Instituto Tecnológico de Costa Rica.

Escuela de Ingeniería Electrónica.

Diseño y Evaluación de Arquitecturas de Enrutador Basado en Tablas de Enrutamiento Estáticas Orientadas al uso en "Network on Chip".

Informe de Proyecto de Graduación para optar por el título de Ingeniero en Electrónica con el grado académico de licenciatura.

Luis Martín Barquero Retana.

Cartago, Noviembre de 2018.

INSTITUTO TECNOLÓGICO DE COSTA RICA

ESCUELA DE INGENIERÍA ELECTRÓNICA

PROYECTO DE GRADUACIÓN

ACTA DE APROBACIÓN

Defensa de Proyecto de Graduación
Requisito para optar por el título de Ingeniero en Electrónica
Grado Académico de Licenciatura
Instituto Tecnológico de Costa Rica

El Tribunal Evaluador aprueba la defensa del proyecto de graduación denominado "Diseño y Evaluación de Arquitecturas de Enrutador Basado en Tablas de Enrutamiento Estáticas Orientadas al uso en Network on Chip", realizado por el señor Luis Martín Barquero Retana y, hace constar que cumple con las normas establecidas por la Escuela de Ingeniería Electrónica del Instituto Tecnológico de Costa Rica.

Miembros del Tribunal Evaluador

Dr. Renato Rimolo Donadío

Profesor lector

M.Sc. Sergio Arriola Valverde

Profesor lector

M.Sc. Ronny García Ramírez

Profesor asesor

Cartago, 27 de noviembre de 2018

Declaración de Autenticidad

Declaro que el presente Proyecto de Graduación ha sido realizado. en su totalidad, por mi persona, utilizando y aplicando literatura referente al tema e introduciendo conocimientos propios.

En los casos en que he utilizado material bibliográfico. he procedido a indicar las fuentes mediante citas.

En consecuencia. asumo la responsabilidad total por el trabajo de graduación realizado y por el contenido del correspondiente informe final.

Luis Martín Barquero Retana.

Cédula: 304850577

1 Resumen

En este trabajo se realizó la implementación de una arquitectura parametrizada de enrutador basado en tablas estáticas para su uso en "Network on chip"; Para ello se diseño un bus paralelo de entrada de datos igualmente paralela y con arbitraje central, el cual posteriormente fue descrito en lenguaje System Verilog, este bus es sintetizado lógica y físicamente para comprobar y comparar los resultados de consumo de potencia, uso de área y latencia de datos, con otros dos buses presentes en la biblioteca del proyecto. Una vez realizadas las pruebas funcionales sobre el bus, este se utiliza en conjunto con decodificadores, y las memorias de tipo FIFO de Flip-Flops de la biblioteca del proyecto para desarrollar la arquitectura del enrutador de datos. Finalmente se describe en lenguaje System Verilog un generador de interconexiones de tipo malla bidimensional, esta arquitectura mantiene flexibilidad en los parámetros de: ancho y alto de la malla, ancho de palabra de datos, bits de direcciónamiento de dispositivo de destino y tamaño de FIFOs.

Palabras Clave: Bus Paralelo, Enrutador, Tablas estáticas, NoC, Network on Chip, Redes en chip.

2 Abstract

This work presents an implementation of a parameterized router architecture based on static tables for its use on Network on Chip; For this, a parallel bus of parallel data input with central arbiter was designed, this was later described on System Verilog language, this bus was synthesized logically and physically to check and compare the results of power consumption, area usage and data latency, against other two buses of the project library. Once the functional test were done on the bus, it is used along with decoders, and the FIFOs made of Flip Flops present in the project library to develop a data router architecture. Finally an interconnection generator of type bidimensional net is described in System Verilog language, this architecture keeps flexibility in parameters of: width and height of the net, data word width, destiny devise address bits and FIFOs depth.

Keywords: Parallel Bus, Router, Static Tables, NoC, Network on Chip.

Contenidos

Lista de Figuras

Lista de Tablas

3 Introducción

3.1 Entorno de la Tesis

El Laboratorio de Diseño de Circuitos Integrados, DCILab, forma parte de la Escuela de Ingeniería Electrónica del Instituto Tecnológico de Costa Rica, en este, los estudiantes desarrollan proyectos de investigación bajo la tutela de los profesores de la escuela.

En el desarrollo de circuitos integrados, la comunicación entre los distintos módulos es un punto clave para el óptimo funcionamiento del sistema, es por esto que en el DCILab se busca tener a disposición un repertorio de estructuras de interconexión, el diseño propuesto añade a este repertorio un bus paralelo con árbitro central, una implementación de este en un enrutador de datos y finalmente una NoC de tipo malla bidimensional.

Las redes en chip brindan una gran flexibilidad para cumplir con los requisitos de potencia, ancho de banda de datos y latencia que se necesitan entre los distintos módulos que conforman la arquitectura de un dispositivo, esto se debe a que hay completo control sobre la ruta que siguen los datos, además también pueden cumplir un rol fundamental al realizar la disposición de los módulos en el silicio.

3.2 Objetivos

3.2.1 Objetivo General

Desarrollar en lenguaje System Verilog una arquitectura de enrutadores estáticos parametrizados y con bus interno, posteriormente realizar una evaluación comparativa entre la arquitectura desarrollada y los buses existentes en la biblioteca el proyecto, comparando el consumo de potencia, uso de área en chip y latencia de datos, usando versiones físicas sintetizadas.

3.2.2 Objetivos específicos

1. Desarrollar a nivel de bloques una arquitectura de bus paralelo con árbitro central.

2. Describir en lenguaje System Verilog la arquitectura de bus paralelo con árbitro central.

3. Realizar la síntesis lógica y física del bus desarrollado junto con un bus paralelo sin árbitro y otro serial con árbitro desarrollados que se encuentra en la biblioteca de dispositivos del proyecto.

4. Diseñar una arquitectura de enrutador a partir del bus desarrollado.

5. Describir en lenguaje System Verilog un generador de malla bidimensional para las tres arquitecturas de bus.

4 Marco Teórico

Generalmente para realizar la conexión entre dos dispositivos estos se conectan directamente, pero conforme la cantidad de dispositivos aumenta esto se convierte en una red de conexiones punto a punto Figura 1, si bien este tipo de comunicación es intuitiva y transmite los datos a alta velocidad tiene enormes problemas en el uso de área (crece según $n^2\sqrt{n}$ [7]) y consumo de potencia ya que cada nodo agregado a la red debe conectarse a cada uno de los nodos existentes generando 2(n-1) puertos de conexión (entrada y salida) en cada nodo y 2n(n-1) conexiones totales y sí se agrega un nuevo nodo debe agregarse dos puertos a cada uno de los nodos existentes.

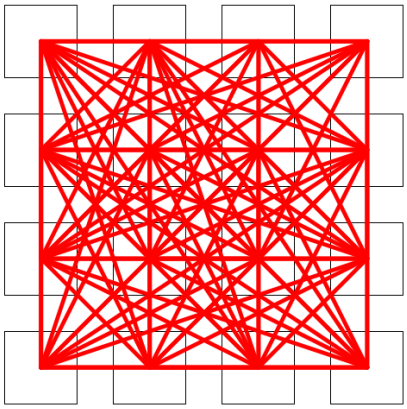

Figura. 1: Conexiones punto a punto entre 16 nodos

Para reducir la cantidad de líneas necesarias en la interconexión, se cambia el esquema de medios de conexión dedicada a medios de conexión compartida como lo son los buses de datos. Los buses de datos utilizan un solo medio para todas las comunicaciones, de esta manera cambian el paralelismo absoluto en transmisión de datos que tiene una red de conexión punto a punto por transmisión secuencial de mensajes, esto tiene la ventaja de disminuir de 2n(n-1) medios a solo 1 a cambio de dis-

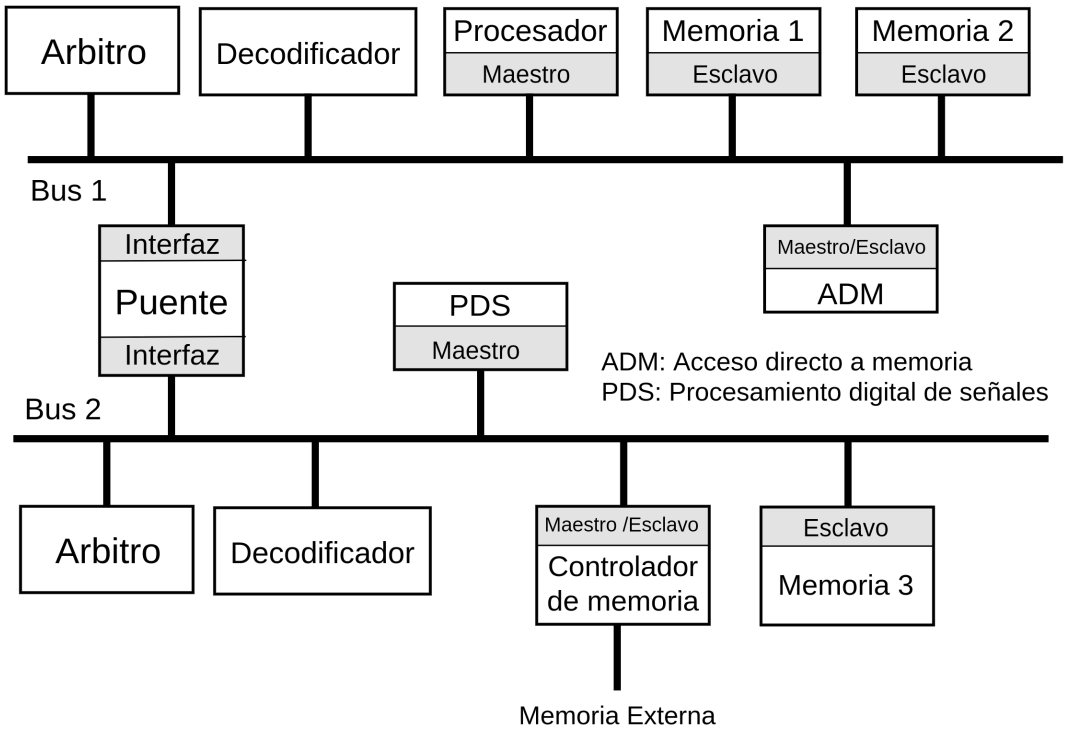

Figura. 2: Interfaz de bus distribuido, Tomado de [1]

tribuir las transferencias en el tiempo, para realizar esto se debe añadir una estrategia de arbitraje (central o distribuido) que indique que dispositivo tiene el derecho de uso del medio y que dispositivo es el destinatario del mensaje.

Con el avance de la tecnología surgen nuevas necesidades de comunicación ya que en un mismo sistema se pueden tener varias unidades con grandes anchos de banda lo cual aumenta la cantidad de transferencias de información que debe manejar el bus. Para solucionar esto se podría aumentar la frecuencia de reloj del bus pero esto no es una solución viable ya que esta limitada por las características de velocidad del hardware y consumo de potencia. Como alternativa se prefiere realizar una arquitectura de bus distribuido en la cual se separan dispositivos según sus necesidades de comunicación, por ejemplo se muestra en la Figura 2 cómo sería posible separar el procesador de la unidad de procesamiento

digital de señales(DSP) del procesador de propósito general del sistema, de esta manera ambas unidades pueden comunicarse con una memoria, realizar operaciones sobre datos para posteriormente guardarlos, en este caso cuando el DSP termina de procesar datos se lo hace saber al procesador y este lee los datos a través de un puente entre los buses que además puede servir para traducir protocolos de comunicación y diferencias de velocidad.

A pesar de la separación de funcionalidades de un bus distribuido esto no es suficiente para cumplir con las nuevas necesidades de procesamiento, paralelismo y velocidad de transferencia de datos que pueden tener algunos sistemas, así que se debe buscar una nueva arquitectura de comunicación para múltiples dispositivos que mejore el ancho de banda de transmisión de datos. Tomando en cuenta que teóricamente la mayor velocidad de transferencia de datos posible la podemos alcanzar usando una red de conexión punto a punto y que el menor uso de líneas de datos se puede alcanzar usando una arquitectura de interconexión tipo bus, se puede pensar en la unión de ambos conceptos, al compartir las líneas de los buses entre los diferentes dispositivos usando paquetes de información, de forma similar a como se hace en las redes de computadoras estándar. de esta manera se introduce el concepto de Networks on chip(NoC).

4.1 Networks on Chip

Las redes en chip surgen como una evolución de los buses distribuidos en estas se busca que todos los dispositivos de la red puedan comunicarse entre si a través de enrutadores, para lograr esto los dispositivos deben adjuntar a la trama de datos un identificador de destino a partir del cual los enrutadores de la red calculan el camino que debe seguir la trama

para llegar efectivamente al destino.

El paradigma que presentan las redes en chip basadas en enrutadores nos da una metodología generalizada para diseñar redes de conexión entre dispositivos en las cuales se puede optimizar el uso de recursos y a la vez seguir teniendo una estructura flexible capaz de entablar comunicaciones de manera concurrente; por ejemplo realizar una transferencia de datos de alta velocidad entre dos dispositivos cercanos y a la vez poder cambiar el direcciónamiento para comunicar otros dos dispositivos que estén físicamente alejados.

4.1.1 Enrutadores

La función básica de un enrutador es direcciónar el flujo de datos a través de sus puertos basándose en algún criterio pre-establecido, comúnmente este criterio es el destino que se encuentra en la trama de datos, de esta manera el enrutador lee el destino y basandose en su algoritmo de direcciónamiento decide por que puerto deben salir los datos.

Normalmente un enrutador esta compuesto por puertos de entrada, puertos de salida, puertos locales (para acceder dispositivos de salida de datos de la red), matriz de direcciónamiento y lógica de direcciónamiento (arbitraje) [8].

Conceptos Básicos:

Flit :
: Máxima cantidad de bits que se puede enviar por un enlace entre enrutadores en un ciclo de reloj.

Lógica de enrutamiento :
: Algoritmo que dicta cómo los enrutadores direcciónan los paquetes de datos desde la fuente hasta el destino, dicho algoritmo debe

resolver problemas relacionados con el flujo de datos en la red, por ejemplo: “Deadlock”,“Livelock” y “Starvation”.

Deadlock :

Cuando un paquete de datos no llega a su destino porque los recursos de la red están ocupados y esperando entre sí a liberar recursos para enviar datos, esto ocurre cuando los caminos seguidos por un paquete de datos se cruzan de una forma cíclica[8].

Livelock :

El paquete de datos no puede llegar a su destino por que se encuentra en un camino cíclico.

Starvation :

Un paquete de datos no llega a su destino porque algún enrutador no le concede acceso (mientras se lo concede a otro).

Interface de Red :

Lógica que traduce protocolos de comunicación entre enrutadores y dispositivos.

Ancho de banda de transferencia(“Bandwidth”) :

Tasa máxima de propagación de datos una vez que el paquete entra a la red se mide en bits por segundo(bps)[8].

Flujo de datos(“Throughput”) :

Máxima cantidad de información entregada por unidad de tiempo se mide en mensajes por segundo o mensajes por ciclo de reloj[8].

Latencia :

Tiempo transcurrido entre el inicio de la trasmisión de un mensaje y su completa recepción en el nodo de destino.

En la Figura 3 se presenta una arquitectura de enrutadores genérica, esta consta conceptualmente de una lógica de transmisión que se encarga

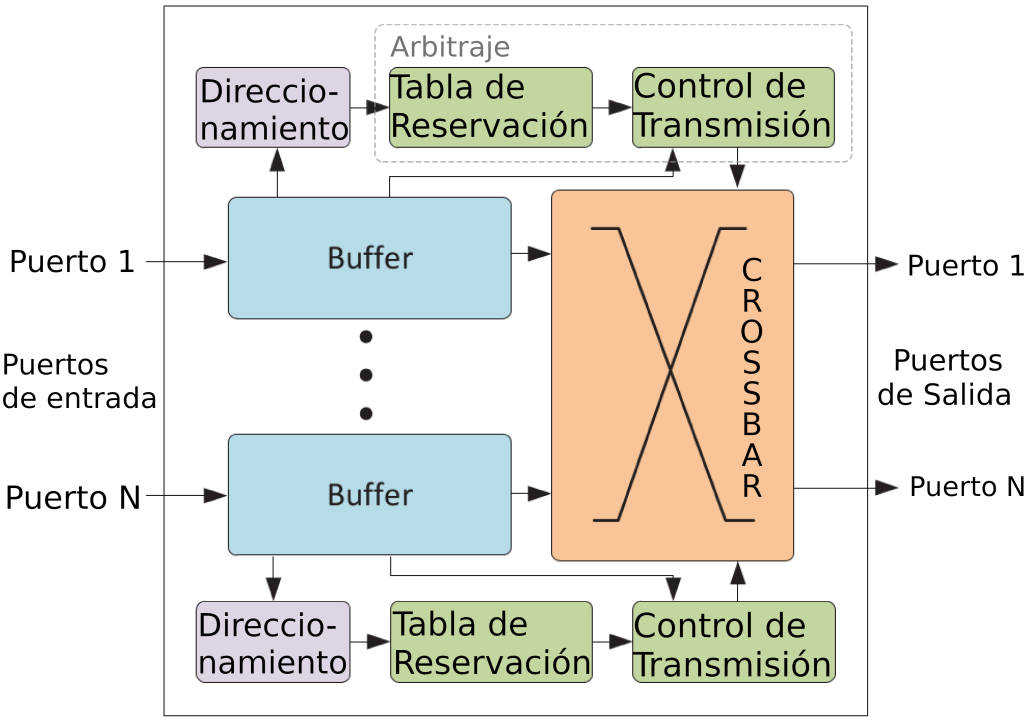

Figura. 3: Arquitectura genérica de enrutador. Tomado de [2]

de direcciónar el mensaje entrante hacia la salida correspondiente, el arbitraje que se necesita cuando varias entradas requieren la misma salida es establecido en las tablas de reservación, finalmente se encuentran los Buffers de entrada de datos, que se encargan de retener información hasta que esta pueda ser procesada y enviada.

En la figura 4 se ilustra una arquitectura de bus que implementa canales virtuales y "lookahead routing logic" que calcula el puerto de salida del siguiente enrutador y lo adjunta a la trama; en esta arquitectura además de la lógica de transmisión ("Switch alocation" en la figura 4) se modifica la lógica de reservación para implementar canales virtuales, estos tienen la función de almacenar los datos entrantes de un mismo paquete en caso de que la salida correspondiente se encuentra recibiendo datos de un tercer puerto.

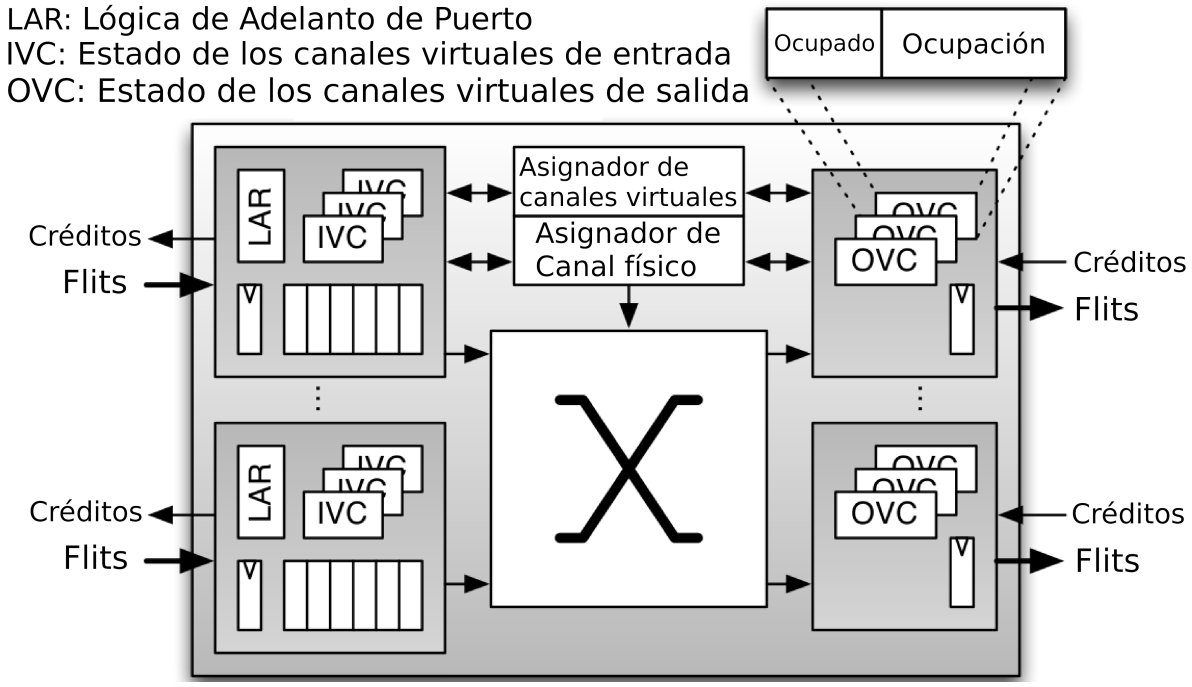

Figura. 4: Arquitectura generica de enrutador con canales virtuales. Tomado de [3]

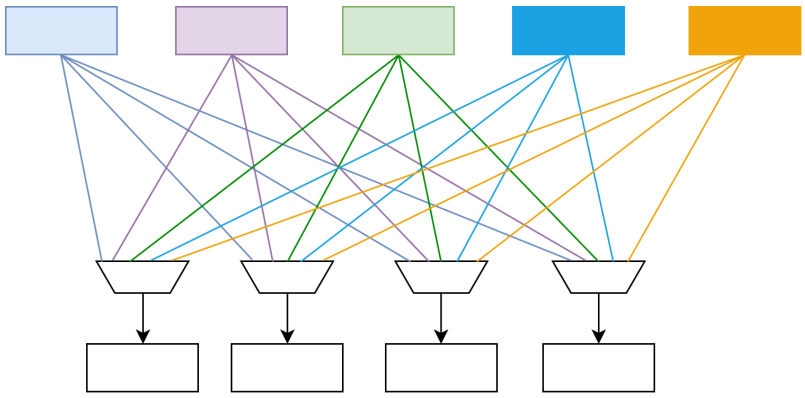

Figura. 5: Arquitectura de Crossbar (solo se ilustra para 4 de 9 puertos)

Una de los módulos principales en un enrutador, es la lógica de enrutamiento, comúnmente, en la literatura [3], [4], [8], [9] y otros, se muestra al "Crossbar" como la arquitectura por defecto para el núcleo de direcciónamiento del enrutador, el diseño de esta implementación consiste en colocar multiplexores en cada uno de los puertos, cuyas entradas

sean todas las salidas de cada uno de los demás puertos esto se ilustra parcialmente en la figura 5, a parte de esto en cada puerto se requiere una lógica que designe que entrada se selecciona en el multiplexor.

4.1.2 Topologías de Red

Topología de red se refiere a la distribución de las conexiones entre enrutadores y entre estos y los dispositivos, la topología de la red define el largo de los cables, el número de puertos de los enrutadores, las posibles lógicas de enrutamiento, y los posibles problemas que pueda tener la red para el tráfico de datos.

Terminología:

Diámetro de red :
: Es la distancia máxima(en cantidad de nodos) para conectar dos nodos cualquiera a través de la ruta más corta [4].

Distancia promedio :
: Es la cantidad promedio de nodos entre todos los dispositivos de la red, afecta la latencia.

Ancho de bisección :
: Cantidad de líneas que tendrían que ser removidas para separar la red de interconexión dejando la misma cantidad de nodos en las dos redes de interconexión resultantes, entre más alto el valor más rápida es la red.

Grado de los nodos :
: Cantidad de puertos que conectan un nodo con sus vecinos.

Cantidad de enlaces :
: Una red con mayor cantidad de enlaces puede tener mayor ancho

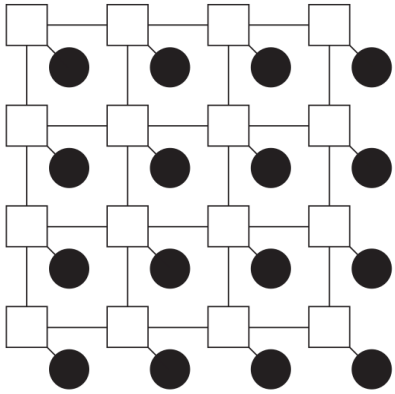

Figura. 6: Topología de Malla simple, tomada de [4].

de banda de datos.

Topología de tipo Malla Rectangular.

La malla rectangular es una de las topologías más comunes, es esta se tiene un dispositivo conectado a cada uno de los enrutadores de la red, posteriormente cada enrutador se conecta con 4 más hasta formar una red se ejemplifica en la Figura 6.

Diámetro de red : (M+N-2).

Distancia promedio : (M+N)/3

Ancho de bisección : min(M,N).

Cantidad de enlaces : 2x[Mx(N-1) + Nx(M-1)].

Cantidad de enrutadores : MxN.

Grado de los nodos : 3(esquinas),4(bordes),5(centro).

Topología de tipo Malla Toroidal.

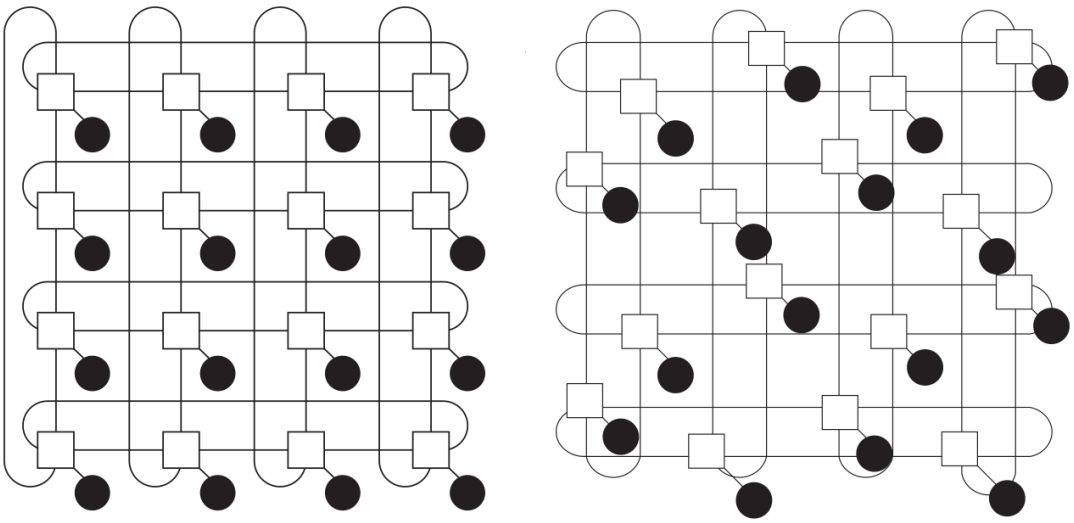

Figura. 7: Topología de Malla Toroidal(a) y Malla Toroidal doblada(b), tomada de [4].

La malla toroidal fue propuesta para reducir el diámetro de red que tiene una malla convencional, para esto conecta los nodos de bordes opuestos entre si, con la desventaja que en una red grande estos enlaces son lo suficientemente largos para causar retrasos en las señales ver figura 7.a, para resolver esto se puede "doblar" el Toroide ver Figura 7.b.

Diámetro de red : M/2 +N/2.

Ancho de bisección : 2 x min(M,N).

Cantidad de enrutadores : MxN.

Grado de los nodos : 5.

Topología de tipo Octagonal.

La malla octagonal se basa en que la comunicación entre cualquier par de nodos sea a través de máximo otros dos nodos ver figura 8, para unir más de 8 nodos varios octágonos pueden ser unidos a través de un puente

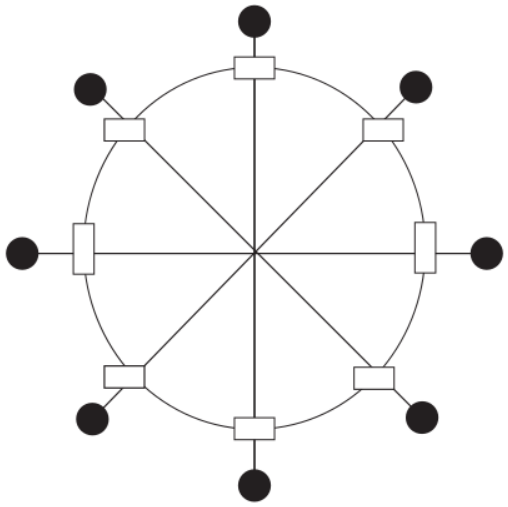

Figura. 8: Topología de Malla Octogonal, tomada de [4].

Diámetro de red : 2 x (N/8)

Ancho de bisección : 6 para N $\leq$ 8 o 6 x (1+ N/8) para N>8

Cantidad de enrutadores : 8 para N $\leq$ 8 o (8+7(N/8)) para N>8.

Grado de los nodos : 4.

Topología de tipo Árbol binario grueso.

Esta topología presenta una estructura recursiva (ver Figura 9) con bajo diámetro pero con las desventajas de un bajo ancho de bisección y que se ocupan enlaces de interconexión largos entre enrutadores y la raíz del árbol lo cual incrementa el consumo de potencia y los retardos.

Diámetro de red : 2 $\times$ (log2(N) $-$ 2).

Ancho de bisección : 1.

Cantidad de enrutadores : (N/2 -1).

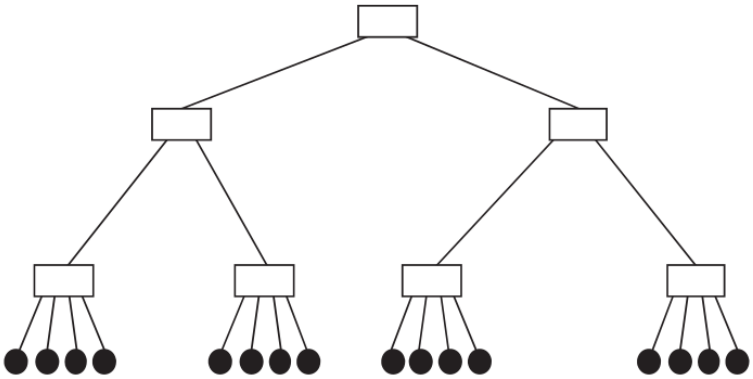

Figura. 9: Árbol binario grueso, tomada de [4].

Grado de los nodos : 5(hojas), 3(rama), 2(raíz).

Técnicas de enrutamiento.

Enrutamiento de circuito: antes de la transmisión un camino físico es reservado desde la fuente hasta el destino de los datos, la latencia de esta técnica esta determinada por el tiempo necesario para reservar el camino y el tiempo para enviar los datos. Esta técnica presenta la desventaja de que no escala bien con el tamaño de la red ya que varios nodos están reservados aun cuando no hay una transferencia a través de ellos además de ser muy propensa a un "Deadlock" [10].

Enrutamiento por paquetes: Cada paquete de datos es dividido en "Flits" y estos son ingresados a la red, siendo entonces direcciónados hacia el destino por medio de la lógica de enrutamiento esta tecnica se puede dividir en 3: almacenar y adelantar, "Virtual Cut-Through (VCT)" y Agujero de Gusano ("Wormhole") [4].

Almacenar y Adelantar: cada paquete es completamente almacenado en cada nodo intermedio entre fuente y destino antes de ser adelantado al siguiente nodo de la ruta. Esta técnica requiere un tamaño de paquete

constante, una enorme área en el silicio ya que se ocupa un buffer en cada puerto del nodo y este debe ser de tamaño suficiente para almacenar un paquete de datos, finalmente la latencia de datos depende del tamaño del paquete [2].

"Virtual Cut-Through (VCT)" : un paquete es adelantado al siguiente enrutador apenas este tiene espacio para recibirlo completo, a diferencia del método anterior el paquete no debe estar completo para empezar a adelantarlo, salvo el problema de latencia y tamaño de paquete, esta técnica comparte los problemas de la anterior [2].

Agujero de Gusano ("Wormhole") : los paquetes de información son divididos en Flits, como flit de cabecera que contiene informacion de fuente, destino y longitud del mensaje, flits de datos y flit de cola, la idea es que cada enrutador almacene solo unos cuantos flit del paquete y éste se mueva en la red como en un "Pipeline". Para mitigar el problema que ocurre cuando una cabecera bloqueada se encuentra con un canal en uso se añaden canales virtuales (VC), estos tienen la función de recibir el paquete cuyo avance esta bloqueado y así aprovechar ancho de banda que de otra manera estaría inactivo [2].

4.2 Estado del Arte

En [5] se propone una arquitectura parametrizada de enrutador con canales virtuales paralelos, latencia de dos ciclos de reloj y control de flujo de datos a través de "crossbar", esta arquitectura implementa una técnica eficiente de enmascarado para filtrar todas las peticiones que no pueden ser enviadas a los puertos de salida, por falta de espacio en alguno de los canales virtuales, además la técnica propuesta provee un uso eficiente de los buffers virtuales.

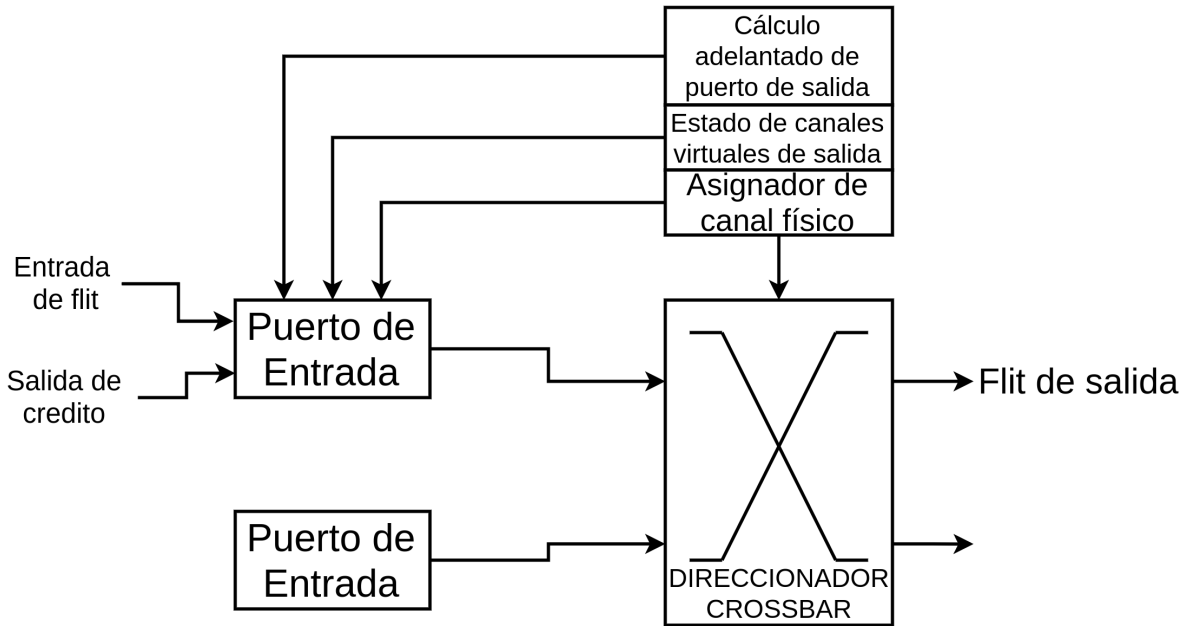

Figura. 10: Arquitectura planteada en [5]

La figura 10 ilustra a nivel de bloques funcionales la arquitectura propuesta, en esta se utiliza direcciónamiento "look-ahead routing" en el cual el puerto de salida es calculado un enrutador antes, y es adjuntado al flit de cabecera, luego de esto y como parte del control, está el módulo de estado de canales virtuales de puertos de salida que registra si un canal virtual ya ha sido asignado, finalmente se ubica el módulo de

asignación de puerto de salida que requiere que haya espacio libre en el siguiente nodo para transferir el dato a través del "crossbar".

En "Efficient Microarchitecture for Network-on-Chip Routers" [3] el autor presenta un extenso análisis de arquitecturas de árbitros, reservadores de canales virtuales y simples, así como la sensibilidad de los indicadores de área, potencia y latencia al variar los parámetros del NoC, también analiza técnicas de manejo de los buffers de entrada; Además provee el código de una implementación totalmente parametrizada de enrutador con canales virtuales, este RTL fue utilizado en númerosas investigaciones de la universidad de Stanford y otros.

Finalmente en [6] presentan un cambio de paradigma con el desarrollo de NoC sin enrutadores, afirmando que la metodología presente de desarrollo de redes significa un alto consumo en área por lógica de direcciónamiento de datos y almacenamiento temporal de datos además señalan que se puede reducir en 9.5x el consumo de potencia, 7.2x el consumo de área, e incrementar en 1.6x el ancho de banda de datos; Esto debido a que afirman que el área y potencia consumidas por los enrutadores de datos pueden ser disminuidas al realizar un uso óptimo de los recursos de conexión, como referencia se muestra que una malla de 6 x 6 nodos solo usa aproximadamente un 3% de los recursos de conexión disponibles.

Según [6] el desarrollo de una NoC sin enrutadores es posible mediante un arreglo de conexiones de tipo lazo que recorra de maneras específicas la red de nodos, esto de forma tal que cada nodo este conectado con otro por al menos un lazo y conectado a máximo n lazos, ya que hay muchas formas de realizar estas conexiones (existen más de 2 millones de lazos distintos para una red de 6 x 6) los autores nos presentan una metodología consistente para realizar los lazos entre los nodos y como es-

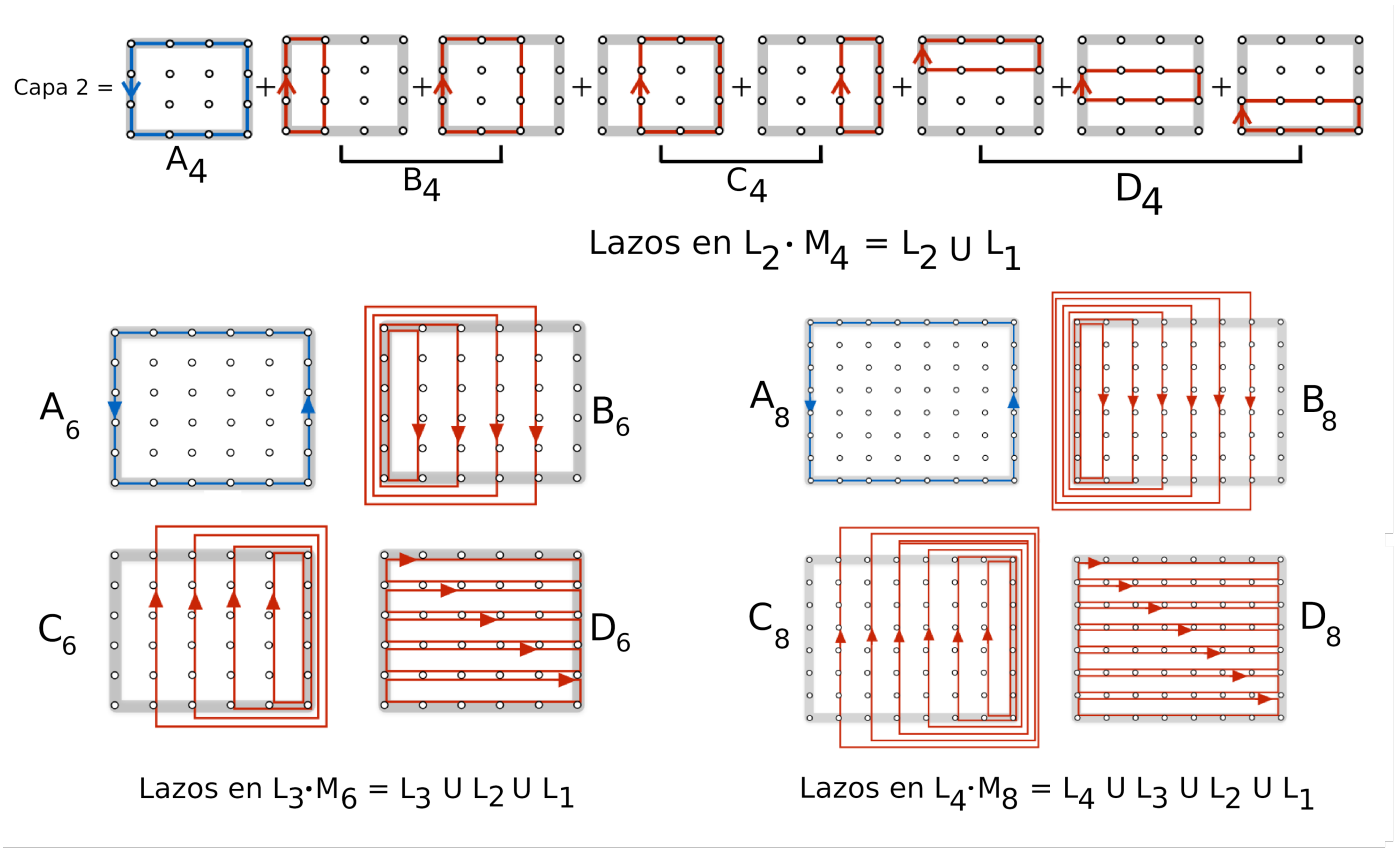

Figura. 11: Metodología de desarrollo de redes Tomado de [6]

calar desde una red de 2 x 2 hasta n x n, este procedimiento se describe brevemente en la Figura 11 (el proceso completo se encuentra en [6]) además en la Figura 12 se presenta una comparativa del área obtenida para una malla 8 x 8 usando distintas técnicas en una tecnología de 15nm

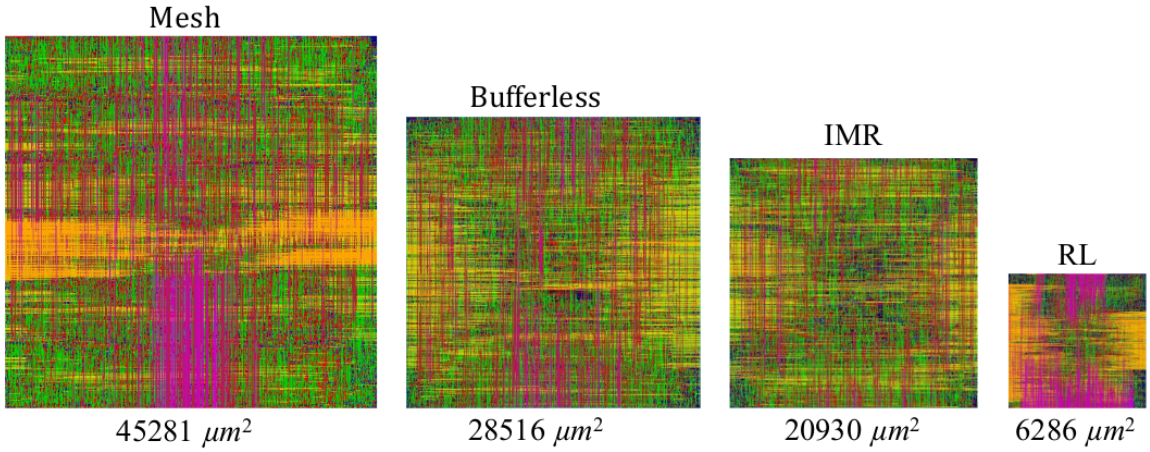

Figura. 12: Comparativa del área obtenida para una malla 8 x 8 usando tecnología de 15nm Tomado de [6]

5 Diseño de Enrutador de Datos

En la figura 13 se detalla la arquitectura del enrutador que consta de 3 partes:

1. FIFOs: Se encuentran en la sección de datos de salida de cada puerto del enrutador, con el fin de compensar las velocidades de procesamiento de datos y retrasos que puedan existir entre dos enrutadores de la red o entre un enrutador y un periférico. En la figura 14 se detalla como las secciones de entrada de datos en un puerto quedan conectadas a la FIFO de salida del enrutador vecino.

2. Decodificadores de direcciónes: Se encuentran en la sección de entrada de datos de cada puerto del enrutador, son diferentes en cada enrutador y se encargan de la lógica de direcciónamiento de datos, para esto el decodificador toma una cantidad(parametrizada) de bits del dato de entrada que corresponden al identificador de dispositivo de destino, con estos datos y mediante una tabla estática de direcciónamiento se genera el valor correspondiente al puerto por donde en dato debe salir para

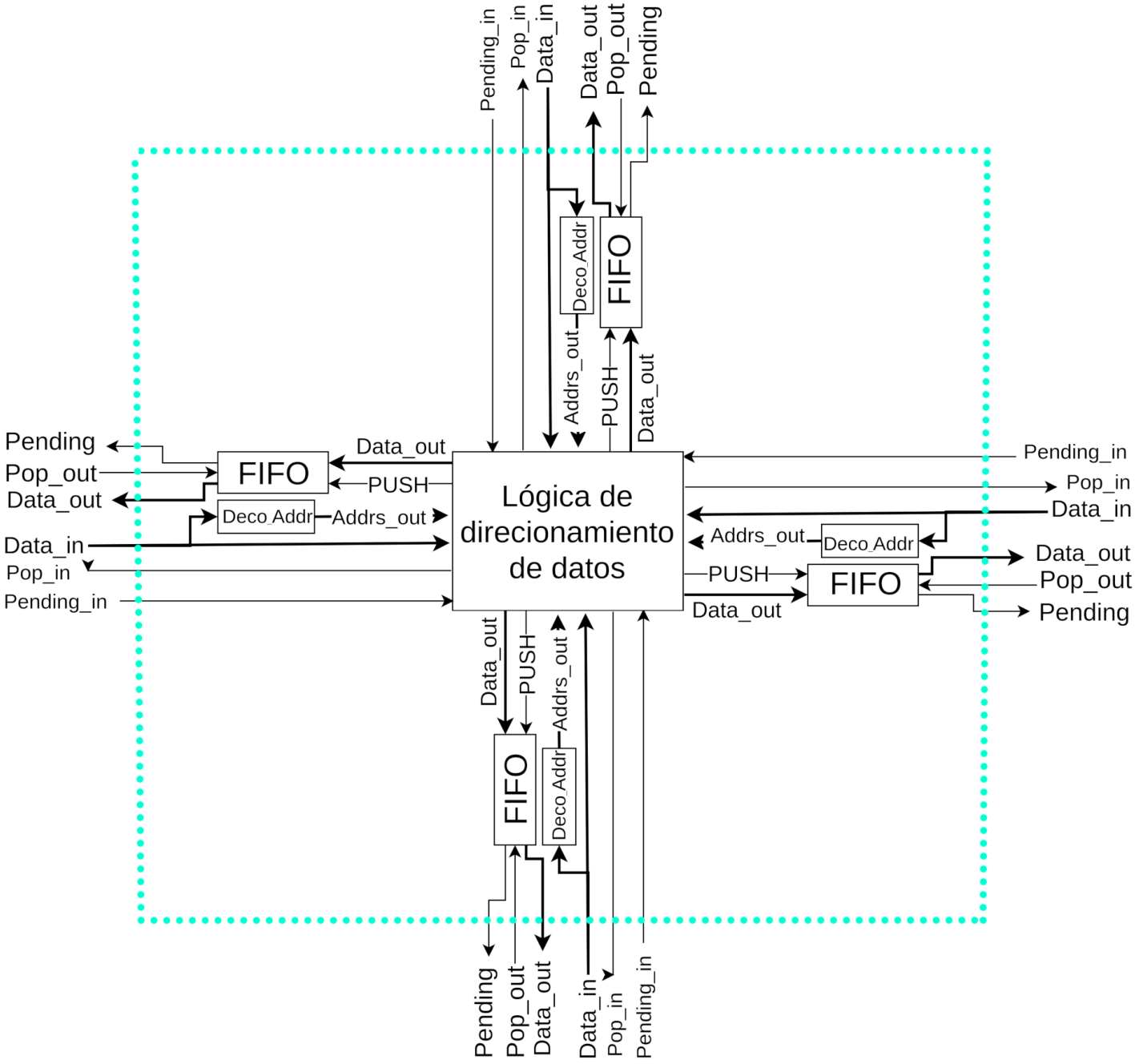

Figura. 13: Arquitectura de Enrutador (4 puertos).

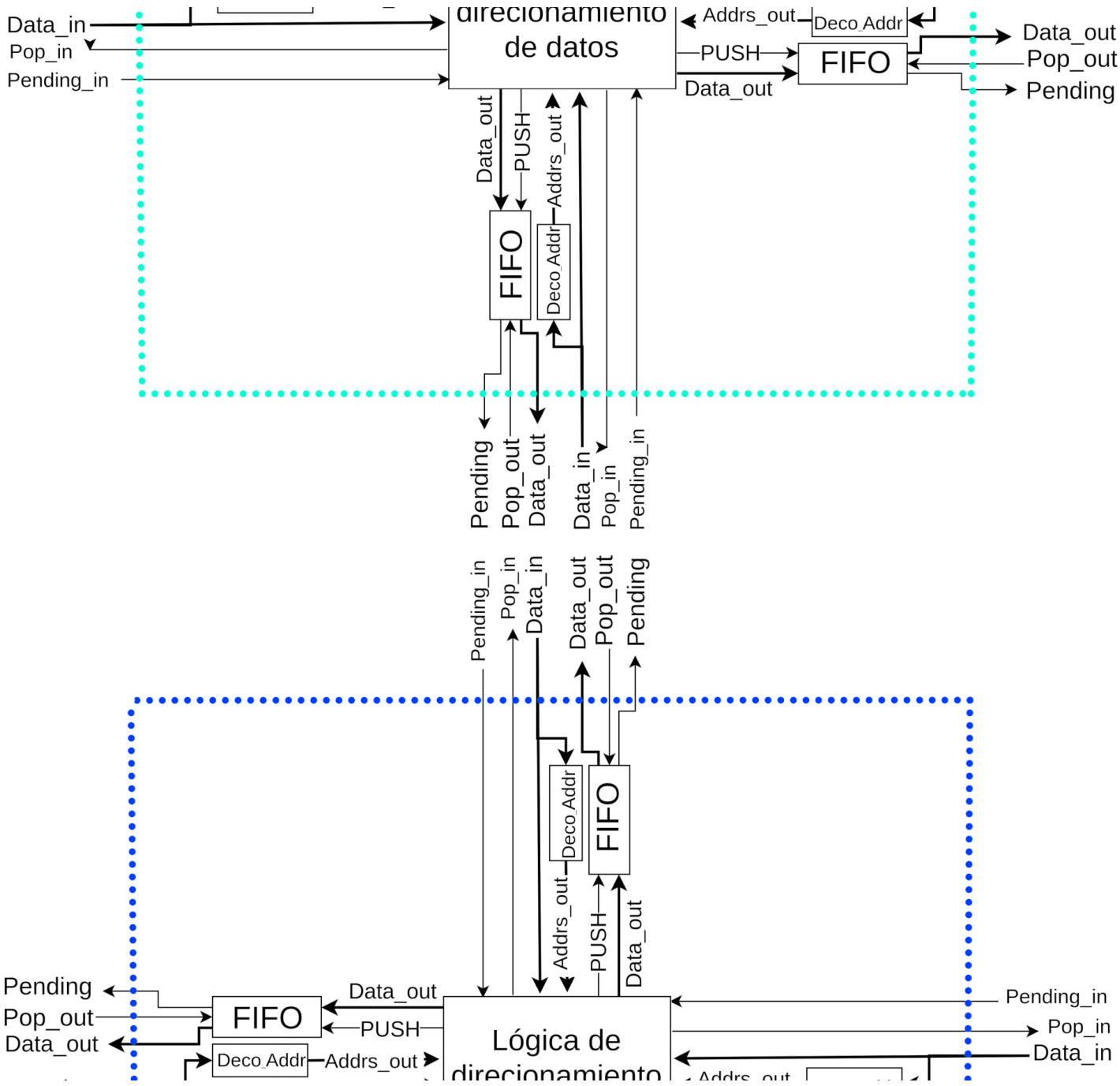

Figura. 14: Detalle de conexión de puertos.

llegar a su destino, este valor de le pasa a la lógica de direcciónamiento de datos.

3. Lógica de direcciónamiento de datos: Constituye la parte principal del enrutador, toma los datos ingresados por un puerto y los envía hacia el puerto designado por el decodificador de dirección.

5.1 Diseño de Lógica de direcciónamiento de datos

Uno de los métodos más usados para realizar el direcciónamiento de datos interno en un enrutador es por medio de una conexión de tipo "Crossbar" o punto a punto, tomando en cuenta que se desea construir una arquitectura parametrizada de propósito general éste tipo de direcciónamiento pese a su gran ancho de banda resulta inconveniente ya que su área crece según $n^2\sqrt{n}$ [7] para n puertos de entrada/salida de datos, además de la lógica de control necesaria; esto limitaría las posibilidades de usar el NoC en sistemas mayores.

Una arquitectura de bus único resulta más conveniente ya que al agregar nodos a la red solo se puede producir uno de los siguientes cambios:

1. El nuevo nodo no tiene un dispositivo asociado: este es un caso en el cual por ejemplo se usa un nodo para conectar varios otros que en términos de transferencia de datos se encuentran alejados, para este caso solo se debe de añadir un puerto a los nodos adyacentes al que se va a agregar a la red.

2. El nuevo nodo tiene un dispositivo asociado: Se debe agregar una entrada a cada decodificador de dirección de cada puerto de cada enrutador (Mejorable ver Conclusiones y Recomendaciones), además se debe de añadir un puerto a los nodos vecinos al que se va a agregar a la red.

Para el diseño de la arquitectura de bus hay dos posibilidades: serial o paralelo, si bien un bus serial tiene la ventaja de usar una sola línea

de comunicación por lo demás resultaría problemático, ya que en caso de aumentar el ancho de palabra se debe aumentar la frecuencia de reloj con el fin de lograr el mismo ancho de banda y esto esta limitado por la tecnología que se use en la implementación física, además un bus serial tiene implicaciones en el uso de área.

El diseño de una arquitectura de bus paralelo resulta más conveniente que las opciones antes citadas ya que la lógica de control permanece parcialmente constante al aumentar el ancho de palabra, y en caso de agregar un puerto los cambios en el árbitro central son menores por lo tanto se puede decir que esta arquitectura "escala" de mejor manera al aumentar el ancho de palabra y la cantidad de puertos, a continuación se detalla el diseño de un bus paralelo con árbitro central.

5.1.1 Diseño del Protocolo de Comunicación

Tomando en cuenta la naturaleza elemental de un bus, se debe de tener en éste un medio para asignar cual puerto puede utilizar el bus y así evitar colisiones, para ello se a designado que el acceso al bus sea controlado por un árbitro central. Para realizar un control sobre el flujo de datos, el árbitro necesita "saber" que un puerto requiere enviar datos a través del bus a estas señales se les denominara "Req", por otra parte la señal con la que el árbitro le indica a un puerto que puede enviar sus datos al bus se le denomina "Grnt", finalmente se necesita indicar a los puertos que hay un dato valido en el bus y que deben de revisar si el identificador de destino presente en el bus coincide con el propio, a esta señal se le denominara "Addr".
De esta manera la interface del árbitro consta de una señal de Addr y tantas señales Req y Grnt como puertos, así cuando un puerto tenga que enviar un dato, primero enviará una señal Req y cuando el árbitro resuelva que el puerto puede enviar sus datos le enviará de vuelta una señal Grnt para que envié los datos de inmediato, junto con esta señal se enviará una señal de Addr a los demás puertos para que el receptor

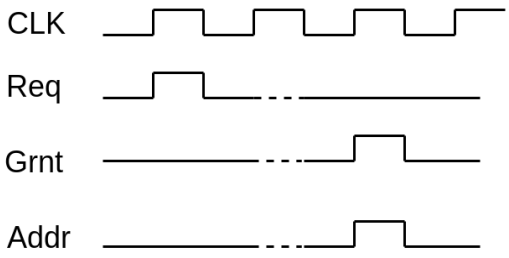

Figura. 15: Protocolo de comunicación árbitro-Puertos.

admita los datos ver Figura 15.

5.1.2 Diseño del árbitro.

En la figura 16 se detalla la lógica RTL del árbitro; Dado que el árbitro tiene 3 funciones : Procesar Peticiones, Enviar Permisos, Enviar Avisos, se analizara el hardware asociado a cada función.

1. Procesar Peticiones: Para evitar "Starvation" el árbitro asigna a través de un contador el derecho del uso del bus puerto por puerto("Turn"), en caso de que el puerto con el derecho de uso del bus no lo ocupe, la lógica asignada a ese puerto genera una señal para avisar al siguiente que no va a usar el turno ("Drop"), el siguiente puerto recibe esa señal ("Catch"), en caso de que este puerto tampoco ocupe el turno, inmediatamente se envía la siguiente señal de "Drop", este proceso continuo se detiene cuando un puerto requiere el turno "Need_turn" y deshabilita la señal de "Drop" inhabilitando a cualquier otro puerto de usar el bus.

2. El envío de la señal Grnt lo realiza cada máquina de estados de cada puerto según las señales generadas por la lógica de procesamiento de peticiones y siguiendo el diagrama de estados de la figura 17.

3. La señal "Addr" es simplemente la unión "OR" de todas las señales Grnt.

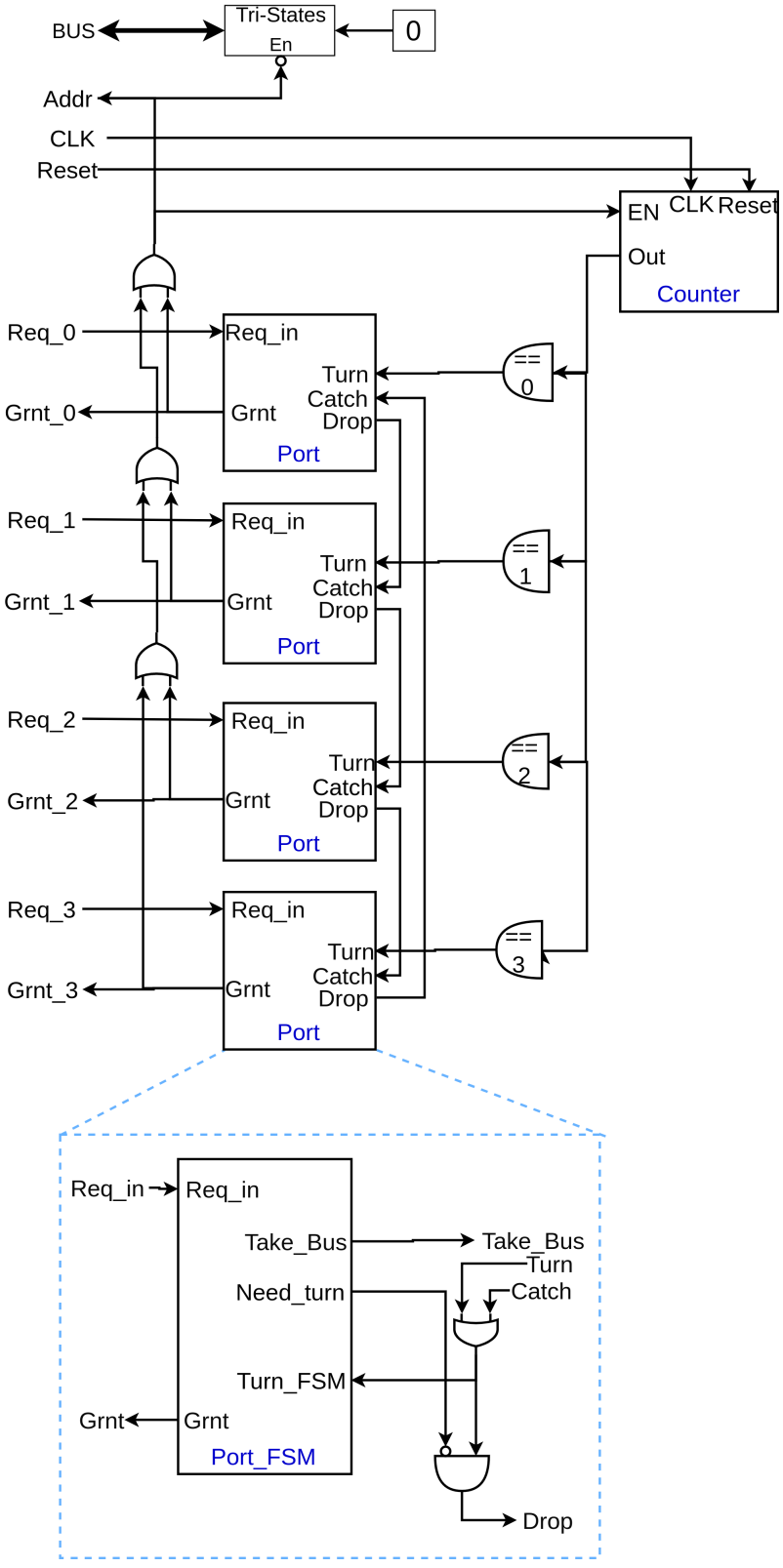

Figura. 16: Lógica RTL del árbitro.

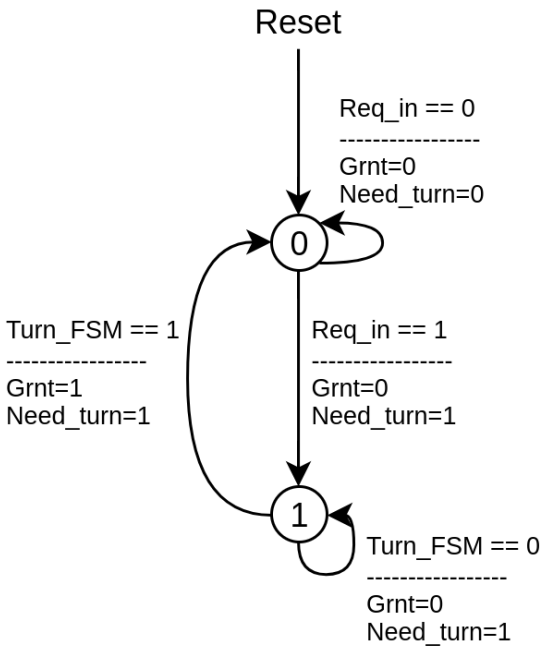

Figura. 17: máquina de estados de cada puerto del árbitro.

La naturaleza del árbitro lo hace independiente de cualquier cambio en el tamaño de la palabra, por otro lado para añadir un nuevo puerto al bus se debe expandir el contador, añadir un juego de "ANDs" correspondiente al identificador del puerto, un "Flip Flop" para la máquina de estados del puerto y su correspondiente lógica de siguiente estado, una compuerta "OR" , una compuerta "AND" y una "NOT" para la lógica "Catch-Drop" y finalmente una "OR" para el Grnt-Addr.

5.1.3 Diseño de Puertos.

En la Figura 18 se detalla la lógica RTL de cada uno de los puertos del bus, en el bloque "FSM_Wrt" se encuentra la máquina de estados que realiza el envío de datos, la recepción de datos se realiza por medio de una lógica simple que valida mediante "ANDs" que haya una dirección en el Bus, que la dirección coincida con el ID del puerto y que no se este escribiendo un dato en la FIFO ("PUSH") en ese momento, si se cumple esto se envia una señal de Push al registro de entrada de la "FIFO" de salida.

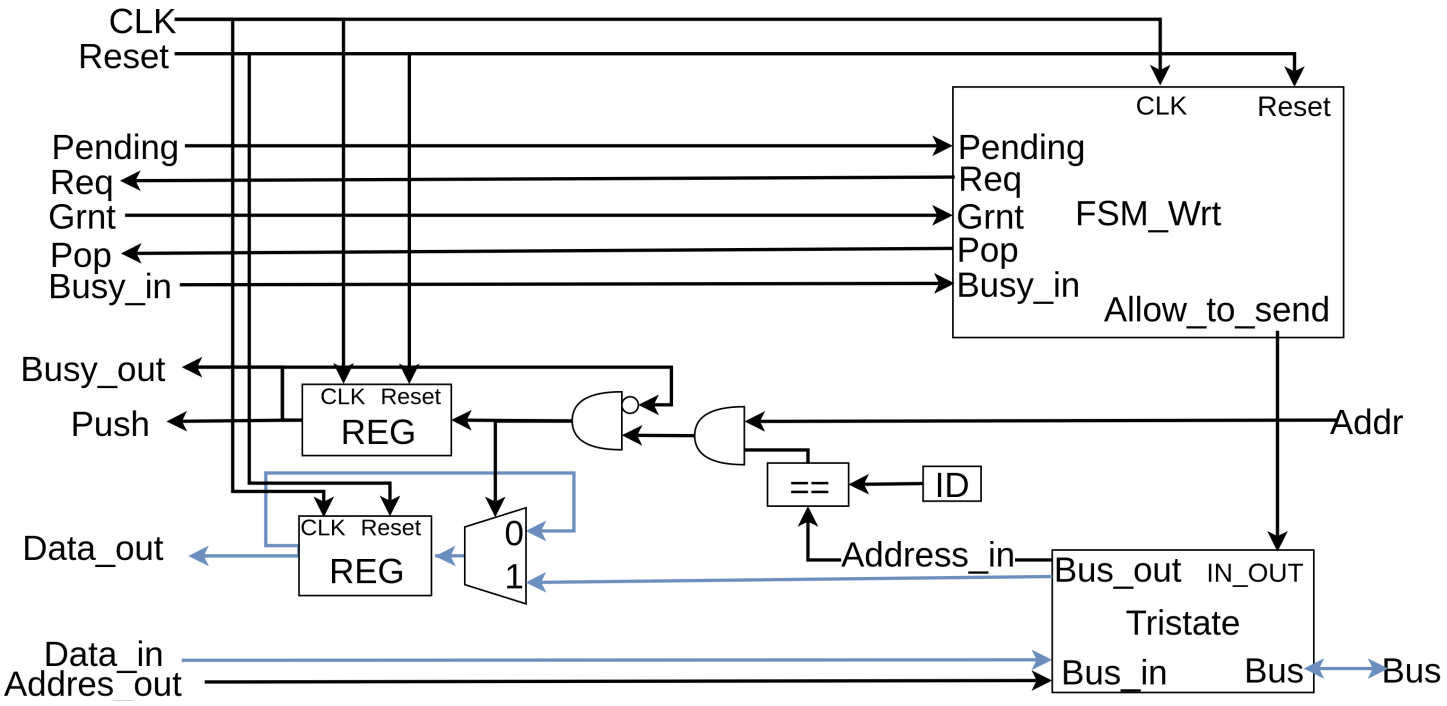

Figura. 18: Lógica RTL de cada puerto del Bus, Ruta de datos en Azul

La máquina de estados (Detallada en Figura 19) espera a tener un dato de entrada pendiente ("Pending") para enviar una petición ("Req") de envío al árbitro, posteriormente espera a recibir el permiso ("Grnt") del árbitro para enviar datos, en este momento se modifica la lógica de siguiente estado para que en caso de recibir la señal de que el puerto de destino esta ocupado("Busy_in") no retirar el dato de la FIFO de entrada ("Pop") en caso contrario se envia el dato por medio de la señal "Allow_to_send".

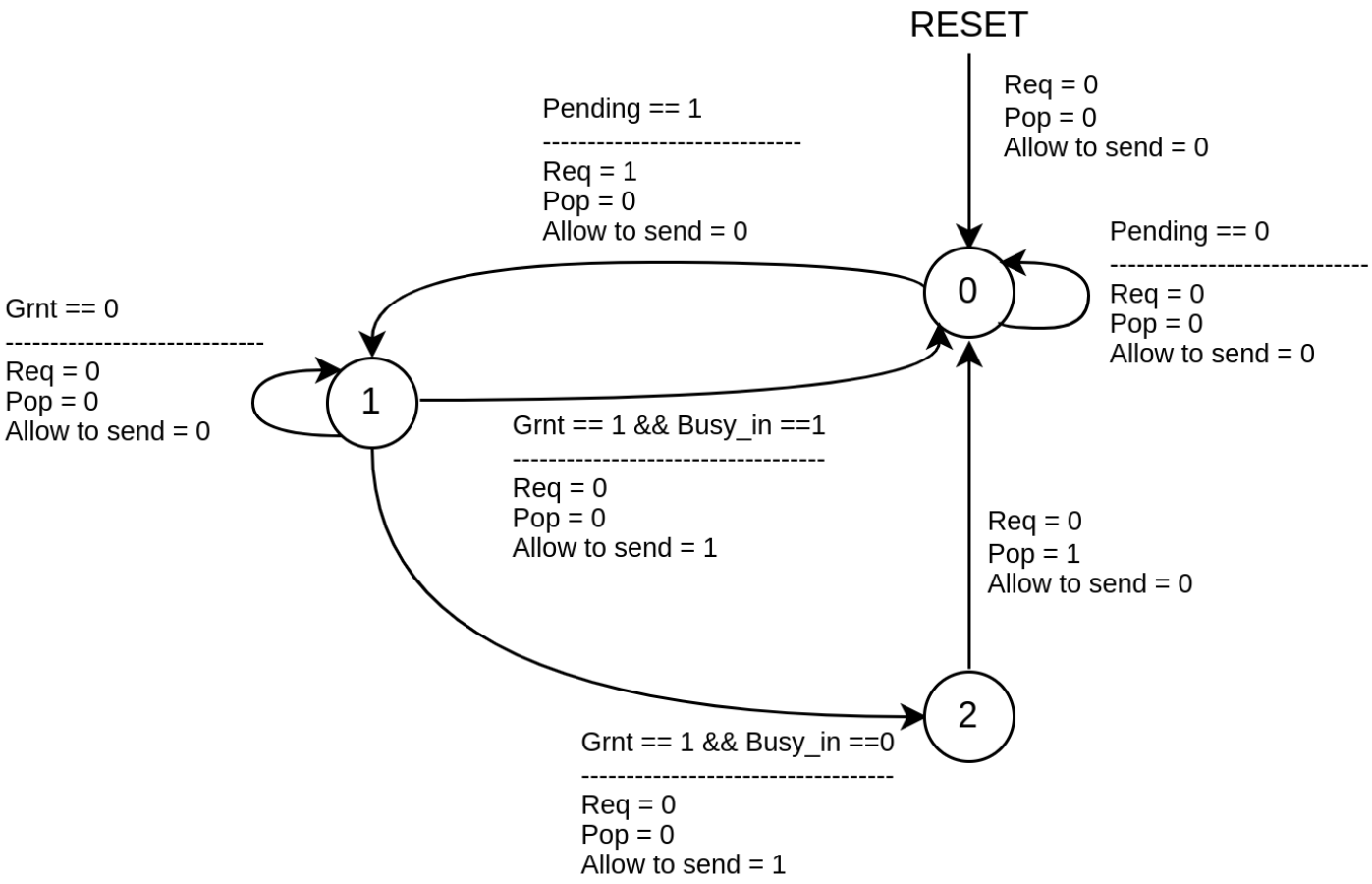

Figura. 19: máquina de estados de cada puerto

En la figura 20 se detalla la conexión entre puertos y árbitro junto con compuertas "OR" para el manejo de las señales "Busy",Ya que se busca que el enrutador sea parametrizado y por lo tanto su bus interno; en el lenguaje System Verilog se utiliza un bloque "generate" con el fin de agregar cuantos puertos sea necesario, de esta manera se completa el bus de datos que constituye le núcleo lógico de enrutador, ver figura 21.

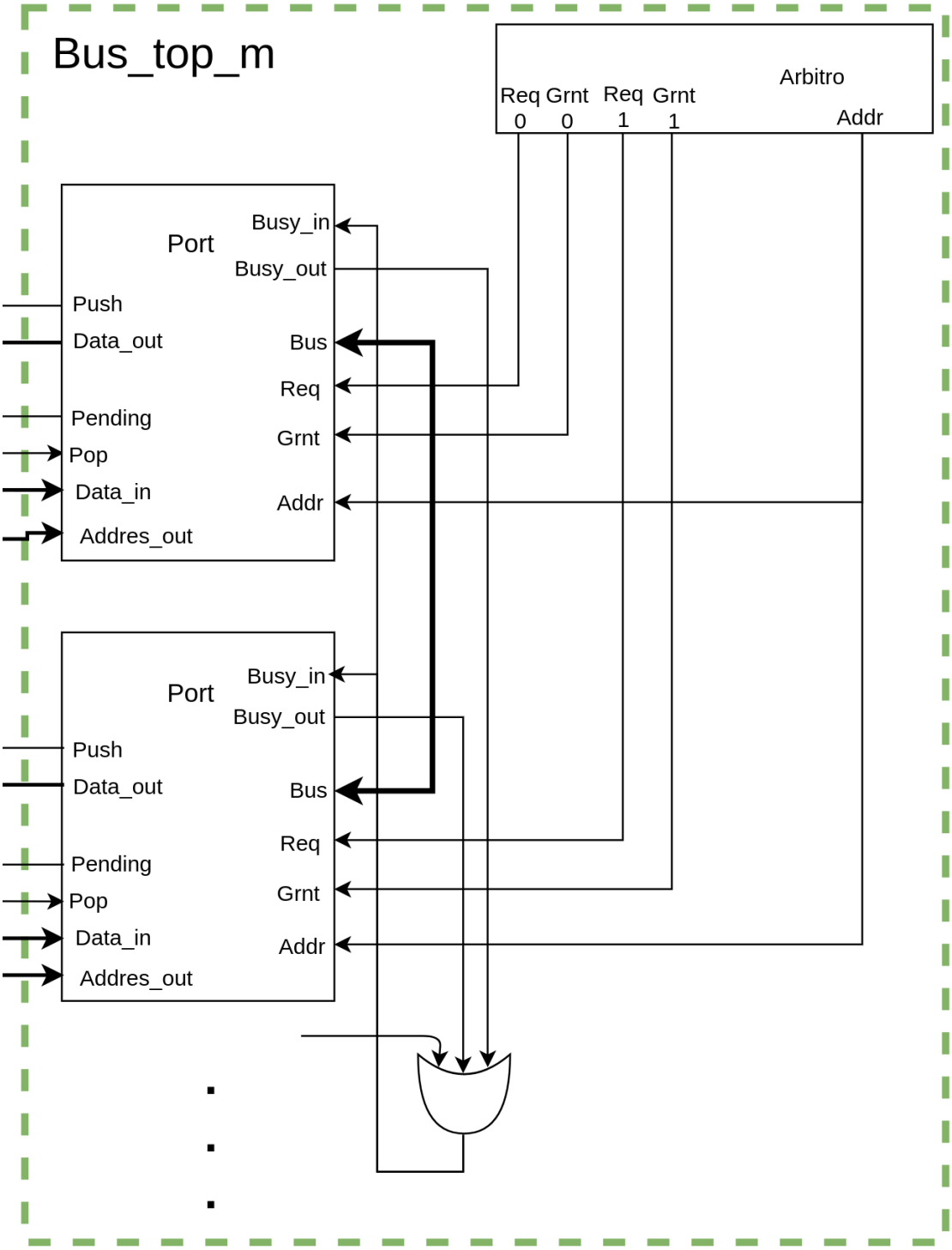

Figura. 20: Detalle de la conexión entre puertos y árbitro

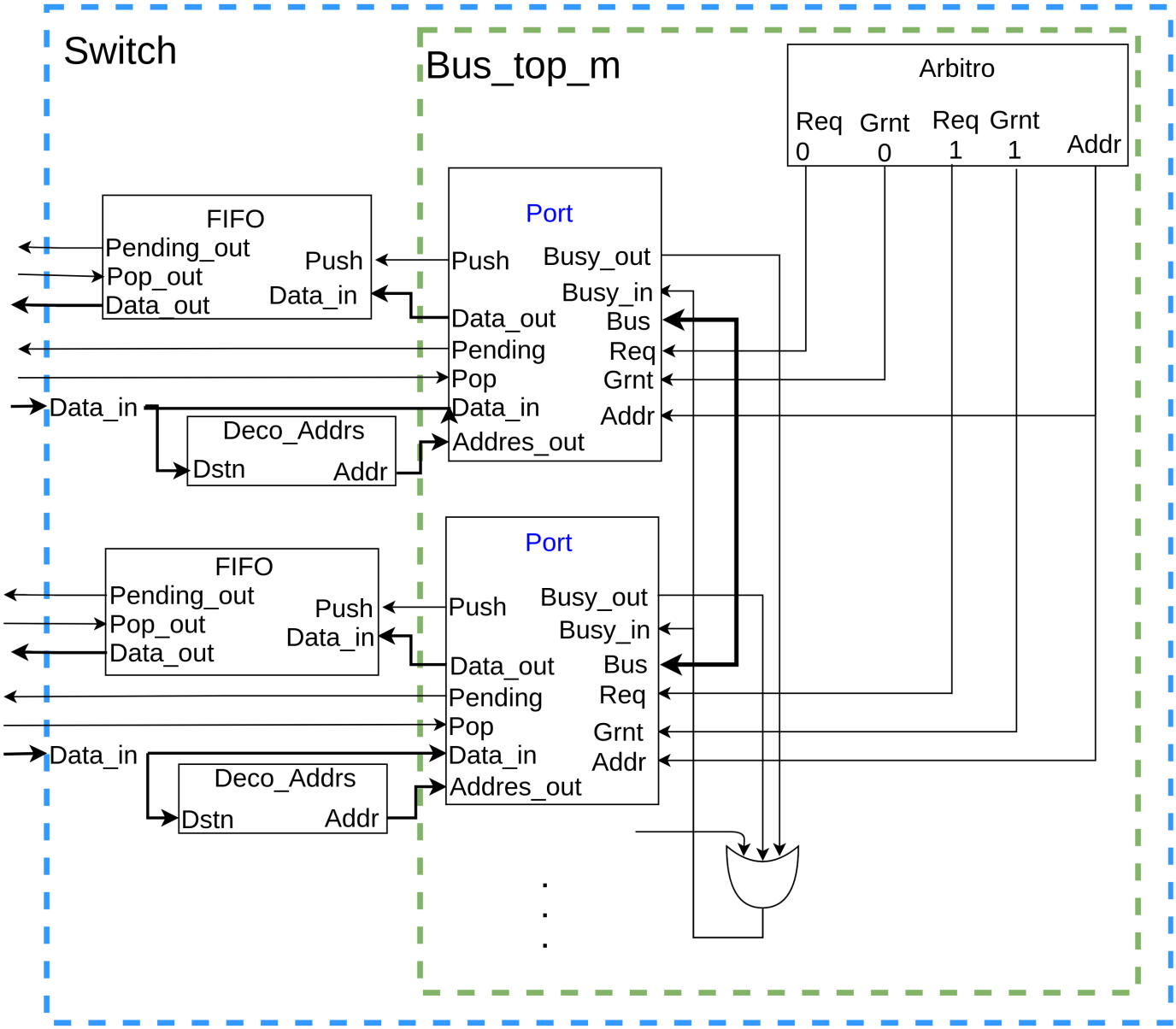

Figura. 21: Detalle de conexión entre Bus, FIFOs de salida y puertos de entrada

6 Diseño de “Network on chip” de tipo Malla bidimensional

Para la implementación del sistema NoC se escogió el diseño de una malla de nodos bidimensional rectangular con un puerto por nodo para la conexión de dispositivos (Figura 22), esto se basa en la capacidad de interconectar los puertos de los enrutadores de la forma que se indica en la Figura 23, teniendo esta base se diseña en lenguaje de descripción de hardware System Verilog un generador de malla de la siguiente manera:

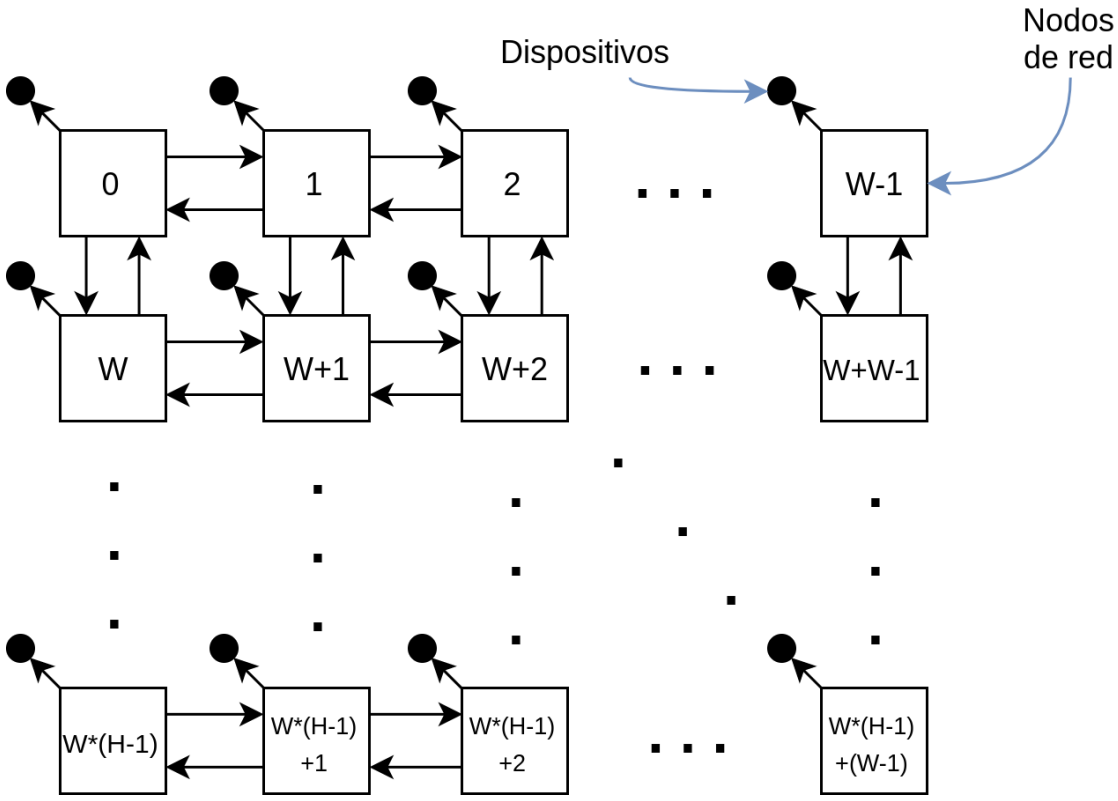

Figura. 22: Malla implementada (W: cantidad de columnas, H: cantidad de filas)

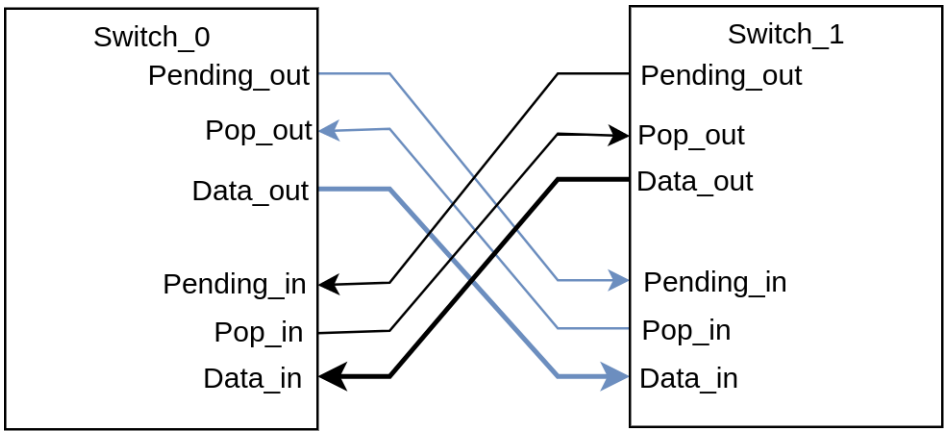

Figura. 23: Detalle de conexiones entre puertos de enrutadores

1. Se toma en cuenta que cada nodo ubicado en las “esquinas” de la red tiene 3 puertos, 1 para dispositivo y 2 para conexión con nodos adyacentes, los nodos en los “bordes” de la malla tienen 4 puertos y finalmente los nodos restantes tienen 5 puertos (ver figura 24); esto se realiza para no generar puertos que no van a ser utilizados.

2. Se utilizan dos bloques “generate” anidados, uno genera filas y el otro columnas, esto se resume en el siguiente código

```
generate // Inicio de un bloque de creación repetida condicional de RTL
    genvar gen_h; // Almacena el número de fila que se esta creando
    //se incrementa el número de fila
    for (gen_h = 0; gen_h <= height-1; gen_h = gen_h + 1)
    begin:_h //para cada fila se genera lo siguiente
        genvar gen_v; //Almacena el número de columna que se esta creando
        //se incrementa el número de columna
        for (gen_v = 0; gen_v <= width-1; gen_v = gen_v + 1)
        begin:_v //para cada columna se genera lo siguiente

            //En este punto se obtienen dos indicadores gen_h y gen_v
            //con los cuales se puede calcular la posición del
            //enrutador, para tomar decisiones sobre la cantidad de
            //puertos, lógica de enrutamiento, decodificadores de
            //direcciónes, conexiones e instanciar FIFOS.

            //Aquí se ubica lo descrito en los siguientes puntos...

        end
    end
endgenerate
```

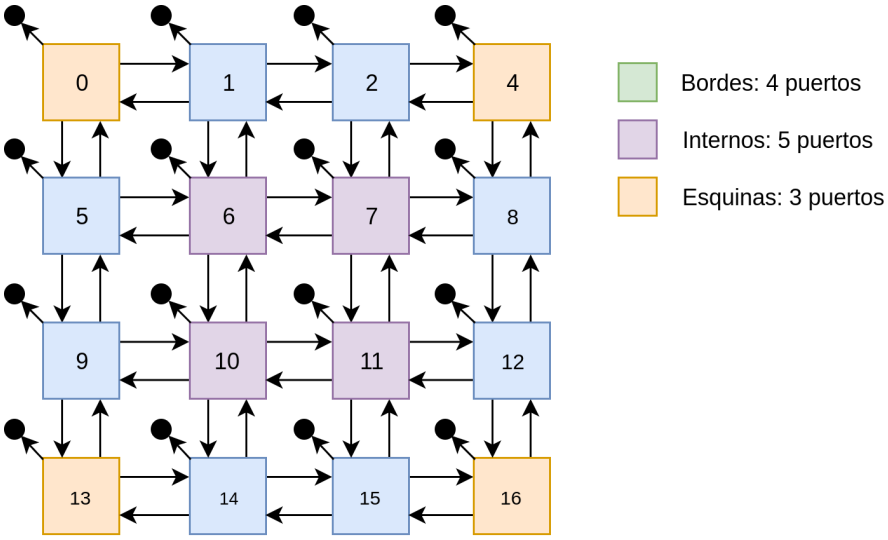

Figura. 24: Ejemplo de malla 4x4

3. Se identifica la posición del enrutador (Esquina, Borde, Interno) y se instancia según la cantidad de puertos que requiera (3, 4, 5); Al instanciar al enrutador se le asigna una posición en la malla ver figura 24, esta posición es utilizada por un generador de decodificadores de dirección que la utiliza para calcular por que puerto debe de salir un dato para llegar a su destino, esto basándose en un algoritmo de direcciónamiento de tipo XY en el cual el dato se mueve en la misma fila hasta que coincida con la columna de destino, posteriormente el dato se mueve en la columna hasta que se encuentre en la misma fila y finalmente sale por el puerto dedicado al dispositivo.

4. Se conecta el puerto de salida de cada enrutador con el puerto de salida de la red correspondiente al dispositivo asociado al nodo.

5. Se instancia una memoria FIFO de “Flip Flops” para la entrada de datos desde el dispositivo y se conecta al enrutador y al puerto de entrada de la red correspondiente al dispositivo asociado al nodo.

6. Como en la red no todos los enrutadores tienen la misma cantidad de

puertos, se genera una lista de casos para conectar el puerto “Derecho” del enrutador generado con el puerto “izquierdo” del enrutador de la siguiente columna.

7. De igual manera al punto anterior se generan conexiones del puerto “inferior” del enrutador generado hacia el puerto “Superior” del enrutador de la siguiente fila.

8. Finalmente la red esta lista para ser conectada con los dispositivos según la numeración mostrada en la figura 22

7 Resultados y Análisis

7.1 Proceso de síntesis

Este es el proceso en el que se interpreta la estructura RTL descrita en el lenguaje System Verilog y se genera un archivo que describe las conexiones en términos de celdas estándar(registros, compuertas lógicas, arreglos de compuestas lógicas) a continuación se describe este proceso de forma simplificada para las herramientas de Synopsys:

1. El proceso se inicia definiendo las rutas de los directorios que contienen los archivos de las celdas estándar.
2. Posteriormente se analiza el código en busca de errores (comando analyze).
3. Se traduce el diseño a uno independiente de la tecnología (GTECH), sustituyendo los parámetros en los módulos, sustituyendo las operaciones aritméticas por componentes genéricos (comando elaborate).
4. Se analiza si todos los módulos instanciados se encuentran correctamente conectados y enlazados al módulo principal (comando link).
5. Se revisa el diseño por errores en consistencia como puertos no conectados, puertos con valores constantes, celdas sin conexiones de entrada o

salida, cables conectados a múltiples salidas, módulos instanciados dentro de si mismos y demás (comando check_design).
6. Se cargan las restricciones de diseño, como frecuencia de reloj, retrasos máximos permitidos, celdas para puertos de salida de datos del chip, "Fan-out" máximo, condiciones de operación y otros, (comando source ...constraints.tcl).
7. Se sintetiza el diseño en uno descrito en términos de compuertas lógicas referentes a la tecnología física que se este usando (Comando compile_ultra).
8. Se calculan predicciones de consumo de potencia a partir de los bancos de pruebas diseñados y pre-compilados en un archivo de transiciones de nodos (.saif) (Comando read_saif _.saif).
9. Se escriben los reportes de consumo de potencia, uso de área, y otros (comandos report_*).
10. Se escriben los archivos de conexiones a nivel de compuertas uno en formato verilog (.v) y otro en uno interno de Synopsys(.ddc) (comando write ...).

7.2 Pruebas Funcionales

Para corroborar el correcto funcionamiento de la arquitectura de bus propuesta se realiza una prueba básica de funcionamiento para un bus de 5 puertos y ancho de palabra igual a 16, la prueba se describe a continuación y se ilustra en la figura 25:

1. Se ingresa en espera un dato igual a h'0001 en el puerto 0 del bus y con el puerto 1 como destino
2. Se espera a que este dato sea procesado y extraído de la entrada del bus.
3. Posteriormente se corrobora que el dato se encuentra en el puerto de salida. Ver Figura 25.

4. Una vez extraído el dato h'0001, se ingresa inmediatamente el dato h'0002 y se repiten los puntos 2 y 3 para el dato h'0002.
Una vez comprobado el funcionamiento del bus para un solo dato de entrada se procede a agregar un segundo dato en un puerto distinto.
5. Se ingresa en espera un dato igual a h'0003 en el puerto 0 del bus y con el puerto 2 como destino y se ingresa en espera un dato igual a h'0004 en el puerto 1 del bus y con el puerto 2 como destino.
6. Se comprueba la salida de un dato.
7. Ya que ambos datos se dirigen al mismo puerto de salida, uno de ellos debe esperar a que se termine la escritura de la FIFO del puerto de salida.
8. El dato que tubo que esperar en el punto anterior es reenviado.

Esta prueba se repite sobre la arquitectura de enrutador cuyos resultados se ilustran en la figura 26, ya que en este caso están presentes los decodificadores de dirección, estos se configuran (por simplicidad) para que el puerto de salida sea 1 + el valor de destino.

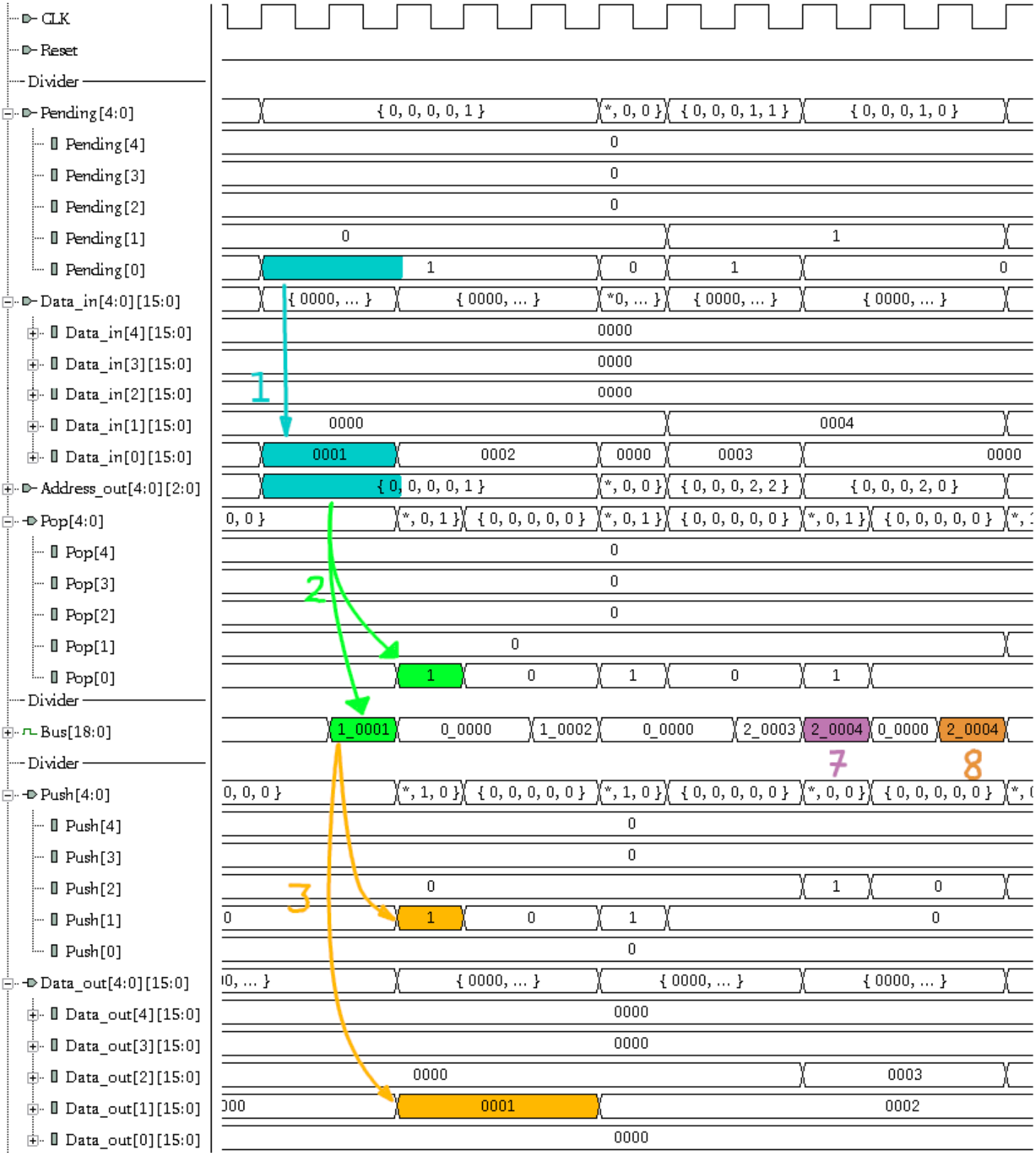

Figura. 25: Prueba sobre el bus propuesto

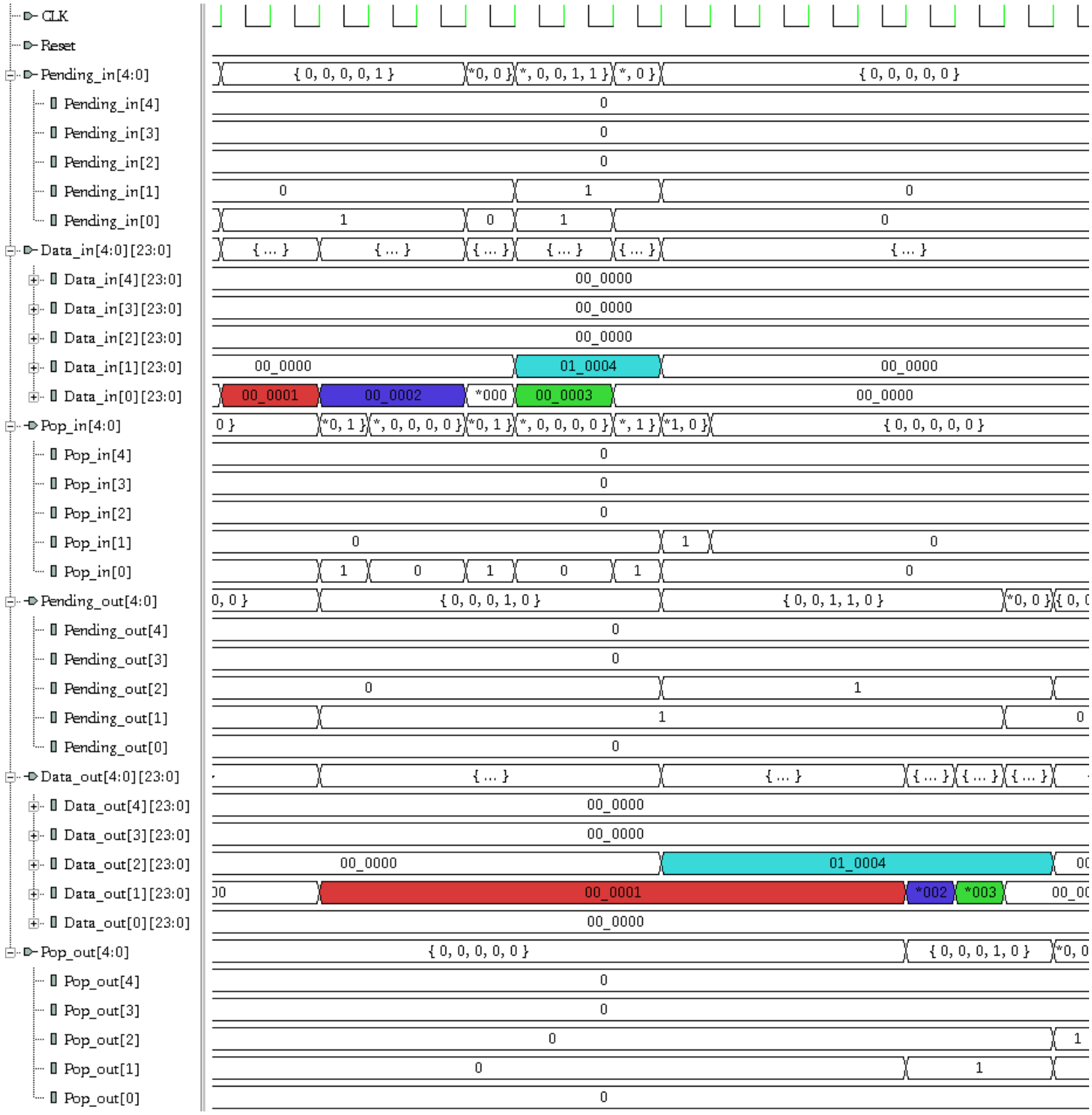

Figura. 26: Prueba sobre el enrutador propuesto, se resaltan las transferencias

Para las pruebas del funcionamiento de la malla bidimensional, se genera una topología de 2 x 2 nodos y se ingresan datos por medio de "Push" a las FIFOs de entrada en los puertos 0,1,2 ; Los datos ingresados tienen el siguiente formatos XXYY_DATA donde XX es el número de dispositivo de salida, YY en general pueden ser datos, en este caso se ingresa el dispositivo de origen del mensaje, este funcionamiento se detalla en la figura 27.

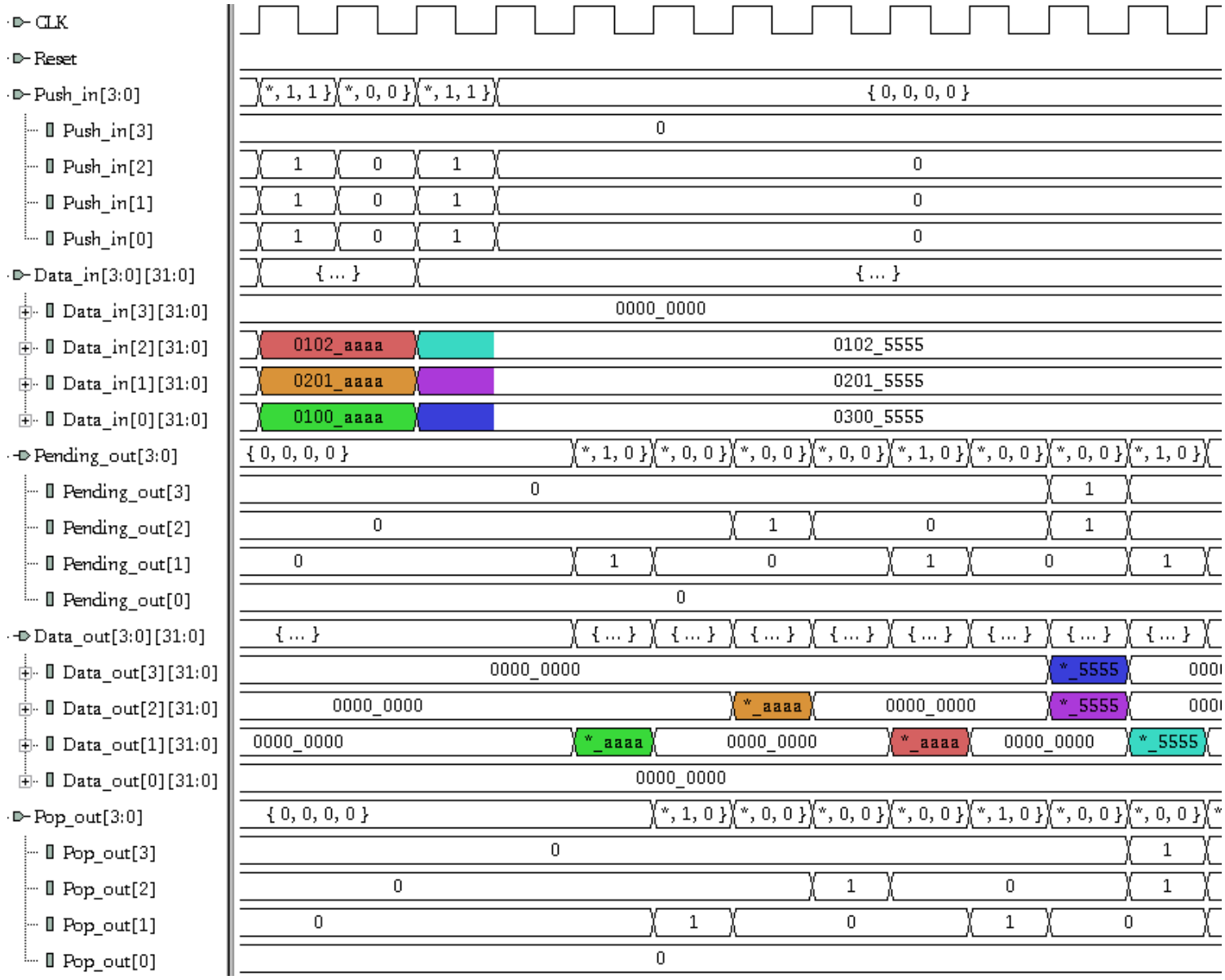

Figura. 27: Prueba sobre el mesh propuesto, se resaltan las transferencias

7.3 Comparativa de Buses en área

Para realizar la comparativa de área se realiza una arquitectura de buses de 2,4,8 puertos y anchos de palabra de 64 y 72 bits, de los cuales los 8 MSB bits son dedicados a el identificador de destino.

Luego de adaptar los scripts existentes en el DCILab para trabajar sobre la estructura de directorios del proyecto, se obtienen los resultados de área para cada uno de los buses, estos se presentan en la tabla 1 y se resumen en la figura 28 , la figura 29 representa el largo de un cuadrado equivalente al área de cada bus.

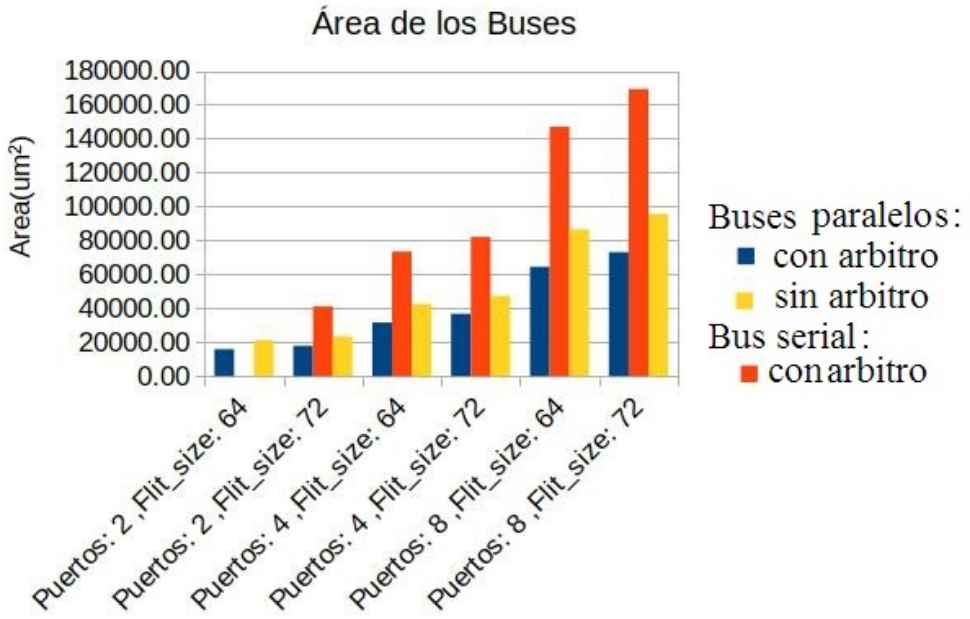

Figura. 28: Resumen de área de los buses evaluados (tabla 1)

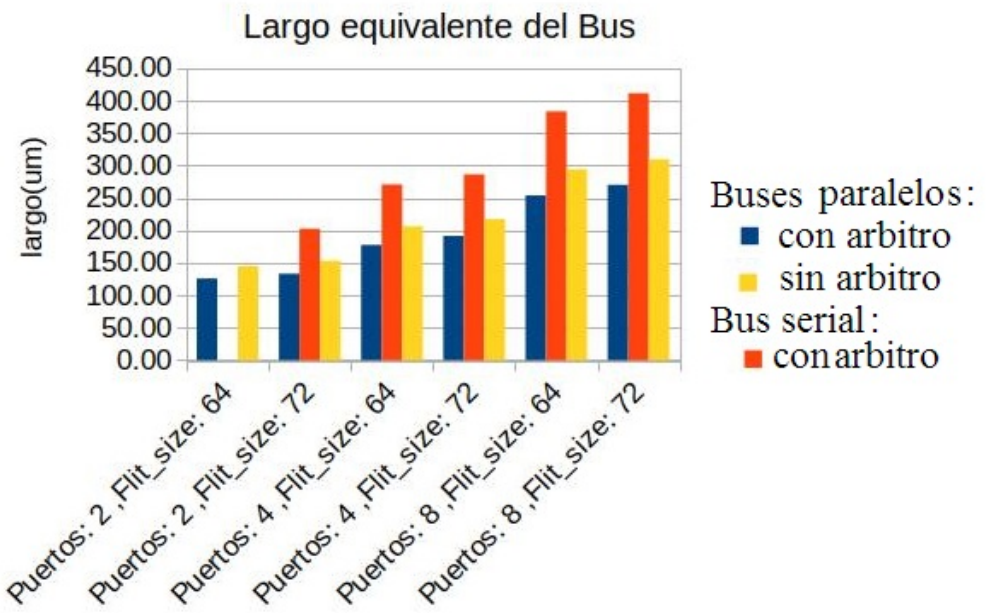

Figura. 29: Resumen de área de los buses evaluados, lado equivalente (tabla 1)

Bus	Características	Área Comb.	Área No Comb.	Área total	Largo eq. μm
Bus_M	Puertos: 2 ,Flit: 64	5245.59	9837.34	15879.04	126.01
Bus_M	Puertos: 2 ,Flit: 72	5764.42	11157.39	17811.83	133.46
Bus_M	Puertos: 4 ,Flit: 64	10182.10	19773.11	31591.22	177.74
Bus_M	Puertos: 4 ,Flit: 72	12445.56	22399.81	36748.40	191.70
Bus_M	Puertos: 8 ,Flit: 64	21417.90	39675.23	64490.12	253.95
Bus_M	Puertos: 8 ,Flit: 72	24264.02	44949.42	73051.85	270.28
Bus_R_a	Puertos: 2 ,Flit: 64				
Bus_R_a	Puertos: 2 ,Flit: 72	5668.96	33529.93	41080.86	202.68
Bus_R_a	Puertos: 4 ,Flit: 64	10361.38	59709.01	73472.75	271.06
Bus_R_a	Puertos: 4 ,Flit: 72	11259.44	67054.84	82097.25	286.53
Bus_R_a	Puertos: 8 ,Flit: 64	20876.31	119360.81	147047.49	383.47
Bus_R_a	Puertos: 8 ,Flit: 72	27357.73	134056.08	169281.53	411.44
Bus_R_p_n_a	Puertos: 2 ,Flit: 64	1166.72	19114.91	21142.31	145.40
Bus_R_p_n_a	Puertos: 2 ,Flit: 72	1190.00	21362.99	23502.00	153.30
Bus_R_p_n_a	Puertos: 4 ,Flit: 64	2345.88	38457.23	42576.08	206.34
Bus_R_p_n_a	Puertos: 4 ,Flit: 72	2408.70	42954.60	47318.64	217.53
Bus_R_p_n_a	Puertos: 8 ,Flit: 64	5338.64	77589.02	86580.02	294.24

Tabla. 1: Áreas de los buses (en μm^2), obtenida en síntesis física

Se puede apreciar en los resultados obtenidos que el área de la arquitectura desarrollada es mucho menor que la del bus serial con árbitro central, esto debido a que en una arquitectura serial se tiene una mayor

cantidad de bloques lógicos, por ejemplo serializadores , contadores para el largo de palabra, lógica de control de secuencias (Puerto de destino, datos), estos bloques por su naturaleza utilizan una mayor lógica secuencial que la presente en un bus paralelo, y esta lógica secuencial aumenta el uso de área como se demuestra en la figura 30.

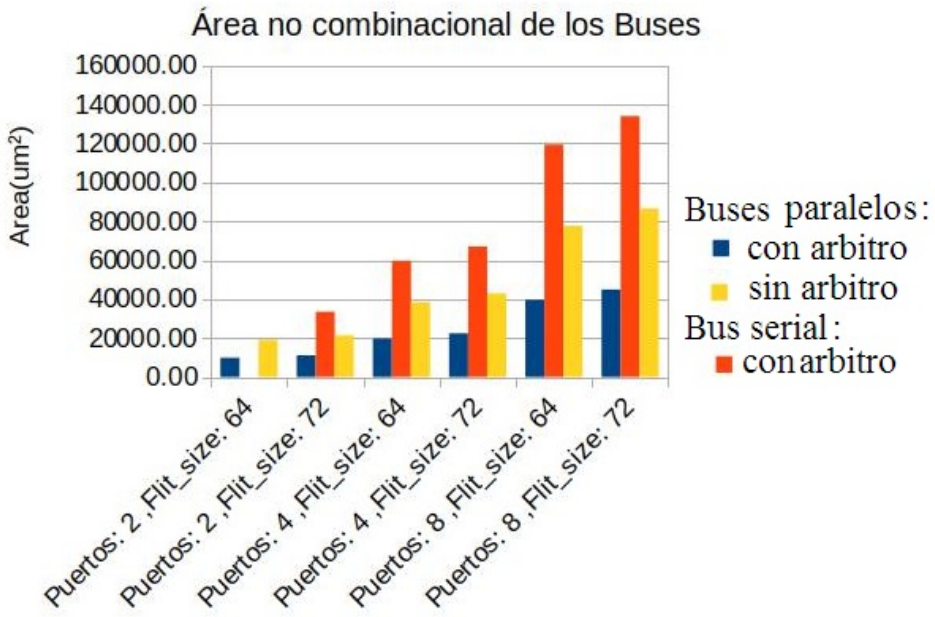

Figura. 30: Resumen de área no combinacional de los buses evaluados (Tabla 1)

Con respecto al uso de área comparado con el bus paralelo sin árbitro, la arquitectura diseñada se acerca considerablemente siendo la diferencia de apenas $50\mu m$ aproximadamente de lado equivalente ver figura 29, Esto se debe principalmente a que en la arquitectura de bus paralelo sin árbitro se requiere que la lógica de arbitraje este presente en cada módulo, esto aumenta el área total del circuito ya que el protocolo de uso del bus debe estar presente en cada puerto, a diferencia del arbitraje central donde el protocolo se concentra en un solo módulo.

7.4 Comparativa de Buses en potencia

Para obtener una métrica comparativa de la potencia consumida por los buses en un escenario crítico se diseña una prueba que consiste en ingresar datos conmutados (...1010 luego ...0101) por todos los puertos, para esto se envían 1024 Flits de datos distribuidos por todos los puertos, además estas pruebas se realizan para buses de 2,4,8 puertos y anchos de palabra de 64 y 72 bits, de los cuales los 8 bits más significativos son dedicados a el identificador de destino.

Bus	Parametros	Combinacional (mW)	No combinacional (mW)	Total (mW)
Bus_M	Puertos: 2 ,Flit_size: 64	10.48	7.08	19.70
Bus_M	Puertos: 2 ,Flit_size: 72	11.08	9.29	22.71
Bus_M	Puertos: 4 ,Flit_size: 64	13.76	14.28	31.78
Bus_M	Puertos: 4 ,Flit_size: 72	21.80	11.71	37.78
Bus_M	Puertos: 8 ,Flit_size: 64	17.88	20.45	45.23
Bus_M	Puertos: 8 ,Flit_size: 72	20.91	23.46	52.11
Bus_R_a	Puertos: 2 ,Flit_size: 64			
Bus_R_a	Puertos: 2 ,Flit_size: 72	6.14	7.63	16.80
Bus_R_a	Puertos: 4 ,Flit_size: 64	14.40	13.04	32.91
Bus_R_a	Puertos: 4 ,Flit_size: 72	17.76	14.84	38.51
Bus_R_a	Puertos: 8 ,Flit_size: 64	34.07	25.78	71.73
Bus_R_a	Puertos: 8 ,Flit_size: 72	42.81	29.34	84.11
Bus_R_p_n_a	Puertos: 2 ,Flit_size: 64	2.29	5.05	8.57
Bus_R_p_n_a	Puertos: 2 ,Flit_size: 72	2.56	5.92	9.67
Bus_R_p_n_a	Puertos: 4 ,Flit_size: 64	3.78	13.05	18.86
Bus_R_p_n_a	Puertos: 4 ,Flit_size: 72	4.36	15.31	21.72
Bus_R_p_n_a	Puertos: 8 ,Flit_size: 64	8.82	27.89	40.78

Tabla. 2: Potencia de los buses, obtenida en síntesis física

Los resultados de las pruebas anteriores se resumen en la tabla 2 y en la figura 31, dado que las pruebas son las mismas para buses con la misma cantidad de puertos, los resultados de potencia son un reflejo de la complejidad del sistema, específicamente de la lógica de control y arbitraje, de esta manera se aprecia que la lógica implementada en el bus serial conmuta más frecuentemente que en los otros dos buses, por otra

parte la comparación de la potencia consumida por los buses paralelos refleja que al aumentar la cantidad de puertos, la potencia consumida por el bus paralelo sin árbitro aumenta más ligeramente más rápido que la potencia de la arquitectura presentada, sin embargo hasta donde se estudió, la potencia del bus paralelo sin árbitro fue siempre menor ver figura 31.

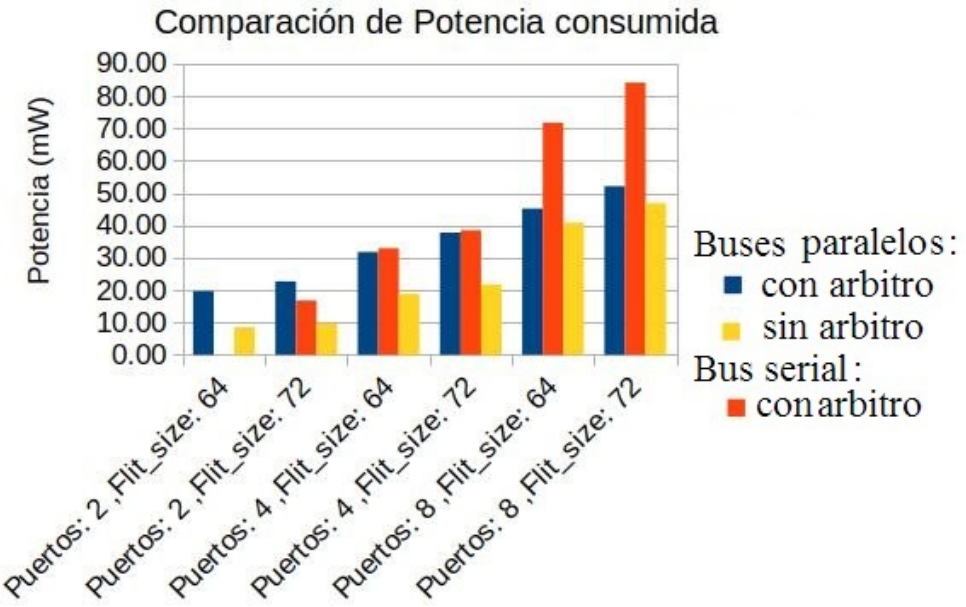

Figura. 31: Resumen de potencia de los buses evaluados (Tabla 2)

7.5 Comparativa de Buses en velocidad

Para el desarrollo de esta prueba se ingresan datos alternando unos y ceros, continuamente a todos los puertos de las implementaciones anteriormente mencionadas, posteriormente se compara la velocidad de salida de datos, en las figuras 32, 33, 34 se detallan los resultados de estas pruebas.

Dada la naturaleza del bus serial, este será el más lento ya que envía los datos bit a bit, como se puede comprobar en la figura 34, tomando en cuenta que el bus debe enviar n bits de datos, la cantidad de pulsos de

reloj restantes entre dos “Push” corresponde al tiempo necesario para realizar el arbitraje en este caso esto corresponde a 3 ciclos de reloj, finalmente se deduce que entre dos “Push” hay 3 + n ciclos de reloj.

Respecto al bus paralelo sin árbitro siguiendo la lógica anterior se encuentra que se requieren 4 ciclos de reloj entre dos “Push”; El bus paralelo con árbitro realiza “Push” en cada ciclo de reloj como se muestra en la figura 32 pero esto es en caso óptimo en el que el bus es usado por múltiples puertos, en el caso de que sea un solo puerto el que requiere enviar datos, esto se realizara cada 2 ciclos de reloj o en el caso de que el destino sea siempre el mismo (Mejorable, ver Conclusiones y Recomendaciones).

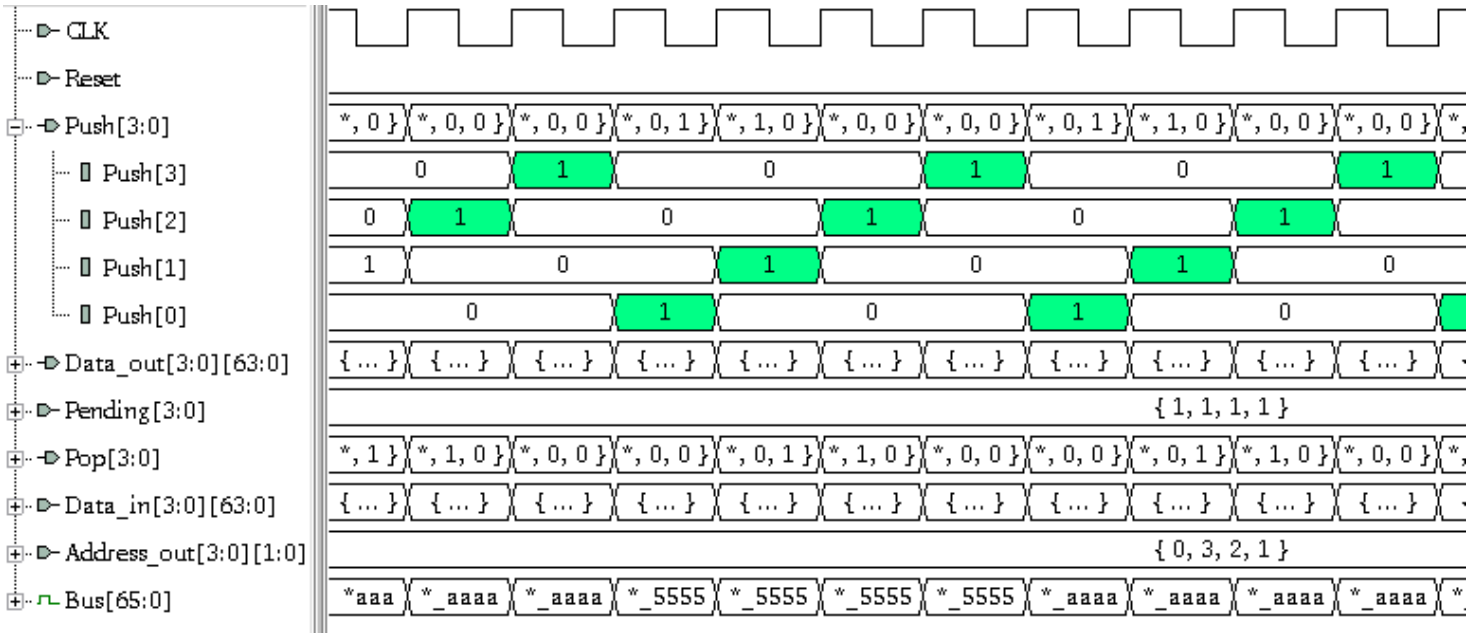

Figura. 32: Señales de puertos de entrada y salida del Bus paralelo con árbitro (4 puertos @72bit), se resaltan las señales de “Push”

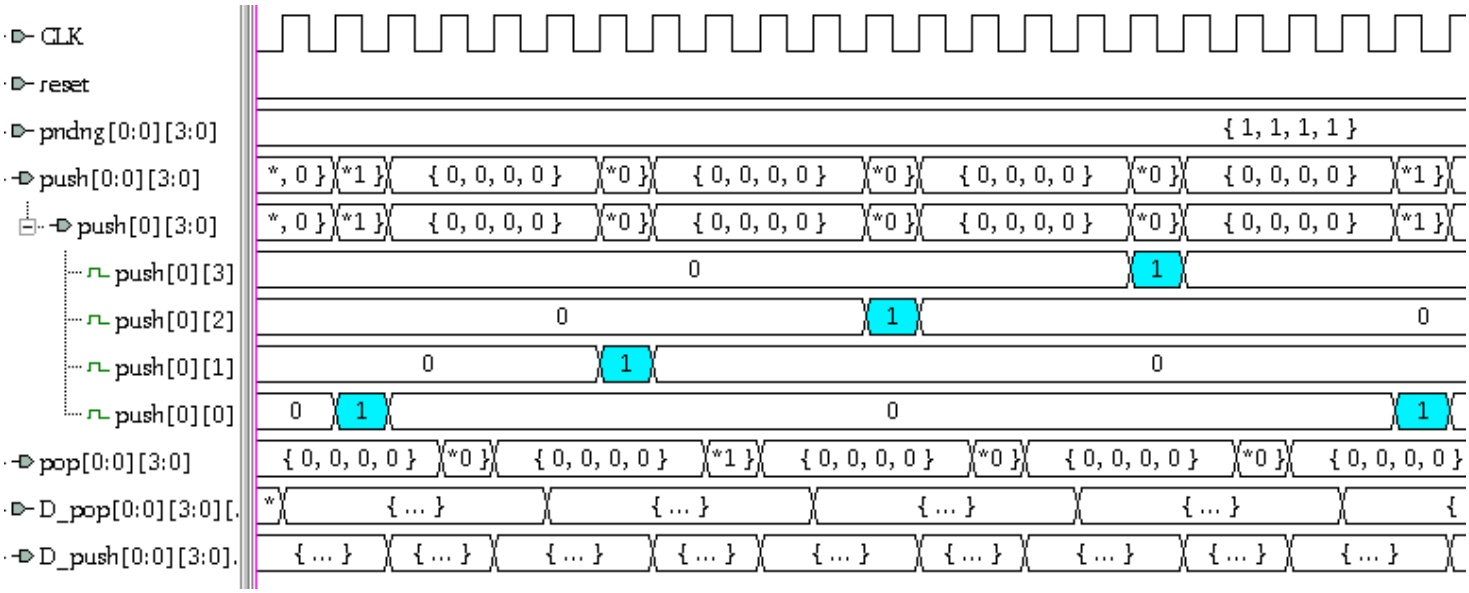

Figura. 33: Señales de puertos de entrada y salida del Bus paralelo sin árbitro (4 puertos @72bit), se resaltan las señales de "Push"

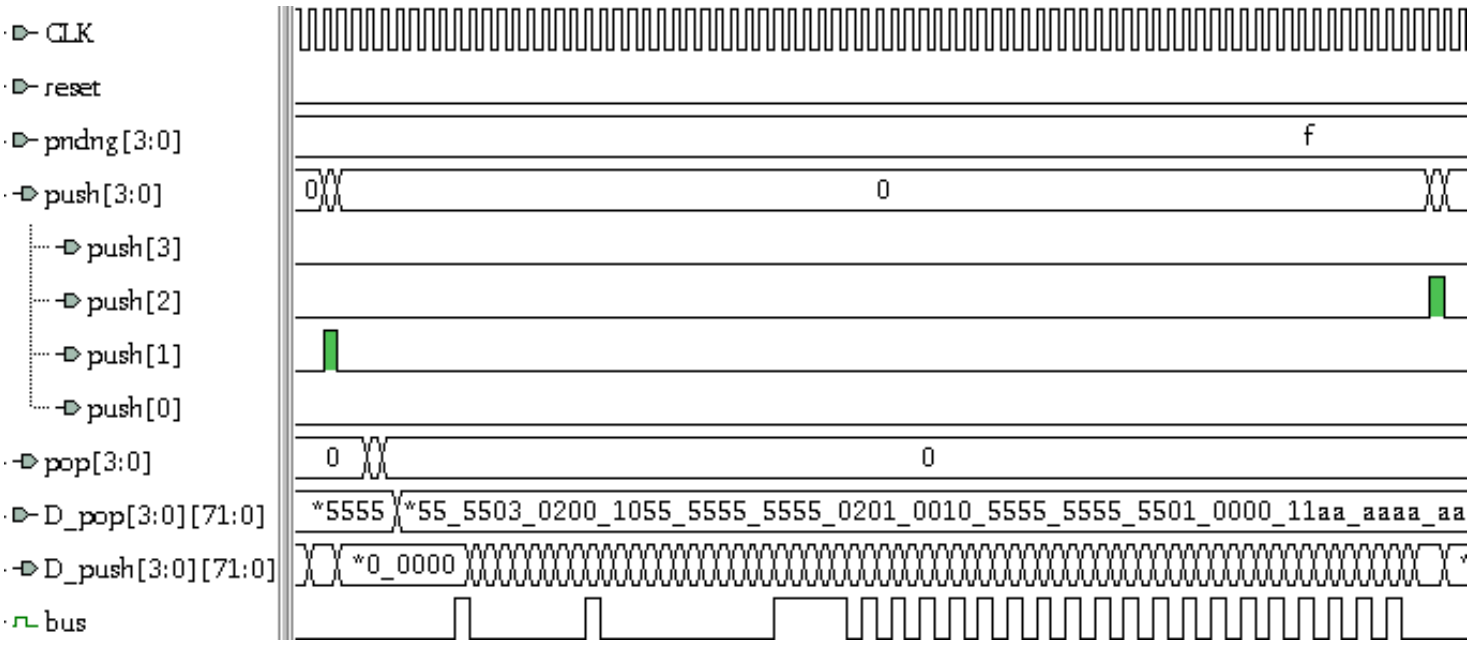

Figura. 34: Señales de puertos de entrada y salida del Bus serial (4 puertos @72bit), se resaltan las señales de "Push"

7.6 Desempeño de Malla rectangular

En la Figura 35 se detallan los resultados de área combinacional y no combinacional para una implementación de malla de 3 x 3 nodos, cabe destacar que el mayor aporte en área proviene de los registros FIFO de flip flops de los cuales se ubican 2 entre cada módulo y sus vecinos (24 en total), esto se evidencia en la gran área no combinacional. Ya que para todas las implementaciones, la cantidad y tamaño de los registros es el mismo, de esto se puede deducir que la diferencia en área vista en las figuras 35 y 36 es proporcional al área aportada por la lógica de los buses de datos.

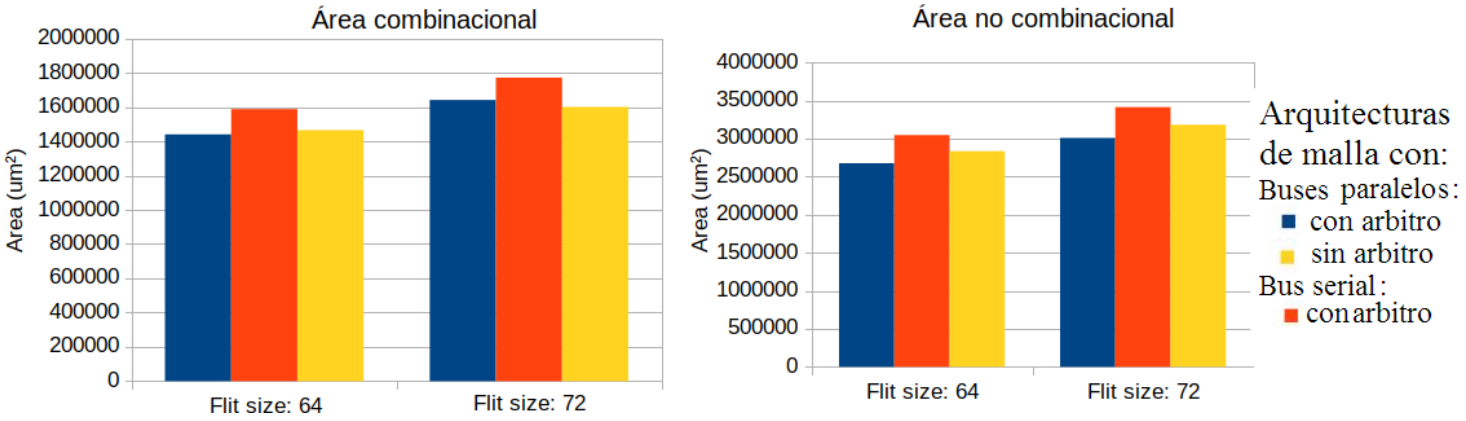

Figura. 35: Áreas combinacional y no combinacional para una malla 3x3 (Tabla 3)

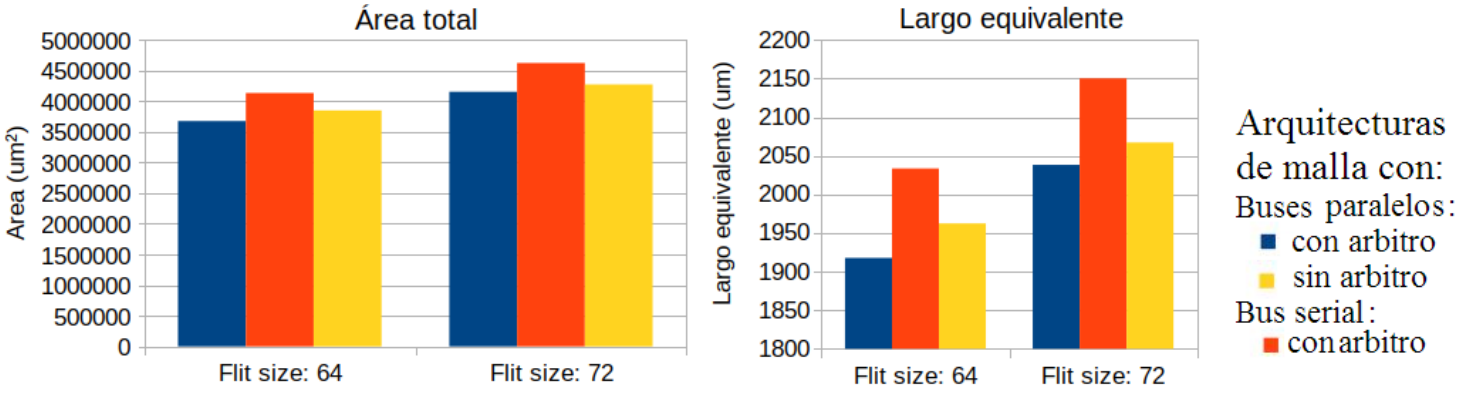

Figura. 36: Áreas de una malla 3x3 y su largo equivalente a un cuadrado (Tabla 3)

Bus	Parametros	Combinacional	No Combinacional	Total	Largo equivalente (μm)
Bus_M	Flit size: 64	1440161.60	2675564.59	3677586.84	1917.70
Bus_M	Flit size: 72	1641058.78	3007141.70	4154900.15	2038.36
Bus_R_a	Flit size: 64	1587488.37	3045454.52	4135646.04	2033.63
Bus_R_a	Flit size: 72	1771463.70	3412078.18	4624212.43	2150.40
Bus_R_p_n_a	Flit size: 64	1464652.51	2834313.05	3851147.26	1962.43
Bus_R_p_n_a	Flit size: 72	1600481.45	3180807.11	4273693.09	2067.29

Tabla. 3: Áreas de las mallas, obtenidas en síntesis lógica expresadas en μm^2

Para probar el consumo de potencia de las implementaciones de malla, se ingreso datos a cada uno de los puertos para dispositivos de la red, y como dispositivos de salida se asigno al dispositivo con el identificador siguiente al puerto, esto se realizo para aproximadamente 1kB de datos. En la figura 37 se detalla el consumo de potencia combinacional y no combinacional, como es de esperar el consumo de potencia no combinacional es mucho mayor al combinacional para cada bus por separado debido a que todos los buses presentan una mayor área no combinacional, además de la comparación entre los buses, resalta que el bus paralelo con árbitro consume mucha más energía que el bus serial y paralelo sin árbitro, esto es parcialmente visible en la figura 31 ya que en la malla implementada existen nodos con 3,4 o 5 puertos, los cuales deben seguir un comportamiento similar a las implementaciones de bus de 4 puertos, sin embargo la marcada diferencia entre las potencias consumidas (ver figura 38) se debe a la velocidad de transferencia de datos y como esta implica conmutar más rápido los registros de las memorias FIFO.

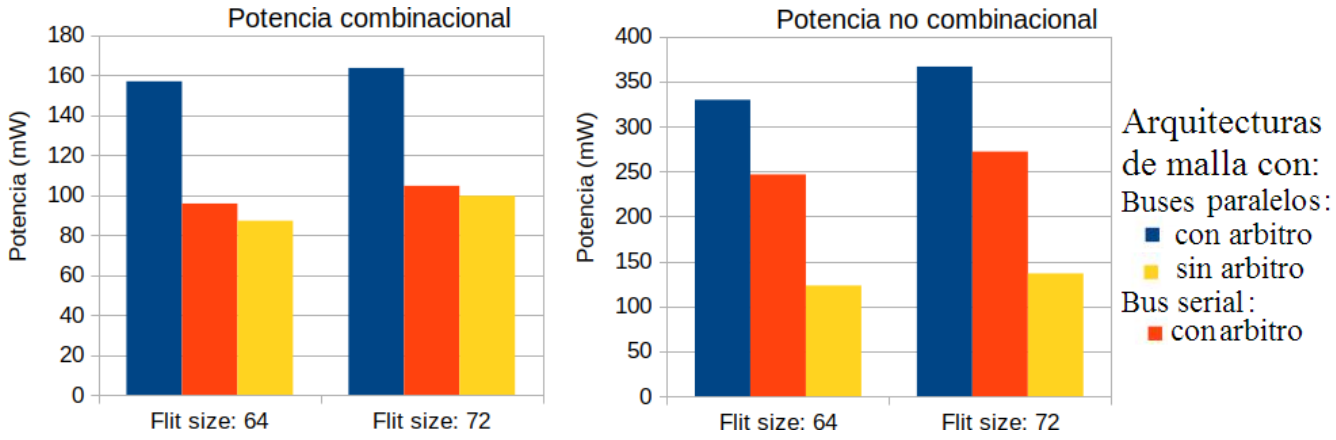

Figura. 37: Potencia combinacional y no combinacional para una malla 3x3@ 1kB (Tabla 4)

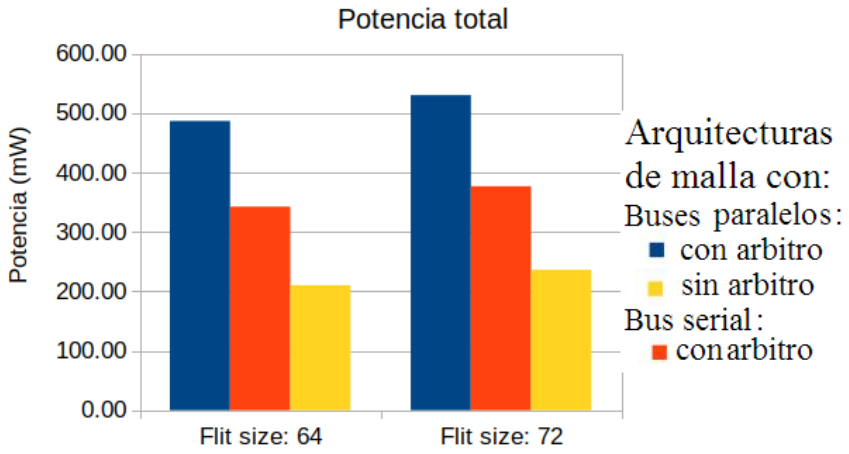

Figura. 38: Potencia total para una malla 3x3@ 1kB (Tabla 4)

Bus	Parametros	Combinacional (mW)	No Combinacional (mW)	Total (mW)
Bus_M	Flit size: 64	156.92	329.74	486.66
Bus_M	Flit size: 72	163.61	366.65	530.25
Bus_R_a	Flit size: 64	95.77	246.71	342.49
Bus_R_a	Flit size: 72	104.64	272.16	376.80
Bus_R_p_n_a	Flit size: 64	87.14	123.14	210.28
Bus_R_p_n_a	Flit size: 72	99.65	136.52	236.17

Tabla. 4: Potencias de las mallas, obtenidas en síntesis lógica

8 Conclusiones y Recomendaciones

8.1 Conclusiones

Las redes en chip tienen como propósito satisfacer necesidades específicas de ancho de banda, comunicación concurrente y latencia en sistemas con múltiples dispositivos, para lograr esto se basan en un modelo de tipo "pipeline" para la transmisión de datos, en el cual entre dos enrutadores solo se transfiere un Flit por ciclo de reloj.

La latencia de datos de una red de enrutadores está principalmente determinada por la lógica de direcciónamiento presente en los nodos de la red, esta lógica determina la velocidad de salida de datos, además de cuantos puertos de salida pueden estar activos a la vez. Otro factor importante es el mecanismo de enrutamiento de datos, su elección es critica ya que de este depende la susceptibilidad de la red a problemas como "Deadlock", "livelock" , "Starvation" y consecuentemente la lógica necesaria para contenerlos.

Con respecto a las arquitecturas de bus analizadas, se destaca que el arbitraje central tiene una importante repercusión en la velocidad de transferencia de datos en los enrutadores, habiendo una diferencia de entre 2 y 4 ciclos de reloj entre los "Push" de los buses paralelos.

La arquitectura de bus serial ocupa una gran área debido a la necesidad del uso de lógica no combinacional (Figura 30), la cual se requiere en serializadores y bloques de control del protocolo de comunicación, además del arbitraje.

8.2 Recomendaciones

Se recomienda realizar un estudio similar al descrito en este trabajo para un módulo de tipo "Crossbar" para evaluar hasta que cantidad de puertos es viable usar un "Crossbar" en un NoC, este dato puede ser usado para implementar topologías pequeñas o usar "Crossbars" en secciones de topologías que requieran una menor cantidad de puertos como por ejemplo en los bordes o esquinas de una malla bidimensional, en otras palabras implementarlos condicionalmente.

Como segunda recomendación, se propone centralizar los decodificadores de dirección que se encuentran en cada puerto, para esto se localiza un único decodificador de dirección que tiene como entrada la dirección del dispositivo de salida, presente en el bus interno del enrutador y su salida se distribuye a todos los puertos por las mismas líneas usadas antes para transmitir el valor del puerto de destino. Esta modificación disminuiría el área, pero debe evaluarse el impacto que pueda tener el retraso temporal sobre la ruta critica.

Como posibles mejoras a la implementación desarrollada se plantea permitir a la máquina de estados de escritura de los puertos, enviar una petición al árbitro en el caso de que al hacer un "Pop" de datos, se encuentre otro dato pendiente, esta mejora no fue implementada al tiempo de finalización de este proyecto.

En el transcurso de desarrollo de este trabajo se usó las memorias FIFO presentes en la biblioteca del proyecto, este módulo recientemente fue actualizado con el fin de permitir la recepción de datos en cada ciclo de reloj, de esta manera el desempeño en términos de velocidad puede ser mejorado modificando el funcionamiento del bus, que actualmente restringe la escritura a ciclo de reloj de por medio para un mismo puerto.

9 Referencias

[1] G. N. Khan, *SoC Bus Interconnect Structures*. Electrical and Computer Engineering Ryerson University, Ontario Canada, 2018.

[2] J. Heißwolf, *A Scalable and Adaptive Network on Chip for Many-Core Architectures*. Ingeniería eléctrica y tecnología de la información. en el Instituto de Tecnología de Karlsruhe (KIT), Alemania, 2014.

[3] D. U. Becker, *EFFICIENT MICROARCHITECTURE FOR NETWORK-ON-CHIP ROUTERS*. Stanford University, 2012.

[4] S. C. Santanu kundu, *Network-on-Chip*. 6000 Broken Sound Parkway NW, Suite 300 Boca Raton, FL: CRC Press by Taylor & Francis Group, LLC, 2014.

[5] M. N. M. Alireza Monemi, Chia Yee Ooi, *Low Latency Network-on-Chip Router Microarchitecture Using Request Masking Technique*. Hindawi Publishing Corporation, 2015.

[6] B. B. L. C. Fawaz Alazemi, Arash Azizimazreah, *Routerless Networks-on-Chip*. Oregon State University, USA, 2017.

[7] G. N. Khan, *NOC: Networks on Chip MPSoC:Multiprocessor System on Chip*. Electrical and Computer Engineering Ryerson University, Ontario Canada, 2018.

[8] M. S. L. Érika Cota, Alexandre de Morais Amory, *Reliability, Availability and Serviceability of Networks-on-Chip*. Springer, New York, 2012.

[9] M. I. M. K. S. E.-A. A. S. M. A. M. W. E.-K. Mahmoud A. Elmohr, Ahmed S. Eissa, *RVNoC: A Framework for Generating RISC-V NoC-based MPSoC*. 26th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, 2018.

[10] G. N. Khan, *NOC: Networks on Chip SoC Interconnection Structures*. Electrical and Computer Engineering Ryerson University, Ontario Canada, 2015.