores de puerto, con lo cual, cada uno de ellos recibe la confirmación para poder proceder a enviar o no el paquete de datos. De este modo, solo se envía un paquete por cada puerto destino y no existe contención entre ellos.



Fuente [8]

Figura 1-25. Ejemplo de conmutación usando el algoritmo de tres fases.

Este diseño de conmutador tiene dos inconvenientes importantes, ambos derivados del tamaño de la red de ordenamiento de Batcher. Primero, que se necesita un número muy elevado de elementos de comparación para implementar una red de Batcher respecto al número de elementos de conmutación requeridos solamente para la red Banyan.

El segundo problema es la pérdida de eficiencia de la red debido a las fases 1 y 2 del algoritmo, principalmente, por el retardo al atravesar la red de Batcher.

1.3.7.2 Conmutador Starlite

Este conmutadorse basa en el uso de una red de conmutación Batcher-Banyan, pero supera el problema de *head-of-line blocking* almacenando los paquetes que no tuvieron éxito en la fase de arbitraje en buffers compartidos ubicados a la salida de la red de clasificación, en lugar de ubicarlos a la entrada [15]. La estructura básica de este tipo de conmutador se muestra en la Figura 1-26.

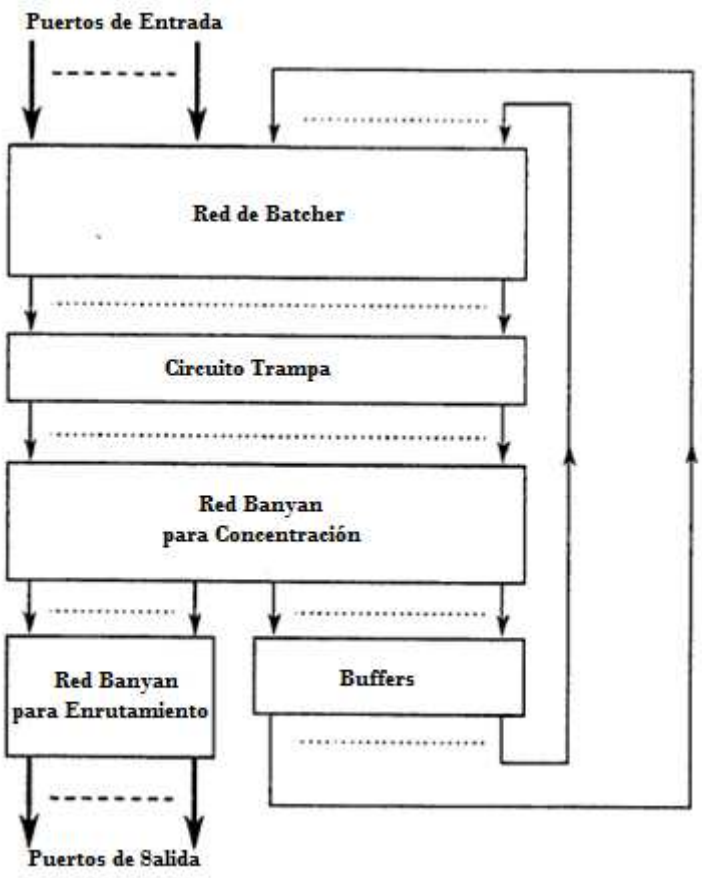
A la entrada del conmutador se coloca una etiqueta a cada paquete, la cual contiene el número del puerto de salida requerido. Estos paquetes entonces son ordenados por la red de Batcher de acuerdo a esta etiqueta, para que luego el circuito trampa realice un arbitraje entre los paquetes que estén contendiendo por el mismo puerto de salida, los cuales se ubican en puertos adyacentes a la salida de la red de Batcher.

Como solamente un paquete puede ser enrutado a cada puerto de salida en una ranura de tiempo, los paquetes que pierden el arbitraje deben ser dirigidos a los buffers para luego poder ser recirculados a la entrada en la siguiente ranura de tiempo. Para lograr esto, el circuito trampa realiza cálculos adicionales tanto para los paquetes que perdieron el arbitraje como para los que ganaron, de modo que añade una etiqueta adicional al inicio de cada paquete.

A continuación, estos paquetes que incluyen la etiqueta adicional, resultado del arbitraje, son dirigidos a una red de concentración, misma que también es de tipo Banyan. Una vez que estos paquetes pasan a través de la red de concentración, resulta que, gracias a esta última etiqueta añadida, los paquetes destinados a los buffers emergen en los puertos del lado derecho, mientras que los paquetes que van a los puertos de salida, aparecen en los puertos del lado izquierdo.

Los paquetes destinados a los puertos de salida, finalmente, son enrutados por la red de Banyan y llegan a su puerto destino, mientras que los paquetes que perdieron el proceso de arbitraje son almacenados en un conjunto de buffers

compartidos y reingresan a la red de Batcher junto con los nuevos paquetes durante la siguiente ranura de tiempo [15, 21].



Fuente [15]

Figura 1-26. Estructura del conmutadorStarlite.

Algunos comentarios que se pueden realizar sobre este diseño de conmutadorson, principalmente, respecto al número de etapas que los paquetes deben atravesar, ya que, debido a la presencia de la red de Batcher y de lasdos redes Banyan(red de concentración y red de enrutamiento), se tiene un gran número de etapas.

Además, en este conmutador se incrementa la complejidad, respecto al conmutador Batcher-Banyan, debido a la presencia del circuito trampa y sus funciones.

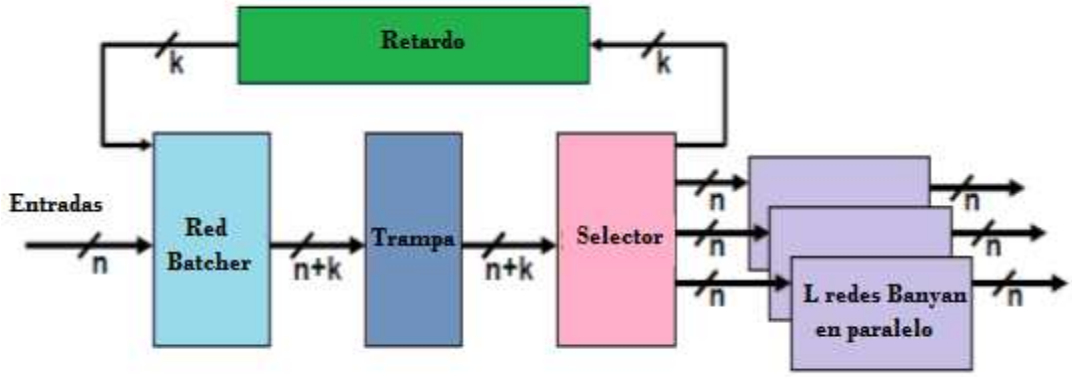
Otro punto a mencionar es que, debido al proceso de recirculación de los paquetes, puede ocurrir que los paquetes lleguen a su destino en desorden, pero esto se puede solucionar agregando una estampilla de tiempo, la misma que daría a los paquetes anteriores mayor prioridad que los nuevos en el proceso de arbitraje.

Esto último, da la pauta para decir que es simple extender un conmutador basado en una red de conmutación Batcher-Banyan, a fin de que maneje cualquier número de niveles de prioridad. Si se añade una etiqueta con el nivel de prioridad de cada paquete, es posible ordenar estos paquetes de acuerdo a su nivel de prioridad usando una red de Batcher, de modo que, la lógica de arbitraje sería más simple. Esto puede ser una característica bastante deseable, principalmente en redes que manejan tráfico multi-servicio.

1.3.7.3 Conmutador Sunshine

El conmutadorSunshine fue diseñado y simulado por la empresa Bellcore como un prototipo de conmutador para redes ATM. Este diseño se basa en una red de conmutación Batcher-Banyan, e inicialmente fue probado con un flujo de celdas ATM a una velocidad de 155Mbps [22]. La arquitectura básica de este tipo de conmutador se muestra en la Figura 1-27.

Como se puede apreciar, es un diseño bastante similar al conmutador Starlite, pero la mayor diferencia radica en que, en este conmutador, se dispone de redes Banyan en paralelo a fin de incrementar la capacidad interna de la red. El inconveniente de esta implementación es quese hace necesario incrementar la velocidad de operación de los controladores de salida, a fin de poder manejar los paquetes provenientes de las múltiples redes Banyan.



Fuente [12]

Figura 1-27. Arquitectura del conmutadorSunshine.

Los principales componentes de este conmutador son:

- **Red de Batcher:** En cada ranura de tiempo, recibe *n* paquetes nuevos y *k* paquetes recirculados, y los ordena de acuerdo al número de puerto de salida requerido.

- **Red Trampa:**Identifica los hasta *L* paquetes que podrán ser entregados al mismo puerto de salida y además, marca hasta *k* paquetes que serán recirculados hacia la red de Batcher, a fin de intentar ser retransmitidos en el siguiente ciclo.

- **Selector:**Distribuye los *L* paquetes, destinados al mismo puerto de salida, y queganaron el proceso de arbitraje, hacia las *L* redes Banyan en paralelo y envía los *k* paquetes para ser recirculados.

- **Unidades de Retardo:**Encargadas de sincronizar el ingreso de los *k* paquetes recirculados, con los nuevos paquetes que ingresan a la red[12, 23].

1.3.7.4 Conmutador Tandem Banyan

Este diseño de conmutador se basa en colocar K_b redes Banyan en cascada, cada una de las cuales tiene sus respectivos puertos de salida conectados, tanto a las colas de los buffers en los puertos de salida del conmutador,como también a la siguiente red Banyan mediante enlaces directos (sin permutaciones).

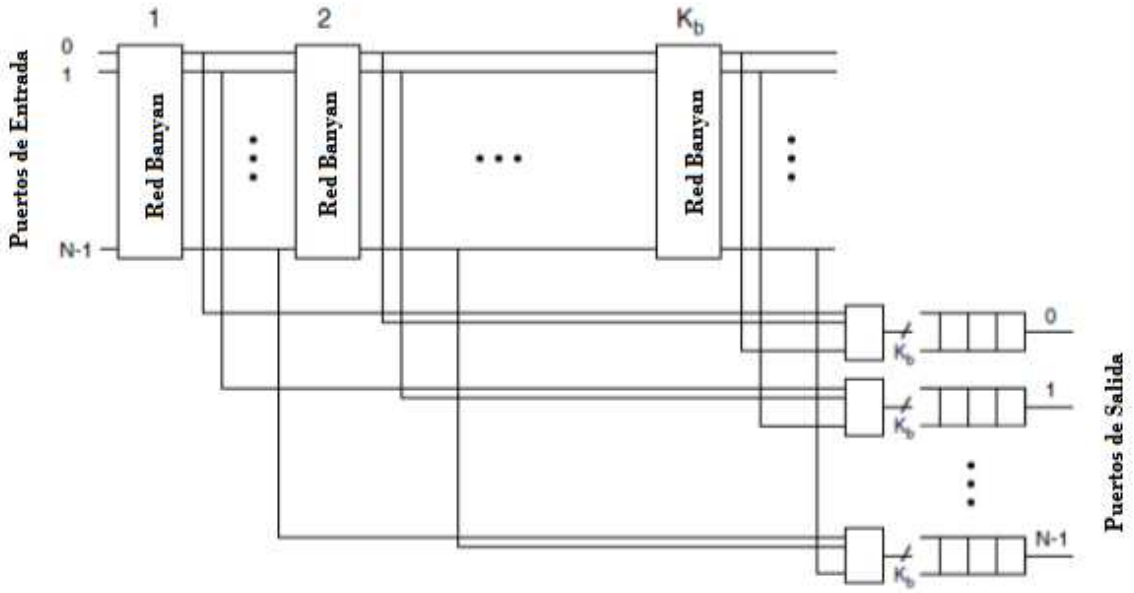
Para un diseño de conmutador Tandem Banyan de tamaño $N\ x\ N,$ cada una de las redes Banyan, también será de tamaño $N\ x\ N$, y constará de $n = \log_2 N$etapas con elementos de conmutación 2x2. Por ello, el conmutador Tandem Banyan consta, en total, de $K = nK_b$ etapas [25]. Este diseño se muestra en la Figura 1-28.

En este diseño de conmutador, la etiqueta que indica el destino delpaquete requiere dos campos, la dirección de salida $o = o_{n-1}, \dots, o_0$ y la bandera *D,*la cual indica si el paquete debe ir al buffer del puerto de salida o a la siguiente red Banyan.

La primera red Banyan enruta los paquetes recibidos en forma simple, en base al puerto requerido de salida. En caso que exista un conflicto entre dos paquetes conteniendo por la misma salida en un elemento de conmutación, se escoge aleatoriamente el ganador, el cual es enrutado adecuadamente, mientras que el perdedor es desviado hacia la otra salida del SE y además, se pone en 1 el bit *D* en la etiqueta del paquete,indicando así que este paquete debe ir a la siguiente red Banyan.

A fin de evitar que un paquete que ha sido desviado en una etapa cause que otros paquetes en las siguientes etapas de la misma red sean desviados, los SE realizan el enrutamiento también fijándose en el bit *D*: si dos paquetes inciden a su entrada y tienen el bit *D* diferente, el SE enruta adecuadamente aquel paquete cuyo bit *D* esté en 0 y envía el otro paquete por la salida restante.

La cola de salida *j,* en este tipo de conmutador, está conectada a la correspondiente salida *j* de cada una de las K_b redes Banyan; en esta cola, se aceptan únicamente aquellos paquetes que no han sido desviados (bit *D*=0).



Fuente [8]

Figura 1-28. Arquitectura del conmutador Tandem Banyan.

De igual manera, la k-ésima red Banyan (k=2, …,K_b), al recibir los paquetes que provienen de la red anterior, solamente acepta aquellos paquetes que hayan sido desviados (*D*=1).

Aquellos paquetes que atraviesen todas las K_b redes Banyan y aún no tengan éxito en su enrutamiento, aparecerán al final con el bit *D*=1 y, por lo tanto, serán descartados.

Como es evidente, en esta arquitectura, los paquetes atraviesan un elevado número de etapas debido a la propia estructura del conmutador.

- *Conmutador Tandem Banyan Dilatado*

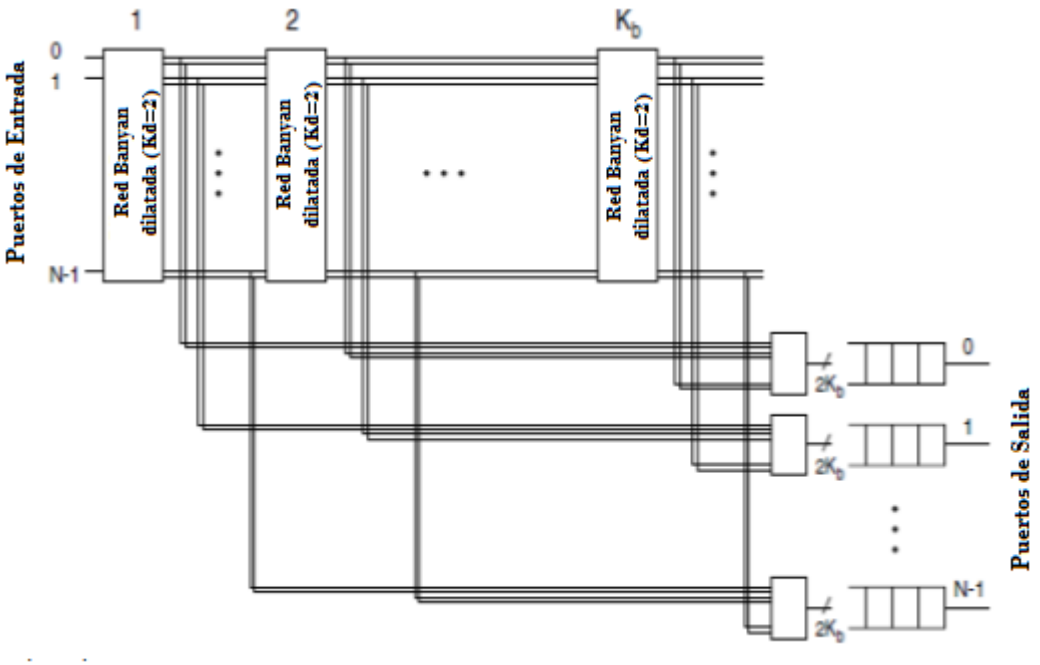
Una opción interesante a fin de limitar el número de redes Banyan que se colocan en cascada en el conmutador Tandem Banyan y, a la vez, satisfacer un nivel

aceptable de paquetes perdidos, es el dilatar la red con un factor K_d, lo cual resulta en un conmutador Tandem Banyan Dilatado (DTBSF, *Dilated Tandem Banyan SwitchFabric*).

Este proceso consiste en incrementar el número de enlaces entre las diferentes etapas de la red con un factor K_d. Un ejemplo de esta arquitectura se muestra en la Figura 1-29, en la cual se ha escogido K_d=2.

Con esta implementación, cada una de las K_b redes Banyan es en sí misma una red Banyan Dilatada con factor de dilatación K_d. Las conexiones entre las redes Banyan ahora incluyen NK_d enlaces, y cada cola de salida del conmutador es alimentada por K_bK_d enlaces.

En esta configuración, cada red Banyan es capaz de conmutar hasta $K_d$paquetes hacia cada puerto de salida, de modo que se necesitan menos redes, en comparación con la estructura no dilatada del conmutador.



Fuente [8]

Figura 1-29. Arquitectura delconmutador Tandem Banyan Dilatado.

Finalmente, en cuanto a complejidad, dos redes DTBSF pueden tener el mismo producto K_bK_d pero, a pesar de ello, tener diferentes grados de complejidad. Esto debido a que la complejidad para implementar los SE en una red dilatada, crece con el cuadrado del factor de dilatación (K_d) [8].

REFERENCIAS

[1] P. Newman. Multi-stage interconnection networks. University of Cambridge, 1988. Disponible: http://pnewman.com/papers/thesis/chapter4.pdf

[2] Arquitectura e Ingeniería de Computadores. Disponible: http://www.fdi.ucm.es/profesor/llorente/AIC/Leccion-14.pdf

[3] M. García, J. Tubella, A. González, C. Aliagas, L. Esteban. Multiprocesadores Von Neumann. UPC/DAC Report, 1990. Disponible: http://www.atc.uniovi.es/inf_superior/atc2010/PARALELAS/Multiprocesadores%20Von%20Neumann(UPC90-1).pdf

[4] Redes De Interconexión. Disponible: http://www.infor.uva.es/~bastida/Arquitecturas%20Avanzadas/Redes.pdf

[5] Source Routing. Disponible: http://www.iss.net/security_center/advice/Underground/Hacking/Methods/Technical/Source_Routing/default.htm

[6] La Importancia De La Red. Disponible: http://www.wikilearning.com/tutorial/el_manual_para_el_clustering_con_openmosix-la_importancia_de_la_red_i/9756-11

[7] D. C. Vasiliadis , G. E. Rizos , C. Vassilakis. Performance Analysis of blocking Banyan Switches.Disponible: http://www.sdbs.uop.gr/files/banyanSwitch.pdf

[8] A. Pattavina. Switching Theory: Architecture and Performance in Broadband ATM Network.Ed. John Wiley & Sons Ltd, 1998.

[9] M. Hidell, P. Sjödin, O. Hagsand. Router Architectures.Disponible: http://web.ict.kth.se/~mahidell/pubs/networking04_tutorial_final.pdf

[10] C. P. Kruskal, M. Snir. A Unified Theory Of Interconnection Network Structure. Disponible: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.45.3068

[11] Telecommunication Switching and Network System. Disponible: http://www.cmlab.csie.ntu.edu.tw/~freetempo/CN2011/slides/Tutorial_banyan.pdf

[12] Switch Construction. Disponible: http://www.cs.wmich.edu/~alfuqaha/Fall07/cs6030/lectures/SwitchDesign-rev2.pdf

[13] Redes De Interconexión Para Computadores Masivamente Paralelos. Disponible: http://ditec.um.es/~jmgarcia/papers/escuela94.pdf

[14] ATM Switch. Disponible: http://lyle.smu.edu/~nair/courses/7344/7_atm_2_v2.pdf

[15] P. Newman. Fast Packet Switch Architecture.University of Cambridge, 1988.http://pnewman.com/papers/thesis/chapter3.pdf

[16] E. Modiano. Conmutación Rápida De Paquetes. Disponible: http://mit.ocw.universia.net/6-263JData-Communication-NetworksFall2002/NR/rdonlyres/Electrical-Engineering-and-Computer-Science/6-263JData-Communication-NetworksFall2002/C1C0738E-C7B6-447E-AF48-914F63DE022B/0/Lectures17_18.pdf

[17] Switch Concept & Architecture. Disponible: http://speed.cis.nctu.edu.tw/~ydlin/course/cn/srouter/Lesson3.PDF

[18] C. Bouras, C. Gkantsidis. Cost of implementing Banyan networks for use in ATM switching fabrics. Disponible: http://ru6.cti.gr/ru6/publications/8526611.pdf

[19] M. Youssef, M. El-Derini, H. Aly. Performance Enhancement Techniques of a Banyan Network Based Interconnection Structure. Disponible: http://wrc.ejust.edu.eg/people/moustafa/papers/aej99.pdf

[20] Redes De Interconexión. Disponible:http://verde.lma.fi.upm.es/wwwwver/pvp/Tema4.Redes_interconexion.pdf

[21] R. F. Liao, A. Leon-Garcia. Fault Tolerant Design of Batcher-Banyan Class ATM Switches.

[22] http://www.cnri.reston.va.us/gigafr/section-4-06.htm

[23] M. Song. Switches.Disponible: http://www.ece.odu.edu/~msong/switches.pdf

[24] http://htmlhelp.com/~liam/Hawaii/BigIsland/Hilo/

[25] M. H. Khansari, A. Movaghar-Rahimabadi, S. Hessabi. Design and Simulation of a Tandem ATM Switch based on FPGA. Disponible: http://ce.sharif.ac.ir/~khansari/docs/ATMSwitch.pdf

[26] http://pg-server.csc.ncsu.edu/mediawiki/index.php/CSC/ECE_506_Spring_2010/ch_12_PP

[27] O. Zadedyurina, Y. Ofek, A. Pattavina. Space and Time Blocking versus Cost in All-Optical Banyan Networks.Disponible: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4534037

CAPÍTULO 2

ANÁLISIS MATEMÁTICO DE UNA RED DE CONMUTACIÓN TIPO BANYAN

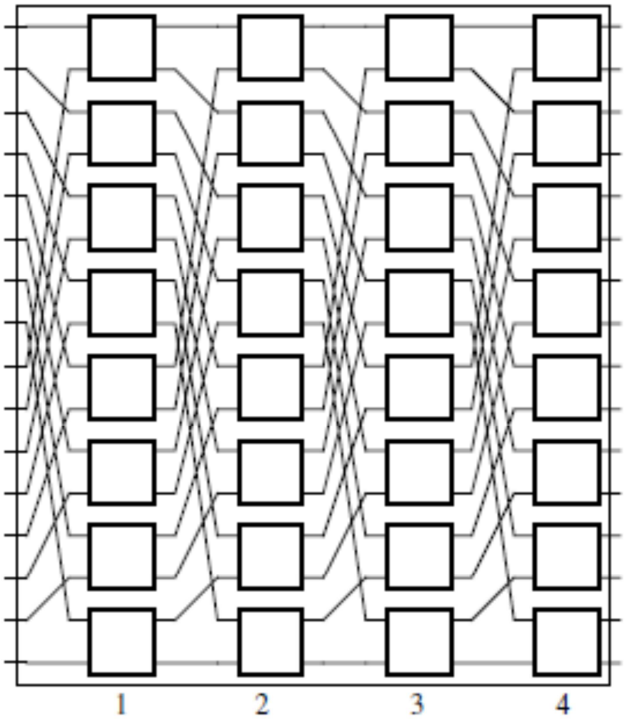
2.1 DESCRIPCIÓN DE LA RED A UTILIZAR

2.1.1 CARACTERÍSTICAS

En el presente proyecto, se usará una red Banyan, cuya topología específica es de tipo Omega(Ω) por ser ésta una de las estructuras más comunes y sencillas.

Además, como se explicó en la Sección 1.3.5, las cuatro topologías básicas de red Banyan son equivalentes entre sí, es decir que, una red Banyan B puede obtenerse en base a otra red Banyan A, simplemente moviendo los SE de A de tal manera que la red obtenida tenga los mismos patrones de interconexión entre etapas que la red B y, de ser necesario, se aplica en la entrada y/o salida de la red una de las dos permutaciones δ (Bit Switch) o ρ (Bit Reversal) [1, 2]. Esto implica que para el análisis de performance que se pretende realizar más adelante, la topología específica no es un factor relevante.

En la Figura 2-1 se puede apreciar un ejemplo de esta red de tamaño 16X16, en la cual se aprecia el patrón de interconexión entre etapas y se ha usado elementos de conmutación de tamaño 2X2, debido a que, en la mayoría de aplicaciones prácticas se trabaja con datos digitales expresados en formato binario.



Fuente [1]

Figura 2-1. Red de Conmutación Omega de tamaño *16 x 16*.

Por tratarse de un tipo de red Banyan, por definición, existe una y solo una ruta posible entre cada puerto origen y destino, lo cual hace que, estas redes sean sensibles a fallos ya que, por ejemplo, si una de las conexiones se corta no será posible disponer de muchas de las rutas.

Además, se sabe que las redes Banyan presentan la propiedad de auto-enrutamiento, es decir que, cada uno de los elementos de conmutación toma la decisión de enrutamiento de forma autónoma, sin que haya un ente de control centralizado.

También, todas las redes Banyan son fácilmente escalables, por ello, para el presente proyecto se usará una red de tamaño 8X8, sabiendo que los resultados pueden fácilmente adaptarse a los posibles incrementos de tamaño que se desee realizar en la red.

La principal desventaja que tienen todas las redes Banyan, es que son bloqueantes, es decir que en un momento dado puede darse que dos paquetes

entren en contención por un puerto en alguno de los elementos de conmutación, aunque los paquetes vayan destinados a diferentes puertos destino [4]. Como se verá en la Sección 2.2, se hará uso de un tipo especial de elementos de conmutación, y con ello se intenta minimizar la probabilidad de bloqueo.

2.1.2 ENRUTAMIENTO

Aunque el algoritmo de auto-enrutamiento usado en redes Banyan ya se mencionó en la Sección 1.3.3, vale la pena recordarlo brevemente.

En esta red, el algoritmo de auto-enrutamiento, es muy simple: a cada paquete que ingresa a la red, se le añade una etiqueta, que consiste en la dirección del puerto destino de dicho paquete.

Los elementos de conmutación de cada etapa toman en cuenta el bit correspondiente de la etiqueta, empezando desde los más significativos y, si este bit es cero, envían el paquete al puerto de salida superior, pero si el bit es uno, envían los datos por el puerto inferior. Con esto resulta que, el bit más significativo de la etiqueta se usa en la primera etapa, el siguiente bit se usa en la segunda etapa, y así sucesivamente, hasta que el bit menos significativo se lo usa en la última etapa.

Una vez que el conmutador usa el bit respectivo de la etiqueta, este bit se lo puede retirar o se puede rotar toda la etiqueta, a fin de que siempre el primer bit disponible sea el que, el conmutador de la siguiente etapa vaya a usar[3,5].

2.2 DESCRIPCIÓN DE LOS ELEMENTOS DE CONMUTACIÓN

Como se mencionó anteriormente, el principal inconveniente con las redes Banyan es que son bloqueantes, lo cual degrada su nivel de desempeño en gran manera. Para tratar de mitigar este problema, en forma general, se han propuesto varias alternativas [5], entre las principales se tiene:

- Colocar buffers en diferentes lugares de la red (entrada, salida, en cada SE) y de diferente tamaño.

- Anteponer redes de ordenamiento a la red Banyan (Conmutador Batcher-Banyan).

- Usar redes Banyan en paralelo (redes replicadas).

- Usar redes con más de una línea de conexión entre los SE de las diferentes etapas (redes dilatadas).

- Usar redes Banyan en cascada (*Tandem Banyan Switch Fabric*).

Para fines de este análisis matemático y su respectiva implementación, se ha escogido la opción de usar buffers en cada elemento de conmutación; para ello se usarán elementos de conmutación Banyan duales, los cuales han demostrado tener un desempeño superior a los elementos de conmutación normales [6] gracias a la técnica de almacenamiento que usa, *Virtual Output Queuing*, misma que se explicará con más detalle en la Sección 2.2.2. A continuación, se analizará la estructura y funcionamiento de este tipo de elementos de conmutación.

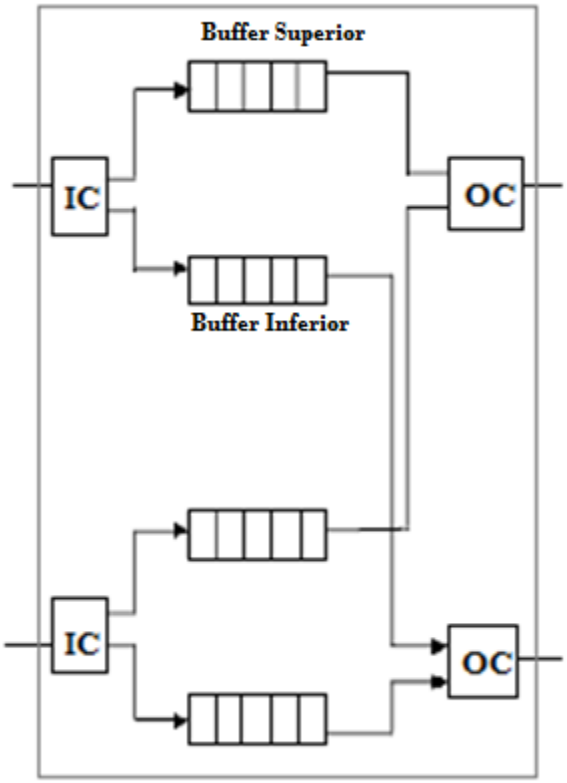
2.2.1 ARQUITECTURA DEL ELEMENTO DE CONMUTACIÓN BANYAN DUAL

La red de conmutación a implementar se basa en elementos de conmutación Banyan duales de tamaño 2X2 de tipo *Crossbar*(Sección 1.2.2), cuya arquitectura interna se aprecia en la Figura 2-2.

En este tipo de elementos de conmutación, se han implementado dos colas luego de cada entrada del SE, a fin de almacenar los paquetes que llegan de acuerdo al puerto de salida requerido; por tanto, en cada elemento de conmutación existen cuatro colas, como se aprecia en la Figura 2-2. Todas estas

colas trabajan con una política de atención FIFO (*First In, First Out*) y son de igual tamaño.

Como se aprecia, también existen dos controladores de salida (OC), los cuales están conectados, cada uno, a un par de buffers de los cuales reciben los paquetes y los envían a la salida del elemento de conmutación. A continuación se explicará en detalle el funcionamiento de este elemento de conmutación.



Fuente [6]

Figura 2-2. Arquitectura de un elemento de conmutación Banyan dual.

2.2.2 FUNCIONAMIENTO

La manera cómo funciona este elemento de conmutación es que, se realizan los procesos de conmutación y arbitraje de forma separada, esto es, en dos lugares diferentes del SE; entendiéndose como conmutación el proceso mediante el cual, a cada paquete que ingresa a una entrada se lo identifica y clasifica de acuerdo al puerto de salida requerido; mientras que arbitraje es el proceso por el

cual se escoge a uno de los dos paquetes disponibles y se lo envía a la salida correspondiente.

Debido a la presencia de las colas duales luego de cada entrada, el proceso de arbitraje para cada uno de los puertos de salida del SE es independiente, lo cual permite que cada SE envíe dos paquetes en cada ranura de tiempo, esto incrementa considerablemente el desempeño global de la red.

En cada una de las dos entradas del SE existe un controlador de entrada (IC), el cual realiza la función de conmutación, es decir, decide a cuál de las dos colas envía la paquete que llega, basándose para ello en el puerto de salida requerido. De modo que es evidente que el proceso de conmutación se realiza a la entrada, antes de almacenar los paquetes en los buffers correspondientes.

Existen también los controladores de salida (OC), que son los encargados de realizar el arbitraje entre las dos colas de los buffers que van dirigidas al mismo puerto de salida. Para realizar este arbitraje de manera equitativa, se usa un esquema *Round Robin*[4], es decir, para seleccionar cuál de los paquetes se enviará a la siguiente etapa se usa un orden cíclico, si el buffer correspondiente a la entrada superior fue tomado en cuenta primero, en la siguiente ranura de tiempo, será el buffer asociado a la entrada inferior el primero en ser considerado para enviar el paquete. En caso que, el primer buffer no tenga paquete que enviar (buffer vacío), se tomará en cuenta al segundo buffer a ver si éste tiene paquetes para transmitir en esa ranura de tiempo. De este modo, como ambos controladores de salida funcionan de forma independiente, es posible transmitir dos paquetes en cada ranura de tiempo [6].

Es precisamente gracias a este esquema de funcionamiento, que el desempeño de la red basada en estos elementos de conmutación duales es superior al de una red usando elementos de conmutación comunes.

[4] ***Round Robin***:Método para seleccionar todos los elementos en un grupo de manera equitativa y en un orden racional, normalmente comenzando por el primer elemento de la lista hasta llegar al último y empezando de nuevo desde el primer elemento.

A pesar de esta ventaja, estos SE siguen siendo de tipo bloqueante, ya que dos paquetes pueden tratar de acceder al mismo puerto de salida en un instante dado, y en este caso, uno solo puede ser transmitido, mientras el otro debe ser almacenado y esperar a la siguiente ranura de tiempo. Aquí, también puede haber paquetes perdidos si hay una elevada carga de tráfico y el tamaño de los buffers es reducido.

2.3 ANÁLISIS MATEMÁTICO

A continuación, se realizará el análisis matemático correspondiente al comportamiento de los elementos de conmutación, y específicamente de los buffers del SE. Debido a que se considera que todos los buffers son idénticos, se analizará solamente uno y posteriormente, en la Sección 2.3.2.5, se ampliará los resultados a todos los buffers del SE y finalmente, a toda la red de conmutación.

Los parámetros a analizar son: el retardo promedio de un paquete al atravesar la red, medido en términos de ciclos de reloj, y la tasa de paquetes perdidos en cada ranura de tiempo en el interior de la red de conmutación.

2.3.1 CONDICIONES DEL SISTEMA

Las premisas, en base a las cuales se realizará todo el análisis matemático, son las siguientes:

- Todos los buffers de los SE son de igual tamaño y trabajan con política de atención FIFO; el tamaño de los buffers se lo denotará con la letra B.

- Los controladores de salida de cada SE, operan con el esquema *Round Robin*.

- Todos los buffers operan en un modo tal que, el paquete que encabeza la cola del buffer está siempre disponible, y la señal de ‘habilitación de

lectura' actúa más bien como una señal que indica que ya se ha realizado la lectura del paquete y que se lo puede retirar del buffer, este es el modo de operación llamado FWFT (*First Word Fall Through*) [8].

- Se tiene un patrón de tráfico igualmente distribuido entre los puertos de entrada y salida de la red [6]. Esto, a su vez implica que:

    - Todas las entradas de la red son independientes entre sí y tienen igual probabilidad de tener un paquete, en otras palabras, la llegada de paquetes a cada entrada de la red constituye un proceso de Bernoulli [5] con parámetro λ.

    - Un paquete que ingresa a una de las entradas del SE, tiene igual probabilidad de ir destinado a cualquiera de los puertos de salida. Para efectos del presente análisis, se denotará como P_u , la probabilidad de que un paquete que ingresa al SE, vaya destinado al puerto de salida superior [6].

Una vez establecidas estas bases, se procede con el análisis.

2.3.2 DESARROLLO MATEMÁTICO

El sistema a analizar es uno de los cuatro buffers existentes en cada SE de la red de conmutación, por ejemplo el buffer superior del primer puerto de entrada del SE. Entonces, sea $M(n)$ el número de paquetes almacenados en la cola del buffer al terminar la *n*-ésima ranura de tiempo.

Además, solamente puede llegar como máximo un paquete a la entrada del buffer en cada ranura de tiempo y, de igual manera, solamente puede salir como máximo un paquete del buffer en cada ranura de tiempo. Entonces, es claro que, si se tienen *p* paquetes en el buffer en una ranura de tiempo, en la siguiente

[5] **Proceso de Bernoulli:** Secuencia de pruebas independientes, en las cuales, el resultado se puede clasificar como 'éxito' o 'fracaso'. Además, en cada prueba, la probabilidad de éxito, λ, permanece constante.

ranura, dependiendo de los procesos de llegada y salida de paquetes, se podrá tener *p-1, p* ó *p+1* paquetes en el buffer.

Bajo estas condiciones, se deduce que, el estado actual del sistema, $M(n)$, depende únicamente del estado inmediato anterior $M(n-1)$. Esto justamente define la propiedad de Márkov[6], la cual dice que: "*si se conoce la historia del sistema hasta su instante actual, su estado presente resume toda la información relevante para describir en probabilidad su estado futuro"*[7]. Con lo cual, se llega a la conclusión que, el sistema en cuestión, se trata de una cadena de Márkov[7]**.**

2.3.2.1 Diagrama de Transición de Estados [8]

Por tratarse de una cadena de Márkov, lo primero a realizar será el diagrama de transición de estados, es decir, el diagrama en el cual se muestran todos los posibles estados del sistema, así como también, las probabilidades de transición de estado, mismas que se explicarán detalladamente en la Sección 2.3.2.2. En este diagrama, que se observa en la Figura 2-3, se han representado con círculos los estados posibles del sistema, y con arcos de flechas las transiciones entre estados, mostrando también las respectivas probabilidades de transición de estado, recordando que, solo pueden existir transiciones entre estados vecinos, esto debido al hecho que solamente puede llegar como máximo un paquete a la entrada del buffer en cada ranura de tiempo, y de igual manera para el proceso de salida, solamente puede salir un paquete como máximo en cada ranura de tiempo.

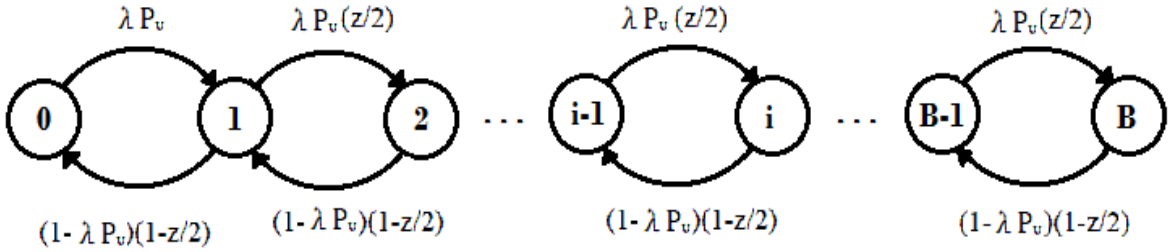
Cabe recordar, que los estados del sistema, $M(n)$, no son más que el número de paquetes almacenados en el buffer al final de la *n*-ésima ranura de tiempo, y

[6] **Andréi Márkov** (1856 - 1922): matemático ruso conocido por sus trabajos en la teoría de los números y la teoría de probabilidades.

[7] **Cadena de Markov:** es un sistema matemático que experimenta transiciones de un estado a otro, entre un número finito de posibles estados. Es un proceso aleatorio caracterizado por la propiedad de no tener memoria, es decir, el estado siguiente depende únicamente del estado actual del sistema y no de la secuencia de eventos precedentes.

[8] **Diagrama de Transición de Estados:** Herramienta usada en teoría de teletráfico para la formulación y solución de modelos de sistemas. Para mayor información sobre este procedimiento, ver referencia [7], Sección 7.4.

como el buffer es de tamaño *B*, se tiene que los posibles estados para el sistema son $M(n) = 0, 1, \dots, B$. Entendiéndose que, $M(n) = 0$ implica que el buffer está vacío, mientras que $M(n) = B$, quiere decir que el buffer está lleno.



Fuente [6]

Figura 2-3. Diagrama de Transición de Estados del elemento de conmutación Banyan Dual.

2.3.2.2 Probabilidades de Transición de Estado

Para esta sección, se debe tener en mente el significado de cada parámetro que aparece en la Figura 2-3, entonces cabe recordar que: λ representa la probabilidad de que a la entrada del SE exista un paquete; P_u representa la probabilidad de que el paquete que ingresa a la entrada del SE vaya destinado al puerto de salida superior; y finalmente, el parámetro *z* es la probabilidad de que el otro buffer asociado al mismo controlador de salida, no esté vacío.

Con estas aclaraciones, se procede a realizar los análisis correspondientes, en base a los cuales se obtienen las probabilidades de transición de estado que se observan en la Figura 2-3.

- **Transición $0 \rightarrow 1$**

Para pasar del estado cero (buffer vacío) al estado uno, se requiere simplemente que, en esa ranura de tiempo, llegue un paquete a este buffer. Para que esto suceda, es necesario que llegue un paquete a la entrada correspondiente del elemento de conmutación, cuya probabilidad es λ, y que dicho paquete vaya destinado al puerto de salida superior, esto es, P_u. Estos dos eventos son

independientes entre sí, y por ende, la probabilidad de que sucedan ambos se la obtiene como el producto de sus probabilidades individuales, esto es: λP_u.

- **Transición** $\boldsymbol{i \rightarrow i+1} \quad (\boldsymbol{i = 1, \ldots, B-1})$

En el caso de los otros estados, para pasar al estado inmediato superior se requieren dos condiciones: que, en esa ranura de tiempo, llegue un paquete a dicho buffer y, que ningún paquete sea transmitido en esa misma ranura de tiempo.

La primera condición ya se analizó en el apartado anterior, por ello se tiene que la probabilidad de que llegue un paquete al buffer es λP_u.

Para el análisis de la segunda condición, es decir, que ningún paquete sea transmitido en esa ranura de tiempo, a su vez, deben suceder dos eventos independientes entre sí, que son:

1. Que, de acuerdo al esquema *Round Robin* del controlador de salida, sea el otro buffer (buffer inferior) el que deba transmitir el paquete en esa ranura de tiempo.

2. Que el buffer inferior, tenga paquetes para transmitir, es decir, no esté vacío.

Para el análisis de estas condiciones, se hace necesario definir el parámetro *z*, que representa la probabilidad de que el buffer inferior no esté vacío:

$$z = \sum_{i=1}^{B} p(i) = 1 - p(0) \qquad Ec.(1)$$

Siendo *p(i)* la probabilidad de estar en el estado *i*, es decir, que existan *i* paquetes almacenados en el buffer inferior.

Además, como en el esquema *Round Robin*, ambos buffers tienen la misma prioridad, resulta que, la probabilidad de que sea el turno de transmitir de cualquiera de los dos es ½.

Dadas ambas condiciones, se concluye que, la probabilidad de que ningún paquete sea transmitido por el buffer superior en esa ranura de tiempo es $\frac{z}{2}$.

Finalmente, debido a que los eventos de llegada y salida de paquetes al buffer son independientes, la probabilidad de transición de estado resulta ser $\lambda P_u \frac{z}{2}$.

- **Transición** $\boldsymbol{j \rightarrow j-1 \quad (j = B, \ldots, 1)}$

Para las demás probabilidades de transición de estado, simplemente se hace el análisis opuesto, es decir, para ir de un estado superior al inmediato inferior se requiere que, en esa ranura de tiempo, se cumplan dos condiciones:

1. Que no llegue ningún paquete al buffer, esto es $(1-\lambda P_u)$.

2. Que un paquete sea transmitido, esto es $\left(1-\frac{z}{2}\right)$.

Como ambas condiciones son independientes entre sí, la probabilidad de transición de estado queda: $(1-\lambda P_u)(1-\frac{z}{2})$

2.3.2.3 Ecuaciones de Corte[9]

Como en este caso, el buffer es de tamaño finito, es claro que se trata de un sistema con pérdidas, ya que si la carga de tráfico es elevada, pueden existir paquetes perdidos.

Ahora bien, todo sistema con pérdidas siempre podrá presentar equilibrio estadístico[10], si el proceso de llegada es independiente del estado del sistema

[9] **Ecuaciones de corte:** Ecuaciones que describen el estado de los sistemas bajo la hipótesis de equilibrio estadístico.

[7], lo cual se cumple para este caso, ya que no existe algún mecanismo de control de flujo que detenga la transmisión de paquetes hacia un determinado buffer cuando el mismo está lleno.

Una vez determinadas las probabilidades de transición de estado, y bajo la premisa que el sistema ha alcanzado el estado de equilibrio, se procede a escribir las ecuaciones de corte. Estas ecuaciones se las obtiene realizando un corte ficticio entre dos estados vecinos, bajo la condición de equilibrio, misma que indica que el proceso de tráfico cambia del estado *i-1* al estado *i* la misma cantidad de veces que cambia del estado *i* al estado *i-1* [7].

En la Ecuación 2, se ha realizado el corte ficticio entre los estados 0 y 1, mientras que en la Ecuación 3, el corte se lo ha realizado entre los estados genéricos *i* e *i-1.*

$$\lambda P_u P(0) = (1 - \lambda P_u)\left(1 - \frac{z}{2}\right) P(1) \qquad Ec.(2)$$

$$\lambda P_u \frac{z}{2} P(i-1) = (1 - \lambda P_u)\left(1 - \frac{z}{2}\right) P(i) \qquad 1 < i \leq B \qquad Ec.(3)$$

Cabe mencionar, que en las Ecuaciones 2 y 3, $P(i)$ representa la probabilidad de que el buffer analizado se encuentre en el estado *i,* es decir, que existan*i* paquetes almacenados en ese buffer en un determinado instante de tiempo.

Ahora, es necesario encontrar una expresión para los diferentes $P(i)$ en función de $P(0)$.

De las Ecuaciones 2 y 3, se observa que es necesario expresar el parámetro *z* en función de $P(0)$, ya que el resto de elementos son constantes.

---

[10]**Equilibrio estadístico:** Estado de un sistema, en el cual la tasa de ocurrencia de los procesos que producen el poblamiento de un nivel o estado debe ser igual a la tasa de ocurrencia de todos los procesos que despueblan dicho estado.

Para ello, cabe hacer una aclaración: al comparar las Ecuaciones 2 y 3 con la Ecuación 1, se ve que la notación para las probabilidades de estado, es diferente ($p(i) \rightarrow P(i)$), esto se lo ha hecho debido a que, la Ecuación 1 se refiere al buffer inferior; mientras que, en las Ecuaciones 2 y 3 se trata del buffer superior. Pero, debido al hecho que el comportamiento de todos los buffers es idéntico, es posible realizar el siguiente remplazo, basado en la Ecuación 1:

$$z = 1 - P(0)$$

Resolviendo la Ecuación 2, con el remplazo mencionado, se tiene:

$$P(1) = \frac{\lambda\, P_u}{(1 - \lambda\, P_u)(1 - \frac{z}{2})}\, P(0) = \frac{2\, \lambda\, P_u}{(1 - \lambda\, P_u)}\left(\frac{P(0)}{P(0) + 1}\right) \qquad Ec.(4)$$

Realizando el mismo procedimiento en la Ecuación 3, queda:

$$P(i) = \frac{\lambda\, P_u \left(\frac{z}{2}\right)}{(1 - \lambda\, P_u)(1 - \frac{z}{2})}\, P(i-1) = \frac{\lambda\, P_u}{(1 - \lambda\, P_u)}\left(\frac{1 - P(0)}{1 + P(0)}\right) P(i-1) \qquad Ec.(5)$$

A continuación, se aplicará las Ecuaciones 4 y 5 para los estados 2 y 3, como se aprecia abajo:

- Estado 2:

$$P(2) = \frac{\lambda\, P_u}{(1 - \lambda\, P_u)}\left(\frac{1 - P(0)}{1 + P(0)}\right) P(1) \qquad Ec.(6)$$

$$P(2) = \frac{\lambda\, P_u}{(1 - \lambda\, P_u)}\left(\frac{1 - P(0)}{1 + P(0)}\right) \frac{2\lambda\, P_u}{(1 - \lambda\, P_u)}\left(\frac{P(0)}{P(0) + 1}\right) \qquad Ec.(7)$$

- Estado 3:

$$P(3) = \frac{\lambda P_u}{(1-\lambda P_u)}\left(\frac{1-P(0)}{1+P(0)}\right)P(2) \qquad Ec.(8)$$

$$P(3) = \frac{\lambda P_u}{(1-\lambda P_u)}\left(\frac{1-P(0)}{1+P(0)}\right)\frac{\lambda P_u}{(1-\lambda P_u)}\left(\frac{1-P(0)}{1+P(0)}\right)\frac{2\lambda P_u}{(1-\lambda P_u)}\left(\frac{P(0)}{P(0)+1}\right) \quad Ec.(9)$$

En base a la Ecuaciones 7 y 9, es posible realizar una generalización para cualquier estado entre 1 y B, así se obtiene la siguiente fórmula:

$$P(i) = 2\left(\frac{\lambda P_u}{(1-\lambda P_u)}\right)^i \left\{\frac{[1-P(0)]^{i-1}}{[1+P(0)]^i}\right\}P(0) \quad 1 \le i \le B \quad Ec.(10)$$

Dado que el sistema siempre se va a encontrar en alguno de los estados, entre 0 y B, se tiene la restricción de normalización:

$$\sum_{i=0}^{B} P(i) = 1 \qquad Ec.(11)$$

Finalmente, aplicando la Ecuación 10 en la Ecuación 11 se tiene:

$$\sum_{i=0}^{B} P(i) = P(0) + \sum_{i=1}^{B} 2\left(\frac{\lambda P_u}{(1-\lambda P_u)}\right)^i \left\{\frac{[1-P(0)]^{i-1}}{[1+P(0)]^i}\right\}P(0) = 1 \qquad Ec.(12)$$

Como los valores tanto de λ como de P_u son constantes, la única incógnita de la Ecuación 12 es $P(0)$, para resolverla se hace uso del software *Matlab*, de lo cual se expondrán los resultados más adelante.

2.3.2.4 Parámetros de interés

Asumiendo que ya se han determinado los valores de las probabilidades de estado, $P(i)$ ahora es posible encontrar el número promedio de paquetes que se encuentran en el buffer, al cual se lo denotará con la letra L. Para ello se hace uso de la definición de valor esperado o media de una variable aleatoria[11], de lo cual se obtiene lo siguiente:

$$L = \sum_{i=1}^{B} i\, P(i) \qquad Ec.(13)$$

Ahora, se denotará como S al número promedio de paquetes perdidos en cada buffer en una ranura de tiempo. Recordando que, como máximo se podría perder un paquete en cada ranura de tiempo, resulta ser que S representa también la probabilidad de que, en una ranura de tiempo, exista un paquete perdido en el buffer en cuestión.

Para que, en una ranura de tiempo, exista un paquete perdido en el buffer, es necesario que se cumplan simultáneamente las tres condiciones siguientes:

1. Que el buffer esté lleno, es decir, que el sistema se encuentre en el estado B, esto es: $P(B)$.

2. Que llegue un paquete a la entrada de ese buffer, λP_u.

3. Que ningún paquete sea transmitido durante esa ranura de tiempo, esto es: $\frac{z}{2}$.

Como las tres condiciones mencionadas son eventos independientes entre sí, la expresión correspondiente al número promedio de paquetes perdidos en cada buffer en una ranura de tiempo queda:

[11] **Variable aleatoria:** variable que no tiene un valor único o fijo, sino que puede tomar un valor de entre un conjunto de posibles valores, cada uno de los cuales tiene asociada una probabilidad de ocurrencia.

$$S = P(B)\lambda P_u \frac{z}{2} \qquad Ec.(14)$$

A continuación, se procederá a encontrar la expresión correspondiente al retardo promedio que experimenta un paquete en el buffer del SE. Para ello, se hará uso del Teorema de Little[12], el mismo que, considerando el sistema en estado estable y que no use ningún tipo de prioridad; en cuanto a las posiciones de espera en el buffer, dice: *"La longitud media de la fila es igual a la intensidad efectiva de llamada multiplicada por el tiempo de espera medio."* [7] Este teorema se expresa matemáticamente en la Ecuación 15:

$$L = N * W \qquad Ec.(15)$$

Para el caso del buffer, la longitud media de la fila es el promedio de paquetes en el buffer y se denota con la letra *L*, mientras que el tiempo de espera medio de los paquetes en la cola del buffer se representa con la letra *W(Waiting Time*).

En cuanto a la intensidad efectiva de llamada, *N*, representa el promedio de paquetes que efectivamente ingresan al buffer en cada ranura de tiempo, para luego ser transmitidos a la siguiente etapa; pero, como se mencionó anteriormente, solo puede llegar como máximo un paquete en cada ranura de tiempo, por lo tanto la intensidad efectiva de llamada, en este caso, es equivalente a la probabilidad de que, en una ranura de tiempo, ingrese un paquete al buffer, esto a su vez se cumple con las dos condiciones siguientes:

1. Que llegue un paquete a la entrada del buffer, esto es, λP_u.

2. Que ese paquete no sea descartado, esto es, $(1 - S)$.

Con ello, la intensidad efectiva de llamada sería $N = (1 - S)\lambda P_u$.

[12]**John Little** (1928-): profesor universitario en el Instituto Tecnológico de Massachusetts (MIT), conocido por su Teorema referente a teoría de colas.

Reemplazando estas expresiones en la ecuación 15, y despejando el tiempo medio de espera en la cola del buffer, se tiene:

$$W = \frac{L}{(1-S)\lambda P_u} \qquad Ec.(16)$$

2.3.2.5 Generalización para toda la red

Todo el análisis que hasta aquí se ha hecho, está basado en un solo buffer del SE; ahora se deberán extender los resultados para los cuatro buffers del SE y, posteriormente, a todas las etapas de la red de conmutación, esto se lo realizará de forma general para una red de tamaño *N.*

A fin de poder obtener los valores correspondientes a los parámetros de interés en toda la red, se hace necesario realizar un análisis de la carga de tráfico presente a la entrada de cada etapa de la red. Es decir, el valor de la variable λ no será el mismo en todas las etapas de la red, sino que se debe considerar el tráfico perdido en cada etapa de conmutadores de la red para obtener el nuevo valor de λ que se usará en la etapa siguiente.

De modo que, en las entradas de la red se tendrá un valor inicial de la variable λ, al cual se lo denotará λ_1, y el mismo que servirá para realizar los cálculos correspondientes a los parámetros de interés en la primera etapa de conmutadores. En base al valor obtenido del promedio de paquetes perdidos en la primera etapa es posible determinar la nueva carga de tráfico que se tendrá en cada entrada de la segunda etapa, a este parámetro se lo denotará λ_2, y a su vez este servirá para calcular los parámetros de interés en la segunda etapa, y así, sucesivamente, en todas las etapas de la red.

Específicamente, estando en la etapa *i*, se tiene el valor actual λ_i. A fin de obtener el valor correspondiente a la siguiente etapa, λ_{i+1}, se procede de la siguiente manera:

1. Con el valor actual, λ_i, se calcula el promedio de paquetes perdidos en un buffer de la etapa actual, s_i.

2. Se multiplica el valor de s_i por el número total de buffers existentes en esa etapa de la red, a este valor se lo denotará S_i (nótese S mayúscula).

3. Se obtiene el número promedio de paquetes que ingresan a las *N* entradas de la red en cada ranura de tiempo, esto es $\lambda_1 N$.

4. Se obtiene también el número promedio de paquetes existentes a la salida de la etapa actual, es decir el valor inicial, $\lambda_1 N$, menos el promedio de paquetes perdidos en la red hasta la salida de la etapa actual, esto es, $\lambda_1 N - \sum_{j=1}^{i} S_j$.

5. Finalmente, se divide este valor para el número de puertos existentes en la red, *N*, y esto corresponde a la nueva carga de tráfico que existirá en las entradas de la siguiente etapa de la red, esto se expresa en la Ecuación 17:

$$\lambda_{i+1} = \frac{\lambda_1 N - \sum_{j=1}^{i} S_j}{N} \qquad Ec.(17)$$

A continuación se explicará brevemente la forma cómo encontrar los valores de los parámetros de interés para toda la red.

- **Promedio de paquetes perdidos, S_T.**

En cuanto al promedio de paquetes perdidos en cada ranura de tiempo en toda la red se deberá tomar los resultados parciales obtenidos en cada etapa, sumarlos y multiplicarlos por el número de buffers existentes en cada etapa de la red. Esto se puede expresar de la siguiente forma en la Ecuación 18.

$$S_T = \left(\sum_{i=1}^{n} s_i \right) x\ 4 \frac{buffers}{SE}\ x\ N^o\ de\ SE\ en\ una\ etapa\ de\ la\ red \qquad Ec.(18)$$

Donde:
n representa el número de etapas de la red de conmutación.
$s_i$representa el promedio de paquetes perdidos en un buffer de la etapa i.

- **Retardo promedio en atravesar la red, W_T.**

Para este parámetro, simplemente se debe sumar los resultados parciales obtenidos en base a la Ecuación 16 para cada etapa de la red de conmutación. Esto, debido a que la Ecuación 16 se la obtuvo considerando uno de los buffers del SE, pero el paquete en cada etapa solamente puede atravesar por uno de dichos buffers, por ello se tiene:

$$W_T = \sum_{i=1}^{n} W_i \qquad Ec.(19)$$

Donde:
n representa el número de etapas que componen la red.
$W_i$representa el retardo promedio en atravesar un buffer de la etapa i.

2.4 CÁLCULOS NUMÉRICOS

En la presente sección se especificarán los valores iniciales de los parámetros correspondientes a las condiciones de tráfico de la red, esto es, los valores de las variables λ y P_u.

Con estos valores, se realizará el respectivo reemplazo numérico en la Ecuación 12, a fin de determinar el valor de $P(0)$; en base a este valor, se resolverá también la Ecuación 10 para así determinar todas las probabilidades de estado del sistema.

Finalmente, con los valores de estas probabilidades se procederá a calcular los valores numéricos de los parámetros de interés que determinan el nivel de

desempeño de la red de conmutación, esto es, retardo promedio, W_T, y tasa de paquetes perdidos, S_T, en base a las fórmulas obtenidas en las Secciones 2.3.2.4 y 2.3.2.5.

Este proceso se repetirá para diferentes valores de la variable B, es decir, para diferentes tamaños de buffer, manteniendo el resto de parámetros constantes.

2.4.1 CONDICIONES DE TRÁFICO

Como ya se mencionó en la Sección 2.3.1, se considerará que se tiene un patrón de tráfico uniformemente distribuido en los puertos de entrada de la red. Entonces, la llegada de paquetes a cada puerto de entrada de los SE, es un proceso de Bernoulli con parámetro λ.

También, se considerará una distribución uniforme de destinos de los paquetes que ingresan a la red, esto quiere decir, que todas las salidas tienen igual probabilidad de ocurrir. De ahí, es posible decir que al entrar un paquete al SE, esta tiene igual probabilidad de ir dirigido a cualquiera de los puertos de salida, es decir, $P_u = 0.5$.

En cuanto a la carga del tráfico, se analizará el sistema bajo la suposición de tener la máxima carga a la entrada, es decir, que en cada ranura de tiempo llegan a la entrada de la red tantos paquetes como puertos de entrada existan, de modo que $\lambda_1 = 1$.

En lo referente al tamaño del buffer, se hará los cálculos numéricos correspondientes a valores de B= 2, 4, 6, …, 32. Cabe mencionar que, en la simulación, solo será posible obtener los resultados correspondientes a valores de B que sean potencia de 2, esto es: 2, 4, 8, 16 y 32. En cambio, en la implementación práctica de la red, debido a limitaciones de memoria del dispositivo a usar, solamente se usará los valores de B=2 y B=4.

2.4.2 ECUACIÓN A RESOLVER

Dadas las condiciones arriba descritas, se reemplaza los valores de λ y P_u en la Ecuación 12, con lo cual se obtiene:

- **Para: $\boldsymbol{\lambda = 1}$ y $\boldsymbol{P_u = 0.5}$**

$$P(0) + 2 * P(0) \sum_{i=1}^{B} \left\{ \frac{[1 - P(0)]^{i-1}}{[1 + P(0)]^{i}} \right\} - 1 = 0 \qquad Ec.(20)$$

Cabe mencionar que esta ecuación se la usará en la primera etapa de la red, mientras que para las siguientes etapas, se reemplazará el nuevo valor de λ en la Ecuación 12, manteniendo P_u constante.

Para resolver estas ecuaciones, se hará uso del software *Matlab*, y en base a ello se obtendrá el valor de *P(0)* para cada valor de B mencionado. Con este valor ya es posible determinar todos los demás parámetros de interés, los cuales se muestran en la Sección 2.5.

2.5 RESULTADOS OBTENIDOS

En el Anexo 2 se exponen de manera detallada los resultados teóricos obtenidos bajo las condiciones ya descritas. En la Tabla 2-1 se muestran los valores de los siguientes parámetros para los diferentes tamaños de buffer:

- *B:* tamaño delos buffers en los elementos de conmutación.

- S_T*:* promedio de paquetes perdidos en toda la red en cada ranura de tiempo.

- W_T*:* retardo promedio de un paquete en atravesar toda la red.

Recordando que la red de conmutación analizada es de tamaño 8x8 y, por ende, consta de 3 etapas, cada una de las cuales tiene 4 elementos de conmutación, se obtuvo los resultados mostrados abajo.

Tabla 2-1. Parámetros de interés para diferentes tamaños de buffer de los SE.

B	W_T	S_T
2	4,79	1,131
4	7,15	0,399
6	9,03	0,213
8	10,70	0,138
10	12,24	0,099
12	13,70	0,076
14	15,09	0,061
16	16,44	0,050
18	17,75	0,042
20	19,05	0,036
22	20,29	0,031
24	21,51	0,028
26	22,74	0,025
28	23,93	0,022
30	25,11	0,020
32	26,25	0,018

Finalmente, a continuación se muestran las gráficas, resumiendo los parámetros que determinan el nivel de desempeño de la red.

En la Figura 2-4 se aprecia la variación del promedio de paquetes perdidos en la red en cada ranura de tiempo (S_T) en función del tamaño del buffer de los SE (B).En la Figura 2-5 se muestra el retardo promedio de un paquete en atravesar la red (W_T) igualmente en función del tamaño del buffer de los SE.

Cabe recalcar que, en la Figura 2-5 el retardo promedio, W_T, está expresado en ranuras de tiempo, con lo cual su valor exacto depende de la frecuencia de operación de la red.

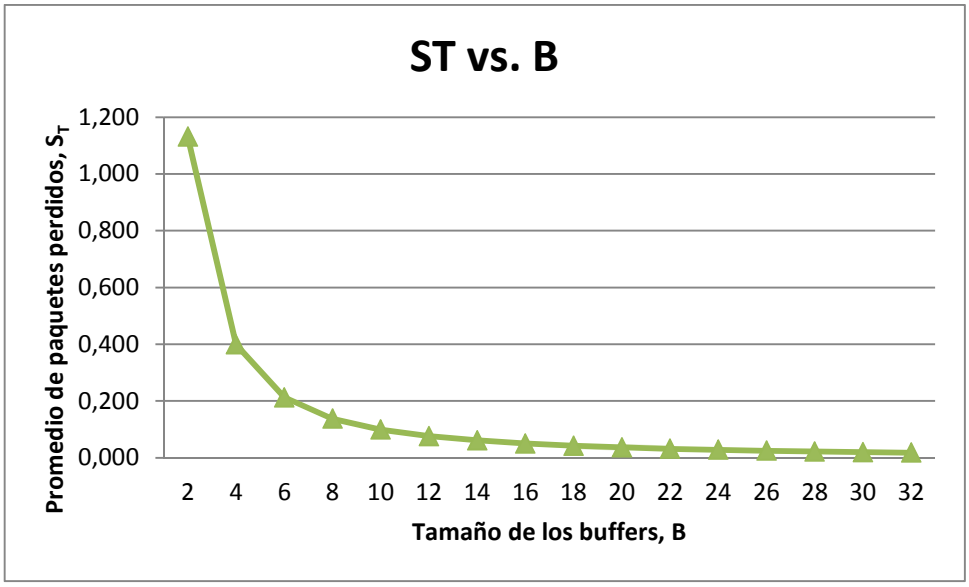


Figura 2-4. Promedio de paquetes perdidos en la red en función del tamaño delos buffers.

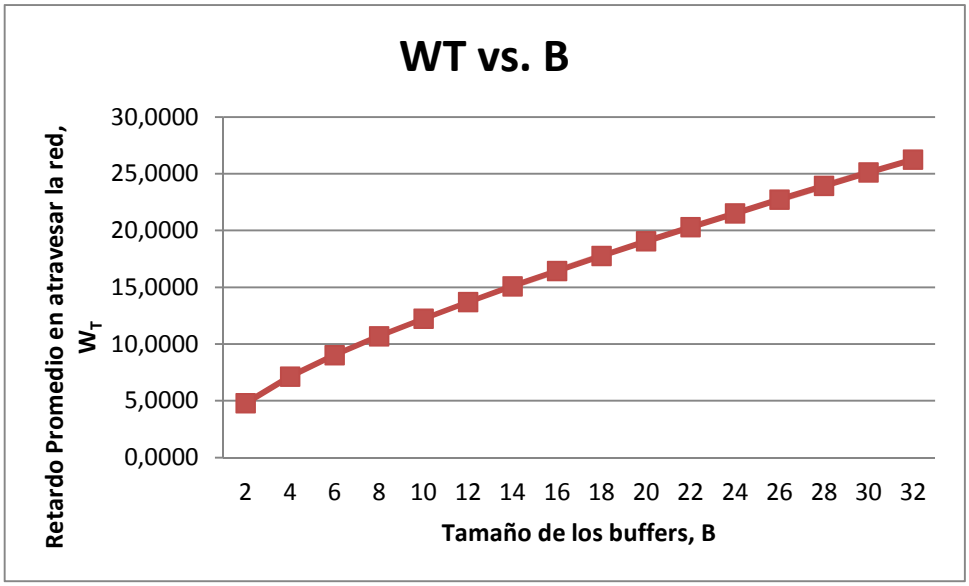


Figura 2-5. Retardo promedio en atravesar la red en función del tamaño delos buffers.

Como se puede observar en las Figuras 2-4 y 2-5, el promedio de paquetes perdidos en la red disminuye rápidamente al aumentar el tamaño de los buffers en los SE, pero como es de esperarse el retardo en atravesar la red se incrementa consecuentemente, de ahí se podría deducir que el tamaño ideal de los buffers en cada SE, a fin de obtener el mejor rendimiento de la red, estaría en el rango entre $B = 4$ y $B = 8$.

REFERENCIAS

[1] A. Pattavina. Switching Theory: Architecture and Performance in Broadband ATM Network.Ed. John Wiley & Sons Ltd, 1998.

[2] Telecommunication Switching and Network System. Disponible: http://www.cmlab.csie.ntu.edu.tw/~freetempo/CN2011/slides/Tutorial_banyan.pdf

[3] Redes De Interconexión. Disponible: http://www.infor.uva.es/~bastida/Arquitecturas%20Avanzadas/Redes.pdf

[4] M. Hidell, P. Sjödin, O. Hagsand. Router Architectures.Disponible: http://web.ict.kth.se/~mahidell/pubs/networking04_tutorial_final.pdf

[5] C. Bouras, C. Gkantsidis. Cost of implementing Banyan networks for use in ATM switching fabrics. Disponible: http://ru6.cti.gr/ru6/publications/8526611.pdf

[6] T. Huang. Mathematical Modeling of a Dual-Banyan Switch in ATM networks.Disponible: http://www.npue.edu.tw/adm/research/%E5%AD%B8%E5%A0%B1/13/10.pdf

[7] UIT-D. Manual sobre Ingeniería de Teletráfico. Ginebra. 2002.

[8] Xilinx. LogiCORE IP FIFO Generator v8.1.User Guide. Marzo 2011. Disponible: http://www.xilinx.com/support/documentation/ip_documentation/fifo_generator_ug175.pdf

CAPÍTULO 3

DISEÑO Y SIMULACIÓN DE LA RED DE CONMUTACIÓN BANYAN

3.1 GENERALIDADES

El diseño e implementación de la red de conmutación Banyan comprende dos fases bien definidas: 1) diseño y simulación de la red Banyan propiamente dicha y, 2) diseño de módulos adicionales e interfaz gráfica de control para permitir la interacción de la PC con el FPGA. En la Figura 3-1 se muestra, en forma resumida, el diagrama de bloques de todo el sistema a implementar.

En el presente capítulo se aborda la primera fase del proceso, es decir, el diseño de la red, junto a su simulación, mientras que en el siguiente capítulo se completarán los componentes faltantes.

En la primera parte del presente capítulo se realiza el diseño en VHDL de cada una de las partes constitutivas de la red de conmutación Banyan, se explica su funcionamiento y se muestra el diagrama correspondiente a dicha entidad, incluyendo una explicación de cada uno de los puertos de entrada y salida de la entidad. Tras interconectar las partes constitutivas, se muestra la entidad correspondiente a la red diseñada.

A continuación, se procede a realizar la simulación dela red de conmutación haciendo uso del simulador *ISim* de Xilinx, se muestran los resultados obtenidos y, finalmente, se procede también a compararlos con los resultados teóricos obtenidos en el Capítulo 2.

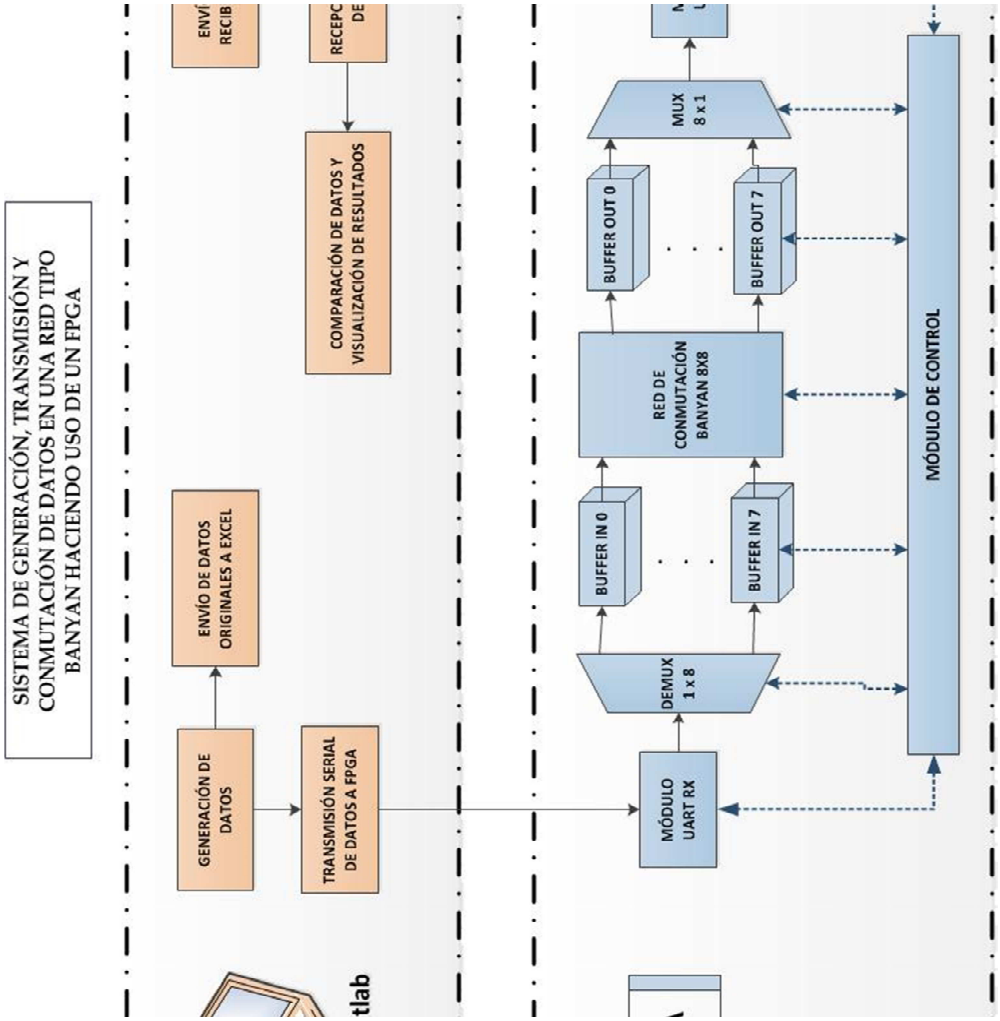


Figura 3-1. Diagrama de bloques del sistema a implementar.

Cabe mencionar que en el Anexo 7 se encuentra el código VHDL debidamente comentado de cada uno de los módulos diseñados, mientras que en el Anexo 3 se muestran las imágenes correspondientes a los resultados de la simulación de cada módulo.

3.2 DISEÑO DE LA RED DE CONMUTACIÓN BANYAN

La red a diseñar es una red Banyan con topología Omega e implementada con elementos de conmutación Banyan duales, como ya se explicó en el Capítulo 2; esta estructura se observa en la Figura 3-2.

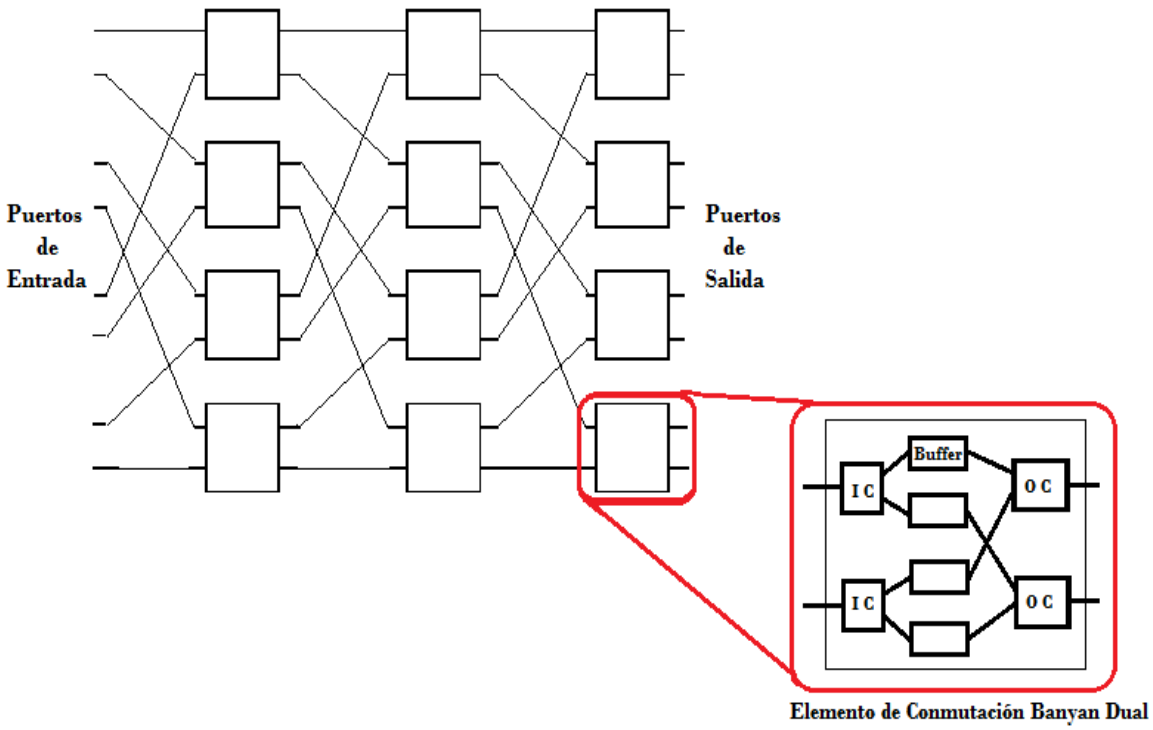


Figura 3-2. Estructura interna de la red Banyana diseñar.

A continuación se realizará el diseño de cada una de las partes arriba mostradas, empezando con el diseño del controlador de entrada (IC), controlador de salida (OC) y buffer interno.

Con el propósito de tener simplicidad en el diseño y agilitar los cálculos, se ha asumido un paquete de datos de 1 byte, al cual se le anteponen 5 bits, inicialmente todos en cero, para la medición del retardo, y se añaden también 3 bits de enrutamiento, esto por tratarse de una red constituida por 3 etapas, con lo cual se tiene un paquete final de 2 bytes como el que se aprecia en la Figura 3-3.

Figura 3-3. Paquete de datos a conmutar en la red.

3.2.1 CONTROLADOR DE ENTRADA

El controlador de entrada(IC) se encuentra presente en cada entrada del elemento de conmutación, y su función es la de escoger la salida correspondiente por la cual debe enviar el paquete que ingresa a su entrada, para ello, este controladortoma en consideración el bit más significativo de los bits de enrutamiento (*routing bits)*. Si dicho bit es 0, enviará el paquete por la salida superior, mientras que si el bit es 1, enviará el paquete por la salida inferior.

Además de escoger la salida adecuada, previo el envío del paquete correspondiente, el controlador de entrada también realiza la tarea de rotar los bits de enrutamiento hacia la izquierda a fin de que, en la próxima etapa, el controlador de entrada correspondiente deba tomar en cuenta el bit más significativo de los bits de enrutamiento, creando así un elemento de conmutación totalmente modular. En la Figura 3-4 se aprecia la entidad del controlador de entrada, mientras que en la Tabla 3-1 se muestra la descripción de los puertos de entrada y salida de dicha entidad.

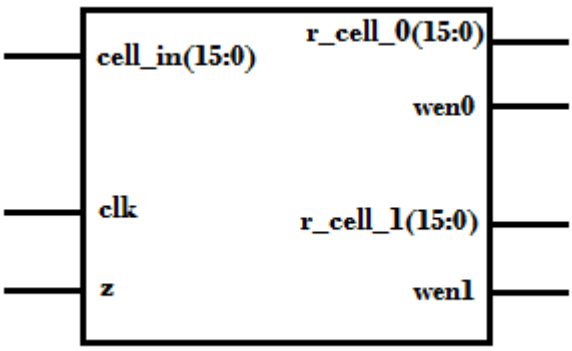


Figura 3-4. Entidad del controlador de entrada.

Tabla 3-1. Puertos de entrada y salida del controlador de entrada.

PUERTO	MODO	TAMAÑO (bits)	DESCRIPCIÓN
clk	In	1	Señal de reloj.
cell_in	In	16	Dato de entrada.
z	In	1	Indicador de que existe o no un dato válido a la entrada.
r_cell_0	Out	16	Puerto de salida superior.

r_cell_1	Out	16	Puerto de salida inferior.
wen0	Out	1	Señal de habilitación de escritura para el buffer superior.
wen1	Out	1	Señal de habilitación de escritura para el buffer inferior.

Como se puede observar, en el diseño de este controlador, se ha incluido la señal *z*, misma que sirve como un indicador de que existe o no un paquete válido en la entrada del controlador. Además, se incluyendos salidas que corresponden a las señales de habilitación de escritura para cada uno de los buffers, obviamente, estas señales no se pueden activar simultáneamente, sino que se activa solamente la del puerto en el cual existe un paquete válido para transmitir.

En cuanto a la descripción en VHDL de éste módulo, por tratarse de un circuito combinacional[13] se hadescrito su funcionamiento usando únicamente sentencias de tipo concurrente; específicamente,el operador de concatenaciónpara la rotación de los bits de enrutamiento, y la sentencia *when* para la asignación de las salidas como se muestra en las líneas a continuación:

```
--rotación de los bits de enrutamiento
cell_rot <= cell_in(cell_in'high downto cell_in'low+etapas) &
            cell_in (etapas-2 downto 0) & cell_in (etapas-1);
--Lógica de salida
 wen0 <= '1' when (z='0' and cell_in(etapas-1)='0') else '0';
```

Como se observa, se ha usado la constante *etapas* en los índices, esto con el fin de tener un diseño escalable en caso de que se quisiera implementar una red de mayor tamaño.

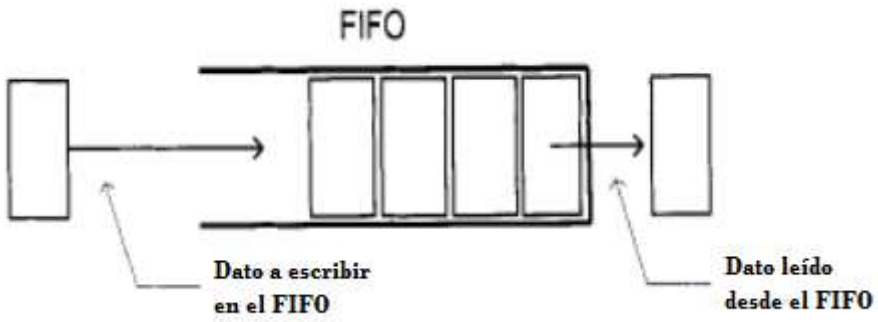
3.2.2 BUFFER INTERNO

3.2.2.1 Funcionamiento

[13] **Circuito combinacional:** circuito en el que sus salidas son función exclusiva del valor de sus entradas en un momento dado, sin que intervengan en ningún caso estados anteriores de las entradas o de las salidas.

Este elemento se encarga de almacenar los paquetes que son procesados por el controlador de entrada y que, posteriormente, serán requeridos por el controlador de salida para su transmisión. Esto corresponde a una aplicación de la técnica de almacenamiento *Virtual Output Queuing*, en la cual se tiene buffers dedicados para cada salida a la que va destinado un paquete, como ya se explicó en la Sección 1.3.6.1.

Estos buffers funcionan con una política de atención FIFO y, para el presente proyecto, se irá incrementando paulatinamente su tamaño (B) a fin de verificar el efecto que esto tiene sobre los parámetros que determinan el nivel de desempeño de la red. En la Figura 3-5 se aprecia el diagrama básico de un buffer FIFO.



Fuente [1]

Figura 3-5. Diagrama básico de un FIFO.

3.2.2.2 Entidad VHDL

En la Figura 3-6 se aprecian los puertos de entrada y salida del buffer, mientras que en la Tabla 3-2 se encuentra una descripción de cada uno de ellos.

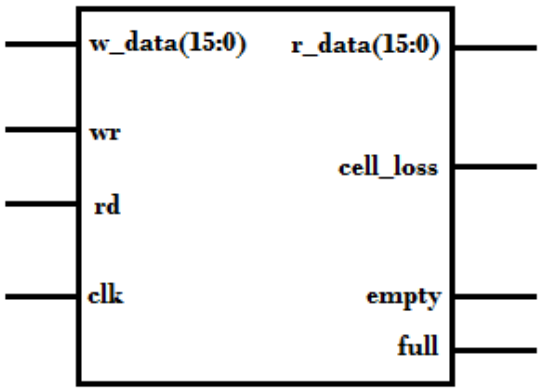


Figura 3-6. Entidad correspondiente a un buffer interno.

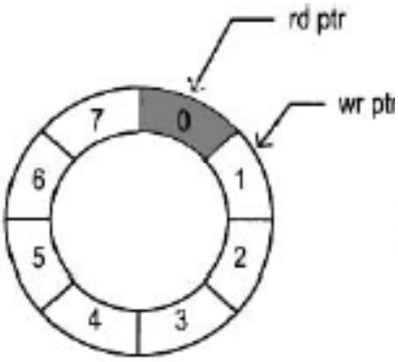
Tabla 3-2. Puertos de entrada y salida del buffer interno.

PUERTO	MODO	TAMAÑO (bits)	DESCRIPCIÓN
clk	In	1	Señal de reloj.
w_data	In	16	Dato de entrada al buffer.
wr	In	1	Señal de habilitación de escritura en el buffer.
rd	In	1	Señal de habilitación de lectura del buffer.
r_data	Out	16	Dato de salida del buffer.
cell_loss	Out	10	Contador de paquetes perdidos.
empty	Out	1	Señal que se activa cuando el buffer está vacío.
full	Out	1	Señal que se activa cuando el buffer está lleno.

3.2.2.3 Diseño del buffer interno

Para este diseño se ha tomado como referencia un esquema de funcionamiento basado en punteros de lectura y escritura (*rd ptr* y *wr ptr,* respectivamente) y un buffer de tipo circular como el que se observa en la Figura 3-7.

El puntero de escritura apunta a la primera posición libre del buffer, y cuando se activa la señal de habilitación de escritura, se escribe el dato de entrada en el buffer y se incrementa el valor del puntero para que apunte a la siguiente posición libre. En caso de que el buffer esté lleno, no se escribe ningún dato en el buffer ni tampoco el puntero incrementa su valor, sino que se incrementa el valor del contador de paquetes perdidos.



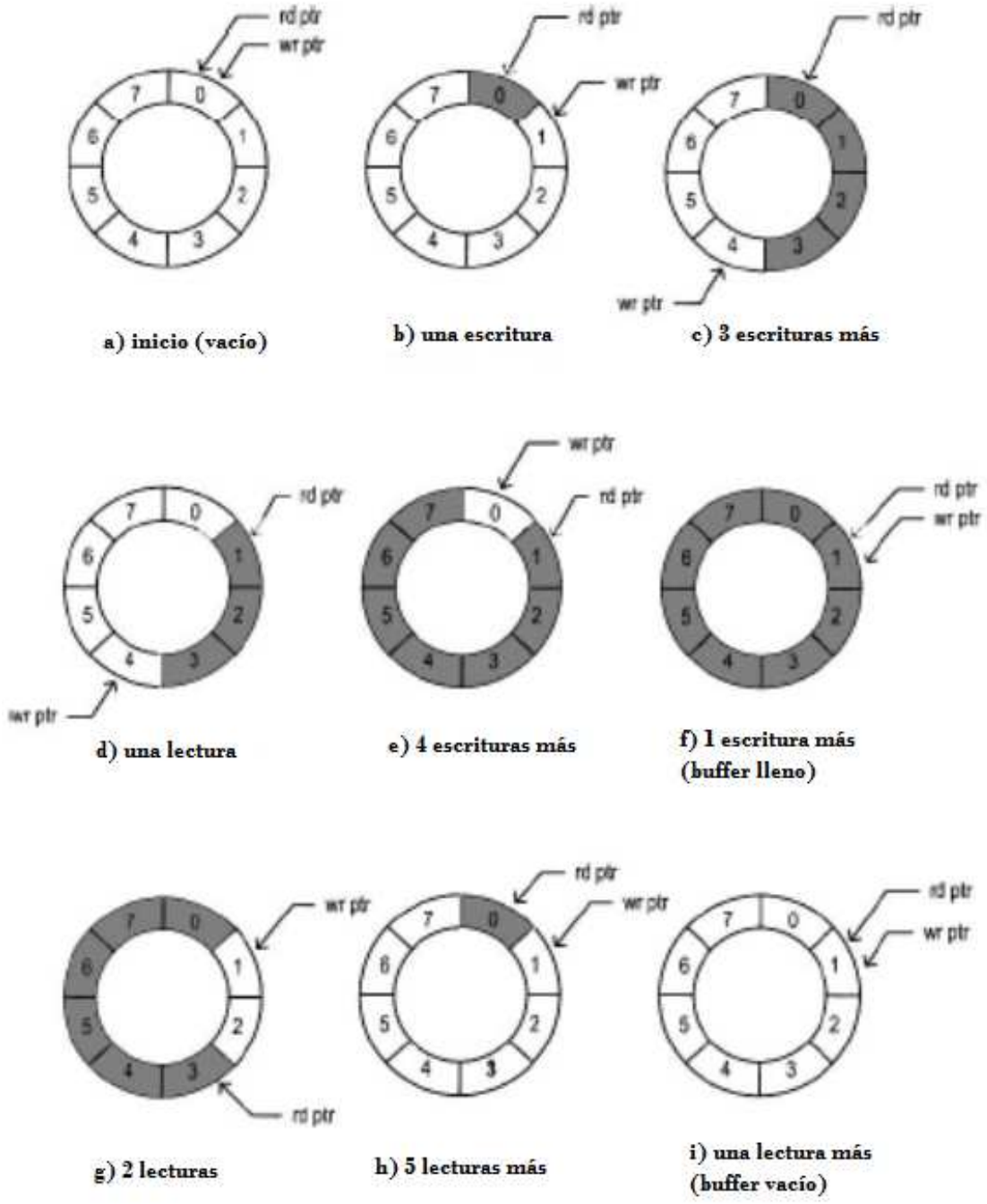
Fuente [1]

Figura 3-7. Buffer circular con punteros de lectura y escritura.

El puntero de lectura, por su parte, apunta a la posición donde se encuentra el paquete que encabeza la cola del buffer, y cuando se activa la señal de habilitación de lectura, el valor de este puntero se incrementa y de esta manera se hace disponible el paquete que se encuentra en la siguiente posición de la cola del buffer.

Cabe aclarar que el paquete apuntado por *rd_ptr*, es decir, el paquete que encabeza la cola del buffer se encuentra siempre disponible para su lectura, de modo que la señal de habilitación de lectura actúa más bien como un indicador de que en esa ranura de tiempo se está realizando la lectura del paquete y que, a continuación, se lo podrá retirar del buffer [1].

En la Figura 3-8 se ilustra el funcionamiento de este tipo de buffer. En esta figura se observa que inicialmente (a) el buffer se encuentra vacío y ambos punteros apuntan a la primera posición del buffer; a medida que se realizan operaciones de escritura (b, c), el valor del puntero de escritura se incrementa, mientras que el puntero de lectura sigue apuntando a la primera posición del buffer. Al realizarse una operación de lectura (d), el valor del puntero de lectura se incrementa; si se realizan más operaciones de escritura, eventualmente, el buffer se llena (e, f) y ambos punteros vuelven a coincidir. Finalmente, se llevan a cabo varias operaciones de lectura (g, h, i) de modo que el buffer vuelve a estar vacío.



Fuente [1]

Figura 3-8. Funcionamiento del buffer circular.

En la descripción VHDL de este módulo se ha hecho uso de la constante B, de modo que, con solamente variar el valor de este parámetro se cambia el tamaño del buffer, es decir el número de paquetes que puede almacenar, esto se lo observa en las líneas a continuación:

```
	--Tamaño del buffer de los SE, B=2,4,...
	constant B: natural := 2;

	--Array de cells para fifo
	type ao_cell is array (natural range<>) of cell;

	--Arreglo para el buffer
	signal array_reg: ao_cell (B-1 downto 0):= (others=>(others=>'0'));
```

Además, es precisamente en los buffers donde se realiza el conteo de paquetes perdidos, así como también, el incremento del valor de los bits de retardo en cada paquete en cada ranura de tiempo, a fin de realizar posteriormente los cálculos de retardo promedio. Abajo se observan las secciones de código VHDL correspondientes a estas tareas:

```
if (clk'event and clk='1') then
	if empty_reg ='0' then-- fifo no está vacío
		--incremento en uno del valor de los bits de retardo
		for i in 0 to 2**W-1 loop--todas las posiciones
		array_reg(i)<=std_logic_vector(unsigned(array_reg(i))+uno);
		end loop;
	end if;

	-- puerto de escritura
	ifwr_op="10" then--solo escritura
		iffull_reg='0' then --buffer no está lleno
			array_reg(to_integer(w_ptr_reg))<= w_data;
		else
			count<= count+1; --conteo de paquetes perdidos
		end if;
	elsifwr_op="11" then--escritura/lectura
		array_reg(to_integer(w_ptr_reg))<= w_data;
	end if;
end if;
```

De las líneas superiores cabe mencionar que las señales *w_ptr_reg* y *rd_ptr_reg* corresponden a los punteros de escritura y lectura, respectivamente.

3.2.3 CONTROLADOR DE SALIDA

El controlador de salida se encuentra presente en cada salida del elemento de conmutación Banyan dual, es decir, existen dos en cada SE. Su función es ir tomando de manera alternada unpaquete de cada uno de los dos buffers que se encuentran conectados a ese controlador y proceder a su envío hacia la siguiente etapa de la red de conmutación o, de ser el caso, al puerto de salida de la red. Como cada controlador de salida opera de manera independiente, cada elemento de conmutación puede enviar dos paquetes en cada ranura de tiempo.

La tarea de arbitraje, la realiza con un esquema *Round Robin* como se explicó en el Capítulo 2,cuyo funcionamiento es como sigue:en la primera ranura de tiempo se verifica si el buffer superior tiene un paquetepara transmitir, de ser así, lo transmite y en la siguiente ranura de tiempo tomará en cuenta primero al buffer inferior; en caso que el buffer superior no tenga paquete para transmitir, se verifica si el buffer inferior tiene alguno que transmitir y de ser así, lo transmite, y de igual forma,en la siguiente ranura será el otro buffer el primero en ser preguntado. De esta forma, el proceso se repite de manera permanente. En la práctica, en caso de que ambos buffers estén vacíos, lo que se envía por el puerto de salida es un paquete con todos sus bits en cero y además se enciende una bandera que indica que no existe un paquete válido. El envío de este paquete vacío es necesario, ya que en caso de no especificar el valor de la salida, en el circuito se generarían *latches*, mismos que no son deseables ya que provocan problemas de sincronismo en el sistema.

En la descripción en VHDL de este módulo, como es de esperarse, se cuenta con un registro (*r0_reg*)para tener seguimiento de cuál buffer está enviando su paquete en la ranura de tiempo actual y así, poder saber cuál buffer debe ser tomado en cuenta primero en la siguiente ranura de tiempo. También, otras señales de entrada importantes son las banderas de *empty* de los buffers (*empty_f*), ya que son las que determinan si existe o no paquete que transmitir en

cada uno de los buffers. A continuación se muestra una sección del código correspondiente al comportamiento de este controlador:

```
if (r0_reg='1') then--ranura actual turno buff_0
      case empty_f is
            when "00" | "10" =>--buffer 1 tiene celdas (e1='0')
                  out_next <= cin1;
                  r0_next <='0';
                  ren0 <= '0';
                  ren1 <= '1';
            when "01" =>--buffer 1 vacío
                  out_next <= cin0;
                  r0_next <='1';
                  ren0 <= '1';
                  ren1 <='0';
            when others =>--ambos buffers vacios
                  out_next <= (others=>'0');
                  r0_next <='0';
                  ren0 <= '0';
                  ren1 <='0';
                  z_next <= '1';
      end case;
else--ranura actual turno buff_1
```

En laslíneas de código mostradas arriba no se ha incluido la sección correspondiente a la rama *else* de la sentencia *if*, por ser similar a la sección mostrada. En el código mostrado, se puede observar la presencia de valores actuales (*_reg*) y valores siguientes (*_next*), esto se hace necesario ya que se trata de un circuito secuencial, en el cual los valores de sus salidas dependen tanto de los valores de las entradas actuales, así como también de valores de las salidas anteriores, por ello, se requiere que existan elementos de memoria para almacenar dichos valores.

En la Figura 3-9 se aprecia el diagrama correspondiente a la entidad del controlador de salida y en la Tabla 3-3 se observa la descripción de los puertos de entrada y salida de esta entidad.

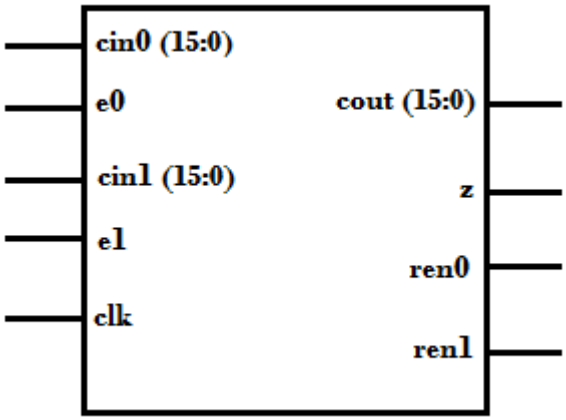


Figura 3-9. Entidad correspondiente al controlador de salida.

Tabla 3-3. Puertos de entrada y salida del controlador de salida.

PUERTO	MODO	TAMAÑO (bits)	DESCRIPCIÓN
clk	In	1	Señal de reloj.
cin0	In	16	Dato de entrada desde el buffer superior.
e0	In	1	Bandera de *empty* del buffer superior.
cin1	In	16	Dato de entrada desde el buffer inferior.
e1	In	1	Bandera de *empty* del buffer inferior.
cout	Out	16	Puerto de salida de datos.
z	Out	1	Indicador de que existe o no un dato válido.
ren0	Out	1	Señal de habilitación de lectura del buffer superior.
ren1	Out	1	Señal de habilitación de lectura del buffer inferior.

3.2.4 ELEMENTO DE CONMUTACIÓN BANYAN DUAL

La función de cada uno de los elementos de conmutación presentes en la red Banyan es la recepción y almacenamiento de los paquetes que recibe en cada una de sus entradas para su posterior transmisión hacia la salida correspondiente.

En la Figura 3-10 se aprecia el esquema del elemento de conmutación Banyan dual, en donde se pueden observar sus partes constitutivas.

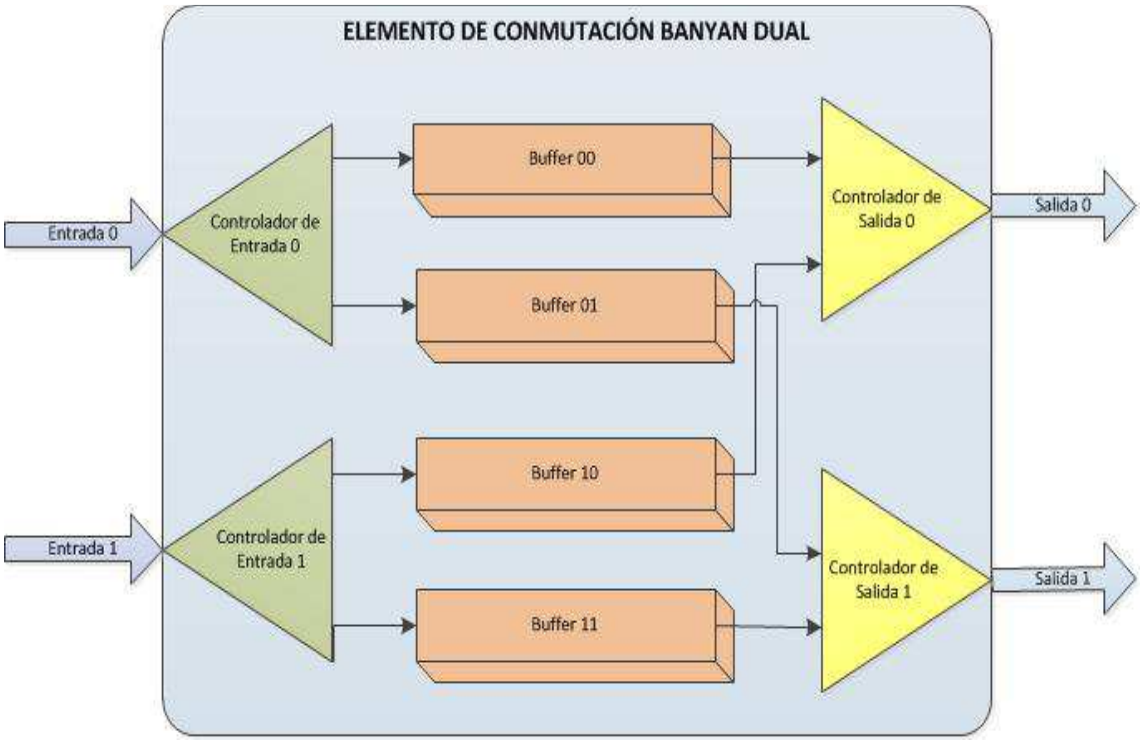


Figura 3-10. Elemento de Conmutación Banyan Dual.

Una vez diseñados los módulos anteriores, es posible interconectarlos a fin de obtener el elemento de conmutación Banyan dual, esto quiere decir, que para la descripción en VHDL de este elemento de conmutación se hace uso del modelo de descripción estructural[14]. A continuación se muestra, a manera de ejemplo, el código necesario para utilizar uno de los controladores de entrada como componente constitutivo del elemento de conmutación:

```
-- Controladores de entrada
entrada_0 :entity work.inp_c(basic)
      port map (clk => clk,
                  cell_in => cell_in_0,
                 z => zin_0,
                  wen0 => w00,
                  wen1 => w01,
                  r_cell_0 => rcell_00,
                 r_cell_1 => rcell_01);
```

[14]**Descripción estructural:** Modo de descripción jerárquico en VHDL, en el cual cada componente debe tener su descripción previa. Aquí, se declaran los componentes y se especifican las interconexiones entre ellos.

En el código mostrado, cabe mencionar que la sentencia *port map* sirve para especificar las conexiones de los puertos de entrada y salida de la entidad llamada *inp_c* con las señales internas del elemento de conmutación.

En la Figura 3-11 se muestra la entidad VHDL correspondiente al elemento de conmutación Banyan dual diseñado, y en la Tabla 3-4 se aprecian los puertos de entrada y salida de esta entidad, junto a su respectiva descripción.

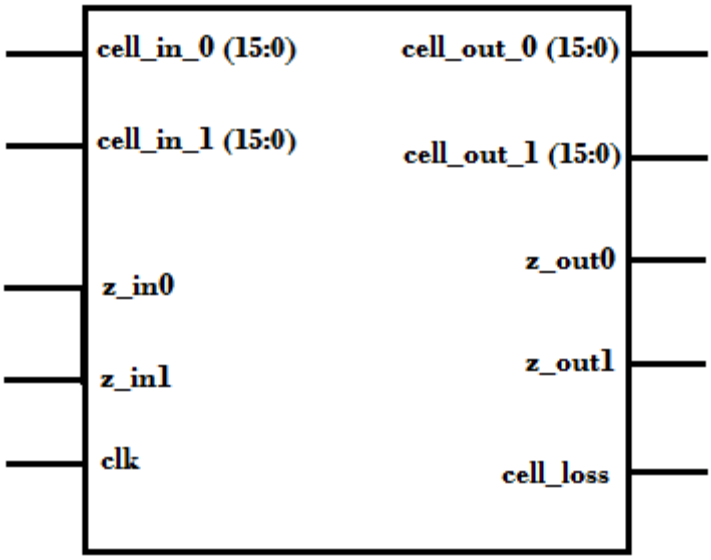


Figura 3-11. Entidad VHDL del elemento de conmutación diseñado.

Tabla 3-4. Puertos de entrada y salida del elemento de conmutación.

PUERTO	MODO	TAMAÑO (bits)	DESCRIPCIÓN
clk	In	1	Señal de reloj.
cell_in_0	In	16	Puerto de entrada de datos superior.
z_in0	In	1	Indicador de que existe un dato válido en la entrada 0.
cell_in_1	In	16	Puerto de entrada de datos inferior.
z_in1	In	1	Indicador de que existe un dato válido en la entrada 1.
cell_out_0	Out	16	Puerto de salida de datos superior.
z_out0	Out	1	Indicador de que existe un dato válido en la salida 0.
cell_out_1	Out	16	Puerto de salida de datos inferior.
z_out1	Out	1	Indicador de que existe un dato válido en la salida 1.
cell_loss	Out	10	Contador de paquetes perdidos.

3.2.5 RED DE CONMUTACIÓN

Se realizará la simulación e implementación de una red de conmutación Banyan con topología específica tipo Omega de 3 etapas, es decir, de tamaño 8x8, como la que se aprecia en la Figura 3-12.

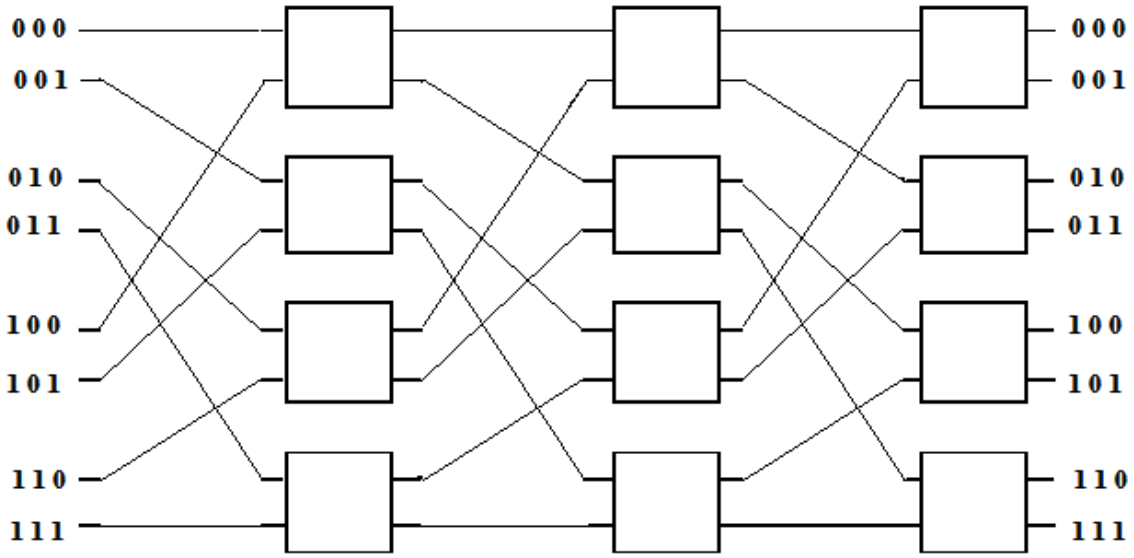


Figura 3-12. Red Omega de tamaño 8x8.

La tarea de la red de conmutación, como su nombre lo sugiere, es la de conmutar los paquetes que recibe en sus entradas a fin de enviarlas por las salidas adecuadas; para ello hace uso de varias etapas constituidas por elementos de conmutación idénticos.

La característica principal de esta red de conmutación es que presenta la propiedad de auto-enrutamiento, es decir, no requiere de un elemento central de control que conozca el estado global de la red en cada momento para, en base a ello, tomar las decisiones de enrutamiento en cada SE, sino que cada elemento de conmutación toma la decisión de enrutamiento de forma totalmente independientemente.

En la Figura 3-13 se observa la entidad correspondiente a la red de conmutación y en la Tabla 3-5 se tiene la descripción de los puertos de entrada y salida de dicha entidad.

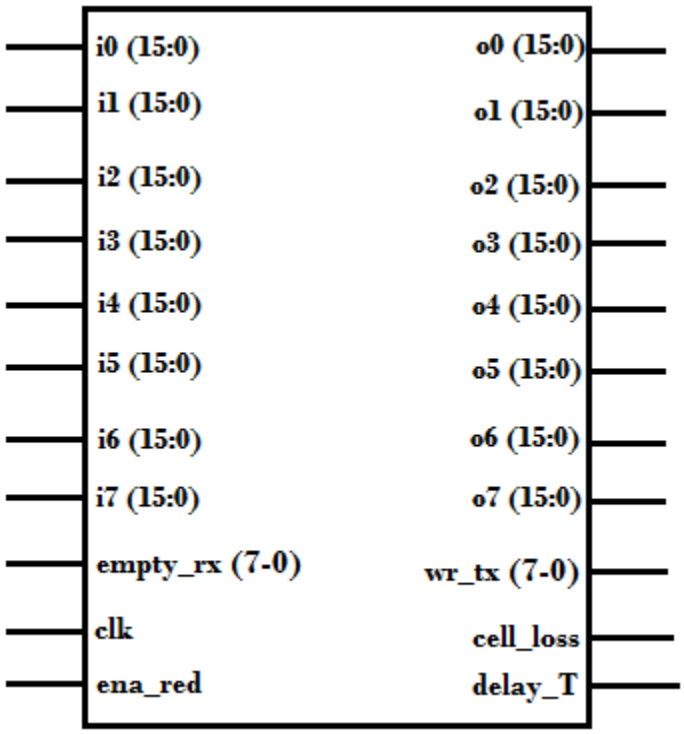


Figura 3-13. Entidad correspondiente a la red de conmutación.

Tabla 3-5. Puertos de entrada y salida de la red de conmutación.

PUERTO	MODO	TAMAÑO (bits)	DESCRIPCIÓN
clk	In	1	Señal de reloj.
ena_red	In	1	Señal de habilitación para la red.
i0 – i7	In	16	Puertos de entrada de datos.
empty_rx	In	8	Banderas de *empty* de entrada.
o0 – o7	Out	16	Puertos de salida de datos.
wr_tx	Out	8	Banderas de escritura de salida.
cell_loss	Out	10	Contador de paquetes perdidos.
delay_T	Out	5	Valor calculado de retardo promedio.

Debido a que en la implementación de la red ésta interactúa con buffers a la entrada y salida de la misma, se ha incluido las señales *empty_rx* y *wr_tx*, que corresponden a las banderas de *empty* de entrada y las señales de habilitación de

escritura para los buffers de salida, respectivamente. Mientras que la señal*ena_red* constituye una señal de habilitación que indica que la red debe empezar a conmutar los datos de entrada. En cuanto a la salida *cell_loss,*ésta se la obtiene como la suma de los paquetes perdidos en cada elemento de conmutación de la red. Así mismo, la señal *delay_T,* constituye el valor calculado del retardo promedio, esto se lo explica en la Sección 3.3.1.

Para la descripción en VHDL de esta entidad, al igual que para el elemento de conmutación, se hace uso del modelo estructural, especificando las conexiones correspondientes entre todos los elemento de conmutación que conforman la red en base a la permutación *Perfect Shuffle*.A fin de obtener un diseño modular y fácilmente escalable, se hace uso de la sentencia *generate*[15] para especificar todos los elementos de conmutación de que consta la red, así como también sus interconexiones. La sección de código correspondiente a esto se aprecia a continuación:

```
--GENERACION RECURSIVA DE SE DE LA RED OMEGA
etapa_gen:
for E in 0 to etapas-1 generate--N° etapas
      SE_etapa_gen:
      for p in 0 to 2**(etapas-1)-1 generate--N° SE por etapa
            omega: SE
            port map (cell_in_0 => ao_inputs(ix(E,2*p)),
                  cell_in_1 => ao_inputs(ix(E,(2*p+1))),
                  clk => clk,
                  zin_0 => z_flag(ix(E,2*p)),
                  zin_1 => z_flag(ix(E,(2*p+1))),
                  zout_0 => z_flag(2*p + (E+1)*(2**etapas)),
                  zout_1 => z_flag(2*p + 1 + (E+1)*(2**etapas)),
                  cell_out_0 => ao_inputs(2*p + (E+1)*(2**etapas)),
                  cell_out_1 => ao_inputs(2*p + 1 + (E+1)*(2**etapas)),
                  cell_loss =>CL(p + E*(2**(etapas-1)))
                  );
      end generate;
end generate;
```

[15]***Generate:***sentencia en VHDL de tipo concurrente utilizada para replicar secciones de hardware.

Del código mostrado, cabe aclarar que las señales *z_flag* y *ao_inputs* corresponden a señales internas de la red declaradas específicamente para realizar las conexiones entre los elementos de conmutación de la red; por otro lado, *ix* es una función que se ha descrito con el fin de calcular los índices correspondientes a la permutación *Perfect Shuffle* de cada uno de los puertos de salida de los elementos de conmutación de la red, tomando en consideración tanto la etapa actual del SE, así como la ubicación del puerto en dicha etapa. El código correspondiente a esta función se aprecia a continuación:

```
--Función para calcular índices (PERMUTACION)
function ix(E, o_ix: natural) return natural is
      variable p: unsigned (etapas-1 downto 0);
      variable et: unsigned (log2c(etapas)-1 downto 0);
      variable s: natural;
begin
      et:= to_unsigned(E,log2c(etapas));
      p:= to_unsigned(o_ix,etapas) ror 1;
      s:= to_integer(et & p);
      return s;
end ix;
```

Como se explicó en la sección 1.2.3 los índices correspondientes a la permutación *Perfect Shuffle* se los obtiene enumerando tanto los puertos de entrada y salida de los elementos de conmutación de etapas adyacentes y expresando estos valores en binario; a continuación se realiza la rotación izquierda de los bits correspondientes al puerto de salida y así se obtiene el valor correspondiente al número de puerto de entrada de la etapa siguiente con el cual se debe conectar. Este mismo principio se lo está usando en la descripción de esta función solamente que de manera inversa, es decir, rotando a la derecha los bits correspondientes al número de puerto de entrada a fin de obtener el número de puerto de salida de la etapa anterior, que corresponde a la señal de entrada para cada uno de los puertos de los elementos de conmutación.

3.3 SIMULACIÓN DE LA RED DE CONMUTACIÓN

En la presente sección, además de mostrar el resultado obtenido de la simulación correspondiente a la red de conmutación, se explica el diseño realizado del módulo para la medición del retardo, así como el análisis que se realizó a fin de determinar aproximadamente el momento en que el sistema alcanza el estado de equilibrio descrito en el Capítulo 2.

3.3.1 MÓDULO PARA CÁLCULO DE RETARDO PROMEDIO

Si bien los módulos hasta aquí mencionados son los indispensables para el funcionamiento de la red de conmutación, se ha elaborado un módulo específico para realizar el cálculo del retardo promedio que experimentan los paquetes en atravesar la red. Este módulo se ubicará a la salida de la red.

Cabe señalar que este módulo se lo utilizará solo con propósitos de simulación, más no para la implementación práctica de la red. Esto debido a que, el operador de división no es sintetizable entre cualquier par de números enteros, sino solamente cuando se trata de potencias de 2 [2].

3.3.1.1 Funcionamiento

La función de este módulo es la de realizar los cálculos necesarios a fin de obtener el valor del retardo promedio que experimentan los paquetes en atravesar todos los buffers de la red de conmutación, este valor se lo expresa en términos de ranuras de tiempo.

Para obtener este valor, se toman los paquetes válidos que existan a la salida de la red en cada ranura de tiempo y se realiza la lectura de los bits de retardo en cada paquete. Recordando que, estos bits de retardo fueron modificados en cada buffer que el paquete atravesó en la red de conmutación. En el código VHDL de

este módulo, se utiliza el arreglo *del_reg* para almacenar el valor de los bits de retardo de cada paquete, un ejemplo se muestra en la siguiente línea de código:

```
del_reg(6) := unsigned( i6(15 downto 16-delay));
```

Con estos bits de retardo, se realiza el cálculo del retardo promedio de los paquetes en esa ranura de tiempo, y ese valor se lo suma a los promedios obtenidos de las ranuras anteriores para, finalmente, dividirlo para el número total de ranuras y así obtener el valor del retardo promedio de los paquetes en atravesar la red de conmutación. La sección de código encargada de esta tarea se muestra a continuación:

```
--conteo de salidas activas
s:= 0;--en base a las banderas z
for i in 0 to 2**etapas-1 loop
      if z_flags(i)= '0' then
            s:= s+1;
      end if;
end loop;
--cálculo del retardo promedio
t := 0;
if s/=0 then
      for k in 0 to 2**etapas-1 loop
            --suma de los valores de retardo
            t:= t + to_integer(del_reg(k));
      end loop;
      --promedio de la ranura actual
      t:= t/s;
end if;
--suma con promedios anteriores
temp:= temp + t;
retardo<= temp;
--valor promedio de retardo
t_delay<= temp/(inc+1);
```

Cabe mencionar que todos estos cálculos se los empieza a realizar desde el momento en que se presume que el sistema haya alcanzado el estado de equilibrio, sobre lo cual setratará en la Sección 3.3.2.

3.3.1.2 Entidad VHDL

En la Figura 3-14 se observa el diagrama correspondiente a la entidad del módulo que mide el retardo de los paquetes, y en la Tabla 3-6 se halla la descripción de los puertos de entrada y salida de esta entidad.

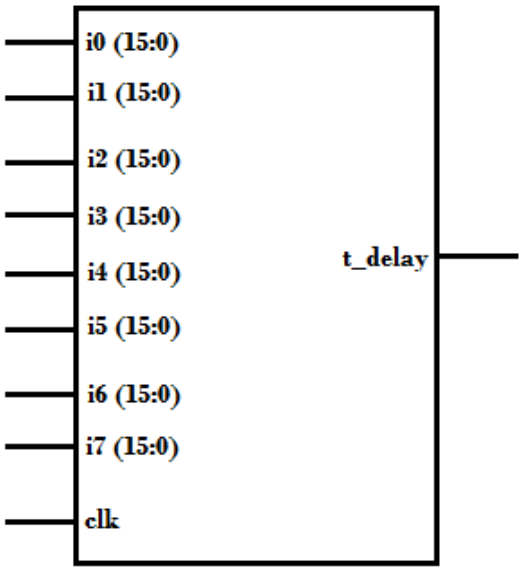


Figura 3-14. Entidad VHDL del módulo para medición de retardo.

Tabla 3-6. Puertos de entrada y salida del módulo de medición de retardo.

PUERTO	MODO	TAMAÑO (bits)	DESCRIPCIÓN
clk	In	1	Señal de reloj.
i0 – i7	In	16	Puertos de entrada de datos.
t_delay	Out	5	Valor calculado de retardo promedio.

3.3.2 EQUILIBRIO ESTADÍSTICO

Este es un punto muy importante a la hora de realizar los cálculos de los parámetros de interés en la red, ya que es necesario que el sistema haya llegado

al estado de equilibrio a fin de poder tomar los datos y realizar los cálculos correspondientes, como se mencionó en el Capítulo 2.

A fin de tratar de determinar el momento en el cual la red llega al estado de equilibrio, se ha realizado un análisis en base al valor promedio de paquetes perdidos en la red y la idea es observar las fluctuaciones en los valores de este parámetro en función del tiempo y, en base a las gráficas que se obtengan,determinar el momento aproximado en el que los valores se estabilizan y por consiguiente, se podría asumir que la red ha alcanzado el estado de equilibrio.

Para ello, en base a los resultados de la simulación obtenida de la red, se han ido descartando paulatinamente las primeras *X* ranuras de tiempo, y se ha encontrado el promedio de paquetes perdidos con los datos restantes. Graficando estos valores se observa el momento aproximado en que se alcanza el equilibrio. En la Figura 3-15 se tiene como ejemplo de este procedimiento, la gráfica correspondiente a los valores obtenidos cuando B=16; en esta figura se observa que, considerando un margen de variación de cuatro centésimas en el valor, aproximadamente se alcanza el equilibrio para el valor de *X=100.*

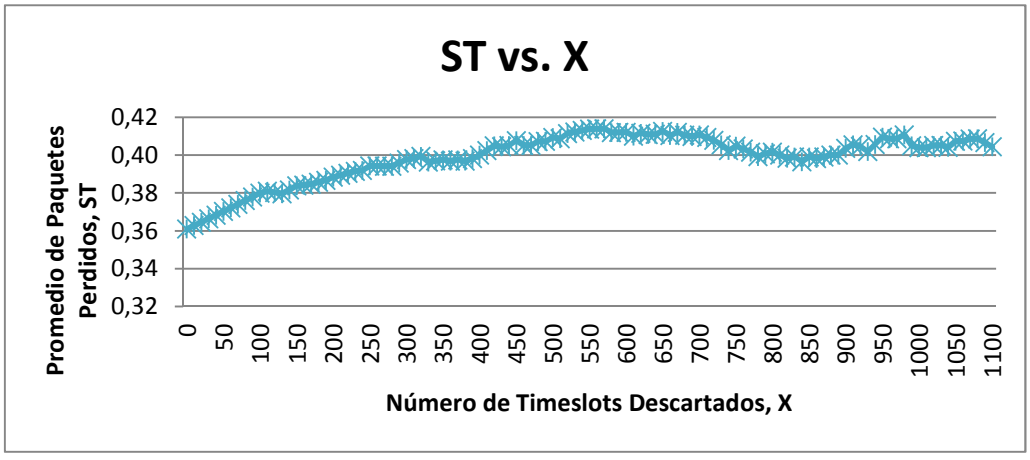


Figura 3-15. Estado de equilibrio para B=16.

Este mismo análisis se lo ha realizado para cada tamaño del buffer, de lo cual se presentan las gráficas en el Anexo 4. En dichas gráficas se puede observar el

momento aproximado en el cual, para cada uno de los tamaños de buffer, el sistema alcanza el estado de equilibrio. En base a este dato, en el código correspondiente al módulo para calcular el retardo promedio, se incluye un contador que, al llegar al valor determinado para alcanzar el equilibrio, indica que es momento de empezar a realizar los cálculos respectivos.

3.3.3 SIMULACIÓN DE LA RED

Como se mencionó anteriormente, en el Anexo 3 se encuentran las gráficas correspondientes a los resultados de la simulación de cada uno de los módulos individuales diseñados. En esta sección, sin embargo, por considerarse necesario, se incluye la Figura 3-16, en la cual se aprecia el resultado obtenido de la simulación de la red de conmutación para un tamaño de buffer B=4.

Como se aprecia, se cuenta con los puertos de salida correspondientes de la red, y además se tiene como salidas, el conteo de paquetes perdidos, así como también el retardo promedio calculado.

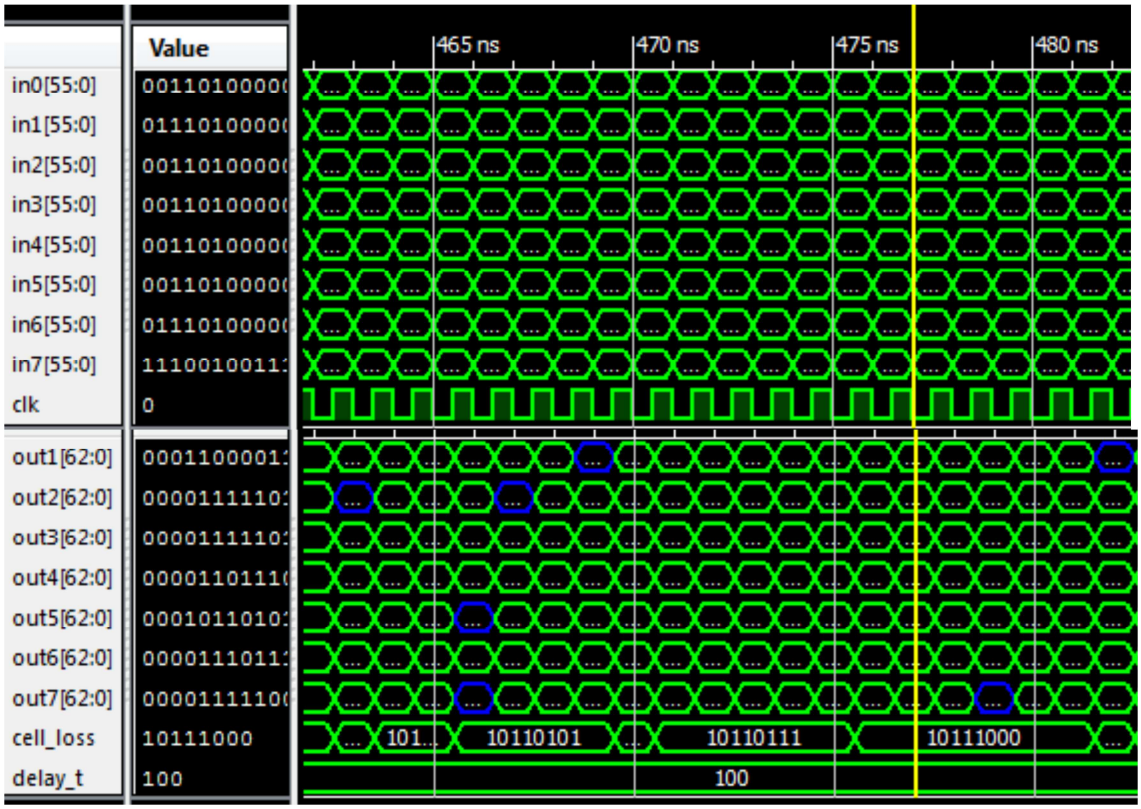


Figura 3-16. Simulación de la red de conmutación con B=4.

Cabe mencionar que en la Figura 3-16, los momentos en los cuales las salidas se encuentran de color azul, corresponde a paquetes vacíos (todos sus bits en cero). Como se observa también, todas las salidas pueden estar activas al mismo tiempo, esto gracias a que los elementos de conmutación Banyan duales pueden enviar dos paquetes en cada ranura de tiempo sin que exista contención.

Cabe recalcar queel archivo que contiene los datos de entrada a la red para la simulación, o banco de pruebas (*testbench*), se lo ha generado usando las condiciones mencionadas en la parte teórica, esto es, máxima carga a la entrada de la red (λ=1), y una distribución uniforme de destinos de los paquetes, es decir, que todas las salidas de la red tienen la misma probabilidad de ocurrencia.

3.3.3.1 Parámetros de interés

En esta sección se presenta el resumen de los resultados obtenidos en la simulación realizada de la red de conmutación Banyan, con la ayuda tanto del software *ISE* como del simulador *ISim*, ambos de Xilinx.

Los dos parámetros de interés, de los cuales se expondrán los resultados son: el promedio de paquetes perdidos en la red en cada ranura de tiempo yel retardo promedio que experimentan los paquetes al atravesar todos los buffers de la red, ambos en función del tamaño de los buffers existentes en cada SE.

En la Tabla 3-7 se resumen los resultados numéricos obtenidos de la simulación realizada a la red de conmutación, y en las Figuras 3-17 y 3-18 se observan estos mismos resultados de forma gráfica. Cabe aclarar que S_T corresponde al promedio de paquetes perdidos en cada ranura de tiempo en los buffers de la red, mientras que W_T representa el retardo promedio que experimenta un paquete en atravesar los buffers de la red, medido en términos de ranuras de tiempo.

Como ya se mencionó, estos resultados se los ha obtenido tomando en cuenta desde el momento en que la red alcanza el estado de equilibrio.

Tabla 3-7. Resultados obtenidos de la simulación de la red de conmutación para diferentes tamaños de los buffers internos.

B	S_T	W_T
2	1,652	4
4	0,717	8
8	0,526	13
16	0,406	13
32	0,397	13

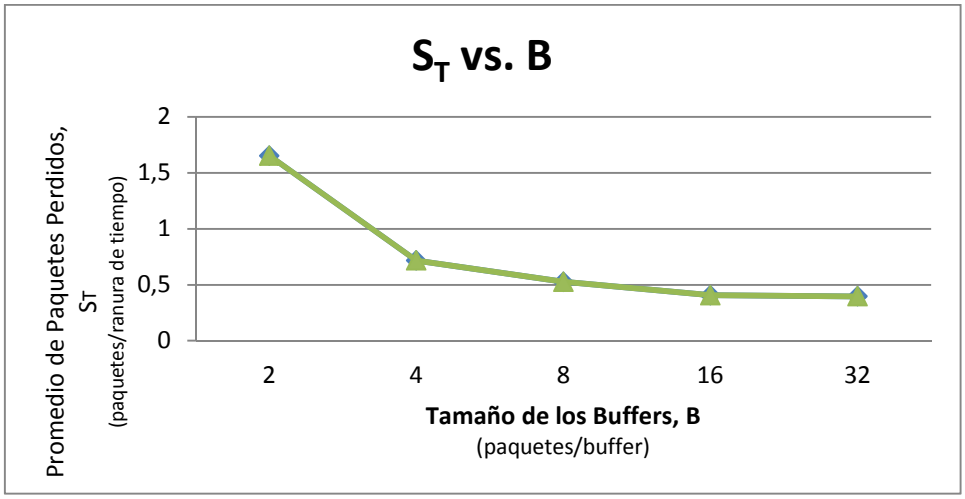


Figura 3-17. Promedio de paquetes perdidos en la red (Simulación).

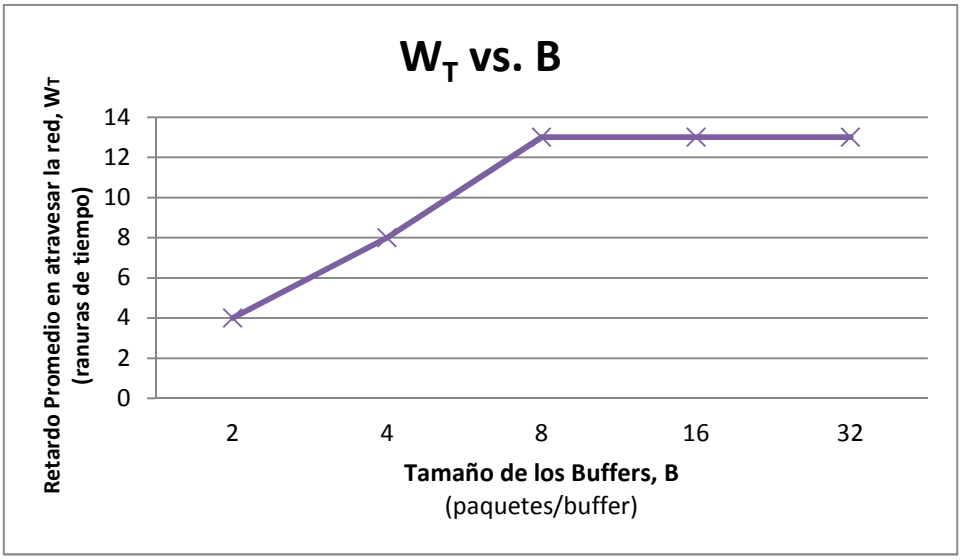


Figura 3-18. Retardo promedio de un paquete en atravesar la red (Simulación).

3.4 COMPARACIÓN DE RESULTADOS TEÓRICOS Y SIMULADOS

En la presente sección se realiza la comparación entre los resultados teóricos obtenidos en el Capítulo 2 con los obtenidos de la simulación de la red de conmutación, tanto en forma de tabla (Tabla 3-8) como en forma gráfica (Figuras 3-19 y 3-20).

Tabla 3-8. Resultados comparativos de los parámetros de interés en la red de conmutación Banyan.

B	Teoría		Simulación	
	W_T	S_T	W_T	S_T
2	5,2225	2,201	4	1,652
4	7,5609	1,203	8	0,717
8	11,1255	0,762	13	0,526
16	16,8215	0,504	13	0,406
32	26,5760	0,321	13	0,397

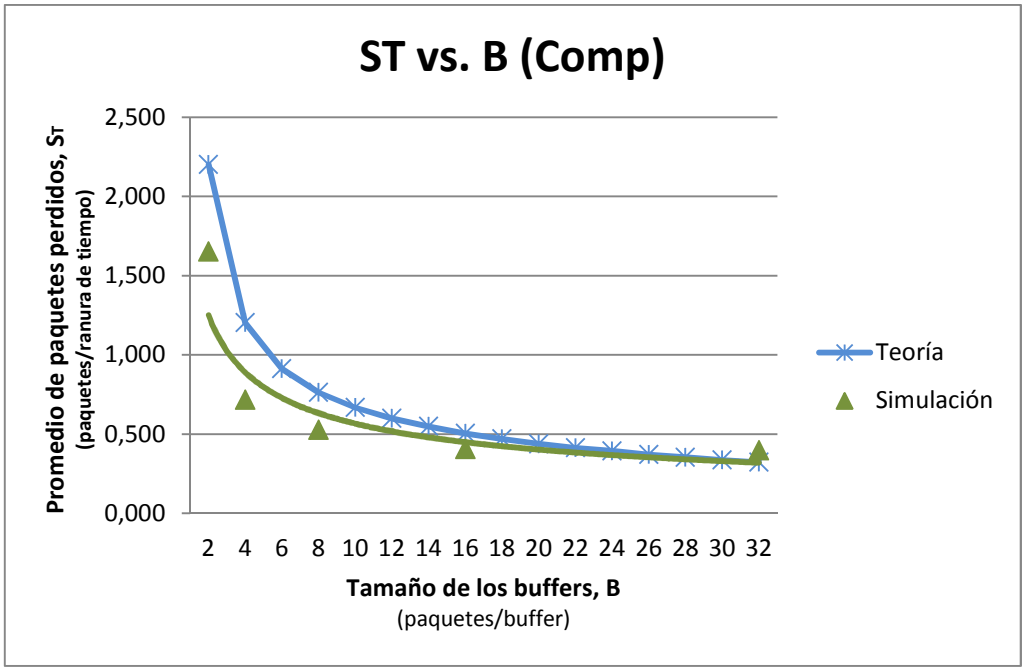


Figura 3-19. Comparación del valor promedio de paquetes perdidos en la red.

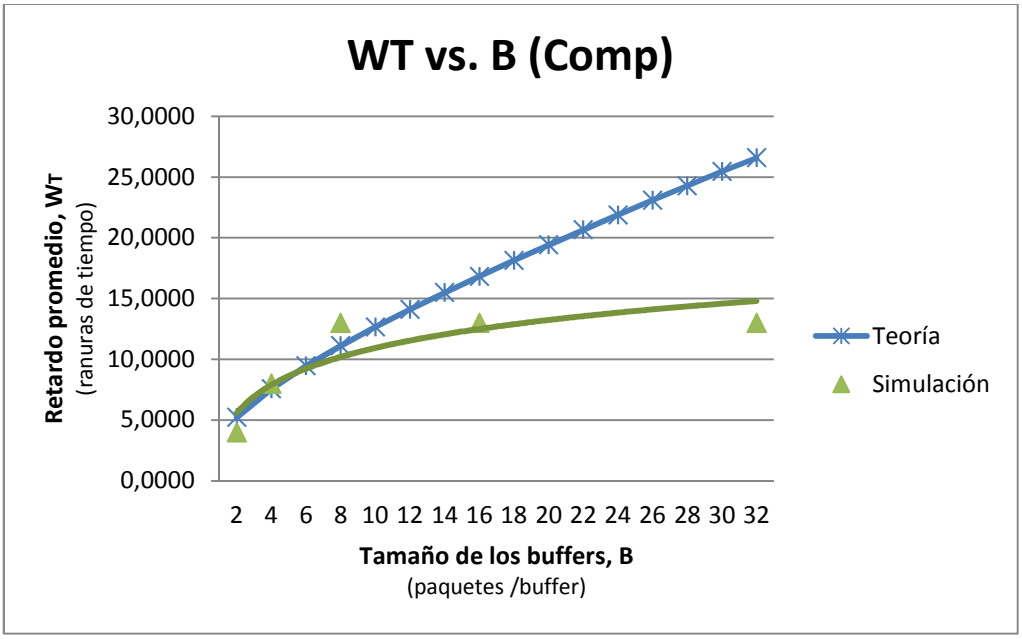


Figura 3-20. Comparación del retardo promedio de un paquete en atravesar la red.

3.4.1 JUSTIFICACIÓN DE DIFERENCIAS

Existen varios factores que influyen en los resultados obtenidos, entre las principales causas que provocan que existan diferencias notorias entre los resultados teóricos con los de la simulación se pueden destacar las siguientes:

- Debido a las limitacionesdel simulador usado en cuanto a resolución y tiempo máximo de simulación, no es posible utilizar una mayor cantidad de ranuras de tiempo en la simulación, lo cual sería deseable a fin de obtener resultados más confiables, y así poder verificar no solamente los resultados de los parámetros de interés, sino principalmente el momento en que el sistema alcanza el estado de equilibrio estadístico.

- En cuanto a los resultados de la simulación, se puede ver que para los valores de B=8, 16 y 32,el número de paquetes perdidos no varía de forma significativa, debido a esto, el valor del parámetro W_T también permanece prácticamente constante.

- Una cuestión muy importante a tomar en cuenta a fin de poder comprender las diferencias existentes entre la parte teórica y la simulación, es que, para el análisis realizado en la parte teórica se asumió que el comportamiento del tráfico era uniforme, se hizo el análisis para un buffer y luego se generalizaron los resultados para el resto de buffers. Esto, en la realidad no es así, ya que si bien para la primera etapa de conmutadores se puede, en cierta medida, garantizar que el tráfico es uniforme y que todos los puertos de salida tienen igual probabilidad, esto no se cumple para las siguientes etapas de la red, en donde, debido al tráfico perdido en las etapas anteriores, ya no se tiene un patrón de tráfico uniforme, ni tampoco se puede garantizar que todas las salidas tienen igual probabilidad de ocurrencia. Esto, obviamente, degrada el comportamiento de la red, principalmente en cuanto al promedio de paquetes perdidos, es por ello que en los resultados de la simulación, la disminución del promedio de paquetes perdidos no es tan pronunciada como el resultado teórico obtenido.

- En cuanto a las notorias diferencias que se presentan entre los resultados teóricos y simulados del parámetro W_T, sería un caso profundo de estudio y principalmente, análisis matemático minucioso con el fin de poder determinar la razón exacta por la cual los valores de las gráficas no convergen al irse aumentando la longitud de los buffers internos de la red.

REFERENCIAS

[1] Pong P. Chu. FPGA Prototyping by VHDL Examples. Editorial John Wiley & Sons, Inc. 2008.

[2] Volnei A. Pedroni. Circuit Design with VHDL.Editorial MIT Press. 2004.

CAPÍTULO 4

IMPLEMENTACIÓN DE LA RED DE CONMUTACIÓN Y PRUEBAS DE FUNCIONAMIENTO

4.1 GENERALIDADES

En el presente capítulo se muestra el diseño de la interfaz gráfica de control, implementada en *Matlab*, explicando cada una de sus funciones. Además, se explica el funcionamiento de los módulos VHDL adicionales necesarios para la implementación práctica de la red, se muestran los diagramas correspondientes a sus entidades junto a una explicación de los puertos de entrada y salida de cada entidad. Una vez diseñados estos módulos se los interconecta para obtener la entidad VHDL de más alto nivel, misma que será sintetizada y, finalmente, implementada en el FPGA.

Para la implementación del sistema diseñado se dispone del FPGA SPARTAN 3-AN de Xilinx, cuya hoja de datos se encuentra disponible en el Anexo 6. Además, en la Figura 4-1 se tiene nuevamente el diagrama de bloques del sistema completo a diseñar a fin de poder identificar cada una de las partes que lo conforman.

Un detalle importante a mencionar es que el diseño del sistema se lo ha realizado de forma paramétrica, haciendo uso de constantes globales colocadas en un paquete[16] dentro de la librería de trabajo, de este modo se logra tener un diseño totalmente flexible y escalable ya que, si se desea modificar alguna característica del sistema, no es necesario realizar cambios en el código sino únicamente especificar el valor deseado del parámetro en cuestión, volver a sintetizar el diseño y descargarlo en el FPGA.

[16]**Paquete:** Unidad funcional en VHDL que se usa para declarar y contener piezas de código de uso frecuente, como por ejemplo: funciones, procedimientos, tipos de datos, constantes globales, etc., las cuales pueden ser usadas por diferentes entidades de un diseño. De esta manera, se proporciona flexibilidad y reusabilidad al código.

Los parámetros susceptibles a modificación son los siguientes:

- Longitud de los buffers internos de la red (B).
- Tamaño de la red (N). Número de puertos de entrada y salida.
- Número de bytes que componen el paquete de datos.
- Longitud de los buffers de entrada y salida de la red.

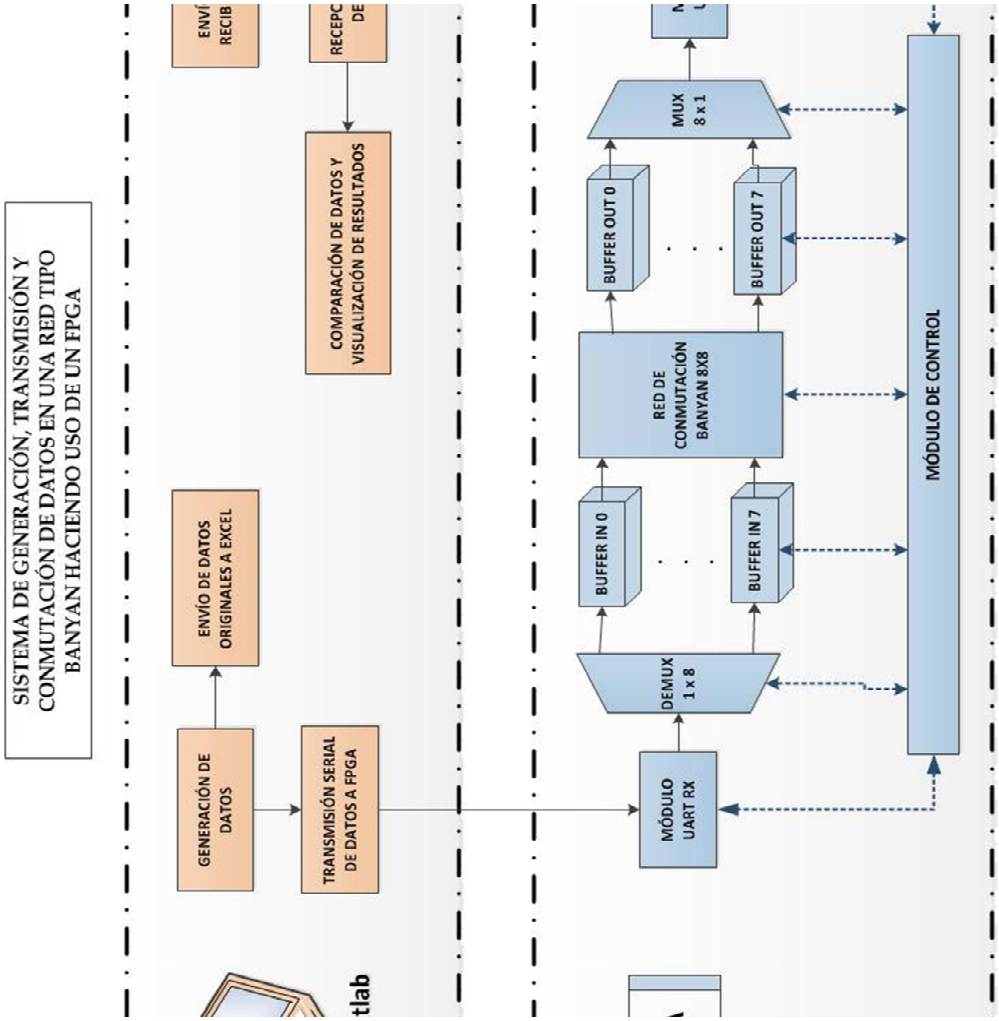


Figura 4-1. Diagrama de bloques del sistema a implementar.

Si bien, en teoría se podría usar cualquier valor para estos parámetros, en la práctica sus valores estarán limitados por las características del FPGA a usarse para la implementación, y especialmente, por la capacidad de memoria RAM disponible.

En el caso de este proyecto, la FPGA a usar dispone, dentro de su arquitectura, de 20 bloques de memoria RAM embebida[17]de 18 kbits cada uno; estos bloques de RAM se los ha usado para la implementación de los FIFOs de entrada y salida de la red, un bloque por cada FIFO, con lo cual se han usado 16 bloques, como consecuencia, el tamaño de la red en este caso queda limitado a solamente 8 puertos de entrada y salida como máximo.

En cuanto a la longitud de los FIFOs de entrada y salida de la red, hay que tomar en cuenta que los FIFOs de salida son del doble de longitud que los de entrada a fin de evitar que se pierdan paquetes ya conmutados por la red. Además, su longitud depende del tamaño de los paquetes a usar; por ejemplo, si se usa paquetes de 16 bits de longitud, y como la capacidad de cada bloque de RAM es de 18kbits, se podrían almacenar hasta 1024 paquetes en cada FIFO de salida y 512 paquetes en cada FIFO de entrada.

Finalmente, los otros dos parámetros, esto es, el tamaño de los buffers internos y la longitud del paquete de datos, son limitados por los propios recursos del dispositivo a usar, específicamente por el número de *flip-flops* disponibles.

En el caso del presente proyecto, a fin de comprobar el adecuado funcionamiento del sistema diseñado,se ha decidido generar, inicialmente, un bloque de 128 paquetes de datos cada vez, esto es, 16 paquetes por cada entrada de la red, recordando que, como se mencionó en la Sección 3.2, el paquete a conmutar tiene una longitud de 16 bits, entre los cuales se incluyen 3 bits de enrutamiento y 5 bits para medición de retardo.

Una vez generado, este bloque de datos será transmitido al FPGA mediante comunicación serial; y, una vez en el FPGA, se procederá a realizar la conmutación de los paquetes a través de la red Banyan y el incremento de los bits de retardo en cada uno de los paquetes.

[17]**RAM Embebida (*Block RAM*):** Bloques de memoria RAM que se encuentran distribuidos en medio del arreglo de CLBs (Bloque Lógico Configurable) del FPGA o a lo largo de su superficie.

Luego de ser conmutados, estos paquetes son nuevamente enviados por comunicación serial a la PC a fin de comprobar que se ha realizado la conmutación adecuadamente y, finalmente, se visualizan los resultados obtenidos.

A continuación, se empezará a explicar el diseño de la interfaz gráfica de control, así como de los módulos VHDL a implementar.

4.2 INTERFAZ GRÁFICA DE CONTROL

Esta interfaz se la ha realizado utilizando la herramienta GUIDE incluida en el software *Matlab*, y cumple las siguientes funciones:

- Comprobación de conectividad con el módulo de desarrollo y el FPGA.
- Edición de parámetros susceptibles a modificación.
- Generación de los datos de forma aleatoria.
- Envío de datos originales a Excel.
- Transmisión serial de los datos generados hacia el FPGA.
- Recepción serial de datos conmutados desde el FPGA.
- Envío de datos recibidos a Excel para su comparación.
- Visualización de los parámetros de interés obtenidos como resultado de la conmutación de los paquetes enviados.

En la Figura 4-2 se observa la interfaz de usuario creada, cabe mencionar que el código correspondiente a esta interfaz se encuentra en el Anexo 5.

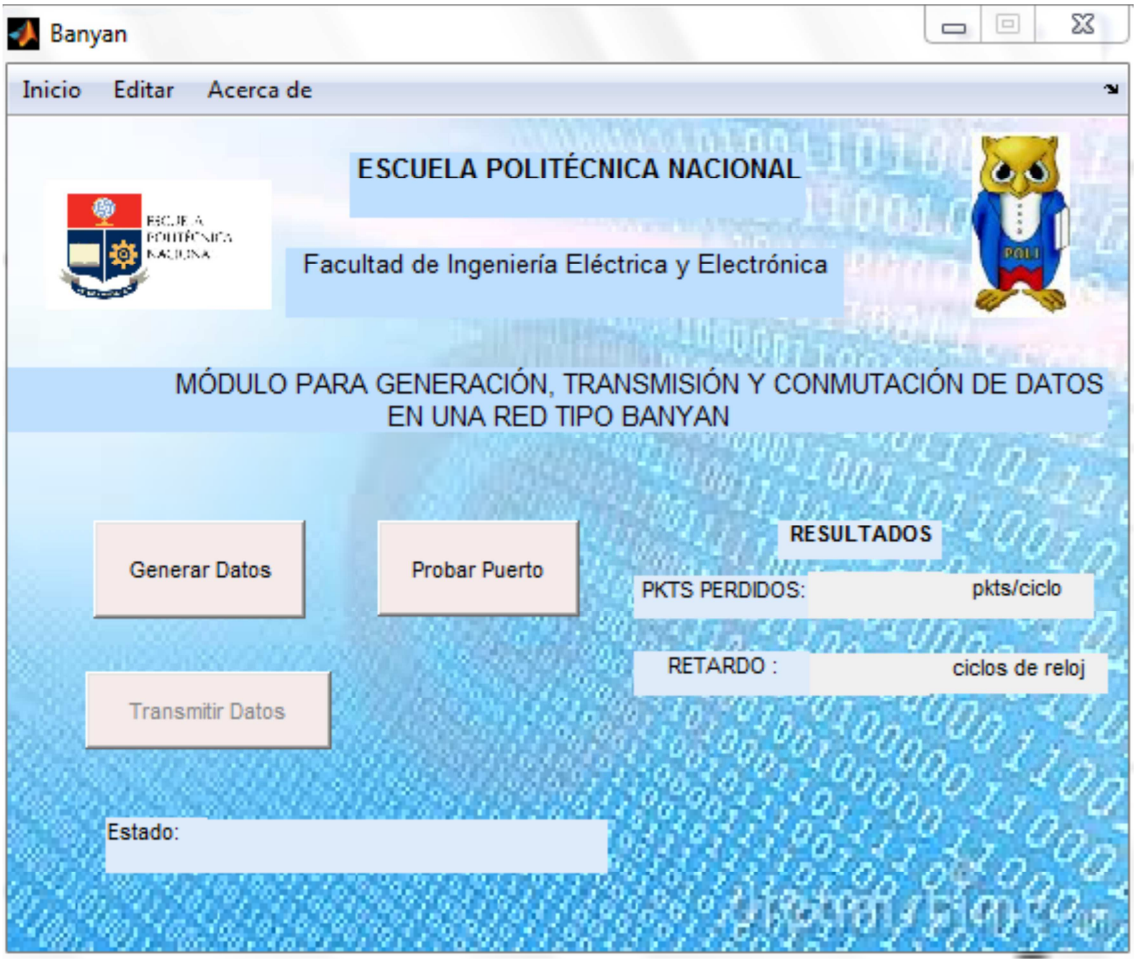


Figura 4-2. Interfaz gráfica de control.

A continuación, se procederá a describir el procedimiento con el que se ha desarrollado cada una de las funciones arriba mencionadas de esta interfaz de control.

4.2.1 COMPROBACIÓN DE CONECTIVIDAD ENTRE EL COMPUTADOR Y EL FPGA

Esta función se la ejecuta en la interfaz gráfica el presionar el botón 'Probar Puerto' o bajo el Menú Editar-> Probar Puerto, y su finalidad es la de verificar que la comunicación serial con el FPGA se haya establecido adecuadamente.

Básicamente lo que se hace es enviar un caracter específico hacia el FPGA y esperar recibir del FPGA otro caracter específico como respuesta; en caso de que

se reciba el caracter correcto, se dice que se ha establecido la conexión, como se aprecia en la Figura 4-3, caso contrario, se dice que la conexión ha fallado.

Figura 4-3. Conexión establecida con el FPGA.

4.2.2 EDICIÓN DE PARÁMETROS SUSCEPTIBLES A MODIFICACIÓN

Si bien inicialmente se estableció que se enviarían 16 paquetes por cada entrada de la red y que la longitud del paquete sería de 16 bits (8 bits de datos y 8 bits de control), estos valores predeterminados se los puede cambiar en la interfaz gráfica; esto es gracias a que se ha realizado un diseño paramétrico tanto de la red de conmutación como de la interfaz gráfica, de modo que el usuario está en capacidad de modificar los siguientes parámetros:

- Número de paquetes a enviar por cada puerto de entrada de la red: los valores posibles de este parámetro están en el rango entre 16 y 512 paquetes, recordando que dichos valores deben ser potencias de dos.

- Número de bits de datos por paquete: este parámetro puede tomar los valores de: 8, 16 y 24 bits de datos por cada paquete, a los cuales se les añadirá los bits para medición de retardo y para enrutamiento; con lo cual resulta que se podrá tener un paquete de longitud máxima igual a 32 bits.

La modificación de estos parámetros se la realizabajo el Menú Editar -> Editar Parámetros, con lo cual aparece la ventana que se observa en la Figura 4-4.

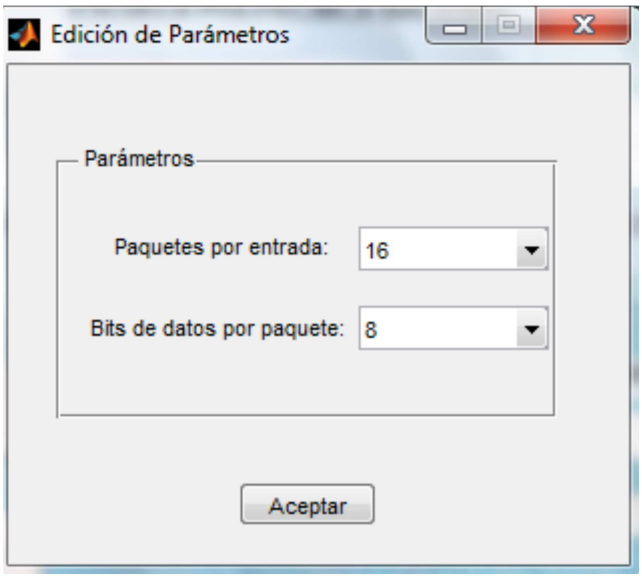


Figura 4-4. Ventana para edición de parámetros.

Cabe mencionar que a fin de que el sistema funcione adecuadamente, previo a la modificación de parámetros en la interfaz gráfica, se debe verificar que el FPGA esté programado con el archivo de configuración adecuado para dichos parámetros.

4.2.3 GENERACIÓN DE DATOS

Para la generación de los paquetes a transmitir, se ha tomado como referencia el formato de paquete mencionado en la Sección 3.2, según el cual se tienen 5 bits (inicialmente en 0) para la medición de retardo, 8 bits de datos propiamente dichos y 3 bits de enrutamiento.

Como ya se mencionó, se generarán 16 paquetes por cada entrada de la red, es decir, que se deberá generar un total de 128 paquetes. Para ello, se procede a generar el vector con los bits para medición de retardo, se genera también un vector con los bits de datos, que corresponden a enteros aleatorios generados entre 1 y 255, a fin de poder expresarlos en 8 bits;finalmente, se genera de manera aleatoria el vector con los 3 bits de enrutamiento para cada paquete. Finalmente, haciendo uso del comando *cat*, se concatenan los 3 vectores antes generados para así obtener los paquetes finales de datos.

En la interfaz gráfica de control, esta función se la ejecuta al presionar el botón Generar Datos o bajo el Menú Editar-> Generar Datos, como se aprecia en la Figura 4-5.

Figura 4-5. Generación de datos en interfaz gráfica de control.

4.2.4 ENVÍO DE DATOS ORIGINALES A EXCEL

A esta sección de código se la ejecuta juntamente con la de generación de datos; y, lo que se hace es repartir la matriz generada con los paquetes de datos originales hacia 8 matrices (una por cada entrada de la red de conmutación), cada una de las cuales contiene 16 paquetes. A continuación, se hace uso del comando *xlswrite*, y se envía estas matrices a diferentes hojas de un Libro de Excel, previamente creado. En la Figura 4-6 se tiene un ejemplo de una de las hojas del libro de Excel que contienen los datos generados correspondientes a uno de los puertos de entrada de la red, en la figura se aprecia que se ha dividido cada paquete en sus tres campos: RETARDO: bits para medición de retardo, DATOS: bits de datos, SALIDA: bits para enrutamiento.

	A	B	C	D	E	F
1	DATOS DE ENTRADA			DATOS DE SALIDA		
2	RETARDO	DATOS	SALIDA	RETARDO	DATOS	SALIDA
3	0	22	1			
4	0	67	6			
5	0	205	5			
6	0	8	3			
7	0	237	1			
8	0	187	3			
9	0	125	3			
10	0	148	0			
11	0	61	4			
12	0	118	1			
13	0	246	3			
14	0	140	4			
15	0	133	2			
16	0	60	2			
17	0	125	4			
18	0	160	2			

Figura 4-6. Ejemplo de datos originales en Excel.

4.2.5 TRANSMISIÓN SERIAL DE DATOS GENERADOS

Una vez generados los datos, se habilita la opción ‘Transmitir Datos’ en la interfaz de control. Se debe recordar que cada uno de los paquetes generados está

compuesto por 2 bytes y por lo tanto se hacen necesarios los siguientes procedimientos previos a su transmisión propiamente dicha:

- Transformar a formato binario la matriz con los datos generados.
- Almacenar en forma separada las matrices con los bytes menos significativos y los más significativos, esto debido a que se desea transmitir primero el byte menos significativo de cada paquete.
- Convertir a decimal estas matrices.
- Concatenarlas nuevamente cambiando el orden de los bytes, es decir, primero los menos significativos y luego los más significativos.
- Convertir esta matriz resultante en un vector fila para su transmisión.
- Dar aviso al FPGA de que se va a empezar con la transmisión de datos, esto se consigue enviando el carácter J hacia el módulo de desarrollo.
- Transmitir los datos.

4.2.6 RECEPCIÓN DE DATOS

Una vez finalizada la transmisión de datos, inmediatamente el FPGA realiza la conmutación de los datos enviados y envía los resultados de regreso. Por esto, inmediatamente luego de finalizar la transmisión de los datos desde el PC, se espera la recepción de los resultados enviados por el FPGA, este proceso de recepción consta de las siguientes etapas:

- Recepción de los *data_count* de cada buffer de salida de la red, esto es, el número de paquetes presentes en cada uno de ellos.
- Conversión a decimal de los *data_count* y multiplicación por 2, debido a que cada paquete se compone de 2 bytes.

- Envío del carácter R, indicando al FPGA que inicie la transmisión de los paquetes de datos conmutados.

- Recepción de los datos enviados desde el FPGA.

4.2.7 ENVÍO A EXCEL DE DATOS RECIBIDOS

Una vez que se reciben los datos desde el FPGA, se procede a re ensamblarlos, debido a que estos llegan en forma de un vector de datos continuo, y se los desea tener organizados en forma de matriz, en la cual cada fila sea un paquete. Ahora, es necesario transformar a decimal de forma separada los bits de retardo, los bits de datos y los bits de enrutamiento. Así, finalmente, es posible tener los datos en la forma deseada para su envío a Excel.

	A	B	C	D	E	F
1	DATOS DE ENTRADA			DATOS DE SALIDA		
2	RETARDO	DATOS	SALIDA	RETARDO	DATOS	SALIDA
3	0	12	0	2	110	2
4	0	193	3	2	178	2
5	0	62	2	3	62	2
6	0	113	3	4	227	2
7	0	176	2	4	100	2
8	0	92	1	4	56	2
9	0	188	3	5	176	2
10	0	101	0	4	193	2
11	0	175	1	2	94	2
12	0	180	3	4	172	2
13	0	113	2	5	113	2
14	0	5	3	5	133	2
15	0	85	2	5	85	2
16	0	109	2	4	238	2
17	0	69	0	0	0	0
18	0	51	2	0	0	0
19	0	0	0	0	0	0

IN 0 | IN 1 | **IN 2** | IN 3 | IN 4 | IN 5 | IN 6 | IN 7

Figura 4-7. Ejemplo de datos recibidos en Excel.

En la Figura 4-7 se observa un ejemplo de una de las hojas de Excel correspondiente al puerto número 2 de la red de conmutación, mostrando los valores tanto de los paquetes que ingresaron por este puerto como de los

paquetes recibidos a la salida de la red por este mismo puerto. En esta figura se puede observar que, efectivamente, el valor de los bits de retardo se ha incrementado y que los datos recibidos corresponden únicamente a los paquetes destinados para este puerto específico.

4.2.8 VISUALIZACIÓN DE RESULTADOS

La tarea final de la interfaz gráfica de control es la de mostrar los resultados obtenidos de los parámetros de interés referentes al nivel de desempeño de la red, esto es: número de paquetes perdidos y retardo promedio de los paquetes en atravesar la red.

En cuanto al número de paquetes perdidos, este dato se lo obtiene como la diferencia entre el número de paquetes enviados y el número de paquetes recibidos; mientras que el cálculo del retardo promedio se lo realiza en base al valor de los bits de retardo de los paquetes recibidos. En la Figura 4-8 se observa la interfaz gráfica mostrando los resultados obtenidos de la conmutación de los paquetes.

Figura 4-8. Visualización de resultados obtenidos.

4.3 MÓDULOS VHDL DISEÑADOS

En la presente sección se describen en detalle cada uno de los módulos VHDL diseñados para la implementación del sistema de conmutación de paquetes de datos que recibe el FPGA. Estos módulos son los necesarios para realizar la comunicación serial con la PC, el ordenamiento y el almacenamiento de los paquetes de datos.

Como se mencionó anteriormente, se ha realizado un diseño paramétrico, de modo que los valores de longitud del paquete de datos y longitud de los buffers de entrada y salida de la red que se mencionan en las secciones siguientes corresponden a los valores iniciales asumidos, mismos que son susceptibles de modificación.

4.3.1 MÓDULO DE COMUNICACIÓN SERIAL UART

En el FPGA es necesario el diseño de un módulo para establecer la comunicación serial con la PC. Esto se consigue mediante la implementación de un módulo UART[18]. La descripción y diseño de este módulo se lo realiza en base a la referencia [1]. El sistema de comunicación UART consiste de un subsistema de recepción y uno de transmisión.

El transmisor, básicamente, es un registro de desplazamiento, el cual recibe un byte en paralelo y lo va desplazando bit a bit a una velocidad determinada. En tanto que el receptor va recibiendo bit a bit los datos y los re-ensambla para su posterior uso.

4.3.1.1 Subsistema de Recepción

El subsistema de recepción del UART se compone de 3 elementos:

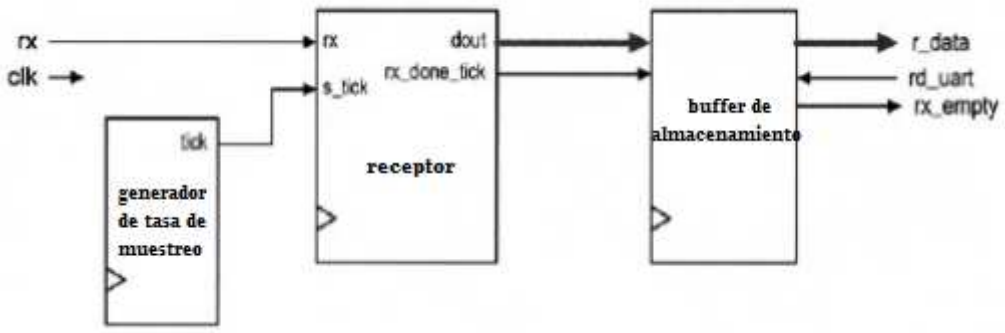
- Generador de tasa de muestreo: Circuito encargado de producir una bandera de habilitación que indica que es tiempo de realizar la lectura del bit recibido.

- Unidad de recepción: Módulo que realiza la lectura de cada bit recibido y recupera los bits de datos.

- Buffer de almacenamiento: Circuito que recibe los bits de datos de la unidad de recepción y los hace disponibles para el resto del circuito.

El esquema básico del subsistema UART de recepción se aprecia en la Figura 4-9. A continuación se detallará el funcionamiento de cada uno de estos componentes.

[18] **UART (*Universal Asynchronous Receiver and Transmitter*):** Circuito capaz de enviar datos paralelos a través de una línea de comunicación serial. Generalmente usado en conjunto con el estándar EIA RS-232.



Fuente [1]

Figura 4-9. Subsistema UART de recepción.

4.3.1.1.1 Procedimiento de muestreo

Se asumirá una tasa de muestreo de 16 veces la velocidad de transmisión, es decir que cada bit que llega se lo muestrea 16 veces, pero se almacena su valor solamente tomando en consideración el valor que tiene dicho bit en su punto medio. Se asumirá también que se tiene N bits de datos y M bits de parada. El procedimiento de muestreo se describe en los siguientes pasos:

1. Esperar hasta que la señal de entrada sea ‘0’, es decir, cuando llega el bit de inicio, entonces se inicia el contador de muestras.

2. Cuando este contador llega a 7 quiere decir que se ha llegado al punto medio del bit de inicio. Se reinicia el contador.

3. Cuando el contador llega a 15 se ha avanzado hasta la mitad del primer bit de datos. Se debe tomar este valor y almacenarlo en un registro. Se reinicia el contador.

4. Se repite el paso 3 un total de N-1 veces más, de modo que se realice la lectura de todos los bits de datos.

5. En caso de que en los parámetros de transmisión se haya establecido el uso de un bit de paridad, es necesario repetir el paso 3 una vez más para obtener el bit de paridad correspondiente.

6. Finalmente, se repite el paso 3, M veces más a fin de obtener los bits de parada, pero ya no es necesario almacenarestos valores.

4.3.1.1.2 Generador de tasa de muestreo

Este módulo es básicamente un contador que genera la señal de muestreo, misma que tiene una frecuencia de 16 veces la velocidad de transmisión del UART.

Para el caso específico de este diseño, se ha escogido una velocidad de transmisión de 115200 bps para el UART, de modo que si se desea tomar 16 muestras de cada bit, la tasa de muestreo debe ser 16 veces este valor, es decir, 1843200 muestras por segundo. Dado que la frecuencia del oscilador presente en la tarjeta de desarrollo es de 50 MHz, se requiere un contador módulo 27=$\left(\frac{50*10^6}{1843200}\right)$, en el cual se activa la señal de muestreo una vez cada 27 ciclos de reloj.

En la Figura 4-10 se observa el diagrama correspondiente a la entidad VHDL de este componente, y en la Tabla 4-1 se tiene la descripción de cada puerto de entrada y salida de esta entidad.

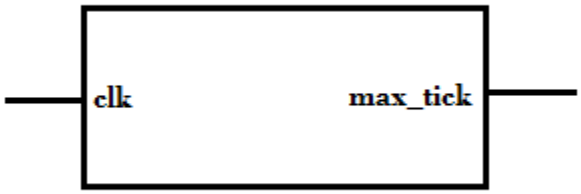


Figura 4-10. Entidad del generador de tasa de muestreo.

Tabla 4-1. Puertos de entrada y salida del generador de tasa de muestreo.

PUERTO	MODO	TAMAÑO (bits)	DESCRIPCIÓN
clk	In	1	Señal de reloj.
max_tick	Out	1	Señal de muestreo.

Para la descripción en VHDL de este módulo se ha hecho uso de genéricos[19] a fin de poder variar fácilmente el valor de la velocidad de transmisión del UART y, por ende, el valor hasta el cual cuenta este módulo. Por tratarse de un contador, su descripción VHDL es bastante sencilla y se observa a continuación:

```
--Lógica de estado siguiente
r_next <= (others => '0') when r_reg=(M-1) else r_reg + 1;

--Lógica de salida
max_tick <= '1' when r_reg=(M-1) else '0';
```

En las líneas de código superiores, el registro contador es *r_reg*, mientras queel parámetro M corresponde al valor hasta el cual se debe contar antes de activar la señal de muestreo (*max_tick*).

4.3.1.1.3 Unidad de recepción de datos

En base al procedimiento de muestreo explicado en la Sección 4.3.1.1.1, se tiene el funcionamiento del receptor, el cual es el encargado de realizar la lectura y almacenamiento de los bits recibidos.

Este módulo se lo ha descrito en VHDL en forma de una máquina de estados finita, FSM[20], cuyo diagrama ASM[21]se observa en la Figura 4-11. En el mismo se

[19] **Genérico:** Parámetro cuyo valor puede ser fácilmente modificado y adaptado a diferentes aplicaciones. Su propósito es proporcionar al código mayor flexibilidad y reusabilidad.

[20] **FSM (*Finite State Machine*):** Técnica especial de descripción de circuitos lógicos secuenciales, especialmente útil en sistemas en los cuales, sus funciones forman una secuencia bien definida.

puede observar que existen 4 estados bien definidos: *idle, start, data, stop.* Como sus nombres lo sugieren, el estado *idle* representa el momento cuando no hay ningún dato que recibir. Los estados de *start, data y stop*, en cambio, corresponden a los momentos cuando se está procesando los bits de inicio, datos y parada, respectivamente.

En el diagrama se observa que se dispone de 3 registros, *b, n* y *s*. El registro s sirve para contabilizar las señales de muestreo recibidas desde el generador de tasa de muestreo; este registro cuenta hasta 7 en el estado *start*, y hasta 15 en el resto de estados.

El registro*n*, por su parte, sirve para contabilizar el número de bits de datos recibidos, de modo que, cuando se reciben todos los bits de datos, el sistema puede pasar al estado *stop.*

Finalmente, el registro *b* es justamente el encargado de almacenar el valor de los bits de datos muestreados; el funcionamiento de este registro corresponde a un registro de desplazamiento hacia la derecha.

Se cuenta también con la bandera *rx_done_tick* que se activa cada vez que se finaliza el proceso de recepción.

En la Figura 4-12 se observa la entidad del receptor de datos, y en la Tabla 4-2 se tiene la descripción de los puertos de entrada y salida de la misma.

[21]**ASM (*Algorithmic State Machine*):** Diagrama en el que se representa de manera gráfica todos los parámetros de interés relacionados a una Máquina de Estados, esto es: entradas, salidas, estados y transiciones.

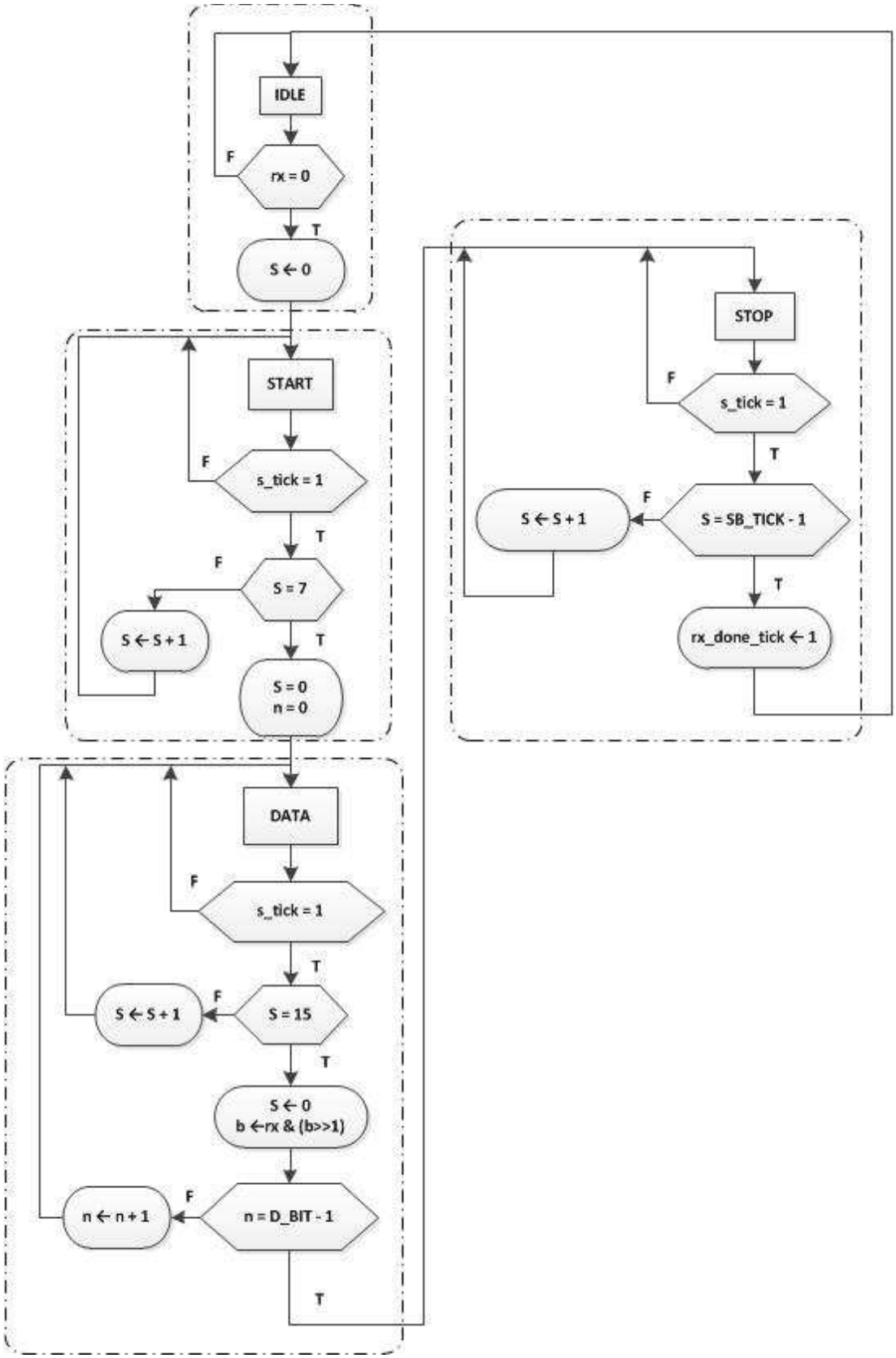


Figura 4-11. Diagrama ASM del subsistema de recepción.

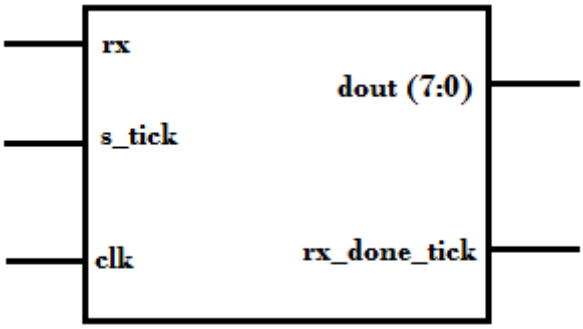


Figura 4-12. Entidad correspondiente al receptor de datos.

Tabla 4-2. Puertos de entrada y salida del receptor de datos.

PUERTO	MODO	TAMAÑO (bits)	DESCRIPCIÓN
clk	In	1	Señal de reloj.
s_tick	In	1	Señal de muestreo.
rx	In	1	Puerto de entrada de cada bit recibido.
dout	Out	8	Puerto de salida de los bits de datos recibidos.
rx_done_tick	Out	1	Indicadorde que se ha concluido la recepción de datos.

4.3.1.1.4 Buffer de almacenamiento

Constituye la interfaz entre el sistema de recepción de datos y el resto del circuito, su función es tomar el dato recibido y almacenarlo hasta que sea usado por el módulo que corresponda. Además, este módulo es necesario, ya que evita que exista sobre-escritura en el receptor. Una vez que se activa la señal *rx_done_tick* del receptor, este buffer inmediatamente toma el dato del receptor a fin de evitar que el mismo se pierda debido a la posible llegada de otro dato antes de que el actual sea leído y procesado.

Para el presente diseño, se ha escogido implementar un buffer con capacidad de almacenar una sola palabra, por considerarse esto suficiente para las necesidades del presente proyecto. En la Figura 4-13 se observa la entidad correspondiente a este módulo, mientras que en la Tabla 4-3 se muestran los puertos de entrada y salida de esta entidad.

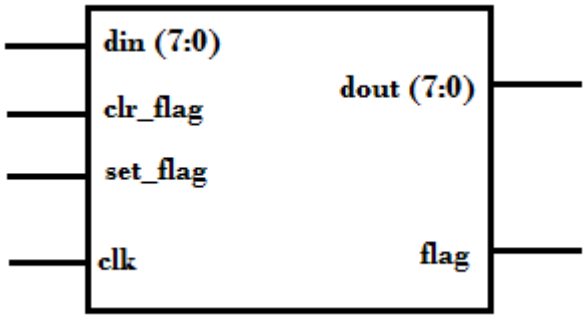


Figura 4-13. Entidad correspondiente al buffer de almacenamiento.

Tabla 4-3. Puertos de entrada y salida del buffer de almacenamiento.

PUERTO	MODO	TAMAÑO (bits)	DESCRIPCIÓN
clk	In	1	Señal de reloj.
din	In	8	Entrada de datos.
clr_flag	In	1	Bandera de habilitación de lectura.
set_flag	In	1	Bandera de habilitación de escritura.
dout	Out	8	Salida de datos.
flag	Out	1	Indicador de si existe dato almacenado o no.

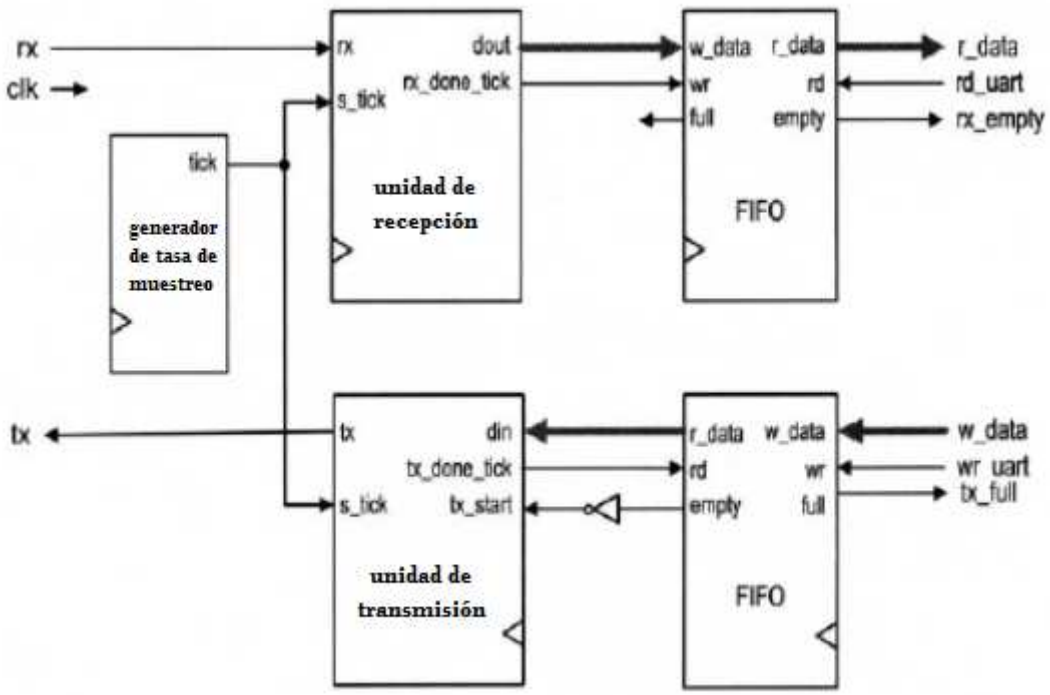
Considerando la Tabla 4-3, es necesario aclarar que el puerto *set_flag*del buffer es justamente al que se conectará el puerto de salida *rx_done_tick* del receptor, es decir que, al activarse esta señal, inmediatamente el dato de entrada es copiado en el buffer y se activa, a su vez, la señal *flag*, indicando que existe un nuevo dato disponible; mientras que al activarse la señal clr_flag, este módulo desactiva la señal *flag*. Este comportamiento se describe en las siguientes líneas de código VHDL:

```
--dato recibido
if (set_flag = '1') then
      buf_next <= din;
      flag_next <= '1';
--dato leido
elsif (clr_flag = '1') then
      flag_next <= '0';
end if;
```

4.3.1.2 Subsistema de Transmisión

Este subsistema tiene prácticamente la misma estructura que el de recepción, excepto que, obviamente, en lugar del receptor se tiene un transmisor. Unificando ambos subsistemas se obtiene el sistema UART completo, que se aprecia en la Figura 4-14.

Dado que los módulos de generador de tasa de muestreo y el buffer de almacenamiento ya se los explicó en las secciones anteriores, solamente se procederá a detallar el funcionamiento del transmisor.



Fuente [1]

Figura 4-14. Sistema UART completo.

4.3.1.2.1 Transmisor

Al igual que para el receptor, se ha hecho la descripción de este módulo mediante el uso de una máquina de estados, cuyo diagrama ASM se observa en la Figura 4-15.

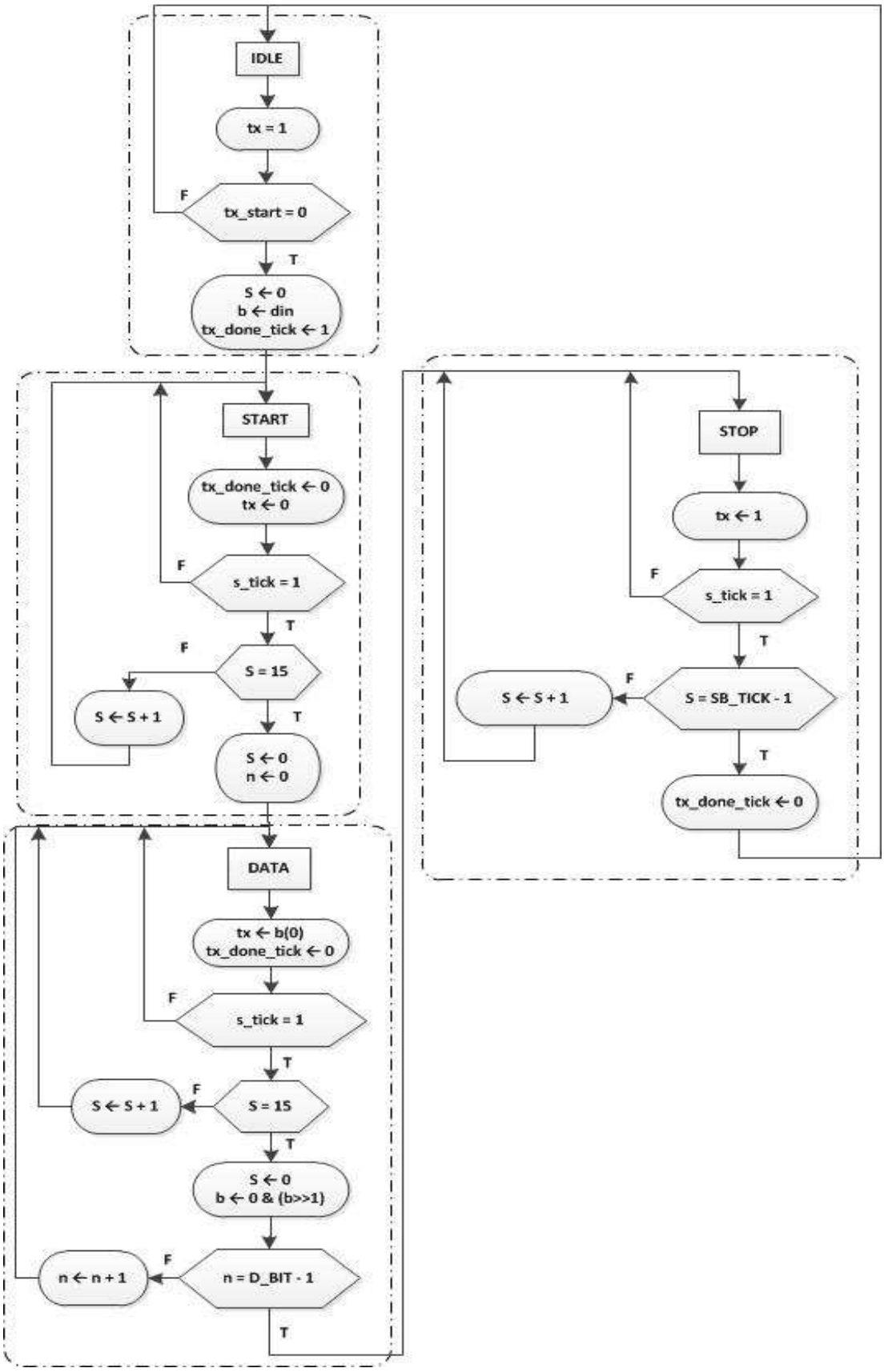


Figura 4-15. Diagrama ASM correspondiente al transmisor.

Como se puede apreciar en el diagrama, se cuenta con los mismos estados que para el receptor: *idle, start, data y stop.*

En el estado *idle:* la línea se mantiene inactiva (1L). Si se verifica que existe un dato para transmitir, se envía la señal de habilitación de lectura al buffer (*tx_done_tick*) y se almacena el dato a transmitir. El sistema pasa al estado *start.*

En el estado *start:* se envía el bit de inicio (0L) y se incrementa el valor del registro s (contador de muestras), hasta llegar a 15. El sistema pasa al estado *data.*

En el estado *data:* se envían uno por uno los bits de datos, a la velocidad determinada por el registro *s*, es decir, cada vez que se tienen 16 señales de muestreo provenientes del generador de tasa de muestreo, se transmite un bit. Una vez transmitidos todos los bits se datos se pasa al estado *stop.*

En el estado *stop:* se envían los bits de parada bajo las mismas consideraciones que para los otros bits. Finalmente, el sistema regresa al estado *idle.*

En la Figura 4-16 se observa la entidad correspondiente al transmisor y en la Tabla 4-4 se muestran los puertos de entrada y salida de esta entidad.

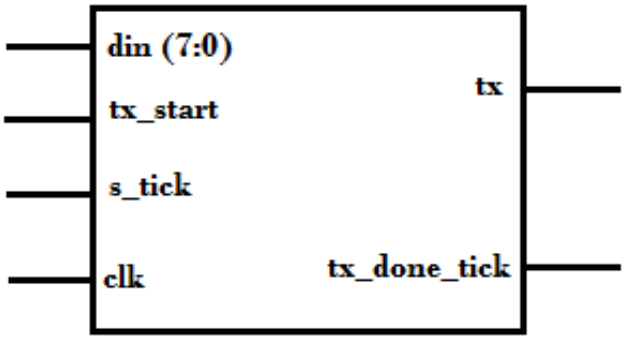


Figura 4-16. Entidad correspondiente al transmisor.

Tabla 4-4. Puertos de entrada y salida del transmisor.

PUERTO	MODO	TAMAÑO (bits)	DESCRIPCIÓN
clk	In	1	Señal de reloj.
din	In	8	Entrada de datos.
tx_start	in	1	Indica que existe un dato que transmitir.
s_tick	In	1	Señal de muestreo.
tx	Out	1	Salida de datos.
tx_done_tick	Out	1	Señal de habilitación de lectura para el buffer.

4.3.2 DEMUX 1X8

Este módulo es el encargado de repartir los paquetes recibidos hacia los 8 buffers existentes a la entrada de la red. En el presente diseño se realizará la transmisión de 16 paquetes para cada entrada de la red y, por lo tanto, la función del demux es entregar 16 paquetes hacia cada buffer en forma ordenada, para que una vez llenos los buffers de entrada, se empiece con el proceso de conmutación de los paquetes.

Cabe mencionar que es el módulo de control el que envía la señal de habilitación al demux cada vez que se dispone de un paquete para ser enviado al buffer correspondiente. Además, es este mismo módulo de control el cual recibe uno a uno los bytes desde el UART y realiza el proceso de ensamblarlos en paquetes de 2 bytes para su envío al demux y su posterior almacenamiento en los buffers de entrada.

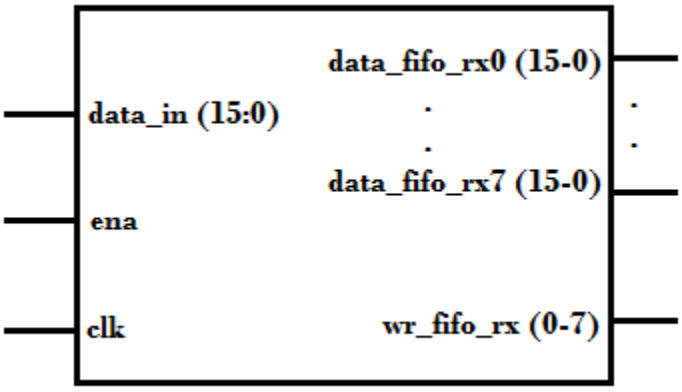


Figura 4-17. Entidad correspondiente al demux 1x8.

En la Figura 4-17 se observa la entidad correspondiente al demux 1x8 y en la Tabla 4-5 se encuentra la descripción de sus puertos de entrada y salida.

Tabla 4-5. Puertos de entrada y salida del demux 1x8.

PUERTO	MODO	TAMAÑO (bits)	DESCRIPCIÓN
clk	In	1	Señal de reloj.
data_in	In	16	Paquete de entrada desde módulo de control.
ena	In	1	Señal de habilitación desde módulo de control.
data_fifo_rx 0 –7	Out	16	Puertos de salida de datos hacia los buffers.
wr_fifo_rx	Out	8	Señales de habilitación de escritura para los buffers.

Para realizar la descripción en VHDL de este módulo lo que se hace es enviar el dato de entrada hacia todos los buffers, pero solamente activar la señal de habilitación de escritura en el buffer correspondiente. Además, se cuenta con dos registros, *b* y *w*. El registro w es el que cuenta el número de palabras que se escribe en cada buffer, mientras que el registro b indica el número del buffer en el cual se debe activar la señal de habilitación de escritura para que éste almacene los paquetes. La lógica de funcionamiento de estos registros y la activación de la bandera de escritura adecuada se observa en la siguiente sección de código:

```
--conteo de palabras escritas en buffer actual
if w_reg = 2**A-1 then
      w_next <= 0;
      b_next <= b_reg + 1;
else
      w_next <= w_reg + 1;
end if;
--activación de la bandera para habilitación de
-- escritura del buffer correspondiente
for i in 0 to 2**etapas-1 loop
      if i=b_reg then
            w_flags_next(i) <= '1';
      end if;
end loop;
```

Cabe aclarar que, la expresión 2**A-1 corresponde al número de paquetes que se debe enviar hacia cada uno de los buffers.

4.3.3 FIFOS DE ENTRADA

Existen 8 buffers, uno para cada entrada de la red de conmutación, los mismos fueron generados usando *IP Cores*[22], con las siguientes características:

- Capacidad de 16 palabras de 16 bits cada una.
- Generados usando bloques de RAM embebida.
- Reloj común para los puertos de lectura y escritura.
- Banderas de salida de *empty y full.*

En la Figura 4-18 se observa el diagrama correspondiente a la entidad de un buffer y en la Tabla 4-6 se tiene la descripción de cada puerto de entrada y salida.

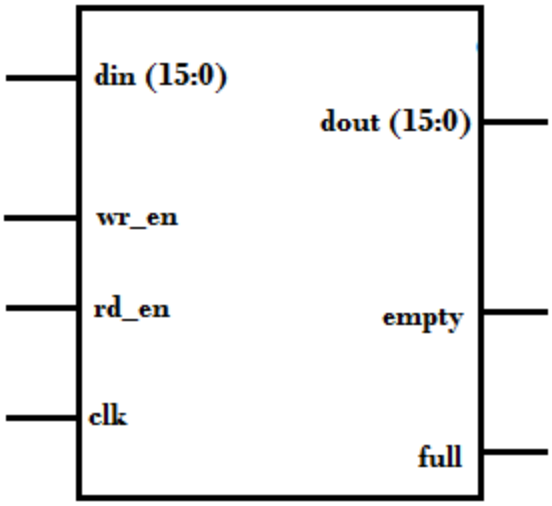


Figura 4-18. Entidad correspondiente a un FIFO de entrada.

[22] ***IP Core*:** Bloque de propiedad intelectual. Son bloques de lógica pre-compilados y reusables, que implementan funciones complejas. Su propósito es minimizar el tiempo de diseño y proveer flexibilidad y reusabilidad a un circuito.

Tabla 4-6. Puertos de entrada y salida de un FIFO de entrada.

PUERTO	MODO	TAMAÑO (bits)	DESCRIPCIÓN
clk	In	1	Señal de reloj.
din	In	16	Puerto para escritura de datos en el buffer.
wr_en	In	1	Señal de habilitación de escritura.
rd_en	In	1	Señal de habilitación de lectura.
dout	Out	16	Puerto para lectura de datos.
empty	Out	1	Indicador de que el buffer está vacío.
full	Out	1	Indicador de que el buffer está lleno.

Por tratarse de *IPCores*, para el uso de estos módulos solamente se los debe declarar como componentes del sistema y realizar sus interconexiones de manera normal.

4.3.4 RED DE CONMUTACIÓN

Este módulo ya fue descrito en la Sección 3.2.6, pero es necesario aclarar que el diseño anterior se lo hizo teniendo en mente, principalmente, la simulación de la red; por ello, se hace necesario realizar una revisión y adecuación del diseño a fin de poder implementarlo.

Para implementar la red se debe eliminar del diseño anterior el módulo para medición de retardo, como se explicó en la Sección 3.2.6.2. Además, se ha decidido también eliminar el conteo de paquetes perdidos en los buffers, esto debido principalmente a que al incluir estos contadores se degrada el desempeño de la red, en términos de frecuencia máxima de operación.

Cabe aclarar que, el hecho de eliminar estas piezas de código no implica que ya no se realizará la medición de los parámetros de interés en la red, sino que los cálculos requeridos para obtenerlos se los llevará a cabo en la interfaz gráfica diseñada en lugar de usar los recursos lógicos del FPGA.

También, para la implementación de la red se hace necesario añadir una señal llamada *end_red* que indique el momento en el cual en ninguna de las salidas de la red existen más paquetes válidos, de modo que se puede concluir que la red ha terminado de conmutar todos los datos de entrada y, por ende, es posible empezar el proceso de transmisión de los datos conmutados de regreso hacia la PC. El comportamiento de esta señal se lo determina en base a las banderas z de la última etapa de la red, mismas que indican si existe dato válido en los puertos de salida; las siguientes líneas de código muestran esta lógica de funcionamiento:

```
--Bandera end_red (indica fin de conmutación de datos)
process (z_flag(etapas*(2**etapas) to (etapas+1)*(2**etapas)-1))
      variable a: std_logic_vector (0 to 2**etapas-1);
      variable x: std_logic;
begin
      a := z_flag(etapas*(2**etapas) to (etapas+1)*(2**etapas)-1);
      x := a(0);
      for i in 1 to 2**etapas-1 loop
            --operación AND entre todas las banderas z
            x := x and a(i);
      end loop;
      end_red <= x;
end process;
```

Otro punto a mencionar del comportamiento de la red, es que las señales de habilitación de escritura para los buffers de salida corresponden a los valores inversos de las banderas z de la última etapa de conmutadores de la red, como se observa a continuación:

```
--Banderas de wr para fifos_tx
wr_tx <= not (z_flag(etapas*(2**etapas) to (etapas+1)*(2**etapas)-1));
```

Una aclaración final es que, en la primera etapa de conmutadores de la red, la señal *z* del controlador de entrada, que indica si existe o no un paquete válido a la entrada del elemento de conmutación, depende tanto de la señal de habilitación para la red como de la señal de *empty* del FIFO de entrada correspondiente, de este modo, al vaciarse estos FIFOs de entrada, los elementos de conmutación de

la primera etapa de la red ya no realizan ninguna operación de lectura de datos desde los FIFOs pero el resto de procesos al interior de la red continúan ejecutándose hasta que finalmente se termina de conmutar todos los paquetes de datos. Esto se observa en las líneas de código a continuación:

```
--RENAME BANDERAS DE EMPTY_RX
rst<= (others=>'1') when ena_red='0' else (others=>'0');
z_flag(0 to 2**etapas-1) <= empty_rx or rst;
```

Con estas consideraciones, se obtiene el diagrama de la entidad correspondiente a la red de conmutación que se observa en la Figura 4-19, y la Tabla 4-7, que describe cada puerto de entrada y salida de la red.

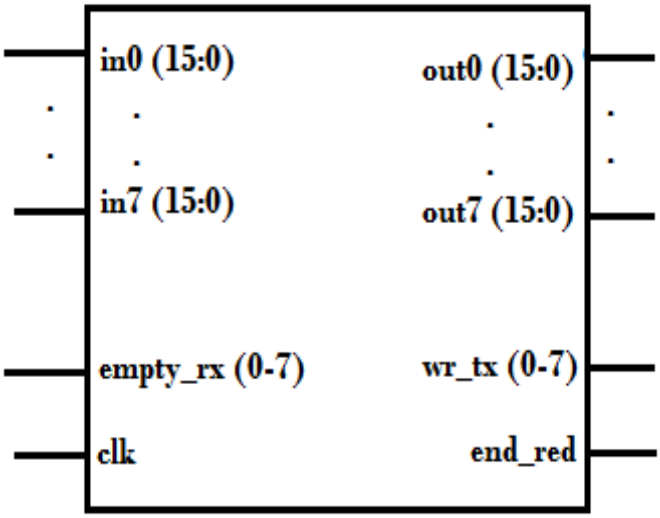


Figura 4-19. Entidad correspondiente a la red de conmutación a implementar.

Tabla 4-7. Puertos de entrada y salida de la red de conmutación.

PUERTO	MODO	TAMAÑO (bits)	DESCRIPCIÓN
clk	In	1	Señal de reloj.
in0 – in7	In	16	Puertos de entrada a la red de conmutación.
empty_rx	In	8	Banderas de *empty* de los FIFOs de entrada.
out0 – out7	Out	16	Puertos de salida de la red de conmutación.
wr_tx	Out	8	Señales de habilitación de escritura para los buffers.
end_red	Out	1	Indicador de fin del proceso de conmutación.

4.3.5 FIFOS DE SALIDA

Aquí también existen 8 buffers, como es de esperarse, uno para cada salida de la red de conmutación, los mismos fueron generados usando *IP Cores*, con las siguientes características:

- Capacidad de 32 palabras de 16 bits cada una.
- Generados usando bloques de RAM embebida.
- Reloj común para los puertos de lectura y escritura.
- Modo de operación FWFT[23].
- Banderas de salida de *empty y full.*
- Salida *data_count* que indica el número de palabras presentes en el buffer.

Hay que aclarar que para estos buffers se escogió la capacidad de 32 palabras debido a que, al generarse los datos de forma aleatoria, puede suceder que haya más de 16 paquetes destinados hacia una misma salida. De modo que, con esta capacidad delos buffers, se evita pérdida de paquetes ya conmutados por la red.

En la Figura 4-20 se observa el diagrama correspondiente a la entidad de un buffer de salida y en la Tabla 4-8 se tiene la descripción de cada puerto de entrada y salida.

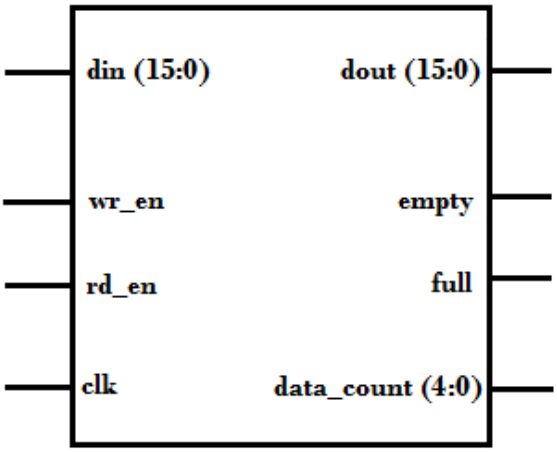


Figura 4-20. Entidad correspondiente a un FIFO de salida.

[23]**FWFT (*First Word Fall Through*):** Modo de operación en el cual el paquete que encabeza la cola del buffer se encuentra siempre disponible para su lectura, y la señal de habilitación de lectura actúa más bien como un indicador de transmisión y de que se debe remover ese paquete del buffer.

Tabla 4-8. Puertos de entrada y salida de un fifo_out.

PUERTO	MODO	TAMAÑO (bits)	DESCRIPCIÓN
clk	In	1	Señal de reloj.
din	In	16	Puerto para escritura de datos en el buffer.
wr_en	In	1	Señal de habilitación de escritura.
rd_en	In	1	Señal de habilitación de lectura.
dout	Out	16	Puerto para lectura de datos.
empty	Out	1	Indicador de que el buffer está vacío.
full	Out	1	Indicador de que el buffer está lleno.
data_count	Out	5	Indica el número de paquetes presentes en el buffer.

4.3.6 MUX 8X1

Este módulo es el encargado de tomar los datos provenientes de los 8 FIFOs de salida en forma secuencial y transmitir cada paquete hacia el módulo de control, para que éste a su vez lo descomponga en bytes individuales y lo envíe al módulo UART para su transmisión serial hacia la PC.En la Figura 4-21 se observa la entidad correspondiente a este mux y en la Tabla 4-9 se tiene la descripción de cada uno de los puertos de entrada y salida.

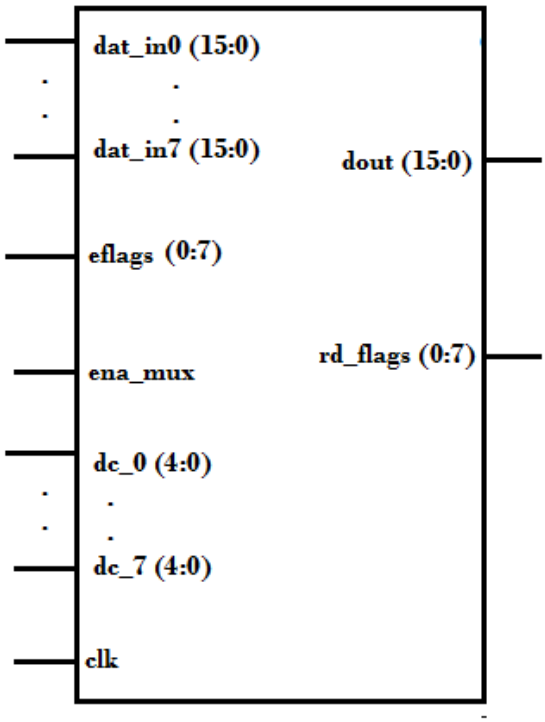


Figura 4-21. Entidad correspondiente al mux 8x1.

Tabla 4-9. Puertos de entrada y salida del mux 8x1.

PUERTO	MODO	TAMAÑO (bits)	DESCRIPCIÓN
clk	In	1	Señal de reloj.
dat_in0 –dat_ in7	In	16	Puertos de entrada de datos desde los FIFOs.
eflags	In	8	Banderas de *empty* de los FIFOs.
ena_mux	In	1	Señal de habilitación para el mux.
dc_0 – dc_7	In	5	Número de paquetes presentes en cada FIFO.
dout	Out	16	Puerto para salida de datos.
rd_flags	Out	8	Banderas de habilitación de lectura para los FIFOs.

El funcionamiento de este módulo es el siguiente:

Este módulo permanece inactivo hasta cuando la señal de habilitación *ena_mux* se pone en 1L, esta señal indica que el módulo UART está listo para recibir un nuevo dato para ser transmitido a la PC.

Al recibir esta señal, el mux empieza a tomar sucesivamente los valores de los *data_count* provenientes de cada buffer, es decir, lee y envía el valor del *data_count_0* hacia el módulo de control, el cual a su vez lo envía hacia el UART para su transmisión. Luego de recibir nuevamente la señal de habilitación, el mux envía el *data_count_1* y así, sucesivamente, hasta completar los 8 buffers. Para ello se cuenta con un registro (*dc_reg*) que indica el número de valores enviados, esto se observa en la siguiente sección de código:

```
--Lógica de salida de los data count
for i in 0 to 2**etapas-1 loop
      if i=dc_reg then
            --se completa el byte con bits en cero
            dout_dc <= zero & d_count(i);
      end if;
end loop;
```

Una vez enviados todos los valores de los *data_count* de los buffers, ahora se procede con el envío de los datos almacenados en cada buffer. Al recibir la señal de habilitación, el mux toma el primer valor proveniente del buffer 0 y enciende la bandera de habilitación de lectura para ese buffer, envía el dato hacia el módulo de control, y vuelve a quedar inactivo hasta que se encienda nuevamente la señal de habilitación indicando que se puede enviar un nuevo dato.

El proceso de lectura y envío de los datos se repite hasta que la bandera de *empty* del buffer 0 se encienda; en ese caso, se procede a tomar los datos provenientes del buffer 1. De esta forma se continúa hasta finalizar el envío de todos los datos presentes en los FIFOs de salida de la red. La descripción en VHDL de este funcionamiento se la muestra a continuación:

```
--Estado de lectura del dato proveniente del fifo
when rd_fifos =>
      for i in 0 to 2**etapas-1 loop
            if i=b_reg then
                  --lectura del dato proveniente del fifo
                  d_next <= data_in(i);
                  -- habilitación de lectura para el fifo correspondiente
                  rd_flags(i) <= '1';
                  --comprobación si se ha vaciado el buffer actual
                  if eflags(i) ='1' then
                        b_next <= b_reg + 1;
                  end if;
            end if;
      end loop;
```

4.3.7 MÓDULO DE CONTROL

Se trata de un módulo esencial para el funcionamiento de todo el sistema, ya que es el encargado de enviar y recibir señales de control desde y hacia el resto de módulos creados. Las principales señales de control necesarias son las que habilitan cada módulo del sistema en el momento adecuado. Este módulo también intercambia señales de datos con algunos de los módulos creados.

En la Figura 4-22 se tiene la entidad correspondiente al módulo de control, y en la Tabla 4-10 se tiene la descripción de cada uno de los puertos de entrada y salida de esta entidad.

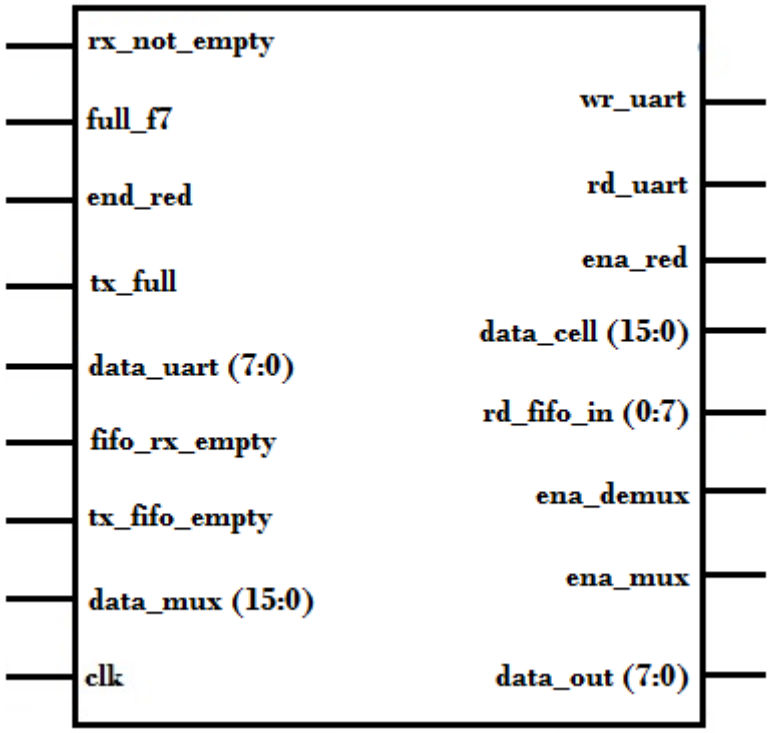


Figura 4-22. Entidad correspondiente al módulo de control.

Tabla 4-10. Puertos de entrada y salida del módulo de control.

PUERTO	MODO	TAMAÑO (bits)	DESCRIPCIÓN
clk	In	1	Señal de reloj.
rx_not_empty	In	1	Indicador de que el UART ha recibido un dato nuevo.
full_f7	In	1	Indicador de que se han llenado los FIFOs de entrada.
end_red	In	1	Indicador de que se ha finalizado la conmutación de paquetes.
tx_full	In	1	Indicador de que el UART está ocupado transmitiendo un dato.
data_uart	In	8	Dato nuevo recibido por el UART.
fifo_rx_empty	In	1	Indicador de que los FIFOs de entrada están vacíos.
tx_fifo_empty	In	1	Indicador de que los FIFOs de salida están vacíos.
data_mux	In	16	Paquete recibido desde el mux.

wr_uart	Out	1	Habilitación de escritura al sistema UART para transmisión de datos.
rd_uart	Out	1	Habilitación de lectura del sistema UART.
ena_red	Out	1	Señal de habilitación para la red de conmutación.
data_cell	Out	16	Paquete enviado hacia el demux.
rf0 – rf7	Out	1	Señales de habilitación de lectura para FIFOs de entrada.
ena_demux	Out	1	Señal de habilitación para el demux.
ena_mux	Out	1	Señal de habilitación para el mux.
data_out	Out	8	Dato a transmitir por el sistema UART.

Este módulo de control se lo ha descrito mediante una máquina de estados finita pero no se ha incluido el diagrama correspondiente debido a su gran complejidad; en lugar de ello, a continuación se explica se funcionamiento simplificado y se muestran brevemente las secciones de código correspondientes a las tareas mencionadas:

1. Inicialmente, el sistema se encuentra en estado inactivo (*zero*); en caso de recibirse un dato desde el UART, se lee el dato y, según el carácter que sea se decide que acción tomar, es decir, a qué estado pasa el sistema.

```
--Estado de lectura del dato recibido por el sistema UART
when rd_data =>
      rd_uart <= '1';
      --Identificación de acciones a realizar
      case data_uart is
            when "01010100" => --T
                  --comprobación de conectividad
                  next_state <= test;
            when "01001010" => --J
                  --recepción de datos
                  next_state <= wait_data;
            when "01010010" => --R
                  --transmisión de datos
                  next_state <= ena_mux;
            whenothers =>
                  next_state <= zero;
      end case;
```

2. En caso de que el dato indique que se debe probar el puerto, el sistema va al estado *testy* lo que se hace es enviar el carácter correspondiente hacia el UART para su transmisión a la PC y volver al estado inicial.

```
--Estado de prueba de conexión
when test =>
      wr_uart <= '1';
      --Se envia el dato "T" + 1
      data_out <= "01010101"; --'U'
      next_state <= zero;
```

3. Si el carácter recibido indica que se va a iniciar la transmisión de datos válidos desde la PC, el sistema va al estado *wait_data* y se inicia el proceso de recepción de los datos de la manera como se explica en el siguiente paso.

4. El módulo UART recibe un dato (*data_uart*) y activa la señal *rx_not_empty*para que el módulo de control almacene el dato; se recibe otro dato y el módulo de control los concatena, activa la señal *ena_demux* y envía el paquete (*data_cell*) hacia el demux.

```
--Estado de lectura de los paquetes a la salida del UART
when read_data =>
      rd_uart <= '1';
      --almacenamiento del byte recibido
      d_next <= data_uart & d_reg(size-1 downto 8);
      Bcount_next <= Bcount_reg + 1;
      --conteo de bytes del paquete
      ifBcount_reg = n_bytes-1 then
            next_state <= store_data;
      else
            next_state <= wait_data;
      end if;

--Estado para habilitación y envío de paquetes al demux
when store_data =>
      data_cell <= d_reg;
      ena_demux <= '1';
```

5. El demux, al activarse la señal *ena_demux*, lee el paquete recibido del módulo de control y lo envía al buffer correspondiente (16 paquetes a cada buffer).

6. Este proceso de recepción (pasos 4 y 5) se repite hasta terminar de recibir todos los datos desde la PC.

7. Cuando todos los buffers están llenos, se activa la señal *full_f7* y el módulo de control activa la señal *ena_red* para iniciar el proceso de conmutación. Se activan también las banderas de habilitación de lectura para los FIFOs de entrada (*rf0 – rf7*).

```
--Se verifica si se han llenado los buffers de rx
if full_f7 = '1' then
      next_state <= wait_red_end;
      --activación de banderas de lecturas de fifos_rx
      rf_next <= (others => '1');
```

8. La red de conmutación empieza a leer los datos de los FIFOs de entrada en cada ranura de tiempo, hasta terminar con los 16 paquetes. Cuando los FIFOs de entrada están vacíos, cada controlador de entrada de los elementos de conmutación de la primera etapa de la red ya no procesa más paquetes. Pero, el resto de elementos de la red siguen funcionando hasta que todos los paquetes sean conmutados.

9. Una vez que ya no existen paquetes válidos en ninguna de las salidas de la red, y los FIFOs de recepción están vacíos (*fifo_rx_empty*) se activa la señal *end_red*.

```
--Estado para habilitar la red de conmutación
when wait_red_end =>
      ena_red <= '1';
      --Se verifica si ha finalizado la conmutación de los paquetes
      if (end_red='1' and fifo_rx_empty='1') then
            next_state <= wait_tx_ready;
            rf_next <= (others => '0');
      else
            next_state <= wait_red_end;
      end if;
```

10. En este punto, el módulo de control llega al estado *wait_tx_ready*, es decir, espera hasta que la señal *tx_full*proveniente del UART esté desactivada para poder iniciar el proceso de transmisión.

11. Una vez que la señal *tx_full* está desactivadase debe empezar a transmitir los *data_count* de los FIFOs de salida, para ello se activa la señal*ena_mux*.

12. El mux, al recibir la señal de habilitación, lo primero que transmite es el valor de los*data_count* correspondiente a cada uno de los FIFO s de salida.

```
--Estado de habilitación y lectura del mux_dc
when ena_dc =>
      ena_dc_mux <= '1';
      d_next <= data_dc_mux;
      next_state <= tx_data_count;
```

13. El módulo de control recibe el paquete correspondiente al *data_count* y envía este valor al UART para su transmisión.

```
--Estado de tx de los data_count enviados por el mux
when tx_data_count =>
      wr_uart <= '1';
      data_out <= d_reg(7 downto 0);
      next_state <= wait_tx_ready;
```

14. Se repite este proceso (pasos 11-13) hasta enviar todos los valores correspondientes a los *data_count* de los buffers de salida.

15. Ahora, se deben enviar los paquetes de datos almacenados en los FIFOs de salida. Para ello, se espera el fin de la transmisión actual y se envía la señal *ena_mux* y el mux, a su vez, envía el primer paquete de datos hacia el módulo de control.

```
--Estado de lectura de datos enviados por el mux
when rd_mux =>
      d_next <= data_mux;
      next_state <= tx_msb;
```

16. El módulo de control recibe el paquete y procede a enviar al UART el byte más significativo del mismo para su transmisión; una vez que se finaliza esta transmisión, se envía el byte menos significativo de ese paquete.

```
--Estado de tx de los msb de los paquetes enviados por el mux
when tx_msb =>
      wr_uart <= '1';
      data_out <= d_reg(size-1 downto size-8);
      Bcount_next <= 1 ;
      next_state <= wait_tx_ready;

--Estado de tx de los lsb del paquete
when tx_lsb =>
      wr_uart <= '1';
      data_out <=
      d_reg(size-(8*Bcount_reg)-1 downto size-(8*Bcount_reg)-8);

      --conteo de bytes transmitidos del paquete
      if Bcount_reg = n_bytes-1 then
            Bcount_next <= 0;
      else
            Bcount_next <= Bcount_reg + 1;
      end if;
      next_state <= wait_tx_ready;
```

17. Se repite el proceso (pasos 15-16) hasta terminar con todos los paquetes de datos, es decir cuando señal *tx_fifo_empty* se activa.

```
      --fin de tx de paquetes conmutados
      if tx_fifo_empty='1' then
            next_state <= zero;
```

18. Al terminar la transmisión de todos los datos, el sistema vuelve a estar inactivo.

4.4 ENTIDAD VHDL DE MÁS ALTO NIVEL

Finalmente, en esta sección se muestra el sistema completo que se obtiene al realizar la interconexión de todos los módulos creados y expuestos anteriormente, esta entidad de más alta jerarquía se observa en la Figura 4-23, mientras que en la Tabla 4-11 se describen sus puertos de entrada y salida.

Cabe mencionar que todos los puertos de salida de los leds no son más que indicadores de avance del proceso de conmutación, mas no son puertos indispensables para el funcionamiento del circuito, por lo cual se los podría omitir.

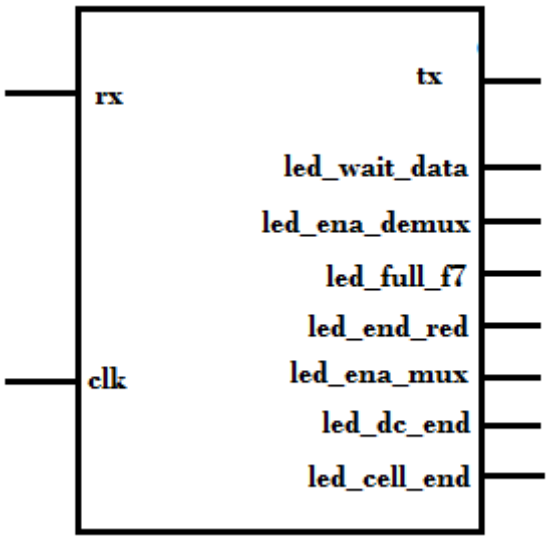


Figura 4-23. Entidad de más alto nivel.

Tabla 4-11. Puertos de entrada y salida de la entidad de más alto nivel.

PUERTO	MODO	TAMAÑO (bits)	DESCRIPCIÓN
clk	In	1	Señal de reloj.
rx	In	1	Entrada de datos seriales.
tx	Out	1	Salida de datos seriales.
led_wait_data	Out	1	Indicador de que empieza la recepción de datos.
led_ena_demux	Out	1	Indicador de que se ha enviado señal de habilitación al demux.
led_full_f7	Out	1	Indicador de que se han llenado los FIFOs de entrada.
led_end_red	Out	1	Indicador de que se ha terminado de conmutar los paquetes.
led_ena_mux	Out	1	Indicador de que se ha enviado señal de habilitación al mux.
led_dc_end	Out	1	Indicador de que se ha terminado de transmitir los *data_count.*
led_cell_end	Out	1	Indicador de fin de la transmisión de datos.

4.5 PROCESO DE IMPLEMENTACIÓN DEL SISTEMA

Una vez que se ha finalizado la descripción en VHDL de todos los módulos que conforman el sistema, para poder verificar el adecuado funcionamiento del circuito se lo debe implementar físicamente en el FPGAy para ello, el software de desarrollo *ISE Design Suite*ejecuta los siguientes pasos:

1. Síntesis:consiste en convertir la descripción VHDL del sistema de alto nivel de abstracción hacia un bajo nivel, esto es, a nivel de compuertas lógicas genéricas. Este proceso es independiente de la tecnología y vendedor escogido[2]. En este paso también, la herramienta de software realiza el proceso de optimización del diseño, de ser posible.

2. Traducción:es el proceso en el cual se realiza un chequeo previo del diseño sintetizado para determinar si existe alguna inconsistencia con la arquitectura del FPGA escogido, principalmente en lo que se refiere a pines y frecuencia del reloj, tomando en cuenta para ello, el archivo de restricciones de usuario (UCF[24]) [3].

3. Mapeo:realiza la distribucióndel diseño sintetizado en los recursos reales que tiene el FPGA a usar. Si el diseño es demasiado grande y los recursos del FPGA escogido no son suficientes, esta etapa no se podrá completar con éxito.

4. Localización y enrutamiento: consiste en ubicar cada uno de los bloques lógicos usados en la posición más conveniente a fin de minimizar los retardos por enrutamiento. Al finalizarse estos procedimientos, se tiene el archivo especificando los recursos lógicos utilizados, su ubicación y sus interconexiones.

[24]**UCF (*User Constraints File*):** archivo que contiene, principalmente, indicaciones de ubicación de los puertos de entrada y salida de un circuito, así como también especificaciones de frecuencia de operación del sistema.

5. Generación del archivo de configuración: convierte el archivo resultante del proceso anterior en un flujo de bits (*bitstream*), necesario para poder realizar la programación del FPGA.

6. Programación del FPGA:descarga el archivo de programación en la memoria escogida del FPGA. En caso que la memoria escogida sea de tipo PROM, se debe generar el archivo adecuado para este tipo de memoria. Este paso lo realiza la herramienta *Impact*, incluida dentro del software ISE.

En las siguientes secciones se muestra y explica el archivo de restricciones creado para el presente proyecto, así como también los principales resultados obtenidos de los procesos de síntesis e implementación del circuito diseñado.

4.5.1 ARCHIVO DE RESTRICCIONES DE USUARIO

Una restricción es una indicación que da el usuario a la herramienta de síntesis e implementación a fin de que los procesos se los realice cumpliendo con ciertas condiciones específicas [1]. Entre las principales restricciones que se usan en los diseños de circuitos digitales son las relacionadas a la ubicación de los pines de entrada y salida del circuito, así como la frecuencia de operación de las diferentes secciones del circuito diseñado.Para el caso del diseño de la red de conmutación, el archivo de restricciones creado se lo observa en la Figura 4-24.

```
NET "clk" LOC = "E12"| IOSTANDARD = LVCMOS33 | TNM_NET = "clk" ;
TIMESPEC "TS_clk" = PERIOD "clk" 20.0ns HIGH 50%;

NET "rx" LOC = "E16" | IOSTANDARD = LVCMOS33 ;
NET "tx" LOC = "F15" | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = SLOW ;

NET "led_test" LOC = "W21" | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = QUIETIO ;
NET "led_wait" LOC = "Y22" | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = QUIETIO ;
NET "led_ena_demux" LOC = "V20" | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = QUIETIO ;
NET "led_full_f7" LOC = "V19" | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = QUIETIO ;
NET "led_end_red" LOC = "U19" | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = QUIETIO ;
NET "led_ena_mux" LOC = "U20" | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = QUIETIO ;
NET "led_dc_end" LOC = "T19" | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = QUIETIO ;
NET "led_cell_end" LOC = "R20" | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = QUIETIO ;
```

Fuente [4]

Figura 4-24. Archivo de restricciones de usuario para el sistema diseñado.

A continuación se explica brevemente el significado de los términos que se observa en este archivo de restricciones:

- **NET:**hace referencia a los puertos de entrada y salida de la entidad de más alto nivel del diseño.
- **LOC:** indica una restricción de ubicación (*location*) para el puerto de entrada o salida en cuestión, es decir, especifica el pin del FPGA al cual va conectado talentrada o salida.
- **IOSTANDARD:**especifica el estándar eléctrico que utiliza el pin del FPGA.
- **TNM_NET:** restricción utilizada para identificar todos los elementos lógicos que conforman un grupo, al cual se le puede posteriormente aplicar una restricción de tiempo.
- **TIMESPEC:**es un marcador de posición que indica que se va a realizar una especificación de tiempo para algún grupo de elementos del diseño.
- **PERIOD:** establece la frecuencia de la señal de reloj. Este valor se lo puede especificar en términos de frecuencia o de periodo.
- **DRIVE:** establece la corriente que maneja el buffer de entrada o salida del pin del FPGA. Este valor está expresado en mA.
- **SLEW:** define la velocidad de transición del voltaje de las señales de salida del circuito.

4.5.2 RECURSOS UTILIZADOS

En la Figura 4-25 se observa el resumen correspondiente a los recursos lógicos y de memoria utilizados del FPGA para un diseño de la red de conmutación Banyan con una longitud de los buffers internos igual a dos posiciones (B=2), mientras que en la Figura 4-26 se observan estos mismos resultados para un tamaño de los buffers internos B=4.

Device Utilization Summary			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Flip Flops	3,363	11,776	28%
Number of 4 input LUTs	5,091	11,776	43%
Number of occupied Slices	2,753	5,888	46%
Number of Slices containing only related logic	2,753	2,753	100%
Number of Slices containing unrelated logic	0	2,753	0%
Total Number of 4 input LUTs	5,092	11,776	43%
Number used as logic	5,091		
Number used as a route-thru	1		
Number of bonded IOBs	11	372	2%
Number of BUFGMUXs	1	24	4%
Number of RAMB16BWEs	16	20	80%
Average Fanout of Non-Clock Nets	3.53		

Fuente [4]

Figura 4-25. Recursos utilizados en el FPGA para B=2.

Device Utilization Summary			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Flip Flops	4,899	11,776	41%
Number of 4 input LUTs	8,115	11,776	68%
Number of occupied Slices	4,321	5,888	73%
Number of Slices containing only related logic	4,321	4,321	100%
Number of Slices containing unrelated logic	0	4,321	0%
Total Number of 4 input LUTs	8,116	11,776	68%
Number used as logic	8,115		
Number used as a route-thru	1		
Number of bonded IOBs	11	372	2%
Number of BUFGMUXs	1	24	4%
Number of RAMB16BWEs	16	20	80%
Average Fanout of Non-Clock Nets	3.95		

Fuente [4]

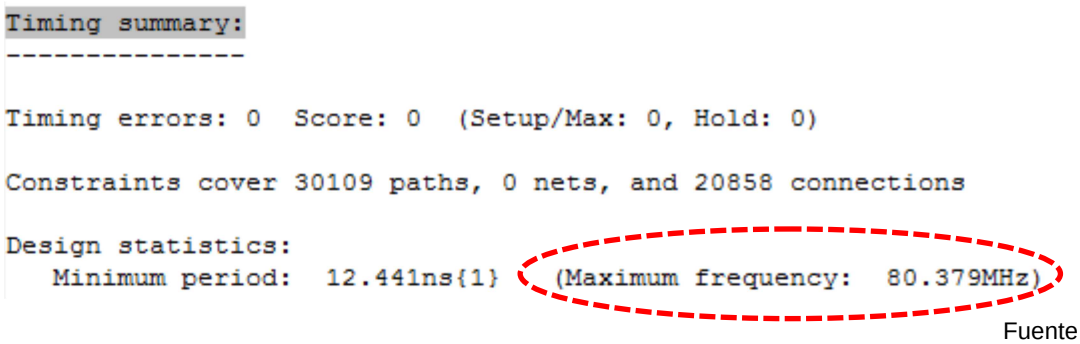
Figura 4-26. Recursos utilizados en el FPGA para B=4.

Como se puede observar, en ambos casos, se han utilizado 16 de los 20 bloques de memoria RAM, esto debido a que existen 8 FIFOs a la entrada de la red y 8 a la salida de la misma.

4.5.3 FRECUENCIA MÁXIMA

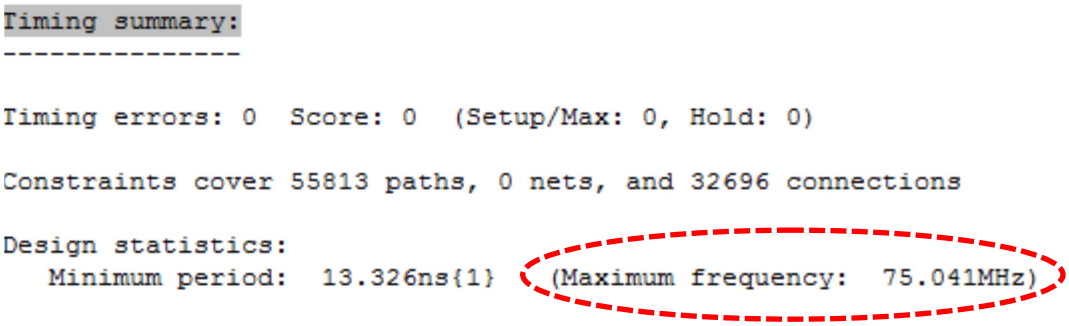
Una vez que se han llevado a cabo los procesos de síntesis e implementación del sistema diseñado, la herramienta de software ISE reporta el valor de la máxima frecuencia real a la que puede operar el circuito. Este valor lo estima tomando en consideración el valor máximo de retardo (debido tanto a lógica como a enrutamiento) que exista en el circuito entre dos elementos sincrónicos consecutivos, lo cual determina el valor mínimo del período requerido para el adecuado funcionamiento del sistema.

Para el caso del sistema diseñado, el resultado de la máxima frecuencia a la que puede operar el circuito,se observa en la Figura 4-27 para un valor de B=2, y en la Figura 4-28 para el valor de B=4.



Fuente [4]

Figura 4-26. Frecuencia máxima de operación del circuito con B=2.



Fuente [4]

Figura 4-27. Frecuencia máxima de operación del circuito con B=4.

Cabe mencionar que el módulo de desarrollo utilizado cuenta con 2 osciladores de 50 MHz y 133MHz; para el diseño realizado se ha utilizado el oscilador de 50

MHz, con lo cual no existe ningún inconveniente en la implementación del circuito ya que esta frecuencia de reloj es menor a la máxima frecuencia posible de operación del circuito.

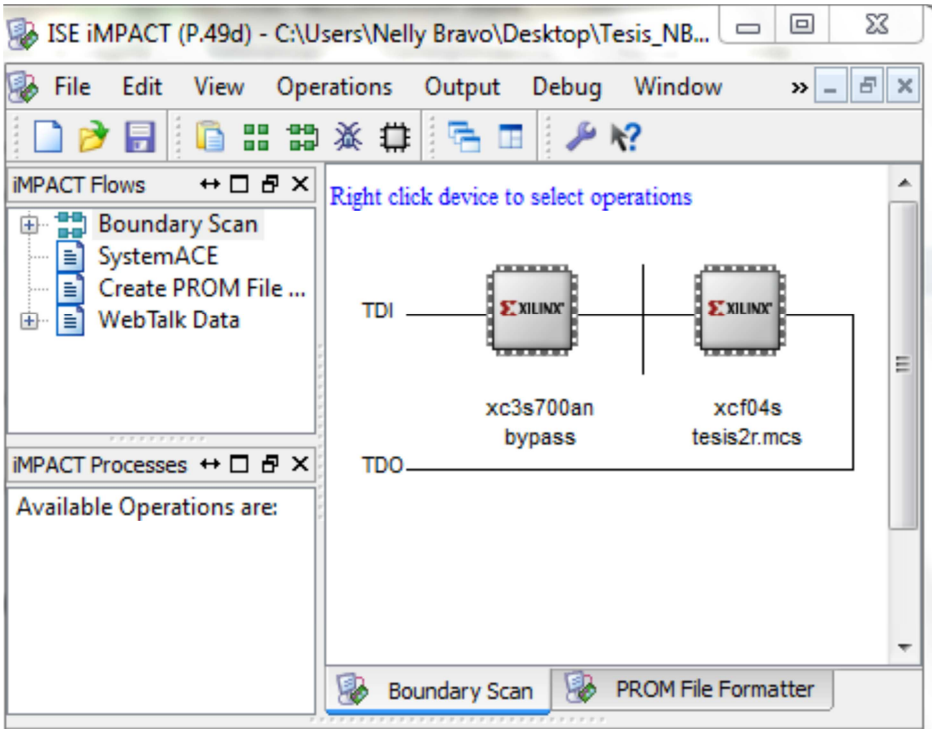
Revisando de forma más detallada el reporte generado luego del proceso de ubicación y enrutamiento, se puede ver que los mayores retardos son generados debido al enrutamiento más que a los niveles de lógica utilizados, esto se debe principalmente al hecho de que para implementar los buffers de entrada y salida de la red se están usando los bloques de memoria RAM embebidos en la arquitectura del FPGA, y por ende su ubicación es fija, lo cual puede conllevar un incremento en los retardos por enrutamiento.

4.5.4 PROGRAMACIÓN DEL FPGA

Este es el paso final para poder implementar físicamente el circuito diseñado, para este fin, el software *ISE Design Suite* cuenta con la herramienta *Impact*, misma que constituye la interfaz gráfica a través de la cual se realiza la conexión con el FPGA y se descarga el archivo de configuración.

En el caso del presente proyecto se ha decidido usar la memoria Flash PROM disponible en el módulo de desarrollo para en ella almacenar el archivo de configuración del FPGA. Por tratarse de una memoria no volátil, se tiene la ventaja de que la configuración del FPGA no se pierde al desconectar la energía del circuito, sino que cada vez que se encienda el circuito, éste se inicializará con la configuración almacenada en la memoria PROM.

En la Figura 4-29 se observa la interfaz gráfica de la herramienta *Impact* una vez que se ha establecido la conexión con el módulo de desarrollo que contiene el FPGA a programar.



Fuente [4]

Figura 4-29. Interfaz gráfica de la herramienta de programación *Impact*.

En la figura se observa que la herramienta *Impact* reconoce tanto el FPGA (xc3s700an) como la memoria PROM (xcf04s) incluida en el módulo de desarrollo. Debido a que la programación se la realizará directamente en la PROM no es necesario descargar el archivo de configuración también en el FPGA, por ello se ha escogido la opción *bypass* para el FPGA y solamente se programa la mencionada memoria PROM.

4.6 PRUEBAS DE FUNCIONAMIENTO

Finalmente, una vez que se ha realizado la programación del FPGA, se procede a realizar las respectivas pruebas de funcionamiento de la red de conmutación. Para ello, se ejecuta el programa con la interfaz gráfica elaborada en *Matlab* y se procede a realizar la generación y transmisión de los datos para luego recibir los datos conmutados y compararlos con los datos originales a fin de determinar el valor de los parámetros de interés, esto es, el número de paquetes perdidos y el retardo promedio de los paquetes al atravesar los buffers de la red de

conmutación. En la Figura 4-30 se observa el equipo usado para la realización de las pruebas de funcionamiento del circuito.

Figura 4-30. Equipo utilizado para realizar las pruebas de funcionamiento.

Anteriormente ya se mencionó que, en un escenario inicial de pruebas se utilizará un bloque de datos que consiste en 16 paquetes para cada entrada de la red, siendo cada paquete de tamaño 2 bytes. Además, se realizarán las pruebas para un tamaño de los buffers internos de B=2. Luego se incrementará este parámetro a un valor de B=4 y se volverá a realizar la transmisión de los datos.

En la Figura 4-31 se observan los resultados obtenidos de los parámetros de interés en la interfaz gráfica de control para las condiciones arriba descritas y B=2. En cambio, en la Figura 4-32 se observan estos mismos resultados pero usando un valor de B=4.

Como se puede observar y era de esperarse, al incrementarse el tamaño de los buffers internos de la red, el promedio de paquetes perdidos es menor, pero el valor correspondiente al retardo promedio en atravesar la red se incrementa.

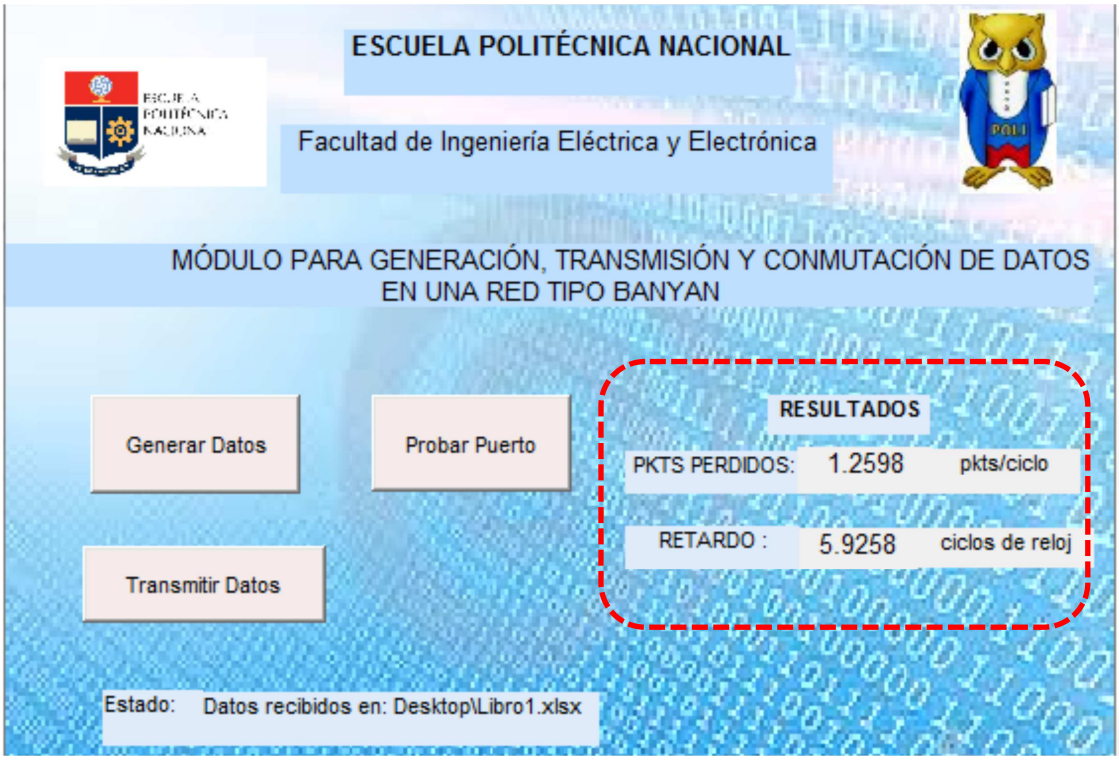


Figura 4-31. Interfaz gráfica mostrando los resultados obtenidos para B=2.

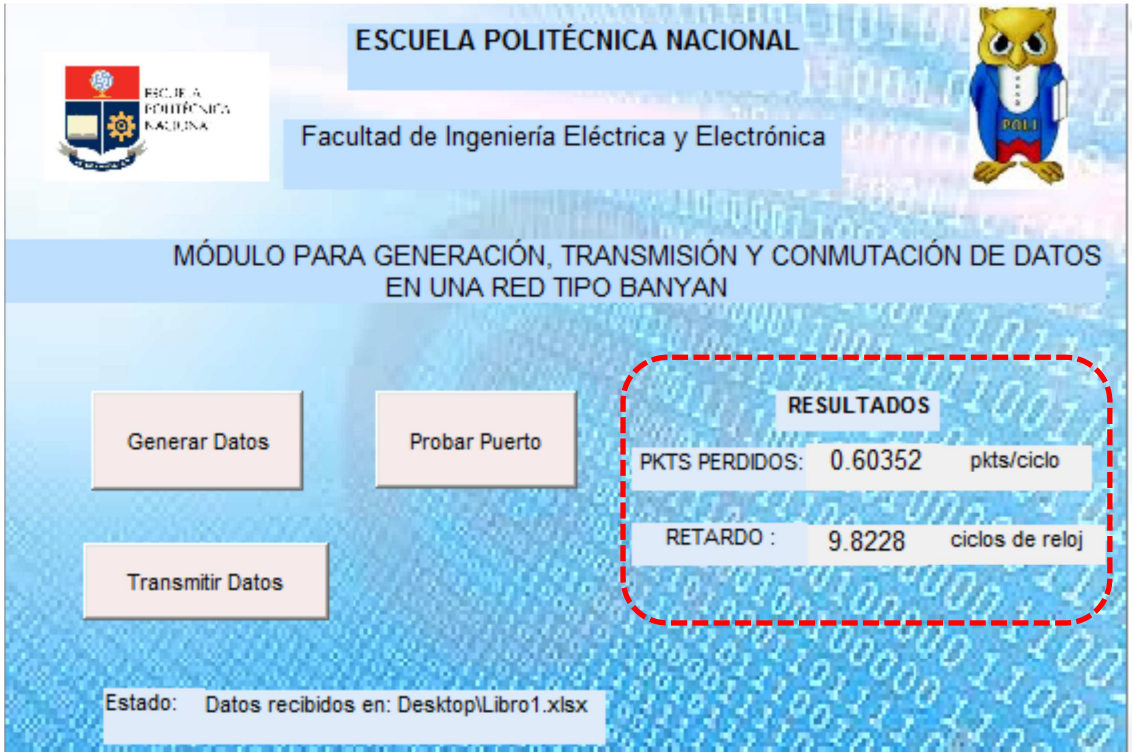


Figura 4-32. Interfaz gráfica mostrando los resultados obtenidos para B=4.

Finalmente, en la Tabla 4-12 se observa un resumen comparativo con los valores obtenidos de los parámetros de interés tanto en la parte teórica como en la simulación e implementación de la red diseñada, para las longitudes de buffer B=2 y B=4, ya que solamente estas longitudes son las que fue posible implementarlas de manera práctica. Cabe señalar que los valores correspondientes a la implementación del circuito se los obtuvo al enviar 512 paquetes por cada entrada de la red a fin de obtener resultados más confiables.

Tabla 4-12. Resultados comparativos entre teoría, simulación e implementación.

B	Teoría		Simulación		Implementación	
	W_T	S_T	W_T	S_T	W_T	S_T
2	5,2225	2,201	4	1,652	5,92	1,26
4	7,5609	1,203	8	0,717	9,82	0,6

De los resultados mostrados en la Tabla 4-12 se puede verificar que los valores obtenidos en la implementación práctica de la red tienen un mayor nivel de concordancia con los valores correspondientes a la simulación; esto se lo puede explicar ya que tanto la simulación como la implementación práctica están basadas en el mismo código VHDL y por ende presentan igual comportamiento.

REFERENCIAS

[1] Pong P. Chu. FPGA Prototyping by VHDL Examples. Editorial John Wiley & Sons, Inc. 2008.

[2] Pong P. Chu. RTL Hardware Design using VHDL. Cleveland State University.Editorial John Wiley & Sons, Inc. 2006

[3] Parnell Karen, Mehta Nick. Programmable Logic Design.Xilinx.Second Edition. 2002.

[4] Project Navigator. ISE Design Suite 14.4

CAPÍTULO 5

CONCLUSIONES Y RECOMENDACIONES

CONCLUSIONES

- Una conclusión muy notoria de este proyecto es que en este tipo de redes en las cuales existen buffers internos, al incrementar la longitud de los buffers, se reduce en gran medida el promedio de paquetes perdidos en la red peroel tiempo que le toma a los paquetes para atravesar la red se incrementa también. Por ello, se debe tratar de buscar un equilibrio entre estos parámetros, ya que no serviría de mucho el incrementar desmedidamente la longitud de los buffers, si el retardo que experimentan los paquetes para atravesar la red se dispara a un valor muy elevado.

- En cuanto al promedio de paquetes perdidos en la red, se puede observar que, al incrementarse el tamaño de los buffers internos, se reduce el valor de este parámetro de una forma no lineal, es decir, que al pasar de tamaño B=2 a B=4 se tiene una reducción del promedio de paquetes perdidos de aproximadamente un 50% lo cual es una gran mejora; pero al pasar de un tamaño de buffer B=8 a B=16, la reducción del promedio de paquetes perdidos es solamente de alrededor del 20%. En base a esta observación se puede concluir que no sería conveniente incrementar mucho la longitud de los buffers ya que la mejora en el desempeño de la red no es significativa y además se eleva mucho la demanda de recursos para memoria en el sistema.

- Las diferencias que existen entre los resultados teóricos con los simulados se deben a varios factores, entre los principales cabe destacar los siguientes:

    - Estado de equilibrio no alcanzado: en el análisis matemático realizado, se asume que el sistema ya ha alcanzado el estado de equilibrio estadístico, pero en la simulación del sistema, debido a la limitaciónen cuanto a la máxima resolución del simulador empleado, no está totalmente garantizado que se haya alcanzado el estado de equilibrio en la red, lo cual puede conllevar pequeñas diferencias en los resultados obtenidos.
    - Tráfico uniforme solamente a la entrada de la primera etapa de la red: si bien, con el banco de pruebas generado para la simulación de la red se trata de generarun tráfico uniforme en las entradas de la red, luego de la primera etapa de conmutadores, ya no es posible garantizar que el patrón de tráfico siga siendo uniforme, con lo cual se generarían diferencias entre los resultados teóricos y los simulados, ya que en la parte teórica se asume que para todos los buffers de la red el tráfico a la entrada es de tipo uniforme.

- De los resultados obtenidos, se puede verificar que a pesar de tener la máxima carga de tráfico en las entradas de la red, el promedio de paquetes perdidos en la red en cada ranura de tiempo es bastante bajo, incluso con el mínimo valor de la longitud de los buffers internos; esto se debe al uso de los elementos de conmutación Banyan duales, ya que la técnica de almacenamiento que usan les permite enviar dos paquetes en cada ranura de tiempo sin que exista contención entre ellos.

- Otro punto a mencionar es quepara el caso de las redes Banyan, la aplicación de la técnica de almacenamiento *Virtual Output Queuing* (VOQ) en el interior de la red, es decir, en cada elemento de conmutación Banyan dual genera excelentes resultados en cuanto al desempeñoglobal de la red; resultados que no serían posibles si se aplicara ésta misma técnica de almacenamiento en las entradas de la red, ya que en ese caso el número

de buffers requeridos sería considerablemente mayor y, se requeriría también la implementación de un algoritmo centralizado que controle el comportamiento de cada uno de los buffers a fin de tratar de minimizar la probabilidad de bloqueo en el interior de la red.

- Como resultado de la realización de este proyecto, se puede también observar que gracias al funcionamiento de los elementos de conmutación Banyan duales empleados, se mejora el nivel de desempeño de la red de conmutación, pero el costo a pagar es, principalmente, el incremento en el requerimiento de memoria necesario para la implementación de los buffers internos; por ello, para cada aplicación específica será necesario hacer el análisis de la capacidad de memoria disponible y el requerimiento del sistema en cuanto a paquetes perdidos a fin de encontrar un equilibrio entre éstos parámetros.

- En la implementación práctica del sistema diseñado, solamente fue posible usar los valores de 2 y 4 para la longitud de los buffers internos, esto debido a las limitaciones de memoria del dispositivo usado y a la gran demanda de recursos de memoria que implica la implementación de los buffers internos de la red de conmutación.

- En base a un análisis detallado del reporte generado tras la implementación del sistema, fue posible determinar que la principal limitante que se tuvo en cuanto a frecuencia fue debido a la ubicación de los diferentes bloques lógicos, es decir que los principales retardos introducidos en el diseño fueron debidos al enrutamiento más que los debidos a niveles de lógica utilizados. Esto principalmente se debe a la presencia de los buffers en las entradas y salidas de la red, ya que estos buffers fueron implementados usando bloques de memoria RAM, mismos que se encuentran en posiciones fijas dentro del FPGA usado.

- Se verificó la funcionalidad del diseño usando un bloque de datos generados y enviados desde la PC mediante comunicación serial, el cual fue luego almacenado y procesado en forma paralela en el FPGA. En el

caso del sistema implementado, no es posible usar un flujo continuo de datos debido a la gran diferencia de velocidad existente entre la interfaz de comunicación serial RS-232 y la frecuencia de operación del FPGA, ya que si por ejemplo se considera una velocidad de transmisión de 19200 bps y la frecuencia de operación del FPGA de 50 MHz, se tendría que cada 26080 ciclos de reloj del FPGA se recibe un byte de datos desde la PC, y en ese caso no tendría sentido alguno el usar la red de conmutación.

- En cuanto a la descripción de los componentes del sistema, otro punto a destacar es que, el diseño aquí presentado es fácilmente escalable, esto gracias al uso de constantes de uso global en el paquete creado, así como el uso de genéricos y lazos *for* en la descripción de la mayoría de los bloques implementados, con ello es posible variar el valor de parámetros tales como: tamaño de la red, longitud de los buffers internos, tamaño del paquete empleado y velocidad de transmisión, sin que ello conlleve mayores complicaciones.

- Respecto al lenguaje de descripción de hardware VHDL, se trata de un lenguaje bastante fácil de aprender ya que tiene una gran similitud con lenguajes de programación muy difundidos tales como C. Pero, a pesar de ello, se debe tener siempre en mente que se trata de un lenguaje para describir hardware, y por ello su ejecución es inherentemente de tipo concurrente. Es por esta razón que al usar este lenguaje para describircualquier sistema digital, es de suma importancia prestar atención a aspectos como el sincronismo entre los módulos que conforman el circuito.

- Se puso en evidencia la gran versatilidad que presenta el lenguaje VHDL para la descripción de circuitos digitales de mediana y gran complejidad. Cada uno de los estilos de descripción tiene sus características y aplicaciones propias, pero a pesar de ello, se los puede combinar en un mismo diseño sin que existan conflictos o problemas para su síntesis y posterior implementación.

- Con la realización de este proyecto, otro punto a destacar es que la simulación funcional es útil para verificar de manera simple el comportamiento del circuito diseñado, pero sus resultados no implican necesariamente que el diseño realizado del circuito sea óptimo o siquiera que sea posible implementarlo en el FPGA, esto debido a que esta simulación no toma en cuenta aspectos fundamentalespara la implementación práctica como son los retardospor enrutamiento y por niveles de lógica usados. Por ejemplo, si se realiza la descripción de un circuito en el cual se incluyen muchos niveles de lógica para una señal, al simular su comportamiento, aparentemente funcionará adecuadamente, pero al momento de tratar de implementarlo usando una frecuencia de reloj específica, la herramienta de compilación nos indicará que la máxima frecuencia a la que puede operar el circuito diseñado es menor que la frecuencia de reloj disponible y, como consecuencia, no se lo puede implementar en el FPGA.

- Las máquinas de estado constituyen una herramienta extremadamente útil y valiosa para la descripción de circuitos bastante complejos y en los cuales su comportamiento se puede describir como una secuencia definida de estados.Al hacer uso de esta herramienta se simplifica mucho la tarea de descripción de un circuito, ya que teniendo un diagrama de estados completo del circuito a diseñar, es bastante sencillo traducir este diagrama a lenguaje VHDL haciendo uso de la sentencia *case* y especificando para cada uno de los posibles estados del sistema, los valores de las salidas y el estado siguiente al que pasa el sistema. En el caso de este proyecto, esta herramienta fue de especial importancia en la descripción del sistema UART así como del módulo de control.

- El uso de *IPCores*para implementar los FIFOs tanto de entrada como de salida de la red Banyan fue una gran ayuda dentro del diseño delsistema, ya que éstos son módulos pre-compilados y descritos de manera que optimizan los recursos lógicos disponibles, además con el uso de este tipo

de módulos se reduce considerablemente el tiempo requerido para realizar la descripción de un sistema complejo y de gran tamaño, logrando así que el diseñador pueda enfocar su atención en las partes medulares del diseño.

- Se ha puesto en evidencia también el gran potencial que poseen los FPGAs, incluso de gama baja como el empleado en este proyecto. Gracias a la gran cantidad de recursos lógicos y de memoria de que disponen estos dispositivos, constituyen una alternativa muy atractiva para la implementación de circuitos digitales bastante complejos y de aplicación en los campos más diversos del conocimiento.

- Al concluir con la realización de este proyecto se pudo verificar el adecuado funcionamiento de la red Banyan y principalmente de los elementos de conmutación duales; se cumplió el objetivo del proyecto de determinar los principales parámetros de desempeño de la red tanto de manera teórica como en la simulación e implementación práctica de la red. Además, en el diseño de la red se incluyó la característica muy interesante de describir el sistema de forma paramétrica, de modo que el código aquí presentado puede ser re utilizado para implementar redes de mayor capacidad, siempre y cuando se cuente con un FPGA con mayor número de recursos lógicos y de memoria.

RECOMENDACIONES

- Al describir cualquier tipo de circuito digital haciendo uso de lenguajes de descripción de hardware, se recomienda en lo posible, el uso de la descripción estructural y jerárquica, ya que ésta permite la reutilización de bloques y la comprobación de su funcionamiento de forma independiente antes de interconectarlos con el resto del diseño, de esta forma se hace posible depurar la descripción de los circuitos fácilmente y encontrar posibles errores de descripción y funcionamiento de cada bloque.

- También, en cuanto a la descripción en VHDL de cualquier circuito se debe tener siempre en mente que tras cada sentencia que se escriba existe un circuito que se genera y, de acuerdo a la descripción que se haga el circuito puede cambiar aunque su comportamiento sea similar. Por ello, es importante no solo conocer la sintaxis correcta de cada sentencia del lenguaje, sino también tener una idea básica del circuito que se está generando, ya que de ello depende en gran medida los resultados obtenidos en el momento de la implementación práctica del circuito en cuanto a frecuencia de operación y cantidad de recursos utilizados.

- Tratar de usar el menor número posible de niveles de lógica en la descripción de los circuitos a fin de minimizar el retardo introducido por la lógica y, de esta forma, evitar problemas de sincronismo en la implementación final.

- Se podría mejorar el diseño aquí presentado con el uso de un FPGA que tenga mayor capacidad de memoria, con ello incluso se podría implementar, no solo una red de mayor tamaño, sino un tipo de red de conmutación más complejo como por ejemplo: redes Batcher-Banyan y redes Banyan dilatadas o replicadas.

- En lo relacionado a las redes Banyan, otra alternativa interesante sería la de tratar de observar su comportamiento ante un patrón de tráfico de otras características, por ejemplo, tráfico de tipo auto-similar, el cual es un modelo que se ajusta más a las características reales del tráfico en redes de datos. Además, se podría también comparar el comportamiento de varios tipos de redes ante este patrón de tráfico y tratar de determinar cuál o cuáles de ellas tienen un mejor desempeño en esta situación y por qué.

- Otra recomendación que se puede hacer en cuanto a la descripción de circuitos digitales con el lenguaje VHDL es que siempre se debe usar Flip-Flops o memorias RAM para el almacenamiento de los datos a fin de evitar la generación de *latches*, ya que estos producen problemas de sincronismo en el momento de la implementación práctica del circuito. La principal causa de los *latches* es el uso de sentencias *if, case, when* en las cuales no se especifiquen todas las alternativas posibles. Por ello, también es importante prestar atención a los reportes generados por la herramienta de compilación y síntesis de los circuitos, ya que en estos reportes se informa de la existencia de *latches* y su ubicación en los módulos específicos del diseño.

- Una recomendación final sería que en nuestra facultad se dé un mayor énfasis y apoyo a proyectos de tipo práctico que hagan uso de los equipos y herramientas tecnológicas disponibles actualmente como son los módulos de desarrollo FPGA, *arduino* y*raspberry pi*, mismos que pueden ser muy útiles para una gran variedad de aplicaciones. Además cabe recalcar que estos equipos mencionados (*arduino* y *raspberry pi*) son de bajo costo y su programación es mucho más sencilla que la del FPGA, sin contar con el hecho de que son de código abierto y han sido específicamente creados con fines académicos.

REFERENCIAS BIBLIOGRÁFICAS

Alverca, Y. Síntesis de Circuitos Digitales utilizando VHDL y FPGAs. Proyecto de Titulación. EPN. 2008.

Ashenden, P. VHDL Tutorial. Elsevier Science. USA. 2004

Ashenden, P. The VHDL Cookbook.Department of Computer Science.University of Adelaide. South Australia. First Edition. July, 1990.

ATM Switch. Disponible: http://lyle.smu.edu/~nair/courses/7344/7_atm_2_v2.pdf

Arquitectura e Ingeniería de Computadores. Disponible: http://www.fdi.ucm.es/profesor/llorente/AIC/Leccion-14.pdf

Bouras, C. Gkantsidis, C. Cost of implementing Banyan networks for use in ATM switching fabrics. Disponible: http://ru6.cti.gr/ru6/publications/8526611.pdf

Chen, T. Liu, S. ATM Switching. Chapter to appear in Wiley Encyclopedia of Telecommunicaction and Signal Processing.John Proakis. Diciembre 2002. Disponible: http://lyle.smu.edu/~tchen/papers/encyclopedia-chap.pdf

Chambers, P. The Ten Commandments of Excellent Design – VHDL Code. VLSI Technology. 1997.

Chu, P. FPGA Prototyping by VHDL Examples. Editorial John Wiley & Sons, Inc. 2008.

Chu, P. RTL Hardware Design using VHDL.Cleveland State University.Editorial John Wiley & Sons, Inc. 2006

Culler, D. Pal Singh, J. Parallel Computer Architecture.Morgan Kaufmann Publishers, Inc. 1999

García, M. Tubella, J. González, A. Aliagas, C. Esteban, L. Multiprocesadores Von Neumann. UPC/DAC Report, 1990. Disponible: http://www.atc.uniovi.es/inf_superior/atc2010/PARALELAS/Multiprocesadores%20Von%20Neumann(UPC90-1).pdf

Grove, M. Introduction to Interconnection Networks. Virginia Tech. October 2011.

Hennesy, J. Patterson, D. Computer Architecture – A Quantitive Approach. Morgan Kaufmann Publishers, Inc. USA. 2003.

Hidell, M. Sjödin, P. Hagsand, O. Router Architectures. Disponible: http://web.ict.kth.se/~mahidell/pubs/networking04_tutorial_final.pdf

Huang, T. Mathematical Modeling of a Dual-Banyan Switch in ATM networks.Disponible: http://www.npue.edu.tw/adm/research/%E5%AD%B8%E5%A0%B1/13/10.pdf

Jimenez, P. Diseño e implementación de un sistema de adquisición, compresión y almacenamiento de imágenes empleando VHDL (VHSIC Hardware Description Language) y FPGAs (Field Programmable Gate Arrays). Proyecto de Titulación. EPN. 2011.

Khansari, M. Movaghar-Rahimabadi, A. Hessabi, S. Design and Simulation of a Tandem ATM Switch based on FPGA. Disponible: http://ce.sharif.ac.ir/~khansari/docs/ATMSwitch.pdf

Kilts, S. Advanced FPGA Design. Architecture, Implementation and Optimization.John Wiley & Sons, Inc. 2007.

Kloth, A. Advanced Router Architectures. Taylor & Francis Group, LLC. 2006

Kruskal, C. Snir, M. A Unified Theory Of Interconnection Network Structure. Disponible: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.45.3068

La Importancia De La Red. Disponible: http://www.wikilearning.com/tutorial/el_manual_para_el_clustering_con_openmosix-la_importancia_de_la_red_i/9756-11

Liao, R. Leon-Garcia, A. Fault Tolerant Design of Batcher-Banyan Class ATM Switches.

Modiano, E. Conmutación Rápida De Paquetes. Disponible: http://mit.ocw.universia.net/6-263JData-Communication-NetworksFall2002/NR/rdonlyres/Electrical-Engineering-and-Computer-Science/6-263JData-Communication-NetworksFall2002/C1C0738E-C7B6-447E-AF48-914F63DE022B/0/Lectures17_18.pdf

Newman, P. Fast Packet Switch Architecture. University of Cambridge, 1988.http://pnewman.com/papers/thesis/chapter3.pdf

Parnell Karen, Mehta Nick. Programmable Logic Design.Xilinx.Second Edition. 2002.

Pattavina, A. Switching Theory: Architecture and Performance in Broadband ATM Network. Ed. John Wiley & Sons Ltd, 1998.Disponible: http://docstore.mik.ua/cisco/pdf/routing/John%20Wiley%20%26%20Sons%20-%20Switching%20Theory,%20Architectures%20and%20Performance%20in%20Broadband%20ATM%20Networks.pdf

Parallel Computing. Disponible: http://en.wikipedia.org/wiki/Parallel_computing

Perry, D. VHDL: Programming by example. McGraw-Hill. 4th Edition. 2002.

Redes De Interconexión. Disponible: http://www.infor.uva.es/~bastida/Arquitecturas%20Avanzadas/Redes.pdf

Redes De Interconexión. Disponible:http://verde.lma.fi.upm.es/wwwwver/pvp/Tema4.Redes_interconexion.pdf

Redes De Interconexión Para Computadores Masivamente Paralelos. Disponible: http://ditec.um.es/~jmgarcia/papers/escuela94.pdf

Song, M. Switches. Disponible: http://www.ece.odu.edu/~msong/switches.pdf

Source Routing.Disponible: http://www.iss.net/security_center/advice/Underground/Hacking/Methods/Technical/Source_Routing/default.htm

Supercomputer. Disponible: http://en.wikipedia.org/wiki/Supercomputer

Supercomputer Architecture.Disponible: http://en.wikipedia.org/wiki/Supercomputer_architecture

Switch Concept & Architecture. Disponible: http://speed.cis.nctu.edu.tw/~ydlin/course/cn/srouter/Lesson3.PDF

Switch Construction. Disponible: http://www.cs.wmich.edu/~alfuqaha/Fall07/cs6030/lectures/SwitchDesign-rev2.pdf

Telecommunication Switching and Network System.Disponible: http://www.cmlab.csie.ntu.edu.tw/~freetempo/CN2011/slides/Tutorial_banyan.pdf

UIT-D. Manual sobre Ingeniería de Teletráfico. Ginebra. 2002.

Vasiliadis, D. Rizos, G. Vassilakis, C. Performance Analysis of blocking Banyan Switches.Disponible: http://www.sdbs.uop.gr/files/banyanSwitch.pdf

Volnei A. Pedroni. Circuit Design with VHDL.Editorial MIT Press. 2004.

Wu, W. Packet Forwarding Technologies.Auerbach Publications. 2008.

Xilinx.LogiCORE IP FIFO Generator v8.1.User Guide. Marzo 2011. Disponible: http://www.xilinx.com/support/documentation/ip_documentation/fifo_generator_ug175.pdf

Xilinx.Spartan-3 GenerationFPGA User Guide. June 2011.

Xilinx.Spartan-3 GenerationFPGA User Guide. October 2009.

Xilinx.Spartan-3A/3ANFPGA Starter KitBoard User Guide. June 2008.

Xilinx.VHDL Coding Style Guidelines.Disponible: http: //users.eecs.northwestern.edu/~seda/coding_guidelines_013003.pdf

Youssef, M. El-Derini, M. Aly, H. Performance Enhancement Techniques of a Banyan Network Based Interconnection Structure. Disponible: http://wrc.ejust.edu.eg/people/moustafa/papers/aej99.pdf

Zadedyurina, O. Ofek, Y. Pattavina, A. Space and Time Blocking versus Cost in All-Optical Banyan Networks. Disponible: https://www-vs.informatik.uni-ulm.de/DE/intra/bib/2008/ICC/DATA/S09S06P06.PDF

ANEXOS

ANEXO 1

EQUIVALENCIA DE REDES

A fin de establecer las posibles relaciones existentes entre redes de interconexión, se debe empezar por mencionar que a las redes de interconexión se las puede modelar mediante grafos[25]. El grafo correspondiente a una red de interconexión multi-etapa se lo obtiene de la siguiente manera: cada SE se lo representa como un vértice del grafo y se representan también los enlaces entre SE (vértices). Inicialmente, no es necesario etiquetar los enlaces ni los vértices, tampoco se representan los puertos de entrada y salida, de modo que en los extremos del grafo solamente se encuentran los vértices correspondientes a los SE de la primera y última etapa. En la Figura A1-1 se observa la red Omega junto a su respectivo grafo.

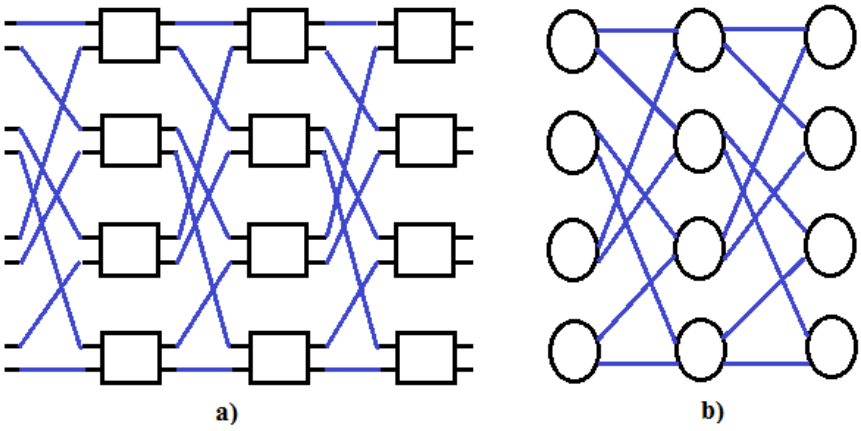


Figura A1-1. a) Red Omega, b) Grafo correspondiente a la red Omega.

Una vez comprendido el procedimiento para obtener el grafo correspondiente a una red, es posible enunciar el concepto del isomorfismo entre grafos.

[25]Un grafo G = (V, E) es una estructura formada por un conjunto no vacío V de elementos llamados vértices y un conjunto E cuyos elementos son pares no ordenados de vértices distintos llamados ramas.

Dos grafos A y B se dice que son isomorfos si, luego de etiquetar los vértices de A con las etiquetas correspondientes a B, el grafo Apuede llegar a ser idéntico a B, al mover sus vértices y por ende los enlaces que los conectan.

En la Figura A1-2 se observa un ejemplo de grafos isomorfos, así como la correspondencia entre los vértices de A y B para que ambos sean idénticos.

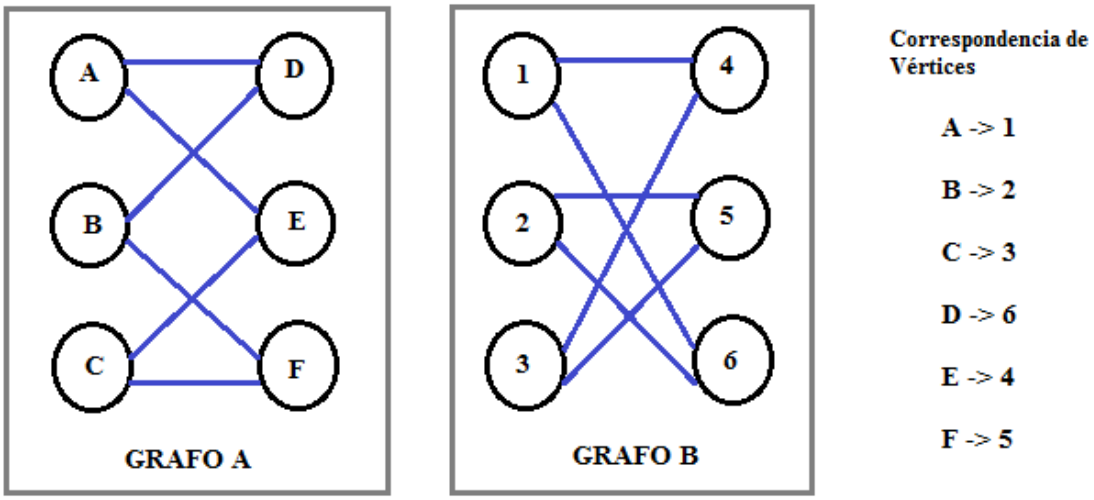


Figura A1-2. Ejemplo de isomorfismo entre grafos.

Ahora bien, con este concepto, es posible definir la primera relación que puede existir entre redes.

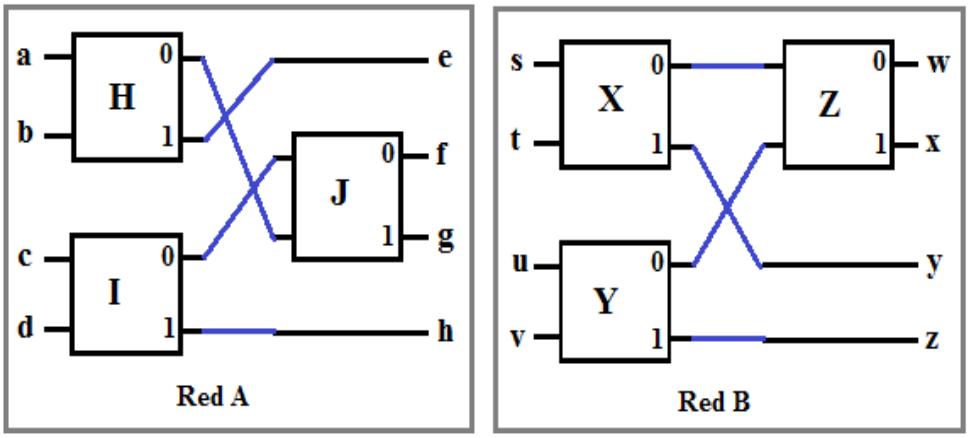
- **Equivalencia Topológica:** Se dice que dos redes son topológicamente equivalentes si existe un isomorfismo entre los grafos correspondientes a dichas redes.

Es claro que, una red de interconexión es una estructura más compleja que simplemente un grafo, ya que en este caso se debe tomar en cuenta no solo la secuencia de nodos, sino también las etiquetas de cada puerto de salida de los SE. De modo que se debe definir un isomorfismo entre redes que tome en cuenta las etiquetas de los puertos de salida de cada SE.

Tomando como referencia una red de tamaño $N \times N$, conformada con elementos de conmutación de tamaño $b \times b$, se etiquetan las entradas y salidas de la red de manera secuencial empezando desde los puertos superiores, se etiqueta también los puertos de salida de los SE desde 0 hasta $b - 1$.

Dos redes A y B se dice que son isomorfas si, luego de etiquetar las entradas, salidas y SE de la red A con las etiquetas correspondientes a B, la red A puede hacerse idéntica a B al mover los SE con sus respectivos enlaces.

Cabe notar que el hecho de etiquetar las entradas y salidas de la red A con las etiquetas de B, es equivalente a añadir una permutación adecuada a la entrada y salida de la red A. Este tipo de isomorfismo entre redes implica que la topología modificada de la red A conserve las etiquetas de los puertos de salida de los SE sin alteración, de modo que se trata de un isomorfismo con preservación de etiquetas. La Figura A1-3 muestra un ejemplo de redes isomorfas, así como la correspondencia entre entradas, salidas y SE, a fin de que la red A sea idéntica a la red B.



Correspondencia entre elementos (A -> B)

Entradas	Salidas	SEs
a -> u	e -> z	H -> Y
b -> v	f -> w	I -> X
c -> s	g -> x	J -> Z
d -> t	h -> y	

Figura A1-3. Ejemplo de isomorfismo entre redes.

Entonces, la segunda relación posible entre redes es el isomorfismo, que se define a continuación.

- **Isomorfismo:**Se dice que dos redes son isomorfas si existe un isomorfismo con preservación de etiquetas entre ellas.

Dos redes isomorfas también son topológicamente equivalentes, aunque lo contrario no siempre se cumple.

En la Figura A1-4 se muestra un ejemplo de redes que son topológicamente equivalentes, pero no son isomorfas. En esa figura se aprecia que el grafo de ambas redes es igual, por ende son topológicamente equivalentes, pero no es posible hallar un isomorfismo con preservación de etiquetas entre ellas. Como se puede apreciar, al mover los SE de la red A, es posible obtener las redes A' y A'', las cuales son similares a B, pero no idénticas.

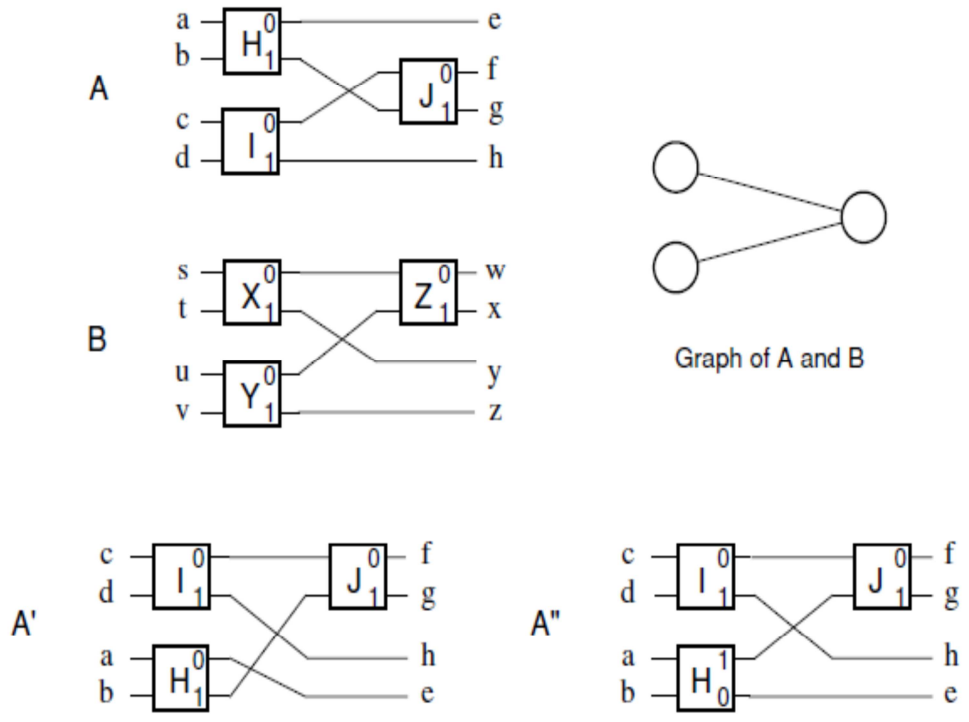


Figura A1-4. Ejemplo de redes topológicamente equivalentes pero no isomorfas.

Finalmente, se enuncia la última relación que puede existir entre redes.

- **Equivalencia Funcional:**Se dice que dos redes son funcionalmente equivalentes si realizan las mismas permutaciones, es decir, es posible obtener el mismo conjunto de conexiones entrada/salida en cada ranura de tiempo.

En la Figura A1-5 se muestra un ejemplo de redes funcionalmente equivalentes. Para demostrarlo, si se trata de realizar, por ejemplo, las conexiones entrada/salida (0,6) y (3,1) simultáneamente en ambas redes, esto es posible. De igual manera, si se desea realizar simultáneamente las conexiones entrada/salida (0,6) y (5,7) en ambas redes, esto no es posible en ninguna de las redes. Se puede verificar que, para cualquier par de conexiones entrada/salida que se desee establecer, ambas redes presentan el mismo comportamiento.

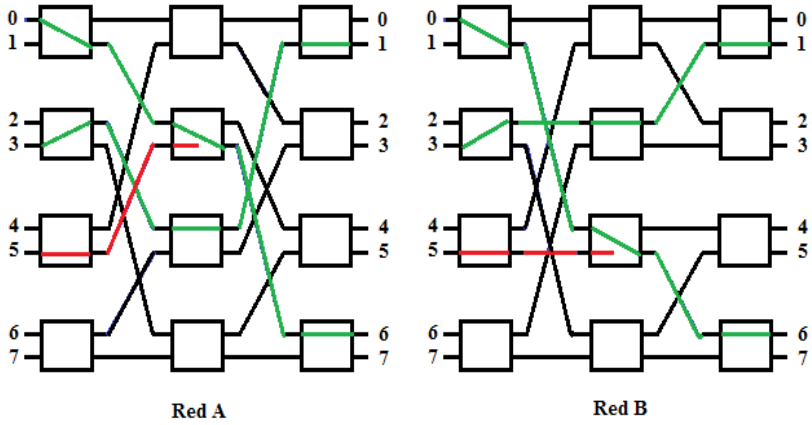


Figura A1-5. Ejemplo de redes funcionalmente equivalentes.

En general, se tiene que dos redes isomorfas o topológicamente equivalentes no son funcionalmente equivalentes.Notemos, por ejemplo, las redes A y B en la Figura A1-3, las cuales son isomorfas, pero no son funcionalmente equivalentes. Para demostrarlo, en la Figura A1-6 se tiene estas mismas redes con sus entradas y salidas numeradas desde 0 hasta 3, luego, por ejemplo, al tratar de realizar las conexiones entrada/salida (0,0) y (1,1); en la red A es posible realizarlas simultáneamente; mientras que en la red B esto no es posible.

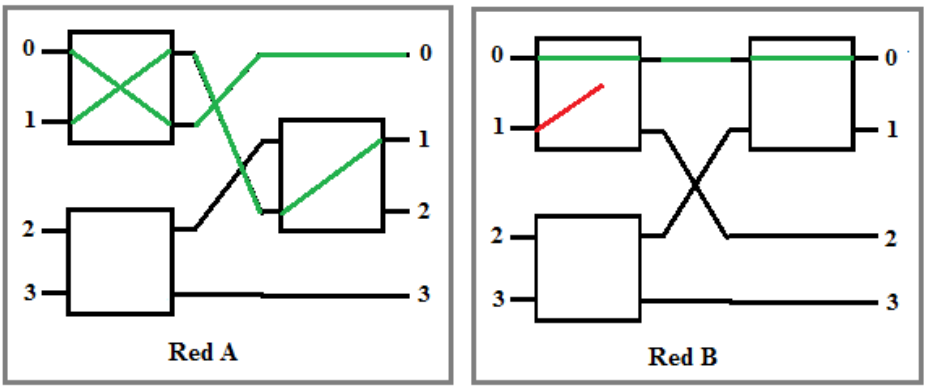


Figura A1-6. Ejemplo de redes isomorfas pero no funcionalmente equivalentes.

ANEXO 2

RESULTADOS TEÓRICOS: PROBABILIDADES DE ESTADO Y PARÁMETROS DE INTERÉS PARA LOS DIFERENTES TAMAÑOS DE BUFFER.

Como se aclaró en la Sección 2.3.2.5, se realizó un análisis de la carga de tráfico presente a la entrada de cada una de las etapas de la red, y los resultados que se presentan a continuación se los ha obtenido bajo la condición de máxima carga de tráfico en las entradas de la red, es decir, $\lambda_1 = 1$.

A continuación se explica la nomenclatura usada en las tablas de este anexo y a la vez se vuelven a presentar las ecuaciones en base a las cuales se han obtenido los valores mostrados:

- $P(i)$: Probabilidad de que el sistema se encuentre en el estado *i*, es decir, que existan *i* paquetes almacenados en el buffer.

$$P(i) = 2\left(\frac{\lambda P_u}{(1-\lambda P_u)}\right)^i \left\{\frac{[1-P(0)]^{i-1}}{[1+P(0)]^i}\right\} P(0) \quad 1 \le i \le B$$

- L_j:Longitud promedio de la cola de un buffer perteneciente a la *j*-ésima etapa de conmutadores.

$$L = \sum_{i=1}^{B} i\, P(i)$$

- s_j: Promedio de paquetes perdidos en cada ranura de tiempo en un buffer perteneciente a la *j*-ésima etapa de conmutadores.

$$s_j = P(B)\lambda P_u \frac{(1-P(0))}{2}$$

- W_j: Retardo promedio que experimenta un paquete al atravesar un buffer perteneciente a la *j*-ésima etapa de conmutadores.

$$W_j = \frac{L}{(1 - s_j)\lambda P_u}$$

- S_j:Promedio de paquetes perdidos en todos los buffers correspondientes a la *j*-ésima etapa de conmutadores de la red. Tomando en cuenta una red de tamaño 8x8, es decir, en cada etapa existen 4 elementos de conmutación, cada uno de los cuales contiene 4 buffers internos.

$$S_j = 16 * s_j$$

- λ_k: Valor que representa la carga de tráfico presente en cada una de las entradas de los conmutadores de la *k*-ésima etapa.

$$\lambda_{i+1} = \frac{\lambda_1 N - \sum_{j=1}^{i} S_j}{N}$$

Cabe mencionar que en las Tablas A2-4 y A2-5 se presentan los valores solamente para la primera y segunda etapa de la red, esto debido a que en los valores correspondientes a la tercera etapa no se tiene cambio respecto a los de la segunda etapa.

Tabla A2-1. Resultados Teóricos para B=2.

λ_1	Probabilidades de Estado			L_1	s_1	W_1	S_1	λ_2
	P(0)	P(1)	P(2)					
1	0,2956	0,4563	0,2841	1,0245	0,0526	2,1627	0,8414	0,8948
λ_2	Probabilidades de Estado			L_2	s_2	W_2	S_2	λ_3
	P(0)	P(1)	P(2)					
0,8948	0,4947	0,4166	0,0886	0,5938	0,0100	1,3406	0,1602	0,8748
λ_3	Probabilidades de Estado			L_3	s_3	W_3	S_3	λ_4
	P(0)	P(1)	P(2)					
0,8748	0,5177	0,4056	0,0766	0,5588	0,0081	1,2880	0,1293	0,8586
					TOTAL	4,7913	1,1309	

Tabla A2-2. Resultados Teóricos para B=4.

λ_1	Probabilidades de Estado					L_1	s_1	W_1	S_1	λ_2
	P(0)	P(1)	P(2)	P(3)	P(4)					
1	0,1991	0,3321	0,2218	0,1481	0,0989	1,6156	0,0198	3,2965	0,3168	0,9604
λ_2	Probabilidades de Estado					L_2	s_2	W_2	S_2	λ_3
	P(0)	P(1)	P(2)	P(3)	P(4)					
0,9604	0,3683	0,4140	0,1470	0,0522	0,0185	0,9386	0,0028	1,9601	0,0449	0,9548
λ_3	Probabilidades de Estado					L_3	s_3	W_3	S_3	λ_4
	P(0)	P(1)	P(2)	P(3)	P(4)					
0,9548	0,3818	0,4156	0,1398	0,0470	0,0158	0,8994	0,0023	1,8884	0,0373	0,9501
							TOTAL	7,1450	0,3990	

Tabla A2-3. Resultados Teóricos para B=8.

λ_1	Probabilidades de Estado					L_1	s_1	W_1	S_1	λ_2
	P(0)	P(1)	P(2)	P(3)	P(4)					
1	0,1279	0,2267	0,1753	0,1356	0,1048	2,8245	0,0082	5,6956	0,1308	0,9836
	P(5)	P(6)	P(7)	P(8)						
	0,0811	0,0627	0,0485	0,0375						
λ_2	Probabilidades de Estado					L_2	s_2	W_2	S_2	λ_3
	P(0)	P(1)	P(2)	P(3)	P(4)					
0,9836	0,3058	0,3886	0,1714	0,0756	0,0333	1,2346	0,0002	2,5108	0,0036	0,9832
	P(5)	P(6)	P(7)	P(8)						
	0,0147	0,0065	0,0029	0,0013						
λ_3	Probabilidades de Estado					L_3	s_3	W_3	S_3	λ_4
	P(0)	P(1)	P(2)	P(3)	P(4)					
0,9832	0,3077	0,3895	0,1707	0,0748	0,0328	1,2255	0,0002	2,4934	0,0033	0,9828
	P(5)	P(6)	P(7)	P(8)						
	0,0144	0,0063	0,0028	0,0012			TOTAL	10,7	0,1376	

Tabla A2-4. Resultados Teóricos para B=16.

λ_1	Probabilidades de Estado					L_1	s_1	W_1	S_1	λ_2
	P(0)	P(1)	P(2)	P(3)	P(4)					
1	0,0791	0,1466	0,1251	0,1068	0,0911	4,7865	0,0031	9,6031	0,0501	0,9937
	P(5)	P(6)	P(7)	P(8)	P(9)					
	0,0778	0,0566	0,0483	0,0412	0,0352					
	P(10)	P(11)	P(12)	P(13)	P(14)					
	0,0352	0,0300	0,0256	0,0219	0,0187					
	P(15)	P(16)								
	0,0159	0,0136								

λ_2	Probabilidades de Estado					L_2	s_2	W_2	S_2	λ_3
	P(0)	P(1)	P(2)	P(3)	P(4)					
	0,2510	0,3537	0,1867	0,0985	0,0520	1,5839	0,0000	3,1878	0,0000	0,9937
	P(5)	P(6)	P(7)	P(8)	P(9)					
	0,0274	0,0145	0,0076	0,0040	0,0021					
0,9937	P(10)	P(11)	P(12)	P(13)	P(14)					
	0,0011	0,0006	0,0003	0,0002	0,0001					
	P(15)	P(16)								
	0,0000	0,0000								
							TOTAL	15,9786	0,0501	

Tabla A2-5. Resultados Teóricos para B=32.

λ_1	Probabilidades de Estado					L_1	s_1	W_1	S_1	λ_2
	P(0)	P(1)	P(2)	P(3)	P(4)					
	0,0476	0,0908	0,0826	0,0751	0,0682	8,9616	0,0011	17,9433	0,0179	0,9978
	P(5)	P(6)	P(7)	P(8)	P(9)					
	0,0620	0,0564	0,0513	0,0466	0,0424					
	P(10)	P(11)	P(12)	P(13)	P(14)					
	0,0385	0,0350	0,0319	0,0290	0,0263					
	P(15)	P(16)	P(17)	P(18)	P(19)					
1	0,0240	0,0218	0,0198	0,0180	0,0164					
	P(20)	P(21)	P(22)	P(23)	P(24)					
	0,0149	0,0135	0,0123	0,0112	0,0102					
	P(25)	P(26)	P(27)	P(28)	P(29)					
	0,0092	0,0084	0,0076	0,0069	0,0063					
	P(30)	P(31)	P(32)							
	0,0057	0,0052	0,0047							
λ_2	Probabilidades de Estado					L_2	s_2	W_2	S_2	λ_3
	P(0)	P(1)	P(2)	P(3)	P(4)					
	0,2005	0,3082	0,1894	0,1164	0,0715	2,0715	0,0000	4,1523	0,0000	0,9978
	P(5)	P(6)	P(7)	P(8)	P(9)					
	0,0439	0,0270	0,0166	0,0102	0,0063					
	P(10)	P(11)	P(12)	P(13)	P(14)					
	0,0039	0,0024	0,0015	0,0009	0,0005					
	P(15)	P(16)	P(17)	P(18)	P(19)					
0,9978	0,0003	0,0002	0,0001	0,0001	0,0000					
	P(20)	P(21)	P(22)	P(23)	P(24)					
	0,0000	0,0000	0,0000	0,0000	0,0000					
	P(25)	P(26)	P(27)	P(28)	P(29)					
	0,0000	0,0000	0,0000	0,0000	0,0000					
	P(30)	P(31)	P(32)							
	0,0000	0,0000	0,0000				TOTAL	26,2479	0,0179	

ANEXO 3

SIMULACIONES CORRESPONDIENTES A LOS DIFERENTES MÓDULOS VHDL DISEÑADOS

CONTROLADOR DE ENTRADA

En la Figura A3-1 se aprecia el resultado de la simulación realizada al controlador de entrada diseñado, en donde se puede observar que, al llegar un paquete válido (señal *cell_in*) se realiza el proceso de conmutación en base al bit más significativo de los bits de enrutamiento, en el caso del ejemplo de la figura este bit es cero y, en consecuencia, se envía el paquete por el puerto de salida superior (señal *r_cell_0*) junto a la señal de *wen (write enable)* para el FIFO correspondiente, además se realiza la rotación hacia la izquierda de los bits de enrutamiento.

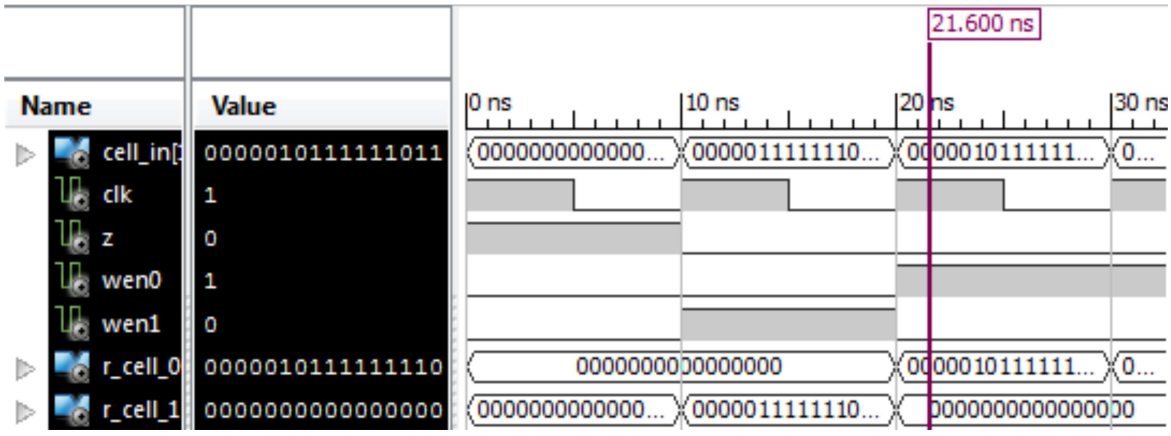


Figura A3-1. Simulación correspondiente al controlador de entrada.

CONTROLADOR DE SALIDA

A continuación se muestra los resultados obtenidos de la simulación hecha al módulo controlador de salida. Como se observa en la Figura A3-2, el comportamiento de este componente es que recibe dos paquetes en sus entradas y envía solo uno de ellos hacia su puerto de salida; además, va tomando de manera alternada los paquetes de sus puertos de entrada, esto se observa en el comportamiento de las señales *ren0 y ren1*, mismas que actúan como señales que indican a los FIFOs que ya se realizó la lectura del paquete correspondiente.

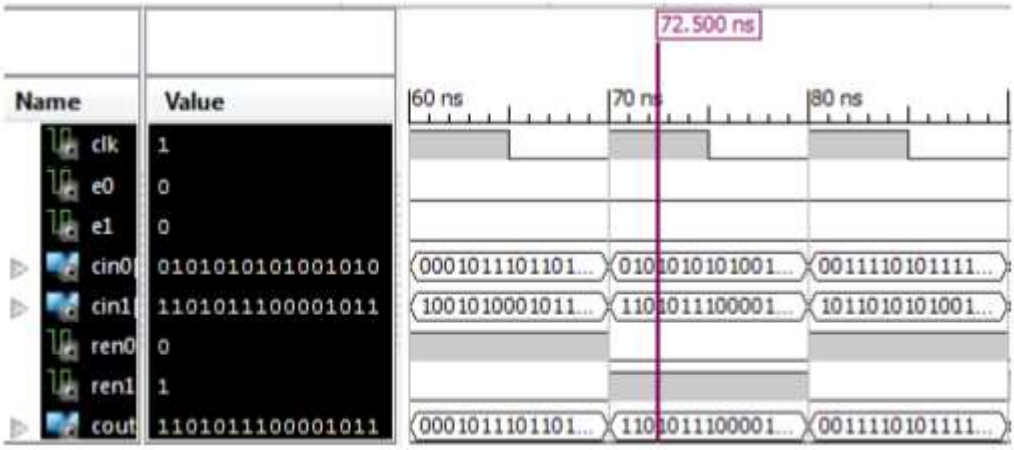


Figura A3-2. Simulación correspondiente al controlador de salida.

BUFFER FIFO

Este módulo corresponde a uno de los buffers FIFO internos existentes en los elementos de conmutación de la red. En la Figura A3-3 se observa el resultado de la simulación realizada al buffer con un tamaño igual a 2. Cabe mencionar que, es precisamente en cada uno de estos buffers, donde se realiza el conteo de paquetes perdidos (señal *cell_loss*), así como también el incremento de los bits de retardo (5 bits más significativos).

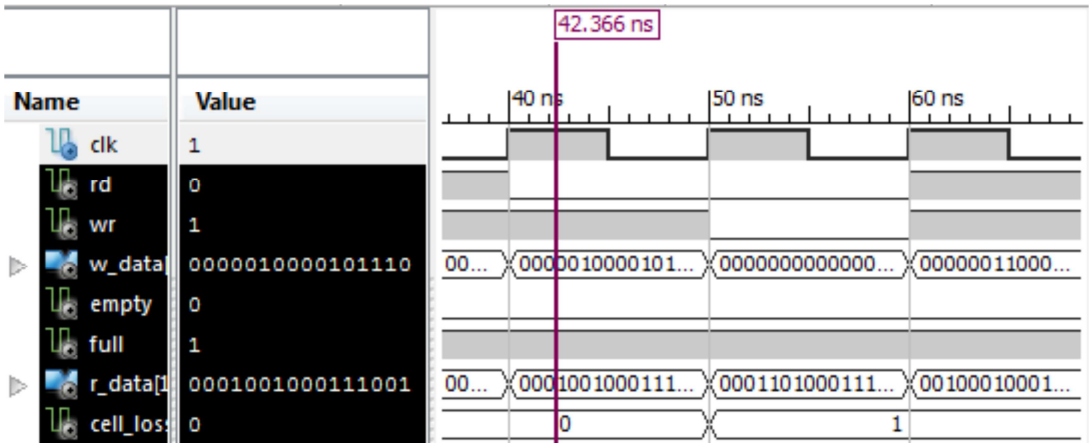


Figura A3-3. Simulación correspondiente al FIFO interno.

ELEMENTO DE CONMUTACIÓN

Uniendo los tres módulos anteriores se obtiene un elemento de conmutación Banyan dual, cuya simulación se observa en la Figura A3-4. En esa figura se observa que existe la señal *cell_loss,* misma que es el resultado de la suma de los paquetes perdidos en los 4 buffers que constituyen cada elemento de conmutación. Se observa también que el elemento de conmutación es capaz de enviar dos paquetes en cada ranura de tiempo y, de esta manera, se mejora el desempeño de la red. Existen también las señales zin y *zout* las cuales indican la presencia de datos válidos en cada una de las entradas y salidas del elemento de conmutación.

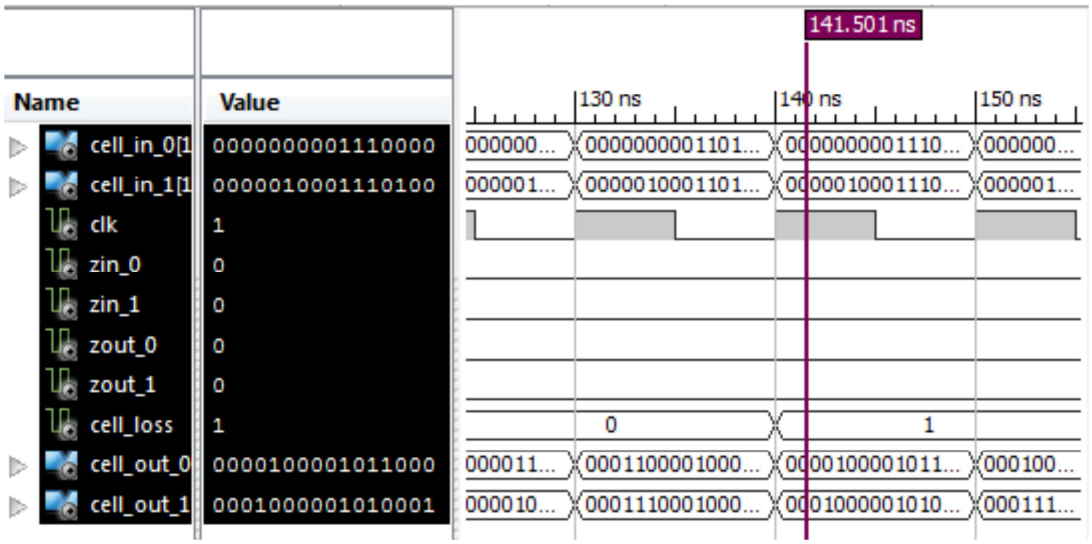


Figura A3-4. Simulación correspondiente al elemento de conmutación Banyan dual.

RED DE CONMUTACIÓN OMEGA

Realizando las conexiones adecuadas entre los puertos de entrada y salida de todos los elementos de conmutación, se obtiene finalmente la topología de la red de conmutación diseñada. En la Figura A3-5 se observa el resultado de la simulación hecha a la red, bajo las siguientes condiciones:

- Máxima carga de tráfico a la entrada.
- Red de tamaño 8x8.
- Tamaño de los buffers internos, B=2.

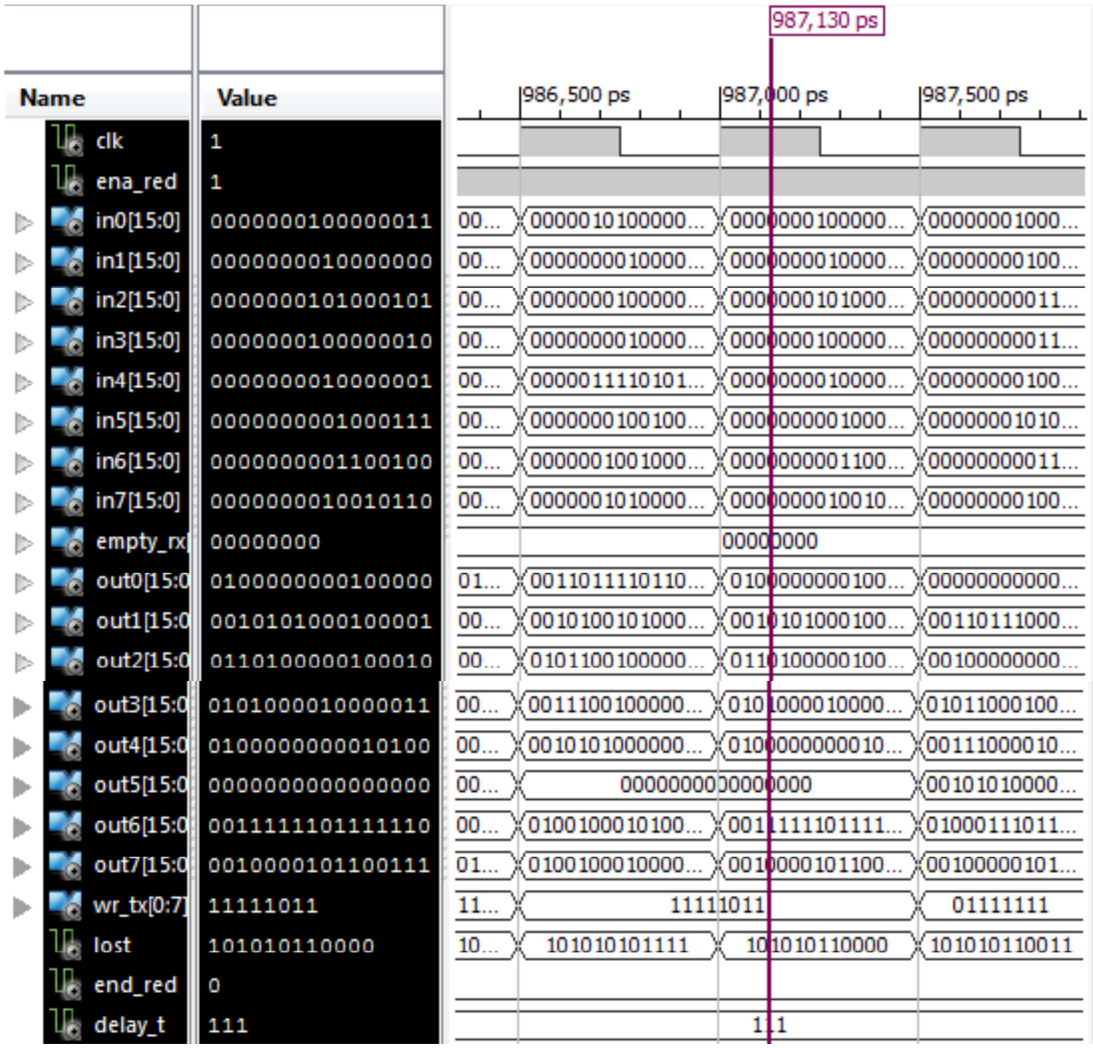


Figura A3-5. Simulación correspondiente a la red de conmutación Omega.

DEMUX 1x8

A continuación (Figura A3-6) se muestra el resultado de la simulación realizada al demux 1x8. La función de este módulo es la de tomar cada uno de los paquetes que se reciben e irlos almacenando de forma ordenada en cada uno de los FIFOs ubicados en las entradas de la red para su posterior conmutación.

En la Figura A3-6 se observa que al llegar un dato (*data_in*) y activarse la señal de habilitación (*ena*) el dato se envía a la salida y a la vez se activa únicamente la bandera correspondiente al puerto de salida adecuado, esta bandera actúa como señal de habilitación de escritura para el buffer respectivo (*wr_fifo_rx*).

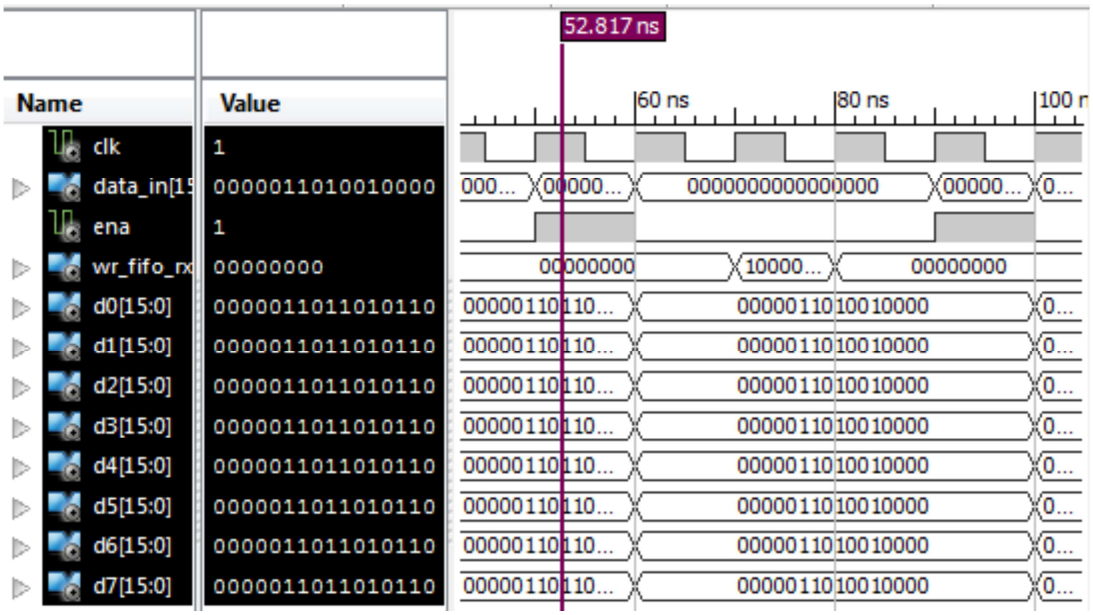


Figura A3-6. Simulación correspondiente al demux 8x1.

MUX 8x1

En la Figura A3-7 se observa la simulación realizada al mux 8x1. Este módulo es el encargado de tomar los paquetes provenientes de los 8 FIFOs a la salida de la red de conmutación e irlos enviando en forma ordenada hacia el módulo de control para su transmisión. Como se observa en la Figura A3-7, al activarse la señal de habilitación (*tx_ready*), este módulo toma el dato correspondiente de sus entradas, lo envía hacia el puerto de salida (*dout*) y a la vez activa la bandera *s_flag*, así como también la respectiva bandera de habilitación de lectura para el FIFO de salida (*rd_flags*) del cual se ha tomado el dato.

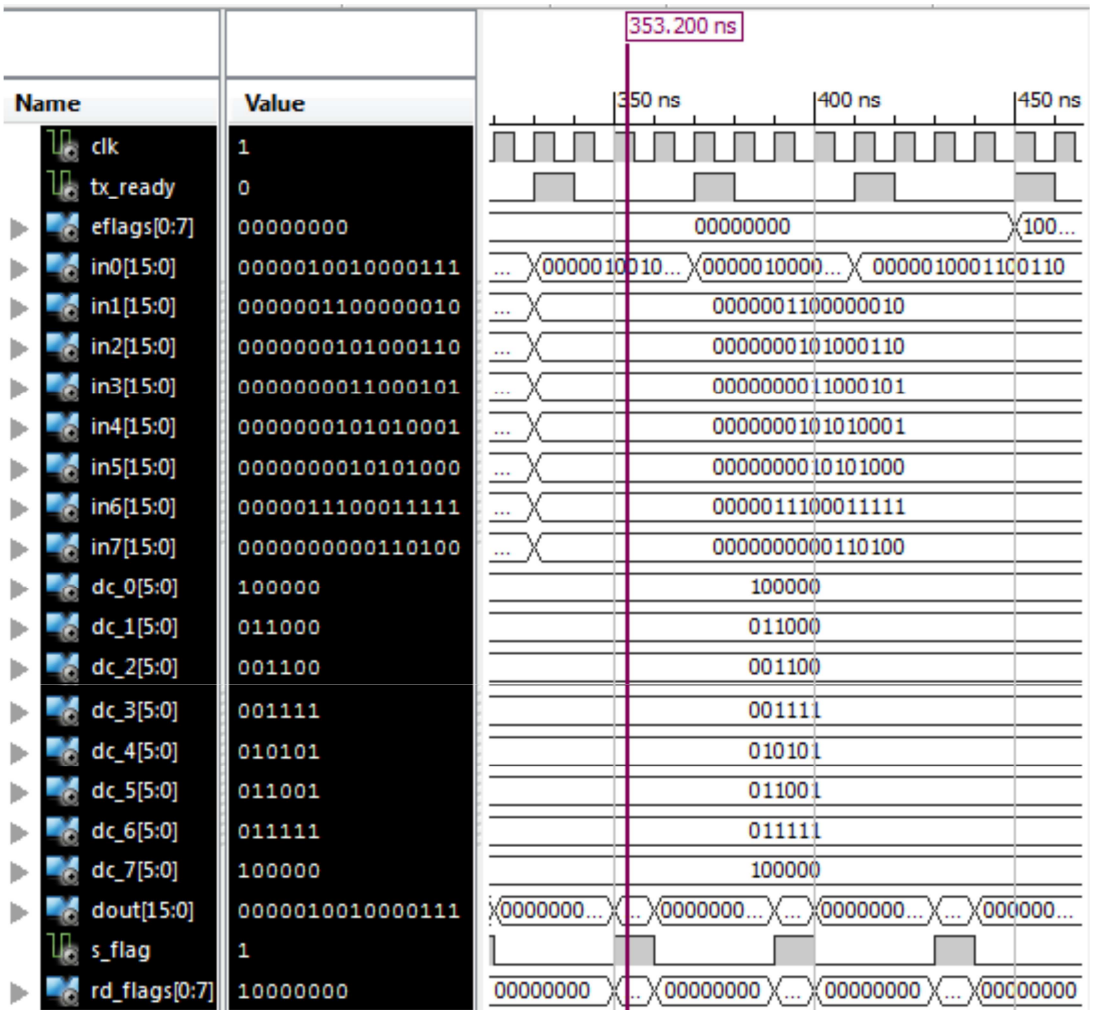


Figura A3-7. Simulación correspondiente al mux 1x8.

ANEXO 4

ANÁLISIS DEL MOMENTO EN EL CUAL EL SISTEMA ALCANZA EL ESTADO DE EQUILIBRIO PARA LOS DIFERENTES TAMAÑOS DE BUFFER

A fin de poder realizar los cálculos de los parámetros de interés de la red, es necesario que el sistema haya alcanzado el estado de equilibrio, mismo que se define como el estado de un sistema en el cual la tasa de ocurrencia de los procesos que producen el poblamiento de un nivel o estado debe ser igual a la tasa de ocurrencia de todos los procesos que producen el despoblamiento de dicho estado.

Como se mencionó en la Sección 3.3.2 de este proyecto, para determinar el momento aproximado en el que la red alcanza el estado de equilibrio se ha realizado un análisis tomando en consideración el parámetro del promedio de paquetes perdidos en la red, descartando paulatinamente las primeras X ranuras de tiempo y graficando los valores restantes a fin de observar su comportamiento y tratar de determinar el momento en el cual los valores se vuelven estables, deduciéndose de esta forma que también la red se halla en estado estable.

A continuación se muestran las gráficas correspondientes a este análisis realizado para cada tamaño de los buffers internos.

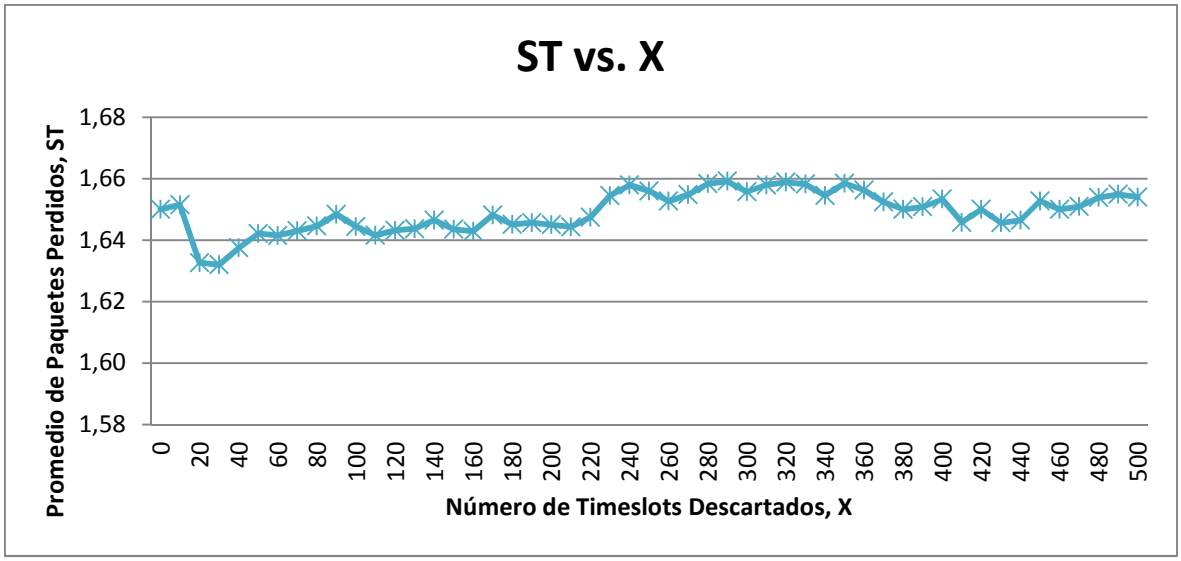


Figura A4-1. Estado de equilibrio para B=2.

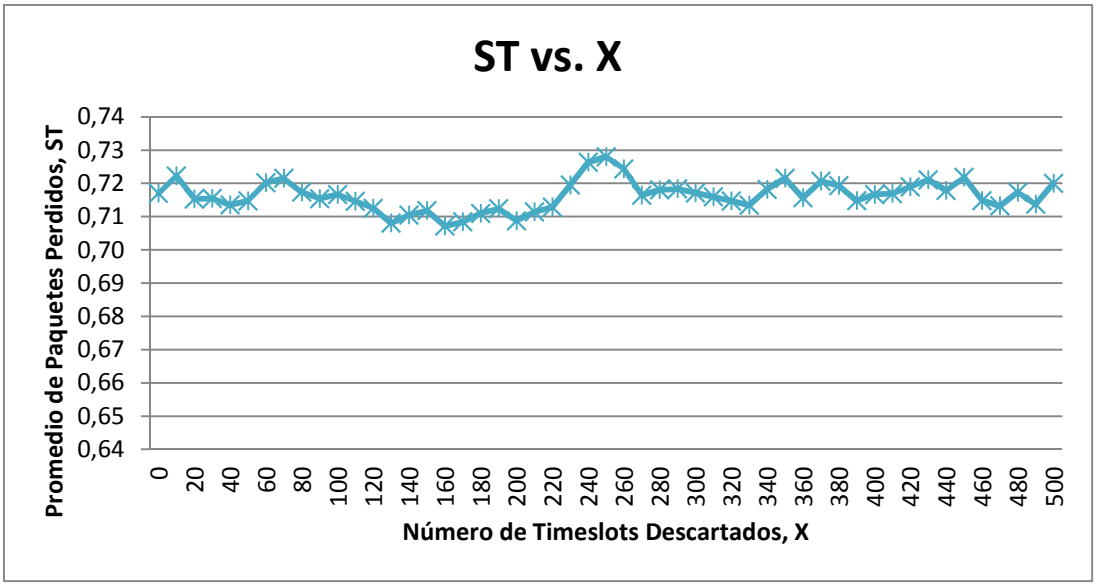


Figura A4-2. Estado de equilibrio para B=4.

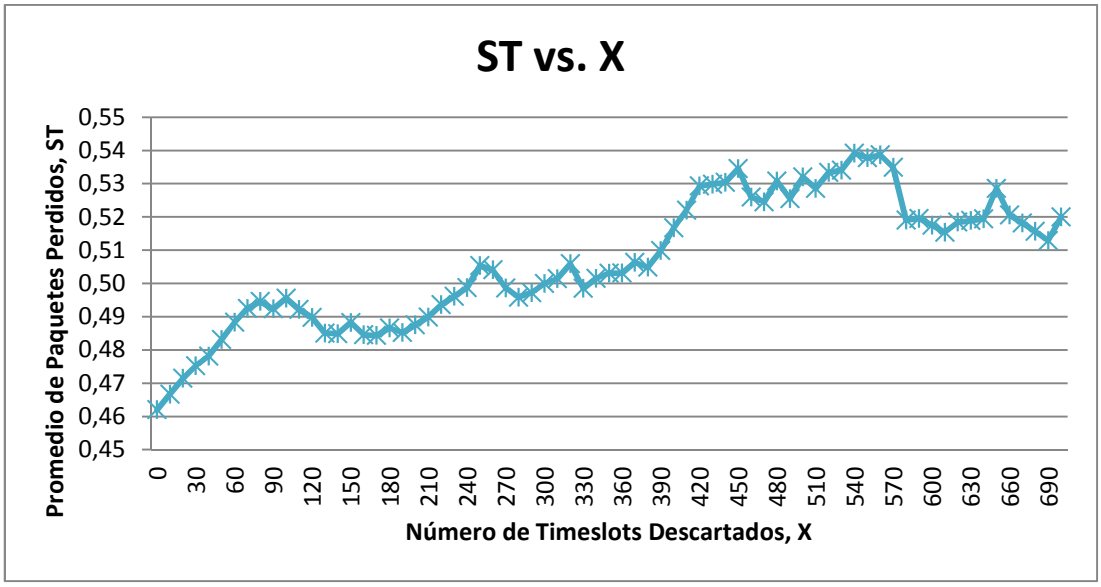


Figura A4-3. Estado de equilibrio para B=8.

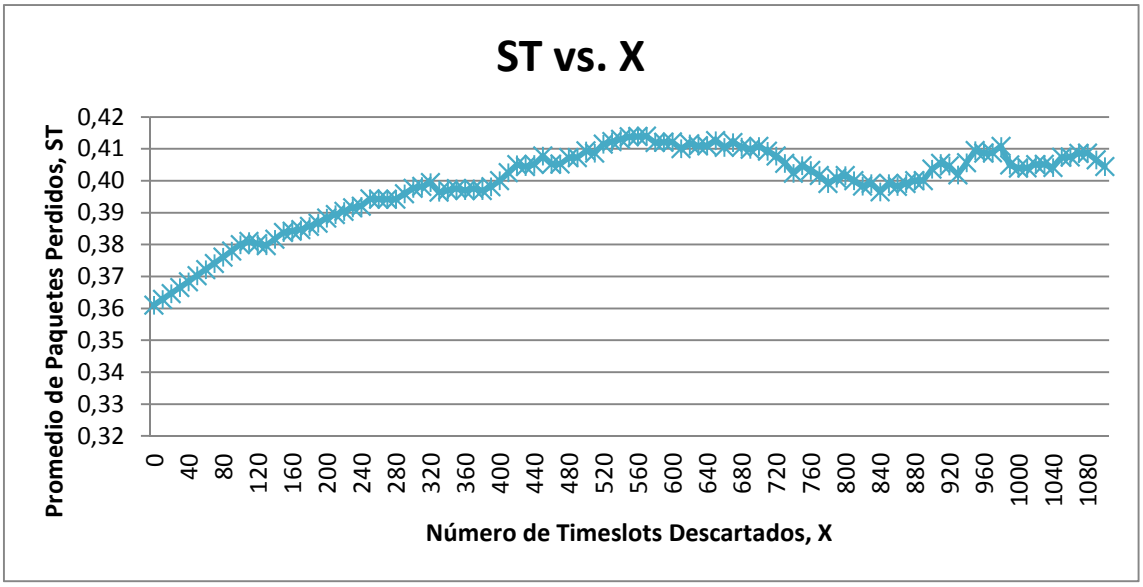


Figura A4-4. Estado de equilibrio para B=16.

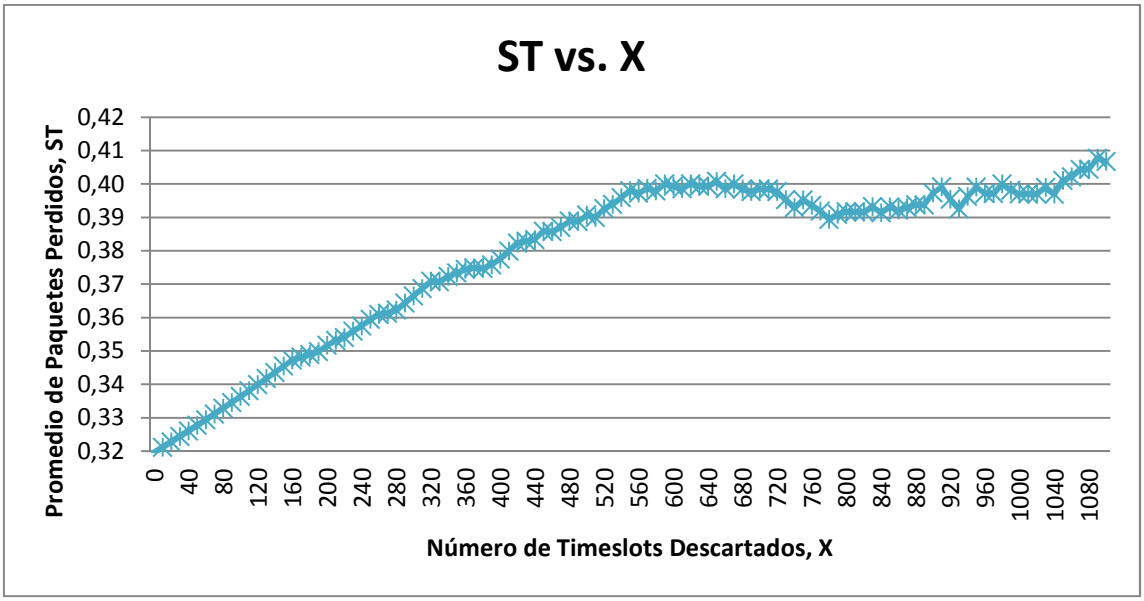


Figura A4-5. Estado de equilibrio para B=32.

En base a la observación de estas gráficas se puede apreciar que para los tamaños de buffer relativamente pequeños (B=2 y B=4) se tiene que los valores son estables casi desde el principio; esto debido a que, al iniciarse la conmutación de los paquetes, éstos empiezan a atravesar la red y a llenar los buffers rápidamente, a medida que todos los buffers de la red tienen paquetes en sus colas se puede considerar que la operación de la red entra en un estado estable; al contrario, si el tamaño de los buffers se incrementa, el proceso de llenado de las colas requiere más tiempoy consecuentementela red tarda más en alcanzar un estado estable de operación.

ANEXO 5

CÓDIGO FUENTE CORRESPONDIENTE A LA INTERFAZ GRÁFICA ELABORADA EN *MATLAB*

```
function Banyan_OpeningFcn(hObject, eventdata, handles, varargin)

%se centra laimagen
scrsw=get(0,'Screensize');
pos_act=get(gcf,'Position');
xr=scrsw(3)-pos_act(3);
xp=round(xr/2);
yr=scrsw(4)-pos_act(4);
yp=round(yr/2);
set(gcf,'Position',[xp yp pos_act(3) pos_act(4)]);
handles.output = hObject;
guidata(hObject, handles);

%imágenes de la presentacion
%Ubicamos ejes en figura
axes('Units','Normalized','Position',[0 0 1 1]);
[x,map]=imread('skyblue.jpg','jpg');
%Representamos imagen en figura, con su mapa de colores
image(x),colormap(map),axis off,hold on
axes(handles.epn_logo)
background = imread('logo_EPN.jpg');
axisoff;
imshow(background);
axes(handles.buho)
background = imread('buho.jpg');
axisoff;
imshow(background);

%CONSTANTES
handles.metricdata.etapas=2; % numero de etapas de la red
handles.metricdata.delay_b=6;  %numero de bits para medir retardo
handles.metricdata.data_b=8;   %numero de bits de datos
handles.metricdata.n_cell=16; % numero de celdas a generar por cada
entrada
guidata(hObject, handles);


% --- Outputs from this function are returned to the command line.
function varargout = Banyan_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
```

```
% --- Executes on button press in gen_data.
function gen_data_Callback(hObject, eventdata, handles)

%CONSTANTES
etapas=handles.metricdata.etapas; % número de etapas de la red
N= 2^etapas; % número de entradas
delay_b=handles.metricdata.delay_b;  %número de bits para medir retardo
data_b=handles.metricdata.data_b   %número de bits de datos HASTA 24
size_cell= delay_b + data_b + etapas; % tamaño de la celda
n_cell= handles.metricdata.n_cell % número de celdas a generar por cada
entrada HASTA 512
n_bytes= size_cell/8; % número de bytes de cada celda

%barra de estado y resultados borrados
set(handles.lost,'string','');
set(handles.delay,'string','');
set(handles.state,'string','');
pause(0.001)

%generar datos de entrada:
%n_cell celdas x N entradas = n_cell*N celdas (16*8=128)
d=randi(2^(data_b)-1,n_cell*N,1);
d_bin=dec2bin(d,data_b);%datos
rb=randi([0,2^etapas-1],n_cell*N,1);
rb_bin=dec2bin(rb,etapas);%rbs
del=zeros(n_cell*N,1);
del_bin=dec2bin(del,delay_b);%delay

%celdas generadas
data=cat(2,del,d,rb);
data_bin=cat(2,del_bin,d_bin,rb_bin);

%envio a Excel de los datos de las 8 primeras entradas
xlswrite('C:\Users\Nelly Bravo\Desktop\Libro1.xlsx',data(1:16,:),'IN
0','A3');
xlswrite('C:\Users\Nelly Bravo\Desktop\Libro1.xlsx',data(17:32,:),'IN
1','A3');
xlswrite('C:\Users\Nelly Bravo\Desktop\Libro1.xlsx',data(33:48,:),'IN
2','A3');
xlswrite('C:\Users\Nelly Bravo\Desktop\Libro1.xlsx',data(49:64,:),'IN
3','A3');
xlswrite('C:\Users\Nelly Bravo\Desktop\Libro1.xlsx',data(65:80,:),'IN
4','A3');
xlswrite('C:\Users\Nelly Bravo\Desktop\Libro1.xlsx',data(81:96,:),'IN
5','A3');
xlswrite('C:\Users\Nelly Bravo\Desktop\Libro1.xlsx',data(97:112,:),'IN
6','A3');
xlswrite('C:\Users\Nelly Bravo\Desktop\Libro1.xlsx',data(113:128,:),'IN
7','A3');

%CONVERSION A SERIAL
%n_cell*N*n_bytes = 16*8*2=256 bytes a tx
data_tx_bin= zeros (size(data_bin));
data_tx= zeros(n_cell*N,n_bytes);

for i=1:n_bytes
   data_tx_bin= cat(2,data_bin(:,(i-1)*8+1:i*8),data_tx_bin(:,1:(n_bytes-
1)*8));
```

```
   data_tx= cat(2,bin2dec(data_bin(:,(i-1)*8+1:i*8)),data_tx(:,1:n_bytes-
1));
end

data_int=uint8(data_tx);

data_vector=zeros(1,n_cell*N*n_bytes);
a=1;
for i=1:n_cell*N
for j=1:n_bytes
        data_vector(a)=data_int(i,j);
        a=a+1;
end
end
handles.metricdata.data_vector=data_vector;
set(handles.tx_data,'enable','on');
set(handles.menu_tx_data,'enable','on');

%barra de estado actualizada
set(handles.state,'string', ' Datos generados en: Desktop\Libro1.xlsx');
pause(0.001)
guidata(hObject,handles);


% --- Executes on button press in test_com.
function test_com_Callback(hObject, eventdata, handles)

%Barra de estado y resultados borrados
set(handles.lost,'string','');
set(handles.delay,'string','');
set(handles.state,'string','Probando puerto...');
pause(0.001)

%CONFIGURACIÓN PUERTO SERIAL
s=serial('COM17');
s.BaudRate= 19200;
s.DataBits= 8;
s.Parity= 'None';
s.StopBits= 1;
s.Timeout=10;

%PROBAR PUERTO
fopen(s);
fwrite(s,'T','uint8');
test=fread(s,1,'uint8');
fclose(s);
delete(s);
clears;
l=length(test);
%comparación del byte recibido
if l==0
set(handles.state,'string', 'Conexion fallida');
else
if test==85 %caracter 'U'
set(handles.state,'string', 'Conexion establecida');
else
set(handles.state,'string', 'Conexion fallida');
end
end
```

```
guidata(hObject,handles);


% --- Executes on button press in tx_data.
function tx_data_Callback(hObject, eventdata, handles)

%CONSTANTES
etapas=handles.metricdata.etapas; % numero de etapas de la red
N= 2^etapas; % numero de entradas
delay_b=handles.metricdata.delay_b;  %numero de bits para medir retardo
data_b=handles.metricdata.data_b;   %numero de bits de datos HASTA 24
size_cell= delay_b + data_b + etapas; % tamaño de la celda
n_cell= handles.metricdata.n_cell; % numero de celdas a generar por cada
entrada HASTA 512
n_bytes= size_cell/8; % numero de bytes de cada celda

%Barras de estado y resultados
set(handles.lost,'string','');
set(handles.delay,'string','');
set(handles.state,'string', 'Transmitiendo...');
pause(0.001)

%Datos a transmitir
celdas=handles.metricdata.data_vector;

%CONFIGURACIÓN PUERTO SERIAL
s=serial('COM17');
s.BaudRate= 19200;
s.DataBits= 8;
s.Parity= 'None';
s.StopBits= 1;
s.Timeout=10;

%envio del caracter "J"
fopen(s);
fwrite(s,'J','uint8');

%ENVIO DE LOS DATOS
fwrite(s,celdas, 'uint8');
set(handles.state,'string', 'Transmision finalizada');
pause(0.001)

%RECEPCIÓN DE LOS DATA_COUNT
if n_cell > 128
dc=fread(s,2*N,'uint8');
L=length(dc);
else
dcount=fread(s,N,'uint8');
   L=length(dcount);
end

if L==0
set(handles.state,'string', 'Conexion fallida');
fclose(s);
delete(s);
clears;
end
```

```
%reensamblaje de bytes correspondientes a data_count
if n_cell > 128
    dc_bin=dec2bin(dc,8);
    x=1;
    dcount_bin=dec2bin(zeros(N,1),16);
for i=1:N
        dcount_bin(i,:)=cat(2,dc_bin(x+1,:),dc_bin(x,:));
        x=x+2;
end
dcount=bin2dec(dcount_bin);
end

%CALCULO DEL NUMERO DE CELDAS PERDIDAS
c_lost= n_cell*N - sum(dc);
Lst=num2str(c_lost);
set(handles.lost,'string', Lst);

%DATA_COUNT: MULTIPLICACIÓN POR n_bytes
T=n_bytes*dc;
Total_bytes=sum(T);
%barra de estado actualizada
set(handles.state,'string', 'Recibiendo datos...');

%ENVIO DE CARACTER R INDICANDO 'READY' PARA RECIBIR CELDAS
fwrite(s,'R','uint8');
%pause(0.001)

%RECEPCIÓN DE LAS CELDAS CONMUTADAS
byte_rx=fread(s,Total_bytes,'uint8');
%msb,.., lsb
%fread-> vector columna
%BORRADO DEL OBJETO SERIAL
fclose(s);
delete(s);
clears;

%Organización en vector columna de celdas
byte_rx_bin= dec2bin(byte_rx,8);

Total_cell=sum(dc);
d=1;

for i=1:Total_cell
    c=1;
for j=1:n_bytes
for k=1:8
            cell_rx_bin(i,c)=byte_rx_bin(d,k);
            c=c+1;
end
        d=d+1;
end
end

delay_rx= cell_rx_bin(:,1:delay_b);
data_rx= cell_rx_bin(:,delay_b+1:size_cell-etapas);
rbs_rx= cell_rx_bin(:,size_cell-etapas+1:size_cell);
```

```
% Calculo de retardo promedio
del_dec= bin2dec(delay_rx);

av_delay= sum(del_dec)/length(del_dec);
t_del=num2str(av_delay);
set(handles.delay,'string', t_del);

%Repartición a las 8 primeras salidas
dat_dec= bin2dec(data_rx);
rb_dec= bin2dec(rbs_rx);

ind(1)=dc(1);
for i=2:N
ind(i)=ind(i-1)+dc(i);
end

B= cat(2,del_dec, dat_dec, rb_dec);

% Envio de datos a Excel
xlswrite('C:\Users\Nelly Bravo\Desktop\Libro1.xlsx',B(1:ind(1),:),'IN
0','D3');
xlswrite('C:\Users\Nelly
Bravo\Desktop\Libro1.xlsx',B(ind(1)+1:ind(2),:),'IN 1','D3');
xlswrite('C:\Users\Nelly
Bravo\Desktop\Libro1.xlsx',B(ind(2)+1:ind(3),:),'IN 2','D3');
xlswrite('C:\Users\Nelly
Bravo\Desktop\Libro1.xlsx',B(ind(3)+1:ind(4),:),'IN 3','D3');
xlswrite('C:\Users\Nelly
Bravo\Desktop\Libro1.xlsx',B(ind(4)+1:ind(5),:),'IN 4','D3');
xlswrite('C:\Users\Nelly
Bravo\Desktop\Libro1.xlsx',B(ind(5)+1:ind(6),:),'IN 5','D3');
xlswrite('C:\Users\Nelly
Bravo\Desktop\Libro1.xlsx',B(ind(6)+1:ind(7),:),'IN 6','D3');
xlswrite('C:\Users\Nelly
Bravo\Desktop\Libro1.xlsx',B(ind(7)+1:ind(8),:),'IN 7','D3');

%barra de estado actualizada
set(handles.state,'string', 'Datos recibidos en: Desktop\Libro1.xlsx');
pause(0.001)

guidata(hObject,handles);


% --MENU------------------------------------------------------------------
function edit_param_Callback(hObject, eventdata, handles)
param% Llamada a la función param
uiwait

global dbits n_pkts% constants globales
handles.metricdata.data_b=dbits;
handles.metricdata.n_cell=n_pkts;

guidata(hObject,handles);
```

```
% --------------------------------------------------------------------
% -VENTANA PARA EDICIÓN DE PARÁMETROS---------------------------------
% --- Executes on button press in aceptar.

% --- Executes on selection change in val_pkt.
function val_pkt_Callback(hObject, eventdata, handles)

% Hints: contents = cellstr(get(hObject,'String')) returns val_pkt
contents as cell array
cont_db=num2str([8;16;24]);
set(handles.val_dbits,'String',cont_db);
v=get(handles.val_pkt,'Value');
switch v
case 1  %16
        handles.metricdata.npkts = 16;
case 2  %32
        handles.metricdata.npkts = 32;
case 3  %64
        handles.metricdata.npkts = 64;
case 4  %128
        handles.metricdata.npkts = 128;
case 5  %256
        handles.metricdata.npkts = 256;
otherwise%512
        handles.metricdata.npkts = 512;
set(handles.val_dbits,'String','8');
end
guidata(hObject,handles);

% --- Executes on selection change in val_dbits.
function val_dbits_Callback(hObject, eventdata, handles)

cont_pkt=num2str([16;32;64;128;256;512]);
set(handles.val_pkt,'String',cont_pkt);
v1=get(handles.val_dbits,'Value');
switch v1
case 1  %8
        handles.metricdata.ndbits = 8;
case 2  %16
        handles.metricdata.ndbits = 16;
otherwise%24
        handles.metricdata.ndbits = 24;
        cont_1=num2str([16;32;64;128;256]);
set(handles.val_pkt,'String',cont_1);
end
guidata(hObject,handles);

function aceptar_Callback(hObject, eventdata, handles)
global dbits n_pkts
dbits=handles.metricdata.ndbits;
    n_pkts=handles.metricdata.npkts;

closereq
```

ANEXO 6

SPARTAN-3AN FPGA FAMILY: DATA SHEET

(El contenido de este anexo se presenta en la carpeta adjunta)

ANEXO 7

CÓDIGO VHDL DE LOS MÓDULOS DISEÑADOS

En la carpeta adjunta se presentan los siguientes archivos, los cuales corresponden al código VHDL de todos los módulos diseñados en este proyecto:

- **av_delay:** Módulo para cálculo del retardo promedio de los paquetes.
- **control:** Módulo de control para todo el sistema.
- **demux_1x8:** Módulo que reparte los paquetes a cada uno de los FIFOs ubicados en las entradas de la red.
- **fifo:** Buffer FIFO interno al elemento de conmutación.
- **flag_buff:** Buffer de capacidad 1 byte para almacenar datos desde y hacia el sistema UART.
- **inp_c:** Controlador de entrada.
- **mod_m_counter:** Generador de tasa de muestreo.
- **mux_8x1:** Módulo que toma los datos de los FIFOs de salida y los envía para su transmisión al PC.
- **mux_dc:** Módulo que toma los valores de los data_count de los FIFOs de salida y los envía para su transmisión al PC.
- **my_pack:** Paquete creado con constantes globales del sistema.
- **omega:** Descripción de la red Banyan con topología Omega.
- **out_c:** Controlador de salida.
- **SE:** Elemento de conmutación Banyan dual.
- **tesis2_ucf:** Archivo de restricciones de usuario para el sistema diseñado.
- **top_red:** Descripción estructural de todo el sistema diseñado.
- **uart:** Módulo correspondiente al sistema UART completo.
- **uart_rx:** Subsistema de recepción del UART.
- **uart_tx:** Subsistema de transmisión del UART.