

# Acelerador en Hardware basado en múltiples núcleos de procesamiento

UNA TESIS PRESENTADA  
POR  
HÉCTOR JESÚS CABRERA VILLASEÑOR  
AL  
DEPARTAMENTO DE DISEÑO ELECTRÓNICO

COMO PARTE DE LOS REQUISITOS  
PARA OBTENER EL GRADO DE  
DOCTOR EN CIENCIAS  
EN EL ÁREA DE  
DISEÑO DE DIGITAL

DIRECTOR DE TESIS  
DRA. SUSANA ORTEGA CISNEROS

CINVESTAV UNIDAD GUADALAJARA  
GUADALAJARA, JALISCO  
ABRIL 2017

# Acelerador en Hardware basado en múltiples núcleos de procesamiento

## RESUMEN

El desarrollo de nuevos procesos para la manufactura de circuitos integrados ha abierto la posibilidad de la fabricación de sistemas digitales contenidos en un solo encapsulado, comúnmente nombrados sistemas en-chip o SoC por sus siglas en inglés (*System on-Chip*). El beneficio de la fabricación de circuitos con alta densidad de transistores ha impactado el diseño de sistemas para dispositivos reconfigurables, abriendo la posibilidad de explorar metodologías de diseño que anteriormente resultaban prohibitivas por los altos requerimientos de elementos lógicos.

La aplicación de los fundamentos del diseño SoC en dispositivos reconfigurables busca extrapolar arquitecturas o conceptos probados en dispositivos de aplicación específica para generar mayores rendimientos en sistemas implementados en FPGAs. Sin embargo, la diferencia en la naturaleza de la tecnología objetivo puede diluir los beneficios si no se toma en consideración las restricciones propias de los dispositivos reconfigurables. El diseño de sistemas en-chip compuestos por múltiples elementos de procesamiento es un área que refleja de manera más drástica la disminución de ganancias de rendimiento al implementar de forma directa la experiencia de su contraparte de propósito específico.

En este documento se presenta una nueva estrategia para la asignación de cargas de trabajo en aceleradores basados en múltiples núcleos de procesamiento, así como una infraestructura de comunicación para el diseño de aceleradores en hardware reconfigurable basados en esta estrategia. La arquitectura desarrollada está compuesta de una *red en-chip*, también referida como *NoC - Network on-Chip*, diseñada tomando en cuenta las restricciones que impone los elementos reconfigurables de un dispositivo FPGA, así como las cargas de trabajo propias de un acelerador en hardware de propósito específico. La red provee de mecanismos para la inclusión de nuevos elementos de procesamiento, ampliando el espacio de aplicaciones al cual puede dar servicio.

# Hardware Accelerator based on Multiple Processing Cores

## ABSTRACT

The development of new integrated circuit fabrication processes has opened the possibility for the design of full digital systems deployed on a single chip. This kind of designs are commonly referred as System on-Chip (SoC). The fabrication of high transistor density integrated circuits has impacted positively the manufacturing process for reconfigurable devices, opening the exploration of new design methodologies which were considered impossible to set up on this kind of circuits due the high demand on logic elements.

Extrapolating architectures previously implemented on application specific integrated circuits to reconfigurable logic is one common mistake while using SoC design fundamentals. The difference on how the logic is deployed physically on a reconfigurable device diluted the benefits shown on other target technologies. The nature of the reconfigurable fabric must be taken on account when using architectures developed for ASIC devices. Systems on-chip containing multiple processing cores are prone to show drastic demises of performance when architectures designed for ASIC devices are used for implementations on reconfigurable devices.

This document presents a workload distribution strategy proposal for hardware accelerators based on multiple processing cores. Also, a communication infrastructure architecture for the design of reconfigurable hardware accelerators based on the proposed strategy is presented. The developed infrastructure is based on a Network On-Chip(NoC) that has been designed taking in account the constraints imposed by the reconfigurable fabric of the FPGAs, and the nature of typical workloads for single purpose hardware accelerators. The network provides a simple interface for the inclusion of new processing elements, broadening the application space of the work presented on this document.

# Índice general

1.	INTRODUCCIÓN	I
1.1.	Objetivos . . . . .	2
1.2.	Motivación . . . . .	3
1.3.	Desafíos . . . . .	7
1.4.	Contribuciones . . . . .	II
1.5.	Estructura de la tesis . . . . .	13
1.6.	Publicaciones . . . . .	14
2.	FUNDAMENTOS	15
2.1.	Sistemas en-Chip . . . . .	16
2.2.	Redes de interconexión . . . . .	20
2.3.	Conceptos básicos de NoCs . . . . .	23
2.4.	Modelo SPMD . . . . .	42
3.	TÓPICOS ESPECÍFICOS	43
3.1.	Topología: Malla . . . . .	44
3.2.	Control de flujo . . . . .	49
3.3.	Planificación de ruta: <i>Modelo basado en giros</i> . . . . .	54
4.	TRABAJO RELACIONADO	61
4.1.	Redes en-chip en FPGA . . . . .	61
5.	NODOS	69
5.1.	Protocolo de comunicacion . . . . .	70
5.2.	Arquitectura del router . . . . .	74
5.3.	Arquitectura de Interfaz de Red . . . . .	89
5.4.	Caso de Estudio: encriptador DES . . . . .	92

6.	ACCELERADOR MULTI-NÚCLEO	96
6.1.	Concepto . . . . .	98
6.2.	Módulos propios del acelerador multi-núcleo . . . . .	104
6.3.	Andamiaje de validación . . . . .	III
6.4.	Pruebas . . . . .	115
7.	RESULTADOS Y CONCLUSIONES	117
	REFERENCIAS	132

# Índice de figuras

1.1.	Secuencia simplificada de operación de un algoritmo <i>block matching</i> . a) La unidad básica de información es un cuadro de una secuencia de video. b) Los cuadros se dividen en macro bloques de dimensiones constantes. c) El macro bloque se especifica solo con el pixel central y la distancia al límite del bloque. Los macro bloques con información relevante son almacenados para la búsqueda de coincidencias en cuadros posteriores. d) Se inicia la búsqueda del macro bloque en cuadros posteriores de la secuencia de video. La búsqueda se limita a un área específica denominada región de interés. . . . .	4
1.2.	Estructura general de una red neuronal convolucional. El número de capas de convolución/submuestreo determinan la complejidad de los patrones a discernir. El resultado del procesamiento llevado a cabo por la red es la clasificación del patron de entrada a una de las categorías definidas durante el entrenamiento de la red neuronal. . . . .	5
1.3.	Detección de bordes en orientación vertical. a) muestra una imagen previa a la aplicación de un kernel de procesamiento. b) presenta el resultado de la aplicación del kernel ( 1-1 ) . . . . .	6
2.1.	Diagrama a bloques de un dispositivo Quark <sup>TM</sup> SoC X1000. El diseño de procesadores para ordenadores de escritorio está fuertemente relacionado con la filosofía SoC, al grado que procesadores de gama alta pueden incluir dentro del mismo silicio módulos de aceleración para el procesamiento de gráficos. . . . .	17

2.2.	Diagrama a bloques de MPSoC Epiphany. Epiphany es un ejemplo de una arquitectura con núcleos de procesamiento homogéneo, algunas aplicaciones donde se a utilizado este dispositivo son: Radios definidos por software ( <i>SDR</i> ), visión por computadora, procesamiento de sonido y video, redes neuronales y simulaciones de sistemas físicos. . . . .	19
2.3.	a) Enlace de 6 nodos mediante líneas de comunicación dedicadas entre terminales. b) Enlace de 6 nodos mediante una red de interconexión en forma de anillo. . . . .	22
2.4.	Nodo de red: N representa un nodo de red, EP simboliza un elemento de procesamiento e IR a una interfaz de red. <i>Canal</i> es el término utilizado para referirse a las líneas físicas de transmisión de datos entre un nodo y uno de sus vecinos de la red. Los nodos miembros de una red pueden ser heterogéneos. No existe un modelo único de interfaz de red, ya que diferentes tipos de elementos de procesamiento requieren la presentación de los datos de trabajo en diferentes formatos. . . . .	24
2.5.	a) Topología tipo torus de 9 nodos. b) Topología tipo anillo con 8 nodos. Cada línea de interconexión en el diagrama representa dos canales en direcciones opuestas, por ejemplo, la línea entre los nodos 1 y 2 de la red tipo anillo representa: un canal saliente del nodo 1 en dirección del nodo número 2 y un canal saliente del nodo número 2 con dirección del nodo 1. . . . .	25
2.6.	Tres instancias de una red con topología tipo malla o <i>mesh</i> . Las direcciones para cada nodo de red se forman de una dupla de coordenadas $\{x y\}$ . a) Rutas trazadas mediante algoritmos de ordenamiento dimensional. b) planificación de una ruta no mínima entre dos nodos de la red. c) Desbalance de carga en la red, el canal que vincula a los nodos $\{1,1\}$ y $\{0,1\}$ . . . . .	29
2.7.	Estructura de un mensaje. La división de un mensaje puede ser física o lógica, dependiendo del contexto de manejo en el cual se esté procesando la información. En la figura no se muestra la división de flits en phits. . . . .	32



2.13. Uso de canales virtuales. Un router con canales virtuales utiliza múltiples buffers, denominados canales virtuales, para evitar la congestión de la red debido a la espera de liberación de recursos. . . . .	41
3.1. Diagrama abstracto de una red en-chip. <i>PE</i> representa un Elemento de Procesamiento. La figura resalta los módulos de la red impactados por los parámetros descritos dentro de este capítulo. . . . .	44
3.2. Topología tipo malla. a) red con topología tipo malla, a cada nodo se le asigna una dirección compuesta de una tupla $(x, y)$ . b) Número de saltos requeridos para comunicar los elementos de procesamiento más distantes de la red. c) Los nodos al centro de la red tienen el mayor <i>grado</i> de conectividad. d) Ancho de banda en la bisección. . . . .	46
3.3. Línea de tiempo de proceso de transmisión de créditos. El nodo emisor (Nodo 1) resta un crédito con cada paquete que envía al nodo receptor. Cada crédito representa espacio de almacenamiento en los buffers del Nodo 2. El nodo receptor envía de vuelta un crédito con la salida de un paquetes de su medio de almacenamiento temporal. El escenario representado en esta figura supone nodos de red con espacio de almacenamiento suficiente para 1 solo paquete. . . . .	52
3.4. Giros previstos durante una transferencia a través de la red.a) giro desde $x$ en dirección a $y$ . b) giro desde $y$ en dirección a $x$ . c) Giro de $180^\circ$ , donde la dimensión de tránsito permanece sin alteración, sin embargo, la dirección de desplazamiento es alterada desde $x+$ a $x-$ . . . . .	55
3.5. Posibles relaciones cíclicas en redes con topologías tipo malla. Los apartados a) y b) muestran la forma más simple de un <i>deadlock</i> . El apartado c) muestra una dependencia cíclica compleja, la cual está formada por 6 giros y tránsito en direcciones opuestas. d) Giros prohibidos por el algoritmo XY para prevenir la formación de dependencias cíclicas. . . . .	56

3.6. Rutas planificadas por medio del algoritmo west-first minimal. Las rutas creadas por medio de este algoritmo permite sortear bloqueos gracias a su capacidad de adaptabilidad. La ruta con inicio en la esquina inferior derecha muestra los cambios de adaptabilidad de la ruta conforme es reevaluada en cada nodo de la red. . . . .	60
4.1. Diagrama a bloques de red PNoC. La comunicación entre islas permite escalar el número de unidades funcionales del sistema. Además, el uso de múltiples isla permite la distribución de la demanda de recursos en diferentes áreas del sistema. . . . .	62
4.2. La estructura central de almacenamiento para los paquetes entrantes a un router CuNoC representa una disminución en los recursos necesarios para su implementación, sin embargo, crea un cuello de botella para el envío rápido de información hacia su siguiente salto en la red. . . . .	63
4.3. RMBoC Utiliza un híbrido entre un medio compartido y una conmutación de circuitos, para incrementar su rendimiento requiere de un aumento de líneas de interconexión que pueden resultar costosas para sistemas en-chip.	
5.1. Cada nodo de red es una isla de procesamiento en el acelerador. Las líneas de interconexión entre nodos se denominan canales. EP: Elemento de procesamiento, IR: Interfaz de red y NR: Nodo Rebotador. . . . .	70
5.2. No existe una restricción en el número de flits para el transporte de información dentro de un paquete, sin embargo, el incremento de estas unidades de transporte acarrea consigo una mayor congestión en la red además de un incremento en la cantidad de almacenamiento temporal requerido por cada uno de los puertos del encaminador. El flit de cabecera contiene la información necesaria para permitir que un paquete sea entregado a su destino. . .	71
5.3. Formato de flit de cabecera. El uso del campo <i>origen</i> permite la implementación de algoritmos de encaminamiento basados en el modelo <i>odd-even</i> . En caso de omitir el uso del campo origen durante el proceso de planificación de ruta, su espacio puede anexarse al campo <i>usuario</i> para ampliar su longitud. .	72

5.4.	El protocolo de comunicación involucra un intercambio de señales ( <i>handshake</i> ) de dos pasos. La transmisión inicia con la inversión de las señales del par diferencial (ciclo 2), el intercambio finaliza con la aserción de la señal rec_crt la cual indica que el router destino tiene espacio disponible para la recepción de un paquete adicional. . . . .	74
5.5.	Diagrama general de un segmento de los caminos de control y datos entre un puerto de salida y uno de entrada. Glosario: UMPE - Unidad de Manejo de Protocolo de Entrada. UCPE - Unidad de Control de Puerto de Entrada. UCC - Unidad de Control de Crossbar. UMPS - Unidad de Manejo de Protocolo de Salida. SCC - Segmento de Control de Crossbar. UGP - Unidad de Generación de Peticiones. PDE - Puerto de Entrada. PDS - Puerto de Salida.	76
5.6.	El buffer dentro de un puerto de entrada funge como registro para el flit en transito durante el ciclo de reloj actual. . . . .	79
5.7.	Forma de onda del intercambio de un paquete entre dos routers. Se requiere de 8 ciclos de reloj para que un crédito realice un viaje redondo entre los dos routers. . . . .	80
5.8.	Todos los puertos de entrada del router son estructuras homogéneas, de manera que el IP es reutilizable y presenta el mismo comportamiento sin importar al canal que se encuentre asignado. . . . .	81
5.9.	a) Maquina de estados finitos de entrada (MEFE). b) Maquina de estados finitos de salida (MEFS). . . . .	83
5.10.	Cada unidad SCC genera el vector de control para cada uno de los segmentos del crossbar. Los segmentos están configurados para mantener sus líneas de salida en cero mientras no se encuentren paquetes en transito, la inclusión de una un valor constante no repercute en el numero de entradas al multiplexor, solo en la señales de control del mismo. . . . .	84

5.11.	Las unidades SCC son responsables de mantener el control del numero de créditos disponibles en el router vecino inmediato, por lo cual reciben de manera directa la señal de recepción de créditos. El tiempo de proceso de un crédito es igual a un ciclo de reloj debido al cambio de estado de la maquina de estados finito que se encarga de administrar el proceso. La señal 'pet_vec' representa el vector formado por todas las peticiones enviadas a la unidad SCC desde los puertos de entrada ligados a ella. . . . .	85
5.12.	a) Implementación tradicional de arbitro con sistema de prioridad dinámica. b) Implementación de arbitro para dispositivos reconfigurables. La eliminación del ciclo combinacional en la cadena de acarreo del arbitro es necesario para su porte a dispositivos FPGA. . . . .	87
5.13.	a) Maquina de estado finito de la unidad MEFCC. b) Maquina de estado finito para la unidad MEFMC. . . . .	89
5.14.	Interfaz de red. a) la interfaz de red se conecta al encaminador como un vecino más de la red. b) Las unidades UCBDE y UCBDS son versiones modificadas de la unidad UCPE del puerto de entrada. . . . .	90
5.15.	Las líneas de comunicación entre interfaz y elemento de procesamiento están definidas para acoplarse al formato utilizado por el elemento funcional. En este ejemplo se favorece una interfaz con de carga paralela de datos. . . . .	93
5.16.	Las líneas de comunicación entre interfaz y elemento de procesamiento están definidas para acoplarse al formato utilizado por el elemento funcional. En este ejemplo se favorece una interfaz con carga de datos paralela. . . . .	94
5.17.	Unidad UODS. . . . .	95
6.1.	Características en aceleradores convencionales. a) El procesamiento de información se lleva a cabo en un nodo específico de la red. b) Cada paquete portando el resultado de un proceso se dirige a una puerta de salida preestablecida, compitiendo con los recursos de transporte con los paquetes de recién ingreso al sistema. c) El uso de políticas desbalanceadas para la asignación de cargas de trabajo de un nodo puede saturar rápidamente varios canales de comunicación en el acelerador. . . . .	100

6.2. Arquitectura propuesta. a) el acelerador está compuesto por una granja de núcleos comunicados por medio de una red en-chip, se hace uso de 3 tipos de nodo: Nodos terminales para la inyección y/o recepción de paquetes de la red, nodos frontera para re circular paquetes que no han sido procesados por una unidad funcional, y nodos centrales, los cuales procesan y envían los resultados fuera del acelerador. b) los nodos frontera solo tienen acceso a la red mediante un canal de comunicación. c) Los nodos terminal comparten micro-arquitectura con los nodos centrales, sin embargo, solo un nodo terminal es capaz de recibir y enviar paquetes fuera de la red. . . . .	101
6.3. Tránsito de un paquete a través del acelerador. a) en primera instancia un paquete ingresa a través de un nodo terminal. El paquete buscará la primera unidad funcional disponible para el procesamiento de información. b) en caso de alcanzar un nodo frontera sin haber procesado su información, el paquete cambia su dirección y continúa la búsqueda de una unidad funcional disponible. c) Una vez terminado el procesamiento de información de un paquete, este reingresa a la red en búsqueda de un nodo terminal para su egreso del sistema. . . . .	103
6.4. Arquitectura de nodo frontera. El camino de datos de este módulo solo lleva a cabo la sustitución de la dirección destino de los paquetes en tránsito. . .	105
6.5. a) Los nodos se etiquetan de acuerdo a su posición en la red. Vale la pena notar que todos los nodos centrales formarán una sub-red tipo malla al centro del acelerador. Las etiquetas para cada nodo sirven para reconocer que trabajo desempeña en el acelerador, además de facilitar la selección de instancia correcta durante el proceso de síntesis del sistema. b) Los nodos frontera deben de tomar en cuenta las restricciones impuestas por el algoritmo de encaminamiento para la selección de direcciones válidas en sus paquetes de salida. . . . .	107
6.6. a) Módulo planificador de ruta para puerto $x-$ del nodo [1,3]. b) Módulo planificador de ruta para puerto $x-$ del nodo [5,3], el nodo en cuestión maneja la salida de red a través del puerto $x+$ . . . . .	110
6.7. Segmento de calculo de ruta en el puerto de entrada $x-$ mediante el uso del algoritmo $XY$ . . . . .	III

6.8. El núcleo de la red integra todos los nodos necesarios para su funcionamiento, sin embargo no incluye un mecanismo para la inyección de paquetes. El arnés provee de los medios para evaluar el núcleo de la red, incluyendo medios para el envío y recepción de paquetes. . . . .	114
7.1. Histogramas de latencia de recepción de paquetes para puertos en dirección $x-$ . . . . .	123
7.2. Histogramas de latencia de recepción de paquetes para puertos en dirección $x+$ . . . . .	124

## Agradecimientos

AGRADEZCO a mis padres por el apoyo brindado durante mi tiempo en el programa de posgrado. A la doctora Susana Ortega Cisneros por el apoyo y dirección durante el desarrollo de este documento. A mis compañeros que los grandes momentos vividos en el laboratorio y a todos los miembros del equipo de CINVESTAV unidad Guadalajara.

Finalmente agradezco a CONACYT por el apoyo otorgado para llevar a cabo este posgrado.



# 1

## Introducción

Los dispositivos reconfigurables han transitado de módulos de interconexión entre miembros de un sistema electrónico(*glue logic*) ha receptáculos para la integración de sistemas digitales complejos bajo un empaquetado único. El éxito de estos dispositivos se deriva de su capacidad para la implementación de múltiples unidades funcionales con ejecución paralela de tareas.

El incremento en la cantidad de elementos programables, así como la integración de bloques dedicados a tareas específicas como: bloques para el procesamiento de señales (*DSP*), unidades de aritmética de punto flotante, procesadores de propósito general y bloques de memoria, han abonado para convertir a los dispositivos reconfigurables en plataformas atractivas

para la implementación de aceleradores en hardware. El espacio de diseño para aceleradores en hardware reconfigurable ha sido fuertemente influenciado por las arquitecturas desarrolladas para tecnologías de fabricación de circuitos integrados de propósito específico, las cuales, no generan los mismos rendimientos al ser portadas a lógica reconfigurable.

El resto del capítulo se desenvuelve en el siguiente orden: en la sección 1.1 se formalizan los objetivos de esta tesis. La sección ?? enumera áreas de oportunidad para el sistema desarrollado en este trabajo de investigación. Dentro de la sección ?? se enumera algunos de los retos que se abordaron durante el desarrollo de la tesis. La sección 1.4 enumera las principales contribuciones obtenidas como resultado de este trabajo. En la sección 1.5 se ofrece un desglose de la organización del contenido del presente documento. Finalmente, la sección 1.6 ofrece un listado de las publicaciones producto del desarrollo de este trabajo.

## OBJETIVOS

El presente trabajo explora el espacio de diseño de aceleradores de procesamiento interconectados mediante redes en-chip para dispositivos reconfigurables. De manera específica, el trabajo busca proponer una plataforma para aceleradores en hardware reconfigurable que explote de manera eficiente los recursos y restricciones propios de esta tecnología.

## OBJETIVOS PARTICULARES

De manera particular, la arquitectura descrita en este trabajo busca incursionar en los siguientes puntos de diseño:

- Diseño de una estructura de interconexión con baja demanda de elementos lógicos para su implementación. Las decisiones de diseño son orientadas por las restricciones

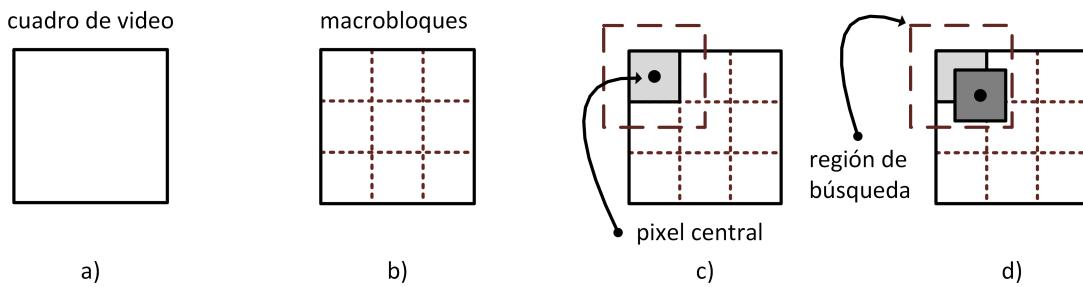
de colocación de recursos dentro del dispositivo reconfigurable.

- Diseño de una interfaz para el intercambio de información entre núcleos de procesamiento y medio de comunicación. El modulo de acoplamiento debe de ocultar los detalles del protocolo de red a las unidades funcionales del sistema.
- Desarrollo de una estrategia de distribución de carga de trabajo entre núcleos. El modelo de tráfico debe de tomar en cuenta las restricciones de área de la tecnología destino, optando por mecanismos eficientes respecto a área para el calculo de trayectoria de paquetes de información a través de la infraestructura de interconexión.
- Propuesta de un modelo de interacción entre un acelerador y generadores de datos a procesar.

## MOTIVACIÓN

El proceso de investigación llevado a cabo para el desarrollo de esta tesis está motivado de manera principal por aplicaciones que pueden descomponer su operación en unidades homogéneas e independientes de procesamiento.

- **ESTIMACIÓN DE MOVIMIENTO - BLOCK MATCHING<sup>25</sup>:** Los algoritmos de estimación de movimiento basados en cotejo de bloques tienen como objetivo el localizar correspondencias de *macro bloques* entre diferentes cuadros de una secuencia de video. La aparición entre cuadros de un patrón representativo de un objeto permite obtener un estimado de la dirección y velocidad de desplazamiento de este mismo. Además, la información proporcionada por el algoritmo puede ser utilizada para la detección de in-

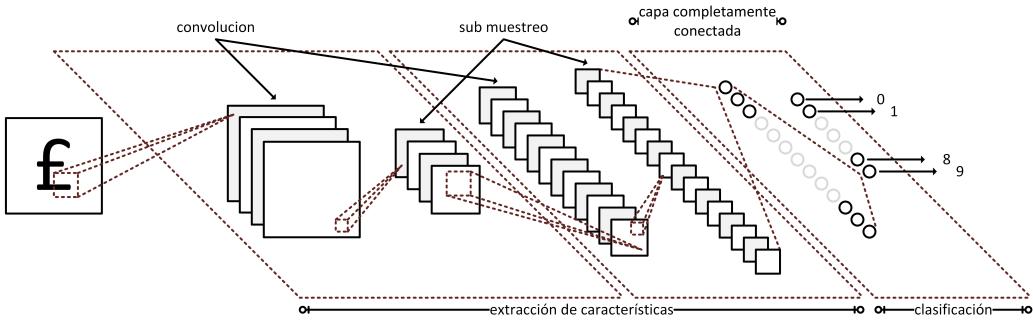


**Figura 1.1:** Secuencia simplificada de operación de un algoritmo *block matching*. a) La unidad básica de información es un cuadro de una secuencia de video. b) Los cuadros se dividen en macro bloques de dimensiones constantes. c) El macro bloque se especifica solo con el pixel central y la distancia al límite del bloque. Los macro bloques con información relevante son almacenados para la búsqueda de coincidencias en cuadros posteriores. d) Se inicia la búsqueda del macro bloque en cuadros posteriores de la secuencia de video. La búsqueda se limita a un área específica denominada región de interés.

formación redundante en la secuencia de video para facilitar el proceso de compresión de la fuente.

Block matching consiste en 3 pasos fundamentales: en primer lugar el cuadro de la secuencia de video se divide en macro bloques de tamaño constante. A continuación se delimita una región de búsqueda, donde el algoritmo llevará a cabo el proceso de comparación de macro bloques. Finalmente, se procede a la búsqueda de correspondencias de bloques entre cuadros. El flujo de procesamiento se muestra en la figura 1.1. La operación de comparación entre bloques resulta particularmente demandante en casos donde las regiones de búsqueda son amplias o cuando se debe de llevar a cabo el seguimiento de múltiples objetos en cada cuadro de video.

- **REDES NEURONALES CONVOLUCIONALES PARA EL PROCESAMIENTO DE IMÁGENES<sup>10</sup>:** Esta clasificación de redes neuronales sigue el esquema de procesamiento denominado *propagación hacia adelante*. El arreglo de neuronas que conforma la red se

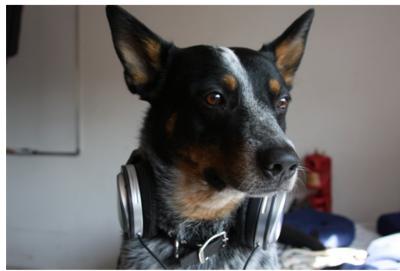


**Figura 1.2:** Estructura general de una red neuronal convolucional. El número de capas de convolución/submuestreo determinan la complejidad de los patrones a discernir. El resultado del procesamiento llevado a cabo por la red es la clasificación del patrón de entrada a una de las categorías definidas durante el entrenamiento de la red neuronal.

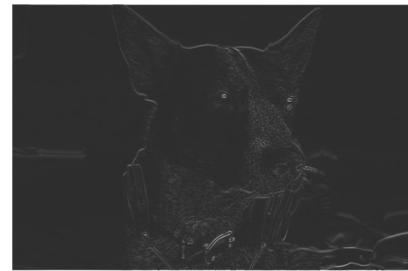
distribuye en forma de una malla que se sobrepone al campo visual de la aplicación. Su funcionamiento está inspirado en un modelo biológico de procesamiento de imágenes, por lo que sus aplicaciones más comunes es el reconocimiento de patrones.

Una red neuronal convolucional es un arreglo jerárquico compuesto por capas de neuronas que efectúan tres operaciones: convolución para la extracción de características de alto nivel, capas de sub muestreo para la extracción de patrones de interés, y una capa con conexión completa para la clasificación del objeto alimentado a la red. Las diferentes capas de una red convolucional operan de manera similar a las células simples y complejas que se encuentran en la capa 1 del córtex visual<sup>28</sup>. El esquema de conexión entre capas de convolución y capas de sub muestreo, así como el tipo de entrenamiento para la red, son características que distinguen diferentes implementaciones de este tipo de sistemas. La figura 1.2 muestra el modelo general de una red neuronal convolucional.

- **PROCESAMIENTO DE IMÁGENES - KERNELS:** En el contexto de procesamiento de imágenes, un kernel o máscara, es una matriz que permite la aplicación de operaciones co-



a)



b)

**Figura 1.3:** Detección de bordes en orientación vertical. a) muestra una imagen previa a la aplicación de un kernel de procesamiento. b) presenta el resultado de la aplicación del kernel ( $\begin{bmatrix} 1 \\ -1 \end{bmatrix}$ )

mo: desdibujado, definición, realce, detección de bordes, entre otras. El uso de kernels implica la ejecución de una operación de convolución entre la máscara y un área de la imagen de entrada. La operación de convolución se denota con el operador de multiplicación, sin embargo la operación no es equivalente al producto de ambos bloques de información, sino a la operación definida en la ecuación I.I.

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \times \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} = (1*a) + (2*b) + (3*c) + (4*d) + (5*e) + (6*f) + (7*g) + (8*h) + (9*i) \quad (\text{I.I})$$

- **CIFRADO - ENCRYPTADORES POR BLOQUES:** Un encriptador por bloques es la implementación de un algoritmo determinístico que opera sobre conjuntos de bits de dimensiones constantes denominados bloques. Esta familia de encriptadores trabaja con llaves simétricas, en otras palabras utiliza la misma contraseña para el cifrado/descifrado de datos. La teoría detrás de este tipo de encriptadores fue presentada por Claude

Shannon<sup>68</sup> en 1949.

Los cifradores por bloques llevan a cabo su operación a lo largo de múltiples rondas, las cuales consta de una combinación de operaciones simples como sustituciones y permutaciones. Cada ronda de encriptación utiliza llave derivadas de la llave original, las cuales también son calculadas ronda a ronda.

Los algoritmos DES<sup>54</sup>(*Data Encryption Standard*) y AES<sup>15</sup>(*Advanced Encryption Standard*) son dos ejemplos populares de esta familia de encriptadores.

## DESAFÍOS

El diseño de aceleradores en hardware reconfigurable ha evolucionado con el desarrollo de nuevos métodos de fabricación y nuevas arquitecturas para los dispositivos, sin embargo cada nueva generación de hardware plantea nuevos desafíos para obtener un mejor rendimiento.

- **RESTRICIONES POR ARQUITECTURA DE HARDWARE:** Un conjunto de recursos predefinidos así como una estructura estática, pero abundante de elementos de interconexión, hacen del diseño de circuitos digitales para dispositivos reconfigurables un área de trabajo distinta al diseño tradicional para dispositivos de propósito específico. Este documento se centra en dispositivos reconfigurables FPGA(*Field Programmable Gate Array*).

Un diseño en hardware reconfigurable debe de tomar en cuenta desde su inicio la organización de recursos internos del dispositivo. Por ejemplo, la arquitectura ASMBL<sup>32</sup> (*Advanced Silicon Modular Block*) de Xilinx organiza los recursos internos del FPGA

en columnas. La distribución de recursos crea una restricción positiva a nivel local, ya que unidades funcionales de un sistema tienen concentrados en un vecindario recursos necesarios para su tendido, entre estos recursos se pueden enumerar: elementos de memoria, multiplicadores, LUTs (*Lookup Table*), árboles de distribución de reloj, etc... Desde el punto de vista de sistema, la restricción resulta desafiante durante el proceso de cierre de tiempos, ya que si dos unidades funcionales del diseño requieren los mismo recursos, una de ellas deberá desplazarse a otra región del dispositivo creando caminos de propagación más largos afectando la frecuencia de operación de todo el diseño. No solo la velocidad del sistema se ve afectada, una planeación pobre durante el inicio del flujo de diseño puede derivar en la pelea por recursos entre unidades, por ejemplo, una unidad funcional puede estar acaparando todo los bloques de memoria embebida de una región obligando a una estructura de interconexión a asentarse en una región remota del dispositivo o simplemente a no poder ser colocada por falta de recursos.

La diferencia estructural entre dispositivos de diferentes fabricantes acentúa este problema, ya que las primitivas específicas de un dispositivo no son portables entre miembros de distintas familias y entre distintos fabricantes.

- **INTERCONEXIÓN DE UNIDADES FUNCIONALES:** Uno de los paradigmas comunes para la construcción de aceleradores consiste en la generación de múltiples unidades funcionales para ejecutar de manera distribuida una tarea. El principal problema con este acercamiento radica en la distribución de la información a ser procesada entre todos los núcleos de ejecución. El uso de estructuras tipo bus ha sido reemplazado por el uso de redes en-chip<sup>3,67,46</sup>, las cuales representan la adaptación del concepto de comunicaciones entre computadoras en una red de área local aplicado a la transmisión de

información entre unidades funcionales de un sistema digital en un solo encapsulado.

Las redes en-chip permiten el transporte información a través de un medio de comunicación distribuido, y se presentan como uno de los candidatos para convertirse en el medio de interconexión estándar para diseños formados por múltiples elementos de procesamiento. Gran parte de los esfuerzos en esta área están enfocados en el desarrollo de mecanismo para proporcionar transacciones de información seguras entre miembros de la red, sin embargo, los mecanismos para proveer un mayor grado de calidad en el servicio de transporte de datos trae consigo un mayor consumo de recursos, al mismo tiempo que agregan latencia derivada los servicios anteriormente referidos.

- **RE USABILIDAD:** El concepto de reutilizar bloques funcionales, comúnmente denominados IP (*Intellectual Property*), es una práctica común el desarrollo de sistemas digitales en FPGAs. Un bloque re utilizable presenta una interfaz definida de entradas y salidas, así como registros de configuración para facilitar su integración en nuevos diseños. Cuando se trata de bloques que interactúan con un medio de comunicación, se requiere el uso de módulos dedicados al manejo del protocolo de transmisión de datos implementado por el medio.

El uso de protocolos complejos conlleva al incremento en los recursos necesarios para implementar la lógica de manejo de protocolo, penalizando la latencia de transmisión de un mensaje con los tiempos necesarios para la decodificación de la información de transporte.

- **HARDWARE RECONFIGURABLE NO ES ASIC:** Gran parte de las arquitecturas de aceleradores en hardware reconfigurable se encuentran fuertemente influenciados por las

reglas de diseño establecidas para circuitos de aplicación específica, las cuales no toman en cuenta las restricciones que un diseño reconfigurable afronta. La extrapolación de arquitecturas exitosas en diseños de aplicación específica no siempre producen los mismo resultados en su contraparte reconfigurable.

El desarrollo de sistemas con múltiples núcleos de procesamiento de propósito general formados con *núcleos-suaves*<sup>33,34</sup> (*Softcores*) es un ejemplo de la degradación de rendimiento de un sistema. El uso de procesadores de propósito general en diseños con múltiples núcleos para dispositivos ASIC toma ventaja de las altas frecuencias de operación de esta tecnología, mitigando en gran parte la penalización de rendimiento generada por el uso de núcleos capaces de ejecutar diversos tipos de tareas. Sin embargo, el uso de softcores implementados en la lógica reconfigurable de una FPGA limita la frecuencia de operación del sistema al orden de los 250 MHz.

El uso de bloques dedicados de un FPGA para la implementación de las unidades funcionales de un sistema provee de reducciones en el área necesaria para su implementación, así como incrementos en las frecuencias de operación de estas unidades.

La organización interna de lógica reconfigurables así como de elementos rígidos dentro de un FPGA son factores que pueden diluir el beneficio obtenido de la aplicación de conceptos de diseño formulados en otras tecnologías de fabricación. Por ejemplo, un diseño en FPGA puede no alcanzar sus objetivos de rendimiento debido al retardo de propagación derivado de la distancia entre bloques DSP. En otro escenario, la disputa entre elementos de procesamiento y módulos de comunicación por recursos de memoria puede llegar a descartar una familia de FPGAs como medio viable para la implementación de un sistema. Finalmente, la falta de flexibilidad para la colocación

de módulos en áreas específicas del dispositivo dificulta la implementación de bloques altamente interconectados de un sistema.

## CONTRIBUCIONES

Las principales contribuciones de esta tesis en el desarrollo de arquitecturas para la implementación de aceleradores en hardware reconfigurable se describen a continuación:

- **MODELO DE PROCESAMIENTO:** Presento un nuevo modelo para la inyección, distribución, procesamiento y recuperación de resultados para aceleradores en hardware reconfigurable. En esta propuesta, la tarea de procesamiento se distribuye en múltiples unidades funcionales, sin embargo, cada paquete de información no busca un núcleo en particular para su procesamiento. En lugar de dirigirse a un punto en particular del sistema, cada paquete de información hace una consulta en cada núcleo del acelerador para conocer si este se encuentra disponible para la recepción de paquetes, en caso de disponibilidad por parte del procesador el paquete de datos abandona el medio de interconexión para ingresar a la unidad funcional.

Los paquetes que han sido procesados por un núcleo del sistema reingresan al medio de interconexión para su transporte a los puertos de salida del acelerador.

- **ALGORITMO DE ENCAMINAMIENTO:** Se alteró el funcionamiento de algoritmos de encaminamiento tradicionales para medios de interconexión a fin de dar soporte al modelo de procesamiento propuesto en este trabajo. Se ofrece la arquitectura en hardware de la implementación de versiones personalizadas de los algoritmos de encaminamiento *DOR XY* y *West First Minimal*.

- **MODELO DE INTERFAZ IP - MEDIO DE INTERCONEXIÓN:** Presento un modelo de interfaz ligera para la interconexión de bloques funcionales preexistentes con el acelerador en hardware de este trabajo. El modelo incluye la descripción de las señales para el manejo del protocolo de transferencia de información a través del medio de interconexión, así como un mapa de registros mínimo para la transferencia de datos entre la interfaz y el núcleo de procesamiento.

El modelo establece sólo el mínimo de reglas para llevar a cabo una transferencia de datos, dejando abierta la arquitectura para la adición de elementos para la mejora en la calidad de servicio, o modelos más complejos de intercambio de información.

- **ARQUITECTURA DE ACELERADOR:** Se presenta una nueva arquitectura a nivel sistema para la implementación de aceleradores de hardware. Esta nueva arquitectura describe la infraestructura de interconexión entre bloques funcionales, la interfaz con los núcleos de procesamiento, y los bloques para la inserción y extracción de datos del acelerador. Como parte del diseño de la arquitectura se ofrece una aplicación a manera de prueba de concepto para exemplificar el funcionamiento del sistema.
- **IMPLEMENTACIÓN:** Toda la investigación realizada durante esta tesis, incluyendo código fuente, se encuentran disponibles en el siguiente enlace: <https://github.com/hcabrera-exa>. El diseño se encuentra liberado bajo la licencia GNU V3<sup>22</sup>.

El repositorio mencionado con anterioridad contiene los archivos de descripción de hardware para cada bloque perteneciente al acelerador. Un proyecto de ejemplo para *Vivado 2015.2*<sup>34</sup> se incluye en el repositorio.

## ESTRUCTURA DE LA TESIS

El trabajo se encuentra organizado con la siguiente estructura: El capítulo 2 presenta una introducción de los conceptos fundamentales para el desarrollo de sistemas contenidos en un solo circuito integrado, así como los fundamentos del uso de redes de interconexión para la comunicación de múltiples núcleos dentro de un dispositivo. El capítulo 3 aborda a detalle conceptos de redes de interconexión para sistemas en-chip. Solo conceptos relevantes para este trabajo se abordan en la sección anteriormente mencionada. El capítulo 4 presenta una captura del estado del arte de redes de interconexión para múltiples núcleos, haciendo énfasis en trabajos desarrollados para dispositivos reconfigurables.

El capítulo 5 presenta la arquitectura estándar para los nodos de procesamiento que forman el acelerador basado en múltiples núcleos. En esta sección se detalla las estructuras de hardware que lo conforman así como la estructura de los paquetes de información utilizados para encapsular los datos de trabajo para los núcleos de procesamiento. Se invita al lector a visitar el repositorio ([github.com/hcabrera-/exa](https://github.com/hcabrera-/exa)) de código fuente de este trabajo para aclarar detalles a nivel de micro arquitectura del modulo descrito.

Finalmente el capítulo 6 presenta al acelerador como un sistema digital, describiendo la filosofía de distribución de cargas de trabajo que lo diferencia de trabajos previos. Se incluye como parte de esta sección los módulos de soporte para llevar a cabo pruebas funcionales y de rendimiento del sistema, así como resultados de escenarios estándar de evaluación.

## PUBLICACIONES

1. S. Ortega-Cisneros; H. J. Cabrera-Villaseñor; J. J. Raygoza-Panduro; F. Sandoval; R. Loo-Yau. Hardware and Software Co-design: An architecture proposal for network-on-chip switch based on bufferless data flow. *Journal of Applied Research and Technology*. Vol. 12, No. 1, pp. 153-163. 2014.
2. Miguel Ángel Carrazco Díaz; Susana Ortega Cisneros; Adrian Pedroza de La Cruz; Hector Cabrera Villaseñor; Juan José Raygoza Panduro; Jorge Rivera Domínguez. Hardware / Software Co-design for Acceleration of Image Processing using FPGA. IX Southern Programmable Logic Conference. Buenos Aires, 5-7 November, 2014.
3. Héctor Cabrera; Susana Ortega Cisneros; Juan José Raygoza Panduro; Rivera J. Implementation of a spiking digital neuron in reconfigurable hardware. XIV Jornadas de Computación Reconfigurable y Aplicaciones (JCRA 2014). Valladolid, 17-19 Sept. 2014. ISBN-10: 84-697-0971-2.
4. Héctor Cabrera; Susana Ortega Cisneros; Juan José Raygoza Panduro; Juan Luis del Valle. Profiling Networks On-Chip Performance with a Software Simulator. Congreso Iberoamericano de Instrumentación y Ciencias Aplicadas (CIICA 2013) Sn. Francisco de Campeche, Campeche, México 28-31 de octubre, 2013.

# 2

## Fundamentos

El objetivo del presente capítulo es el ofrecer una introducción a los conceptos base para el desarrollo de este trabajo. La sección 2.1 Sistema en-Chip ofrece una vista general de esta metodología de diseño, 2.2 Redes de Interconexión ofrece un resumen de los medios utilizados para el transporte de información entre módulos de un sistema digital. La sección 2.3 Conceptos básicos de NoCs presenta las características que definen a estos medios de comunicación intra chip. Finalmente, la sección 2.4 Modelo SPMD, ofrecen un resumen de los conceptos técnicos sobre los cuales se sustenta la arquitectura desarrollada en esta tesis. El lector puede omitir la lectura de estas secciones si está familiarizado con estos conceptos de diseño.

## SISTEMAS EN-CHIP

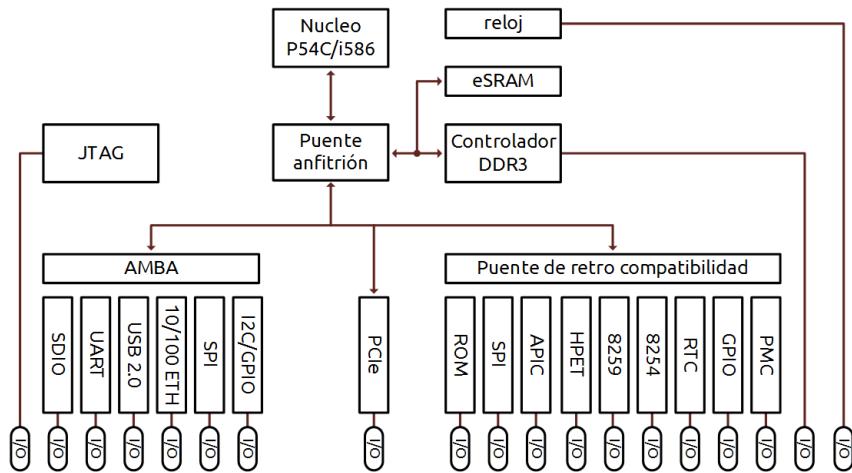
La reducción en la geometría del transistor, unidad mínima en la construcción de circuitos integrados, dio paso a diseños con mayor holgura en sus restricciones de área. El incremento en densidad de transistores en circuitos integrados modificó las tendencias de diseño, donde en lugar de favorecer circuitos integrados que llevarán a cabo una tarea particular dentro de un sistema electrónico se dio preponderancia a circuitos con altos niveles de integración funcional.

Mayores niveles de integración funcional dentro de un solo empaquetado desembocó en una nueva metodología de diseño denominada *System on-Chip* o *SoC*<sup>60,29,63</sup>. Los circuitos integrados diseñados bajo esta metodología se distinguen por implementar en su totalidad o gran parte de las funciones que normalmente llevaría a cabo un sistema electrónico formado de varios componentes.

La metodología SoC ha tenido un fuerte impacto en la filosofía para el diseño de sistemas electrónicos, al grado de estar prácticamente presente en la mayoría de los dispositivos con los cuales convivimos de manera cotidiana.

La solución integrada Intel®Quark™SoC X1000<sup>13</sup> es un ejemplo de un circuito integrado basado en la la filosofía SoC. Este dispositivo cuenta con un solo núcleo de procesamiento e integra fuentes de señal de reloj, reguladores de voltaje, controladores de memoria e interfaces de I/O (entrada/salida) como: ethernet, PCI express, USB 2.0, SD/SDIO, SPI, UART y I<sub>2</sub>C/GPIO. La figura 2.1 muestra el diagrama a bloques de un dispositivo Quark™SoC X1000.

Un circuito SoC puede incluir procesadores de propósito general, bloques de memoria, aceleradores, lógica especializada, medios de interconexión y un sin número de bloques di-



**Figura 2.1:** Diagrama a bloques de un dispositivo Quark™SoC X1000. El diseño de procesadores para ordenadores de escritorio está fuertemente relacionado con la filosofía SoC, al grado que procesadores de gama alta pueden incluir dentro del mismo silicio módulos de aceleración para el procesamiento de gráficos.

gitales que cubran las necesidades del diseño. Actualmente podemos encontrar ejemplos de sistemas electrónicos basados en SoCs prácticamente en cualquier parte de nuestro entorno, desde teléfonos celulares hasta aviones comerciales.

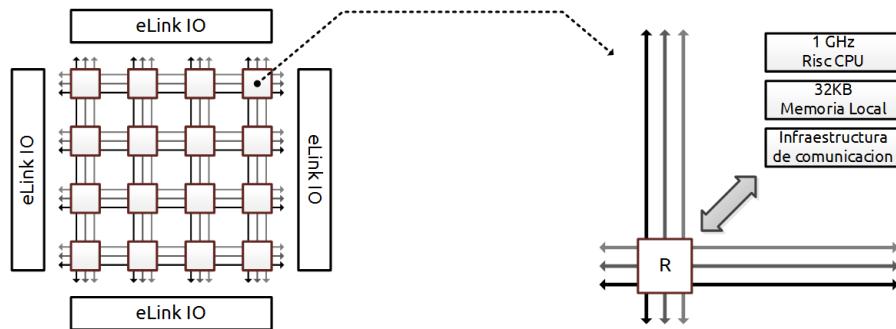
El concepto SoC se diversifica a razón del desarrollo de sistemas eléctricos especializados en tareas específicas. *Multi Processor System on Chip* o *MPSoC*<sup>39</sup> es un sistema SoC formado por múltiples núcleos de procesamiento o unidades funcionales. En general, un circuito MPSoC está formado por un conjunto heterogéneo de núcleos de procesamiento, y aprovecha la concurrencia entre operaciones necesarias para llevar a cabo una tarea específica. El formato de codificación H.264/AVC<sup>71</sup> es un ejemplo popular de una aplicación que aprovecha el concepto de MPSoC, debido al número de tareas independientes que se llevan a cabo para lograr la codificación de video.

El uso de un conjunto de elementos de procesamiento heterogéneos es la arquitectura más popular para la diseño de MPSoCs. Núcleos de procesamiento dedicados a una tarea especí-

fica traen consigo beneficios de reducción de área necesaria para su tendido, eficiencia en el consumo de energía y reducción de complejidad en la validación de cada uno de ellos. Las arquitecturas MPSoC formadas por grupos de núcleos homogéneos, aunque no tan populares, han encontrado un nicho de aplicación donde el intercambio de eficiencia por flexibilidad de aplicación resulta atractivo. El uso de elementos homogéneos requiere que cada núcleo puede ser programado, desde procesadores de propósito general hasta conjuntos de unidades aritméticas capaces de ejecutar un número predeterminado de operaciones son los elementos más comunes encontrados en esta variante de MPSoC. *Epiphany* es un ejemplo de un dispositivo MPSoC con núcleos de procesamiento homogéneos. Epiphany es desarrollado por la compañía Adapteva<sup>30</sup>, el dispositivo cuenta con una arquitectura multi-núcleo escalable hasta 4,095 procesadores. Cada núcleo es un procesador de propósito general con arquitectura RISC con soporte para operaciones de punto flotante bajo el estándar IEEE de 32 bits. Epiphany utiliza un modelo de memoria compartida entre todos sus núcleos de procesamiento. La comunicación entre núcleos de un dispositivo Epiphany descansa sobre una NoC capaz de ejecutar el intercambio de mensajes entre núcleos vecinos con una latencia de 1,5 ns. La figura ?? muestra un diagrama a bloques de la arquitectura de Epiphany.

Los siguientes trabajos presentan casos de estudio de aplicaciones específicas basadas en sistemas MPSoCs: El artículo de Tota *et al.*<sup>70</sup> analiza el uso de un MPSoC homogéneo como acelerador de gráficos, desde una fase de prototipado en un dispositivo reconfigurable hasta la generación de su modelo equivalente en VLSI(Very-large-scale integration); El trabajo de Jalier *et al.*<sup>36</sup> ofrece una comparativa entre un MPSoC heterogéneo y una variante homogénea para la implementación de un módem para redes LTE.

La abundancia de recursos dentro de circuitos integrados no solo beneficio a dispositivos



**Figura 2.2:** Diagrama a bloques de MPSoC Epiphany. Epiphany es un ejemplo de una arquitectura con núcleos de procesamiento homogéneo, algunas aplicaciones donde se ha utilizado este dispositivo son: Radios definidos por software (SDR), visión por computadora, procesamiento de sonido y video, redes neuronales y simulaciones de sistemas físicos.

de propósito específico, gran parte de estos avances impactó en el proceso de fabricación de dispositivos reconfigurables, permitiendo incrementos en la cantidad de celdas lógicas reconfigurables y en la cantidad de bloques rígidos de propósito específico que se integran dentro de estos dispositivos. El incremento de capacidad en dispositivos reconfigurables propició la exploración del espacio de diseño con filosofía SoC/MPSoC, creando nuevas vertientes que incorporan las restricciones y propiedades físicas de los dichos dispositivos. Los términos *System on Programmable Chip* o *SoPC*, y *Programmable System on-Chip* o *PSoC*, se utilizan de manera frecuente para describir sistemas diseñados con filosofía SoC e implementados dentro de dispositivos reconfigurables como *FPGAs*.

El rendimiento y prestaciones de un dispositivo SoC se encuentran en gran medida determinadas por la capacidad de movimiento de información entre los núcleos de procesamiento del sistema. La infraestructura de comunicación interna es un tema complejo, al cual se ha dedicado un gran esfuerzo de investigación para el desarrollo de nuevas arquitecturas más eficientes y con mejor escalabilidad. La relevancia de las estructuras de interconexión para dispositivos SoC toman un rol más preponderante en el diseño de sistemas, debido al constante

incremento de núcleos que deben de comunicarse para llevar a cabo la tarea para la cual fue diseñado el dispositivo. El concepto de interconexión *Network on-Chip* o *NoC*, ha emergido como la solución preponderante en el desarrollo de dispositivos de próxima generación, al grado de crear filosofías de diseño alrededor de la infraestructura de interconexión y dejando en plano secundario a los núcleos funcionales del sistema. *Multi Processor Network on-Chip* o *MPNoC* es un ejemplo de dichas metodologías centradas en la comunicación. El concepto de red en-chip es vasto, por lo que la siguiente sección de este trabajo se presenta una breve introducción a sus fundamentos.

## REDES DE INTERCONEXIÓN

Una estructura tipo bus esta formada por un grupo de lineas de transmisión de datos compartidas entre un grupo de módulos que requieren el intercambio de información. De manera ordinaria, solo dos elementos conectados a las lineas de transmisión de un bus pueden establecer un enlace durante un periodo de tiempo, dejando al resto de los módulos incomunicados de manera momentánea. La característica mencionada anteriormente hace de los buses estructuras de interconexión poco eficientes. Conforme el número de módulos que forman un sistema se incrementa, las desventajas del uso de buses como medios de interconexión se hacen más notorias. Buses jerárquicos<sup>44</sup> y buses segmentados<sup>41</sup> son algunos de los ejemplos de modificaciones estructurales que sufrieron los buses para mejorar su desempeño, sin embargo el incremento en las frecuencias de operación en silicio y un aumento en el número de módulos dentro de un circuito SoC orillo a la búsqueda de nuevas estructuras para la transmisión de datos.

Diseños electrónicos complejos integrados en un solo empaquetado migraron del uso de

buses como infraestructura de comunicación a redes mas complejas para su interconexión. Una *red de interconexión* puede definirse como un *sistema programable para el transporte de información entre módulos o nodos, donde las líneas físicas de transporte de información son compartidas entre todos los miembros de la red*. La principal característica que distingue a una red de interconexión entre otros medios de enlace es su modelo general de operación: Una red trabaja bajo el concepto de intercambio de mensajes entre nodos miembro; Si un nodo desea enviar un mensaje a un nodo vecino, este debe de entregar el mensaje a la red y esta a su vez se encargará de que dicho mensaje sea entregado a su destinatario. Otros medios de enlace, como líneas de comunicación dedicadas, se limitan a operar como simples medios físicos para el intercambio de datos.

Los términos *sistema* y *programable* que forman parte de la definición de red de interconexión, proveen de elementos extra para diferenciar este concepto de otros medios de enlace. Una red de interconexión se considera un sistema porque está formado de varios componentes como: buffers, canales de transmisión, lógica de control y switches. El término *programable* indica la capacidad de la red para crear diferentes conexiones entre nodos, utilizando un conjunto de recursos limitados pero compartidos para la transmisión de datos desde cualquier miembro del sistema.

Las redes de interconexión ofrecen grandes beneficios en comparación con otras estructuras tradicionales, por ejemplo, la figura 2.3 presenta dos estructuras para interconectar seis nodos. A la izquierda se presenta el enlace de todos los miembros del sistema mediante líneas de conexión dedicadas entre cada uno de los nodos. Se requieren de 30 líneas de transporte de datos para conectar cada nodo con sus vecinos. En contraparte, la red de interconexión a la derecha en la figura 2.3 requiere solo de 1/5 de los recursos requeridos por la primera estructura.



**Figura 2.3:** a) Enlace de 6 nodos mediante líneas de comunicación dedicadas entre terminales. b) Enlace de 6 nodos mediante una red de interconexión en forma de anillo.

tura.

Las redes de interconexión no solo ofrecen una reducción en la cantidad de recursos para el enlace entre nodos, imagine que cada nodo de las estructuras presentadas en la figura 2.3 requiere transmitir un mensaje a cada otro nodo en intervalos de 100 ciclos de reloj. En el modelo con enlaces dedicados, cada enlace estará ocupado solo durante el 1 % del tiempo de operación del sistema. El enlace por medio de una red de interconexión con forma de anillo tendrá un factor de trabajo de entre 12 % y 15 % dependiendo de los detalles de organización de la red.

El concepto de red de interconexión y sus fundamentos pueden aplicarse a diferentes escalas: desde redes para la interconexión de computadoras dentro de un edificio, denominadas *redes de área local* o *LAN*; Redes de conexión entre circuitos de un sistema electrónico, o hasta redes de interconexión de módulos dentro de un SoC, denominadas *Networks on-Chip* o *NoC*. Este último término, traducido al español como *Redes en-Chip*, es uno de los fundamentos de este trabajo, y a partir de este punto del documento los términos red de interconexión y *NoC* se utilizaran de manera indistinta. En el trabajo<sup>4</sup> se encuentra un recopilado de el espacio de diseño y campos de investigación en NoCs.

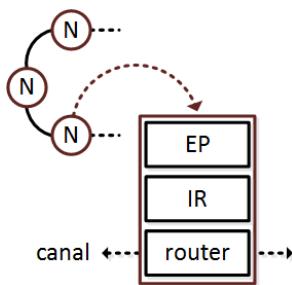
## CONCEPTOS BÁSICOS DE NoCs

Una NoC ofrece el servicio de transporte de mensajes entre nodos miembro de una red. De manera general, todo nodo de una red en-chip está formado por tres elementos principales: un elemento de procesamiento, una interfaz de red y un *router*. El elemento de procesamiento representa una unidad funcional, puede ser un procesador de propósito general, lógica para la ejecución de una operación específica o un elemento de almacenamiento. El router forma parte de la infraestructura de la red, su tarea principal consiste en la recepción, cálculo de ruta y reenvío de paquetes en dirección a un nodo destino. El término *router* se traduce al español como encaminador o enrutador.

La interfaz de red consiste en una etapa de lógica de acoplamiento entre el elemento de procesamiento y el router de un nodo. La interfaz de red traduce paquetes de datos a un formato de trabajo para el elemento de procesamiento, de igual forma, la interfaz se encarga de tomar los resultados del elemento de procesamiento y empaquetarlos de manera que puedan ser transportados a través de la NoC. La figura 2.4 muestra la estructura de un nodo de red estándar. Utilizaremos el término *canal* para referirnos a un conjunto de líneas de transporte de datos que enlazan a un nodo con un vecino inmediato.

Una red en-chip está definida por 3 características principales: topología, algoritmo de planificación de ruta y esquema de control de flujo de datos. Topología se le llama al patrón de distribución de nodos y canales a través de red. El término *grado del nodo* define el número de canales que comunican un nodo con sus vecinos. Los nodos del segmento de red mostrados en la figura 2.4 tienen un *grado 2*, es decir un canal por cada vecino.

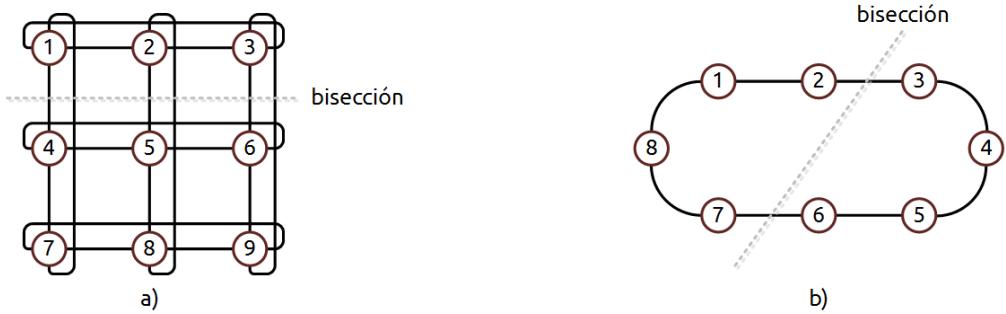
Un mensaje puede tomar diferentes vías a través de la red para llegar a su destino. La selección de uno de los múltiples caminos para el transporte de información está a cargo del



**Figura 2.4:** Nodo de red: N representa un nodo de red, EP simboliza un elemento de procesamiento e IR a una interfaz de red. *Canal* es el término utilizado para referirse a las líneas físicas de transmisión de datos entre un nodo y uno de sus vecinos de la red. Los nodos miembros de una red pueden ser heterogéneos. No existe un modelo único de interfaz de red, ya que diferentes tipos de elementos de procesamiento requieren la presentación de los datos de trabajo en diferentes formatos.

algoritmo de planificación de ruta. Una ruta está formada por el conjunto de nodos y canales que un mensaje deberá de cruzar antes de llegar a su destino. Una práctica común es el referirse como *número de saltos* a la cantidad de routers que un mensaje debe visitar antes de llegar a su destino. Un buen algoritmo de planificación de ruta busca reducir el número de saltos necesarios para entregar un mensaje a su destinatario mientras mantiene a todos los canales de la red ocupados. Si el algoritmo de planificación tiende a saturar algunas rutas de la red, mientras otras se encuentran en estado de reposo, se dice que el algoritmo propicia *desbalances de carga*. Dentro del argot de redes en-chip, suelen utilizarse los términos *ruteo* o *encaminamiento* para referirse a la acción del cálculo de ruta que tomará un mensaje a través de la red, en este documento utilizaremos cualquiera de estos términos de manera indistinta.

El control de flujo de datos son políticas aplicadas en cada router e interfaz de red para administrar el uso de recursos como: canales, puertos de entrada y salida, buffers, y cualquier elemento compartido para la transmisión de mensajes. Los mecanismos de control de flujo se aplican en diferentes elementos de la red, pero en conjunto, buscan reducir el tiempo que



**Figura 2.5:** a) Topología tipo torus de 9 nodos. b) Topología tipo anillo con 8 nodos. Cada línea de interconexión en el diagrama representa dos canales en direcciones opuestas, por ejemplo, la línea entre los nodos 1 y 2 de la red tipo anillo representa: un canal saliente del nodo 1 en dirección del nodo número 2 y un canal saliente del nodo número 2 con dirección del nodo 1.

tarda un mensaje en alcanzar a su destinatario y evitar la pérdida de mensajes durante su traslado.

## TOPOLOGÍA

La topología de una NoC define el arreglo de conexiones entre nodos y canales que forman la red. La topología es análoga al mapa de una carretera, donde cada ciudad o poblado representa un nodo de la red y los tramos de camino representan los canales entre nodos. La figura 2.5 muestra una red con topología tipo torus y una red de topología tipo anillo, ambos ejemplos se consideran *redes directas*, ya que cada router de la red está conectado de manera directa con un elemento de procesamiento. En caso que uno o más routers de la red no estén ligados de manera directa a un elemento de procesamiento, se considera a dicha red como *indirecta*.

La topología tiene un alto impacto en el rendimiento de una NoC, por ejemplo, el máximo *ancho de banda* (Tasa máxima de transferencia de datos a través de un segmento de la red, la

unidad de transferencia de datos utilizada para tasar el ancho de banda es *bits/sec.*) que puede ejercer una red está determinado por el ancho de banda de bisección. La bisección de una NoC conciste en el grupo de canales que son seccionados por una línea imaginaria que divide la red en dos segmentos. La figura 2.5 destaca la bisección de ambas topologías ilustradas.

Utilicemos las dos redes de la figura 2.5 como ejemplo para evaluar cómo la topología repercute en el rendimiento de una NoC. Los parámetros base para ambos ejemplos son los siguientes: Ambas redes tienen una frecuencia de operación de 300 MHz (periodo = 3.33 ns), el ancho de canal para la red con topología torus es de 32 bits mientras que para la red de topología tipo anillo es de 64 bits. Es importante notar que en la figura 2.5 cada linea que conecta dos nodos representa dos canales en direcciones opuestas, es decir, desde el punto de vista de un nodo cada línea representa un canal saliente y un canal ingresando a él.

El *ancho de banda de bisección* para la red torus se calcula como el producto entre el número de canales que cruza la bisección(12), el ancho de palabra de cada canal(32 bits) y la frecuencia de operación de la red (300 MHz). La ejecución del producto anterior da como resultado un total de 115.2 *Gbits/s*. El ancho de banda en la bisección de la topología tipo anillo se calcula de manera similar, sustituyendo los canales en bisección(4) y el ancho de cada canal(64 bits). La máxima tasa de transferencia en la bisección para la red anillo es de 76.8 *Gbits/s*. En este caso particular, la red torus ofrece 1.5x veces mayor capacidad de transferencia de datos en comparación con la red tipo anillo.

La tasa máxima de transferencia de una red no es único indicador de rendimiento. Retomemos el ejemplo anterior definiendo 4,096 bits como el tamaño de los mensajes a transmitir a través de la red. Definamos el término *latencia promedio* como el tiempo promedio que tarda un mensaje en ser entregado a su destinatario. Para el cálculo de la latencia promedio, es nece-

sario la selección de un *patrón de tráfico*, el cual es un modelo de probabilidad y frecuencia de comunicación entre nodos de una red. Los patrones pueden ser generados de manera artificial o a partir de la captura del esquema de comunicación entre nodos de un sistema preexistente. En este ejemplo utilizaremos un patrón aleatorio, donde todos los nodos de la red tienen la misma probabilidad de ser electos como destino de un mensaje. En el trabajo<sup>2</sup>, presentado por Bahn y Bagherzadeh, se aborda la generación de patrones de prueba para redes en-chip. Además, es necesario definir el término *distancia promedio*, el cual representa el número de saltos en promedio que un paquete debe de llevar a cabo para alcanzar cualquier nodo de la red. Para la red torus la distancia promedio es de 2 saltos, mientras que en la red anillo la distancia promedio también es de 2 saltos. Supongamos que para ambas redes, el tiempo promedio para ejecutar un salto entre dos nodos es de 9.99 ns (3 ciclos de reloj).

Mensajes de 4,096 bits de longitud, con canales de 32 y 64 bits de ancho de palabra, hacen necesario la *serialización* de los mensajes, es decir, la división de estos en bloques de tamaño igual al ancho de palabra del canal. El proceso de división de un mensaje trae consigo una penalización de latencia equivalente al tiempo que se requiere para extraer un fragmento del mensaje original. A esta penalización se le denomina *latencia de serialización*. Supongamos que la latencia de serialización para ambas tipologías es igual a un ciclo de reloj por bloque.

El tiempo promedio de transporte de un mensaje en la red de topología torus es igual a la latencia de serialización del mensaje en fragmentos de 32 bits ( $4,096/32 = 128$  ciclos), más el tiempo requerido para cubrir la distancia promedio entre nodos de la red ( $9.99ns * 2 = 19.98ns$ ), es decir la *latencia promedio* para esta red es de 2983.68 ns para el transporte de los 128 paquetes que conforman un mensaje de 4,096 bits. Para la topología tipo anillo, que cuenta con canales de 64 bits, la latencia de serialización es de 213.12 ns. Si a esta última cifra agregamos

el tiempo necesario para llevar a cabo los dos saltos promedios de la red tenemos una *latencia promedio* total de 1491.82 ns por mensaje.

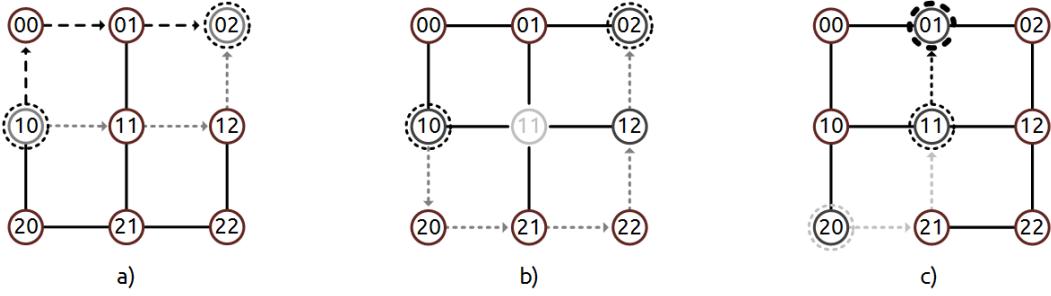
Este último resultado presenta a la topología tipo anillo como la infraestructura que ofrece la menor latencia para la entrega de mensajes a través de la red, aun cuando la red de topología torus ofrece un mayor ancho de banda para la transmisión de datos. La selección de topología depende de las características y requerimientos de la aplicación además de las restricciones físicas del dispositivo que pueden restringir el uso de ciertas topologías.

Sadawarte<sup>66</sup> presenta un compilado de fuentes de información respecto a las topologías más comunes utilizadas en redes en-chip, este documento es un gran punto de inicio para profundizar en el tema.

## PLANIFICACIÓN DE RUTA

La ruta o camino entre dos nodos de la red se representa mediante un conjunto ordenado de canales  $P = \{c_1, c_2, \dots, c_n\}$ . Cada canal  $c_n$  representa un punto de entrada y uno de salida. El punto de salida del canal  $c_n$  es el punto de entrada del canal  $c_{n+1}$ . Físicamente, la topología provee de múltiples rutas  $P$  para la transmisión de mensajes entre dos nodos de la red, sin embargo, es tarea del algoritmo de planificación de ruta la selección del camino para cada transferencia de datos. Los términos *routing*, *ruteo* o encaminamiento se utilizan de manera indistinta para hacer referencia al algoritmo de planificación de ruta de una NoC.

Un algoritmo de ruteo eficiente procura la reducción del número de saltos necesarios para la transferencia de un mensaje, a la vez que procura evitar la saturación de un conjunto reducido de canales de la red. La saturación de ciertos canales mientras rutas alternas se encuentran disponibles se denomina *desbalance de carga*.



**Figura 2.6:** Tres instancias de una red con topología tipo malla o *mesh*. Las direcciones para cada nodo de red se forman de una dupla de coordenadas  $\{x|y\}$ . a) Rutas trazadas mediante algoritmos de ordenamiento dimensional. b) planificación de una ruta no mínima entre dos nodos de la red. c) Desbalance de carga en la red, el canal que vincula a los nodos  $\{1,1\}$  y  $\{0,1\}$ .

Es una práctica común nombrar a los nodos de una red mediante coordenadas que reflejen su posición respecto a sus vecinos. La figura 2.6 muestra redes con topologías tipo *malla*, los identificadores asignados a cada nodo de estas redes consta de una tupla  $\{x,y\}$  que representa su posición en coordenadas de un plano cartesiano. Los algoritmos ordenados por dimensión se caracterizan por realizar avances en una dirección a la vez, por ejemplo: la transferencia entre los nodos  $\{1,0\}$  y  $\{0,2\}$  en la figura 2.6 (a) presenta dos rutas posibles: la primera ruta visita los nodos  $\{1,1\}$  y  $\{1,2\}$ , ejecutando todos los saltos necesarios en la dirección  $x$  antes de entregar su mensaje mediante un último salto en la dirección  $y$ ; De manera contraria, la ruta con saltos a los nodos  $\{0,0\}$  y  $\{0,1\}$ , lleva a cabo en primer lugar un salto en la dirección  $y$ .

El algoritmo de *ruteo XY*<sup>9</sup> es un ejemplo popular de planificación de ruta ordenada por dimensión. Este tipo de algoritmos tienden a ser sencillos y ofrecer *rutas mínimas* para la entrega de mensajes. *Ruta mínima* se denomina a un camino que requiere el mínimo de saltos para alcanzar su objetivo, las dos rutas presentes en la figura 2.6 (a) son rutas mínimas entre los nodos  $\{1,0\}$  y  $\{0,2\}$ .

Los algoritmos ordenados por dimensiones tienen la desventaja de propiciar la saturación

de canales en la red. Por ejemplo, en la figura 2.6 (c) el canal entre los nodos  $\{o,1\}$  y  $\{1,1\}$  es utilizado en todas las rutas con destino al nodo  $\{o,1\}$  que parten desde los nodos en las filas 1 y 2 de la red. Conforme aumente la demanda de transferencia de mensajes al nodo  $\{o,1\}$ , el ancho de banda disponible en los canales de acceso desde la dirección  $y$ - se reduce, mientras la latencia de transferencia de mensajes aumenta. Algoritmos que pueden seleccionar rutas no mínimas como la trazada en la figura 2.6 (b) tienden a reducir el problema del balanceo de carga en la red, sin embargo suelen ser más complejos y pueden generar bloqueos en la red en caso de no estar verificados de manera exhaustiva. Los algoritmos con capacidad de generar rutas tanto mínimas como no mínimas se denominan *algoritmos adaptativos*. Los algoritmos adaptativos dan flexibilidad a la red, ofreciendo un mejor balance en la carga de sus canales y ofreciendo rutas alternas en caso que algunos de los nodo miembro no se encuentre disponible para reenviar un mensaje en dirección de su destino, sin embargo el uso de rutas más largas entre nodos aumenta la latencia promedio para la entrega de los mensajes, además de requerir hardware más complejo para la toma de decisiones. Existen algoritmos exclusivos para su uso en conjunto con topologías específicas, por ejemplo, el algoritmo aEqualized<sup>II</sup> está diseñado para su uso exclusivo con topologías tipo spidergon.

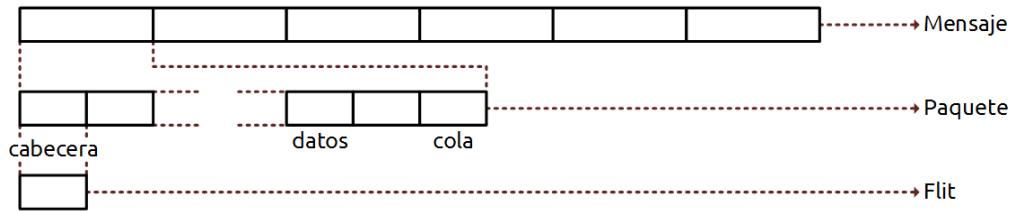
## ESTRUCTURA DE MENSAJES

Hasta este momento se ha utilizado el *mensaje* como unidad básica de transporte de información a través de la red sin embargo la mayoría de redes de interconexión requieren descomponer un mensaje en segmentos de tamaño igual al ancho de sus canales de enlace. Un mensaje puede tener una longitud arbitraria y dependiente de la aplicación, por ejemplo para un sistema de procesamiento de imágenes un mensaje puede contener la información de

todos los píxeles capturados por una cámara, mientras para una red distribuida de sensores, el mensaje puede tener la longitud necesaria para transmitir una lectura de temperatura. La diferencia entre tamaños de mensaje en las aplicaciones anteriormente descrita puede estar entre varios megabytes de información a unos cuantos bytes.

Para facilitar el manejo de información a través de la red los mensajes son fracturados en *paquetes*. Un paquete contiene una sección de la información que forma un mensaje, y es la unidad básica para las operaciones de control y transferencia de datos entre routers. Cuando un router de la red asigna recursos para una transacción, la asignación tendrá una duración equivalente al tiempo necesario para movilizar al siguiente salto todos los datos que forman un paquete. A todo paquete liberado a la red se le anexa información de control para que este sea capaz de navegar a su destino, y para poder reconstruir un mensaje una vez que todos los paquetes hayan sido capturados. Los paquetes deben de tener una restricción sobre el tamaño máximo que pueden alcanzar, de manera que se pueda tener un estimado en la cantidad de recursos y tiempo que se le asigna a cada paquete en la red. Los paquetes de un mensaje pueden contener información de control, datos de trabajo o una combinación de ambos.

Los paquetes a su vez se dividen en *flits*(control flow digit) o dígitos de control de flujo. El flit es la unidad atómica para la asignación de espacio en buffer, por lo que todos los routers de la NoC deben de proveer espacio al menos para un flit en transito. Esta ultima regla no aplica si el router de red utiliza una estrategia de control de flujo sin almacenamiento temporal. La estructura de un paquete segmentado incluye un *flit de cabecera*, uno o más *flits de datos* y un *flit de terminación*. El flit de cabecera de manera general no contiene parte de la información del mensaje, en su lugar, este flit acarrea la información necesaria para abrirse paso a través de la red hasta su destino. Por ejemplo, un flit de cabecera puede contener la dirección del



**Figura 2.7:** Estructura de un mensaje. La división de un mensaje puede ser física o lógica, dependiendo del contexto de manejo en el cual se esté procesando la información. En la figura no se muestra la división de flits en phits.

nodo destino del mensaje. Los flits de datos generalmente cargan un segmento del mensaje enviado a través de la red, sin embargo, también pueden contener algunos bits de control para identificarse como flits de datos. Finalmente, un flit de terminación anuncia el final de un paquete. El flit de terminación es utilizado cuando los paquetes de la red pueden tener un número no predeterminado de flits de datos.

En caso de que el tamaño en bits de un flit sea superior al número de líneas físicas que conforman un canal, será necesario dividir los flits de un paquete en *phits* o *unidades físicas*. El phit es el máximo de información que se puede transferir de un nodo a otro en un solo ciclo de reloj. La figura 2.7 muestra la descomposición de un mensaje en unidades de dimensiones inferiores.

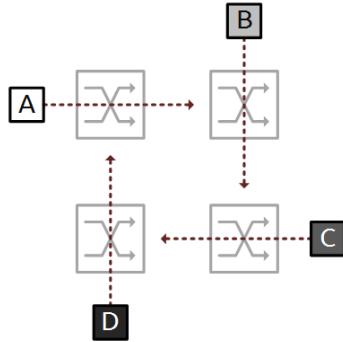
## CONTROL DE FLUJO

El control de flujo de datos de una NoC establece las políticas para la asignación de recursos que permitirán a los paquetes avanzar a través de la red. Las políticas de control de flujo se encargan de administrar los canales entre nodos y los buffers internos de cada router. Los buffers de un router permiten almacenar de manera temporal los paquetes que, por algún motivo, no pueden continuar de manera inmediata su camino.

El funcionamiento de los buffer dentro de un nodo, siguiendo la analogía con una carretera, representan un carril exclusivo para incorporarse a otra intersección. Si un automóvil, que en esta analogía representa un paquete, quiere incorporarse a una intersección saturada, este deberá esperar su turno para ingresar a carril deseado. Durante el tiempo de espera otro automóvil que desea continuar por el camino principal llega a la intersección, este último automóvil no puede continuar su trayecto debido a que el automóvil esperando a incorporarse a la intersección se encuentra bloqueando su paso. Agregando un carril exclusivo para los automóviles que desean dejar el camino principal se previenen posibles bloqueos, permitiendo el avance de automóviles sin obstáculos en las intersecciones.

Las políticas de control de flujo definen el avance de un paquete entre nodos y dentro de sus correspondientes routers. El control de avance entre nodos puede pensarse como semáforos, controlando cuándo es momento para que un automóvil tenga acceso al siguiente segmento de carretera.

Una buena política de control de flujo procura evitar el bloqueo de paquetes en tránsito debido a la falta de recursos, en especial en el caso que existan recursos disponibles en otras áreas de la red. De igual manera, el control de flujo debe evitar situaciones donde varios paquetes en circulación generen un *punto muerto* o *deadlock*, donde el avance de estos paquetes sea imposible debido a dependencias circulares de recursos entre ellos. Otra situación que el control de flujo debe evitar es la creación de *livelocks* o *paquetes errantes*, en esta situación un paquete en la red no recibe los recursos adecuados para alcanzar su destino y por consiguiente se encuentra vagando entre nodos de manera permanente. Estos dos últimos escenarios, *deadlock* y *livelock*, son resultado de políticas injustas al momento de asignar recursos entre paquetes compitiendo por ellos. La figura 2.8 presenta una situación de deadlock en una red

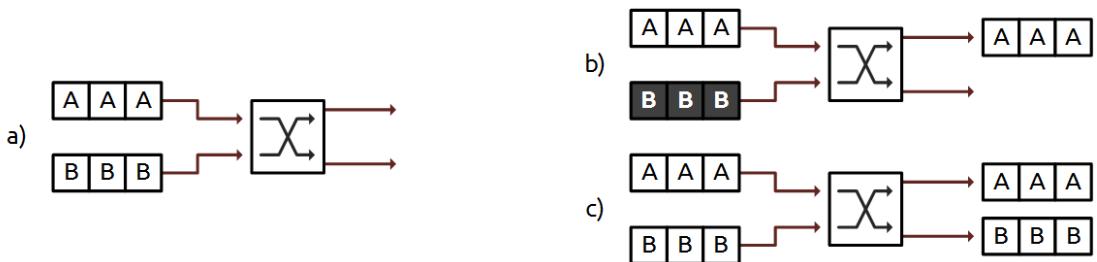


**Figura 2.8:** Deadlock en una NoC. En este ejemplo El paquete A espera los recursos reservados por el paquete B, que a su vez espera los recursos asignados al paquete C, el cual, no puede avanzar por el bloqueo de recursos causado por el paquete D, el cual cierra la *dependencia circular* esperando los recursos del paquete A.

en-chip. El control de flujo puede verse como un problema de asignación de recursos o como una tarea de resolución de conflictos entre paquetes disputando estos recursos.

El mecanismo más sencillo de control de flujo se denomina *bufferless*, el término se deriva de la aplicación de políticas de gestión de recursos que no consideran la disponibilidad de buffers en los diferentes nodos de la red. El manejo de disputas por recursos entre paquetes es solucionada por medio de el *descarte* del paquete no ganador, o mediante el *reenvío* del paquete por cualquier canal de salida que se encuentre disponible en ese momento, inclusive, si el canal disponible envía al paquete en la dirección opuesta de su destino. Estos dos mecanismo de resolución de conflictos pueden ser referidos por los términos *drop* o *misroute* respectivamente. La figura 2.9 ejemplifica las dos estrategias utilizadas con un mecanismo bufferless de control de flujo.

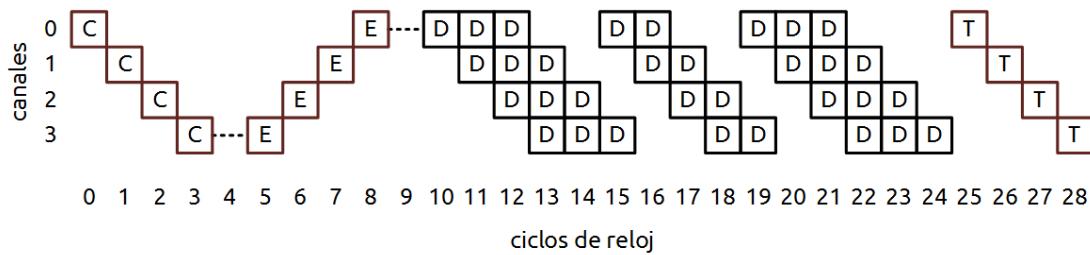
*Commutación de circuitos* o *circuit switching* es una estrategia de control de flujo con mayor compromiso respecto a la fiabilidad de la entrega de mensajes entre nodos. Este mecanismo inicia toda transmisión con el envío de un paquete de cabecera. Los paquetes de cabecera



**Figura 2.9:** Control de flujo bufferless. a) Dos paquetes llegan al router solicitando el puerto de salida superior. b) Bajo una estrategia de *descarte* el paquete A gana control de los recursos para avanzar mientras el paquete B es desechar. b) En una estrategia de *reenvío* el paquete B en lugar de ser descartado se le da salida a través del puerto disponible.

contienen información de encaminamiento de manera exclusiva. La cabecera reserva el uso de los recursos que ha utilizado durante su avance a través de la red, una vez alcanzado su destino, todos los recursos reservados forman un *túnel* entre el nodo origen y el nodo destino. Ningún paquete o flit de datos entrara a la red a menos que el túnel, también llamado *circuito*, esté formado de manera satisfactoria. Se puede llevar a cabo cualquier número de transacciones entre los nodos enlazados siempre y cuando el circuito entre nodos se encuentre activo. Una vez finalizada la transmisión de datos, uno de los nodos enlazados debe de emitir un paquete para la liberación de los recursos reservados. Una gran desventaja del uso de esta estrategia es el bloqueo de recursos durante la vida de un enlace entre nodos, los recursos reservados no pueden ser utilizados por ningún otro paquete hasta que sean liberados. La conmutación de circuitos requiere sólo de asegurar el almacenamiento temporal para un flit.

La figura 2.10 muestra una transacción estándar en una red que hace uso del esquema de conmutación de circuitos, la transmisión inicia con el envío de un flit de cabecera, este tarda 4 ciclos de reloj para llegar al nodo destino de la transmisión. El nodo destino procesa la petición de establecimiento de circuito durante el ciclo 4 y en el siguiente flanco positivo de la señal de sincronización libera un señal de confirmación de establecimiento de enlace. Durante el



**Figura 2.10:** Secuencia de transmisión de datos mediante conmutación de circuitos. El identificador *C* representa la cabecera de un paquete, *E* representa un mensaje de confirmación de creación de enlace, *D* representa la carga efectiva de datos de un paquete y *T* representa el final de un paquete.

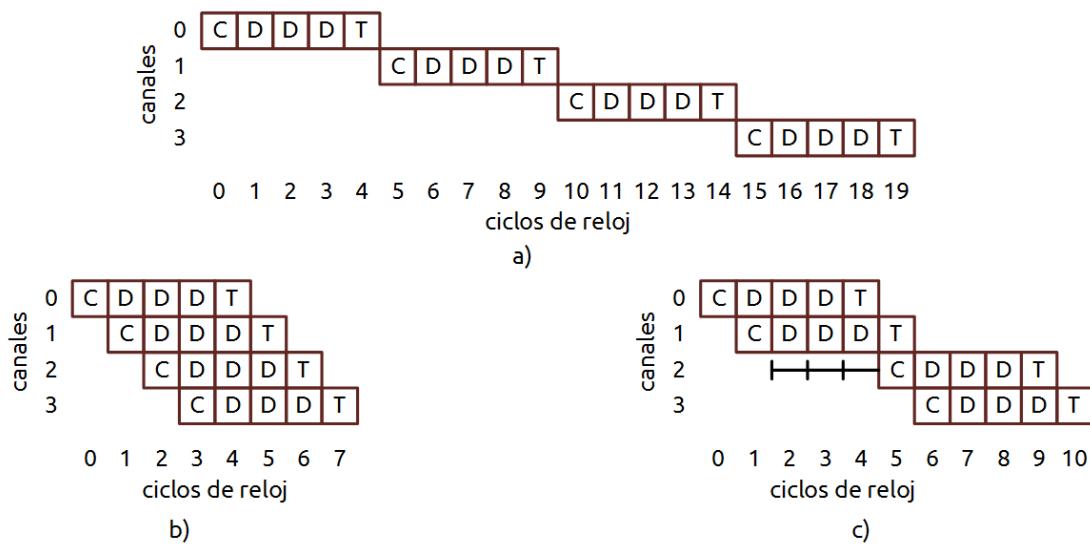
noveno ciclo de operación, el nodo origen recibe la confirmación de la formación del circuito con el nodo destino, a partir de este momento se puede iniciar con la transferencia de datos. La transferencia de datos en la figura 2.10 se lleva a cabo en ráfagas independientes, el envío de datos puede llevarse a cabo bajo cualquier patrón de intervalos de transferencia e inactividad durante el tiempo que se mantenga el enlace entre nodos. Finalmente el nodo origen libera un flit de terminación que libera los recursos del circuito conforme avanza en dirección del nodo destino.

El esquema más popular para el control de flujo en la actualidad es la *comutación de paquetes* o *packet switching*. Esta técnica obtiene ganancias en rendimiento mediante el almacenaje de paquetes previo a la asignación de recursos, desacoplando de manera efectiva el proceso de transmisión de datos y de solicitud de canales de salida. La conmutación de paquetes es un concepto del cual se derivan diferentes esquemas de control de flujo, cada esquema se diferencia por la granularidad de sus buffers y por el mínimo de almacenamiento requerido para permitir el salto de datos entre nodos de la red. Los esquemas más populares de conmutación de paquetes son: *store & forward*, *cut-through*, y *wormhole*.

Los esquemas *store & forward* y *cut-through*, requieren garantizar al menos el espacio su-

ficiente para un paquete completo en el nodo receptor antes de proceder con el reenvío de un mensaje. La principal diferencia entre estos dos esquemas radica en la condición para el inicio de reenvío de paquetes al siguiente nodo, en store & forward se requiere tener almacenado en su totalidad un paquete antes de poder reenviarlo al siguiente salto, mientras que en cut-through, es posible reenviar de manera inmediata parte de un paquete al siguiente nodo siempre y cuando se asegure tener espacio suficiente para recibir el paquete completo en el siguiente router. La figura 2.II(a) muestra una transacción bajo el esquema store & forward, en este ejemplo cada salto entre nodos tiene una duración de 5 ciclos de reloj, debido a que cada paquete debe ser almacenado en su totalidad dentro de un buffer de manera previa a su reenvío a través de un puerto de salida. b) y c) de la figura 2.II muestran el comportamiento de un router con control de flujo cut-through bajo dos escenarios. El primer escenario, ilustrado por la figura 2.II (b), muestra una red en estado de reposo, en este caso la transmisión entre nodos inicia de manera inmediata, retransmitiendo en primer lugar el flit de cabecera al siguiente salto de la ruta. En la figura 2.II (c), el salto entre los canales 1 y 2 presenta un bloqueo temporal, donde el nodo que maneja la salida del canal 1 debe de esperar durante 3 ciclos de reloj para que el nodo receptor libere suficiente espacio para la recepción de todos los flits del paquete. Este segundo escenario es una muestra del comportamiento de la red bajo cargas de trabajo pesadas.

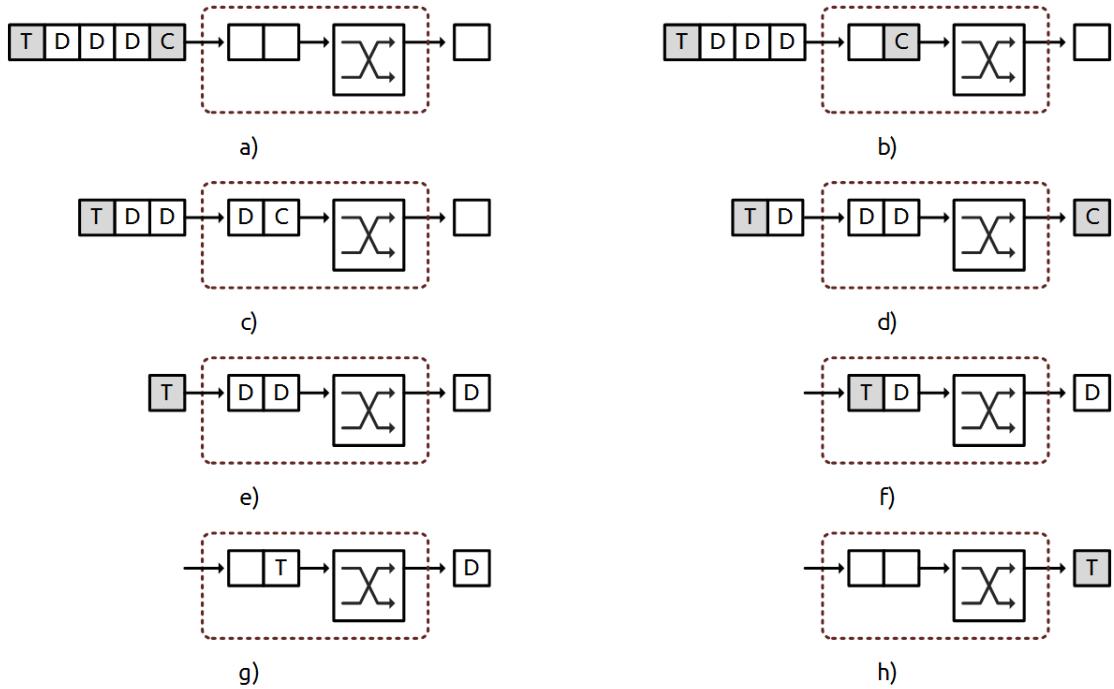
*Wormhole* trabaja a nivel de granularidad de flits, por lo que demanda al menos tener seguro el espacio suficiente para recibir un flit antes de continuar la transferencia de un paquete. La figura 2.III muestra el desenvolvimiento de una operación de tránsito de paquetes a través de un router con control de flujo tipo wormhole. El buffer del router mostrado en la figura 2.III tiene una capacidad de almacenamiento de 2 flits. Durante el primer ciclo de captura, figura



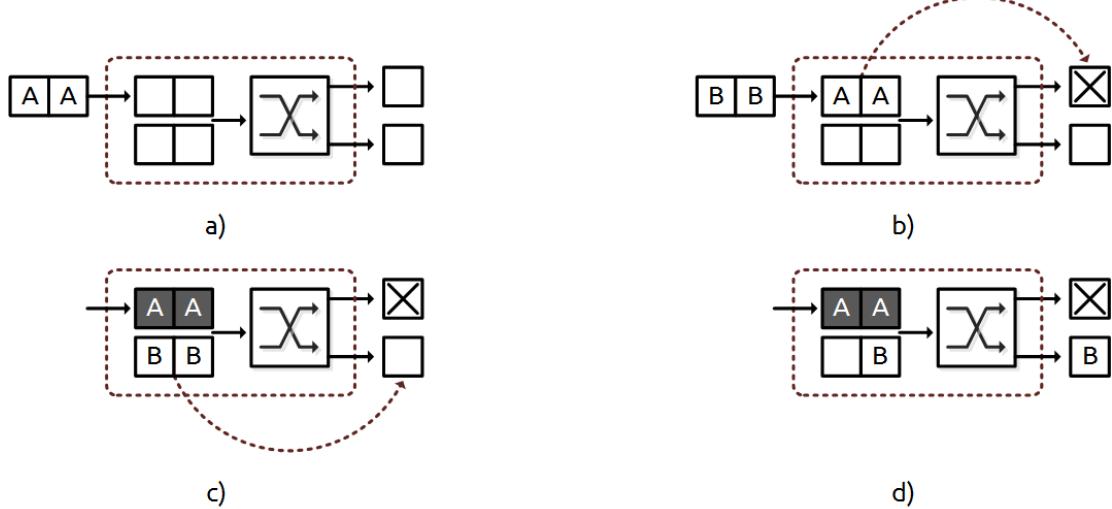
**Figura 2.11:** Transferencias con controles de flujo basados en granularidad de paquete. a) Transferencia *store & forward*, antes de ejecutar el próximo salto en la red, todo el paquete debe de estar almacenado en el router actual. b) Bajo el esquema *cut-through* no es necesario la espera de recepción de todos los flits de un paquete, tan pronto se cuente con espacio suficiente en el siguiente router se puede iniciar la transferencia de flits. c) En caso de no contar con espacio suficiente en el siguiente salto, los flits de un paquete se almacena en buffer sin riesgo de pérdidas, al momento de asegurar recursos en el siguiente router se reanuda la transferencia.

2.12 (a), la cabecera del paquete ingresa al router y de manera inmediata le sigue el primer flit de datos. Hasta el ciclo mostrado en la figura 2.12 (c), no ha habido desplazamiento alguno de flits fuera del router. La lógica de control y planeación de ruta utilizan la información del flit de cabecera, recibido un ciclo previo, para la asignación de recursos al paquete en tránsito. Terminada la asignación de recursos, el flit de cabecera avanza al siguiente nodo en su ruta, el proceso de recepción y asignación de recursos se repite una vez más en el siguiente router. La salida del flit de cabecera del router marca el inicio de la transferencia del paquete a su siguiente salto, la figura 2.12 (e,f,g,h) muestran el envío de los flits restantes del paquete. En caso que el flit de cabecera encuentre una obstrucción de camino en un nodo posterior, la transferencia del resto de flits también sufre de un paro, habiendo la posibilidad que los flits detenidos se encuentren almacenados en buffers de diferentes routers a lo largo del camino.

Virtual channels o canales virtuales<sup>73,47</sup>, es un mecanismo para incrementar el rendimiento de estrategias de control de flujo. Este mecanismo recurre al uso de múltiples estructuras de almacenamiento temporal en cada puerto de entrada de un router, creando buffers alternativos para la recepción de paquetes. En caso de que un paquete almacenado no pueda continuar su camino, el router habilita un segundo buffer para la recepción de paquetes entrantes. Si un paquete almacenado en el buffer secundario del router cuenta con los recursos necesarios para abandonar el nodo de manera inmediata así lo hace. Esta técnica incrementa el rendimiento de la red al proveer un mecanismo para el salto de paquetes en espera por disponibilidad de recursos. Una desventaja en el uso de canales virtuales es el incremento en la complejidad de la lógica de control de cada router, ademas de un incremento de área necesaria para los buffers adicionales. La figura 2.13 ejemplifica el uso de canales virtuales, en este escenario el paquete *A* llega al router y es almacenado en el canal virtual número 1, figura 2.13 (a,b). El paquete



**Figura 2.12:** Transferencia con control de flujo wormhole. Los flits que conforman el paquete presentan los siguientes identificadores: C (cabecera), D (datos o carga) y T (terminación). El concepto de transferencia es similar al control tipo cut-through, su única diferencia es la garantía de espacio en buffer requerido para continuar la transferencia de paquetes. El comportamiento de wormhole es exactamente igual al comportamiento de cut-through cuando este último tiene disponible el espacio de buffer en el siguiente nodo durante la llegada de un nuevo paquete.



**Figura 2.13:** Uso de canales virtuales. Un router con canales virtuales utiliza múltiples buffers, denominados canales virtuales, para evitar la congestión de la red debido a la espera de liberación de recursos.

$A$  desea avanzar por el puerto de salida superior, sin embargo este se encuentra bloqueado por lo que  $A$  debe de esperar para continuar su camino. Un segundo paquete se presenta a la entrada del router, este en respuesta habilita el canal virtual número 2 y empieza la recepción del paquete  $B$ , figura 2.13 (c). El puerto solicitado por  $B$  se encuentra disponible, por lo que el router selecciona el canal virtual número 2 como el canal activo y empieza la transmisión del paquete  $B$ , figura 2.13 (d). Si no se encontraran canales virtuales en este router, el paquete  $B$  tendría que esperar por el avance del paquete  $A$ .

La principal diferencia entre los métodos con granularidades de paquete o flit, radica en el tamaño de medio de almacenamiento que requiere cada uno. En caso de utilizar un esquema basado en paquetes se requerirá un área mayor de silicio para la implementación de los buffers.

## FUENTES DE CONSULTA

*Reliability, Availability and Serviceability of Networks-on-Chip*<sup>14</sup> ofrece un resumen más amplio sobre los fundamentos de NoCs en su segundo capítulo. En el texto *Interconnection Networks: An Engineering Approach*<sup>19</sup> se aborda un amplio espectro de estructuras de interconexión desde buses hasta redes en-chip. *Principles and Practices of Interconnection Networks*<sup>16</sup> es una referencia obligatoria para cualquier persona iniciando en el diseño de redes en-chip. Palesi *et al.*<sup>57</sup> presentan una recopilación de algoritmos de encaminamiento específicos para redes en-chip.

## MODELO SPMD

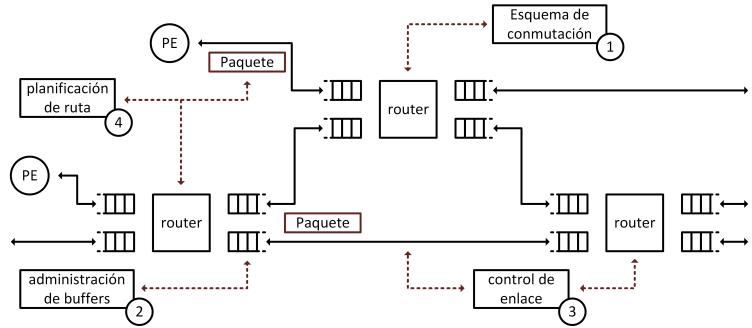
El modelo *SPMD*<sup>7</sup>(*Single Program, Multiple Data*) es una técnica para la explotación de paralelismo en tareas de procesamiento. Dentro de la taxonomía de Flynn<sup>21</sup> se clasifica como una subcategoría del grupo MIMD(*Multiple Instructions, Multiple Data*). Bajo esta organización de procesamiento una misma tarea se ejecutará, sobre diferentes conjuntos de datos, en múltiples unidades de procesamiento de manera paralela. A diferencia del modelo SIMD(*Single Instruction, Multiple Data*), SPMD utiliza múltiples procesadores independientes, los cuales cuentan con sus propias unidades de control. El uso de unidades de control independientes permite a arquitecturas SPMD el utilizar como unidades funcionales procesadores de propósito general, a diferencia de SIMD que requiere una arquitectura de procesador vectorial para implementar su modelo de ejecución paralela. Este modelo de procesamiento empata con arquitecturas MPSoC.

# 3

## Tópicos Específicos

Dentro de este capítulo se abordan técnicas y mecanismos que modelan el comportamiento de la arquitectura propuesta para aceleradores basados en múltiples núcleos de procesamiento. Si el lector no se encuentra familiarizado con los conceptos básicos de redes en-chip, se recomienda cubrir el material presentado en el capítulo 2, en particular las secciones 2.2 y 2.3, antes de internarse en este capítulo.

La figura 3.1 muestra una representación abstracta de una red en-chip. Dentro de esta figura se indica los módulos impactados por los mecanismos tratados en esta sección. El contenido de este capítulo se desarrolla en el siguiente orden: en primer lugar se abordan las características para redes organizadas con una topología tipo malla. De manera subsecuente se describen



**Figura 3.1:** Diagrama abstracto de una red en-chip. PE representa un Elemento de Procesamiento. La figura resalta los módulos de la red impactados por los parámetros descritos dentro de este capítulo.

las técnicas de control de flujo para la arquitectura propuesta en este trabajo. El control de flujo se lleva a cabo a nivel de administración de buffers, puntos 1 y 2 de la figura 3.1, y a nivel de enlace, punto 3 de la misma figura. La sección posterior ofrece una visión general de la familia de algoritmos basados en el *modelo basado en giros* o *turn model*. Los algoritmos de planificación de ruta impactan al módulo con el cual comparten nombre, localizados en la figura bajo el punto 4. Cabe mencionar que este capítulo no muestra detalles de la micro arquitectura de la red, el capítulo 5 aborda dicho nivel de detalle.

## TOPOLOGÍA: MALLA

Una red con topología tipo malla alberga  $N = k^2$  nodos en una cuadricula regular de dos dimensiones, donde cada dimensión contiene  $k$  nodos con enlaces solo a sus vecinos inmediatos. Una malla presenta una organización regular, con canales de comunicación entre nodos de longitud reducida que facilita su colocación en silicio. La figura 3.2 presenta una malla con 16 nodos.

Las mallas son redes asimétricas respecto a sus vértices, ya que los nodos en la periferia

presentan un menor grado de colectividad respecto los nodos centrales (figura 3.2 c)). La asimetría de la topología puede producir desbalances en la distribución de tráfico en la red, ya que los canales centrales tendrán mayor demanda con respecto a los canales de la periferia ya que los primeros forman parte de un mayor numero de rutas entre nodos. Al ser una topología asimétrica el grado de nodo no es homogéneo para todos sus miembros, sin embargo es una practica común definir  $\delta = 4$  para todos los nodos de la red y dejar sin conexión los canales sin vecino inmediato\* como se muestra en la figura 3.2 c).

### DESEMPEÑO DE UNA MALLA

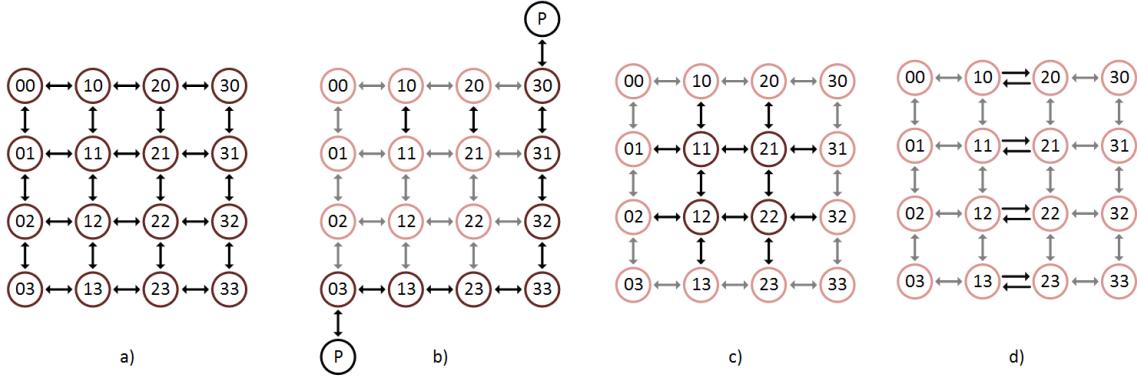
Factores como topología, control de flujo y algoritmo para la planificación de ruta determinan la capacidad de una red para el desplazamiento de información entre sus miembros, el máximo flujo de datos es denominado rendimiento de la red ( $\Theta$ ). En el resto de esta sección se presenta el desempeño ideal de una red tipo malla así como los factores que lo determinan. Los datos presentados en esta sección utilizan la configuración de red de la figura 3.2.

El rendimiento de una red es la relación entre el ancho de banda disponible y la demanda de transporte de información. El máximo rendimiento de la red se presenta cuando uno de sus canales alcanza el estado de saturación, es decir, la carga de trafico en el canal ( $\gamma_c$ ) es igual al ancho de banda del mismo. El rendimiento ideal de una red se representa como la relación en la ecuación 3.1.

$$\Theta_{ideal} = \frac{b}{\gamma_{max}} \quad (3.1)$$

---

\*Los puertos sin conexión son optimizados por las herramientas de síntesis para código de descripción de hardware (HDL en inglés)



**Figura 3.2:** Topología tipo malla. a) red con topología tipo malla, a cada nodo se le asigna una dirección compuesta de una tupla  $(x, y)$ . b) Número de saltos requeridos para comunicar los elementos de procesamiento más distantes de la red. c) Los nodos al centro de la red tienen el mayor *grado* de conectividad. d) Ancho de banda en la bisección.

En la ecuación anterior  $\gamma_{max} = \max(\gamma_C)$ . Para la red en la figura 3.2, bajo un esquema de tráfico uniforme, en todo momento la mitad de los paquetes en transito ( $N/2$ ) deben pasar a través de la bisección. El numero de canales que forman parte de la bisección de una malla esta definida por la ecuación 3.2<sup>\*\*</sup> y se representa gráficamente en la figura 3.2 d). Si el tráfico en la bisección se encuentra distribuido de manera igualitaria entre los nodos que la forman, la máxima carga entre canales de la red esta definida como la carga de tráfico en los canales de la bisección (3.3).

$$B_c = k^{n-1} \quad (3.2)$$

$$\gamma_B = \frac{N}{2B_C} \quad (3.3)$$

Para el caso particular de la malla bajo tráfico uniforme, la carga máxima de trabajo esta

---

<sup>\*\*</sup>Para redes tipo malla el numero de dimensiones  $n$  es igual a 2.

definida por la carga en cualquiera de los canales que forman la bisección de la red\*\*\*

$$\begin{aligned}
 \gamma_{M,U} &= \frac{N}{2k^{n-1}} \\
 &= \frac{N}{2} \cdot \frac{1}{2k^{n-1}} \\
 &= \frac{N}{2} \cdot \frac{k}{2N} \\
 &= \frac{k}{4}
 \end{aligned} \tag{3.4}$$

La ecuación de rendimiento requiere el ancho de banda para el canal que define  $\gamma_{max}$ . El ancho de banda esta determinado por la frecuencia de operación del router así como el ancho de canal del mismo, para una red tipo malla el ancho de canal esta definido por la ecuación 3.5.

$$W_M = \frac{W_n}{\delta_n} \tag{3.5}$$

Sustituyendo las ecuaciones 3.4 y 3.5 en la ecuación 3.1 obtenemos:

$$\begin{aligned}
 \Theta_{ideal,M} &= \frac{b}{\gamma_{max}} \\
 &= \frac{fW_M}{k/4} \\
 &= \frac{B_n/\delta_n}{k/4} \\
 &= \frac{4B_n}{n \cdot k}
 \end{aligned} \tag{3.6}$$

---

\*\*\*Para redes tipo malla  $2k^{n-1} = N/k$ , siempre y cuando solo se encuentre un elemento de procesamiento por nodo.

Donde  $B_n = f \cdot W_M$  representa el ancho de banda de un nodo de una malla.

#### LATENCIA DE UNA MALLA

No solo el rendimiento de una malla determina el desempeño de esta como medio de transporte de información, otro factor a tomar en cuenta es la latencia entre el envío-recepción de un paquete. Una malla es una topología con un numero de dimensiones moderado ( $n = 2$ ), por lo que el tiempo de transporte entre origen/destino esta dominado por el numero de saltos entre nodos.

El mínimo numero de saltos en una malla se calcula mediante el promedio de la distancia mínima entre todos los pares posibles de nodos en la red. En promedio un paquete deberá visitar una tercera parte del total de nodos en cada dimensión, por lo tanto el numero mínimo de saltos puede obtenerse mediante la ecuación:

$$H_{min,M} = \begin{cases} \frac{nk}{3} & \text{si } k \text{ es par} \\ n\left(\frac{k}{3} + \frac{1}{3k}\right) & \text{si } k \text{ es inpar} \end{cases}$$

La latencia no solo está compuesta por el numero de saltos entre nodos, un segundo componente a tomar en cuenta es la latencia derivada de la serialización de un paquete en situaciones donde el ancho de canal ( $W_n$ ) no es suficiente para el envío de datos en una sola transacción. La latencia de serialización ( $T_{s,M}$ ) es igual a la relación entre la longitud en bits de un paquete y el ancho de banda disponible para su transporte.

$$\begin{aligned}
 T_{s,M} &= \frac{L}{b} \\
 &= \max\left(\frac{\delta n L}{W_n}\right)
 \end{aligned} \tag{3.7}$$

## CONTROL DE FLUJO

El control de flujo de una red en-chip se encuentra distribuido a través de la lógica de sus encaminadores. El control se divide en dos secciones principales: administración de buffers y control a nivel de enlace. La técnica para la administración de buffers se encarga de definir la partición lógica del espacio físico en las estructuras de almacenamiento en los puertos de entrada de un router, además de determinar los recursos que se deben de asegurar antes de la retransmisión de un paquete. Por otra parte, la técnica de control a nivel de enlace define estrategias para sincronizar la transmisión / recepción de paquetes entre nodos de la red, priorizando la entrega de paquetes entre nodos, y la velocidad de transferencia de dichas transacciones.

## CONTROL DE FLUJO BASADO EN PAQUETES: VIRTUAL CUT-THROUGHT

La red en-chip de este trabajo utiliza la técnica *virtual cut-through*<sup>40</sup> para la administración de buffers en sus routers. Otros trabajos se refieren a esta técnica simplemente como *cut-through*, en gran medida para evitar ambigüedades respecto al uso del término canales virtuales. Este documento hará uso del término abreviado cut-through.

Cut-through divide de manera lógica los buffers en unidades de almacenamiento del tamaño de un paquete al igual que la técnica store & forward, sin embargo, se caracteriza por su

capacidad de tomar decisiones de planificación de ruta en el momento que se reciba la dirección del destino del mensaje. Aunado al proceso acelerado de resolución de ruta, es posible la propagación de los flits pertenecientes a un paquete de manera inmediata tras finalizar el proceso de cálculo de trayectoria y asignación de puerto de salida.

Otra de las características de la técnica en cuestión, es su dualidad de comportamiento dependiendo del estado de la ruta trazada para un paquete. Cuando la ruta se encuentra libre de bloqueos, el retardo de propagación de un paquete está definido por la siguiente expresión:

$$t_{vct} = D(\Delta_r + \Delta_s + \Delta_w) + P[\max(\Delta_s, \Delta_w)] \quad (3.8)$$

Donde  $\Delta_r$  representa el retardo de la lógica para la codificación y cálculo de ruta,  $\Delta_s$  es el tiempo necesario para que un flit cruce el camino de datos del encaminador,  $\Delta_w$  es el retardo de propagación de las pistas que forman un canal entre routers,  $D$  es la distancia en saltos desde el nodo origen al nodo destino, y finalmente  $P$ , es el número de flits que conforman el paquete. En el segundo escenario, el paquete encuentra un bloqueo en su ruta, y debe de esperar a que el camino se despeje para continuar su tránsito. Para que la ecuación 3.8 sea válida bajo un bloqueo en ruta, es necesario agregar un término con valor equivalente a la duración del bloqueo. Un router cut-through se comporta como un encaminador tipo wormhole durante bajas cargas de trabajo, debido a un número reducido de bloqueos a través de sus canales. Durante cargas altas de trabajo, el mismo router se comportara de manera similar a un encaminador store & forward, capturando en su totalidad el paquete antes de realizar el reenvío del mismo.

Wormhole es la técnica más popular para el control de flujo en redes en-chip de propósito general, sin embargo, trabajos de investigación <sup>69,18</sup> han documentado un rendimiento infe-

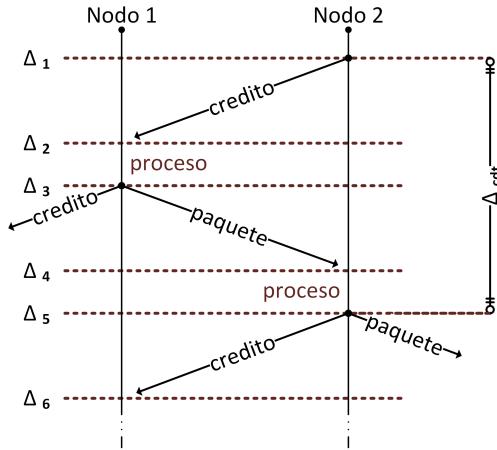
rior de esta técnica frente a cut-through cuando se inyectan cargas de trabajo pesadas a la red. Gran parte de las disminución de rendimiento en redes wormhole se debe al relativamente bajo umbral de saturación derivado de la ocupación de múltiples routers cuando un paquete encuentra un bloqueo en su camino. Durante la saturación de una red cut-through, cuando un paquete se encuentra con un bloqueo, solo un router se ve afectado por la contingencia, mientras los demás canales y encaminadores operan de manera habitual. Vale la pena mencionar que la mayor eficiencia de redes cut-through bajo órdenes altos de trabajo requiere el pago de una penalización en área debido al incremento de espacio de buffer requerido en cada router. El tamaño estático de los paquetes de la red, descrita en este trabajo, limitan el impacto en área resultado de la implementación del control de flujo cut-through.

Una descripción gráfica del comportamiento de una transacción cut-through puede encontrarse en la figura 2.II.

#### CONTROL A NIVEL DE ENLACE BASADO EN CRÉDITOS

Se ha implementado una estrategia de control de flujo a nivel de enlace basado en créditos. El control basado en créditos requiere que cada nodo de la red mantenga un registro del espacio disponible en los buffers de recepción de paquetes de cada uno de sus vecinos inmediatos. El objetivo de este registro es el evitar la pérdida de paquetes debido a la falta de espacio para su recepción. Dado que se utiliza una administración de buffers tipo *cut-through*, la unidad atómica de almacenamiento es el paquete, por lo que un crédito equivaldrá al espacio necesario para almacenar todos los flits de un paquete.

Durante un restablecimiento general de la red, cada uno de los routers inicia sus contadores de créditos con la cantidad de espacios disponibles en los buffers de cada uno de sus nodos



**Figura 3.3:** Línea de tiempo de proceso de transmisión de créditos. El nodo emisor (Nodo 1) resta un crédito con cada paquete que envía al nodo receptor. Cada crédito representa espacio de almacenamiento en los buffers del Nodo 2. El nodo receptor envía de vuelta un crédito con la salida de un paquetes de su medio de almacenamiento temporal. El escenario representado en esta figura supone nodos de red con espacio de almacenamiento suficiente para 1 solo paquete.

vecinos. Con cada envío de paquete, el router decremente su contador de créditos. Antes de iniciar una transmisión, el encaminador emisor es responsable de verificar que aún cuenta con créditos para realizar el envío de un paquete, en caso contrario, este debe de retenerlo hasta que el router receptor cuente con el espacio suficiente para la captura de los datos.

El retorno de créditos es el mecanismo para informar que un router receptor a liberado un espacio en su buffer, se hace uso de una señal dedicada entre nodos para esta tarea. Al momento de la recepción de un crédito, el encaminador emisor incrementa su contador de créditos.

La figura 3.3 muestra el proceso de administración de créditos entre dos nodos de la red. El escenario inicia con la recepción en el nodo número 1 de un crédito durante  $\Delta_2$ . La recepción de un crédito implica el incremento en el contador interno del nodo, esta operación requiere un tiempo equivalente a  $\Delta_3 - \Delta_2$ . El nodo reconoce la recepción del crédito en  $\Delta_3$ , por lo que

de manera inmediata reinicia la operación de envío de paquetes al nodo 2. La salida de un paquete del nodo 1 conlleva a la liberación de espacio en uno de sus buffers de recolección de paquetes, por lo que de manera simultánea con la salida del paquete en dirección del nodo 2 durante  $\Delta_3$ , el nodo 1 regresa un crédito al nodo origen del paquete que se acaba de transmitir. El nodo 2 recibe el último flit transmitido durante  $\Delta_4$ . El nodo 2 regresara un crédito al nodo 1 solo cuando libere el espacio ocupado por el paquete de reciente captura, este proceso puede llevar varios ciclos de reloj en caso de no contar con los recursos necesarios para reenviar el paquete a su destino. En este escenario, el nodo 2 no encuentra problemas para la retransmisión del paquete, por lo que después del tiempo necesario para la decodificación de la cabecera y la gestión de uso de recursos de salida, el paquete es liberado en dirección de su destino. El tiempo de gestión descrito anteriormente es equivalente a la diferencia  $\Delta_5 - \Delta_4$ . El tiempo  $\Delta_{cdt}$ , denominado *tiempo mínimo de retorno de crédito*, es uno de los parámetros críticos para determinar el desempeño de un encaminador de red.  $\Delta_{cdt}$  determina el tiempo necesario para completar un ciclo completo de retorno de un crédito para una misma posición en el buffer del nodo emisor. Para alcanzar un máximo rendimiento, un encaminador debe de proporcionar espacio suficiente en buffer para mantener transmisiones de paquetes de manera continua, sin interrupciones resultantes de la espera de devoluciones de créditos por parte del nodo receptor.

$\Delta_{cdt}$  limita la máxima tasa de transferencia entre nodos, por ejemplo, si un encaminador requiere de un crédito para el envío de un flit, y solo se cuenta con un crédito por puerto de entrada, la tasa de transferencia máxima estará modelada por la ecuación 3.9.

$$b_{cdt} = \frac{L_f}{\Delta_{cdt}} \quad (3.9)$$

Donde  $L_f$  representa la longitud en bits de cada paquete. La unidad de transferencia está dada en bits por segundo ( $bps$ ). Generalizando la ecuación 3.9 para modelar routers con mas de un espacio en buffer, se agrega el termino  $F$  el cual representa el numero de espacios de recepción en el router objetivo. El incremento en el espacio de almacenamiento temporal permite el intercambio de un numero mayor de flits/paquetes antes de ser detenido el proceso debido a la falta de créditos. La tasa de transferencia de flits por  $\delta_{cdt}$  se expresa en la ecuación 3.II.

$$b_{flits} = F \frac{L_f}{\Delta_{cdt}} \quad (3.IO)$$

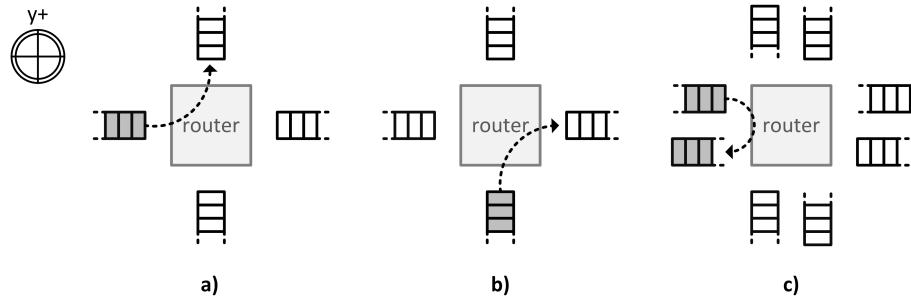
De esta manera se calcula el número de buffers necesarios para garantizar la saturación de cada canal de la red, evitando el incremento de la latencia en el transporte de un paquete debido al tiempo de retorno de un crédito.

Para evitar una degradación del rendimiento es necesario proveer un mínimo de créditos para mantener una transferencia de paquetes constantes, se puede establecer relación en el numero de créditos respecto a  $\delta_{cdt}$  mediante la ecuación 3.II:

$$F \geq \frac{\Delta_{cdt} b_{cdt}}{L_f} \quad (3.II)$$

### PLANIFICACIÓN DE RUTA: *MODELO BASADO EN GIROS*

El modelo basado en giros<sup>23</sup>, denominado *Turn model* en idioma inglés, ofrece un conjunto de reglas para la generación de algoritmos de encaminamiento para redes con topologías tipo malla o hipercubo. Los algoritmos obtenidos siguiendo los lineamiento de este modelo tienen

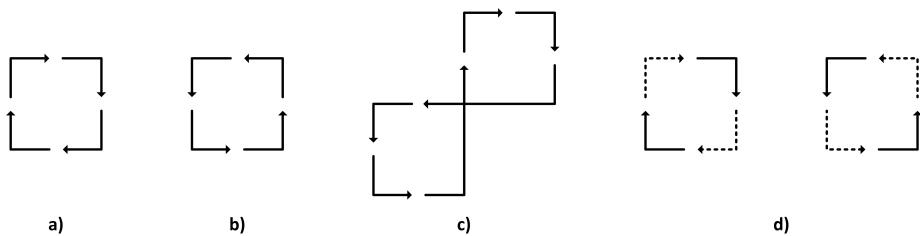


**Figura 3.4:** Giros previstos durante una transferencia a través de la red.a) giro desde  $x$  en dirección a  $y$ . b) giro desde  $y$  en dirección a  $x$ . c) Giro de  $180^\circ$ , donde la dimensión de tránsito permanece sin alteración, sin embargo, la dirección de desplazamiento es alterada desde  $x+$  a  $x-$ .

la característica de ser parcialmente adaptativos, además de poder generar rutas mínimas o no mínimas.

El concepto de giro se refiere al cambio de dimensión durante el desplazamiento de un paquete a través de la red. La figura 3.4 exemplifica cada uno de los giros previstos por el modelo: Un giro simple se considera cualquier cambio de dimensión durante el traslado de un paquete, la figura 3.4 apartado *a*) refleja un giro de la dimensión  $x$  a la dimensión  $y$ , mientras que el apartado *b*) representa un cambio desde la dimensión  $y$  a la dimensión  $x$ . El apartado *c*) muestra un giro de  $180^\circ$ , se considera a este giro como un cambio de dirección mas no de dimensión. El giro de  $0^\circ$  no se encuentra ejemplificado en la figura 3.4, este giro representa un cambio de canal virtual dentro de la misma dirección y la misma dimensión. Si se requiere mayor información sobre este último tipo de giro se recomienda al lector la referencia de los autores *Glass et. al.*

El principal desafío de cualquier algoritmo de encaminamiento es el proveer de rutas libres bloqueos. Un bloqueo o *deadlock*, se presenta cuando se forma una dependencia cíclica de peticiones de recursos entre paquetes como se muestra en la figura 3.5. La estrategia empleada



**Figura 3.5:** Posibles relaciones cíclicas en redes con topologías tipo malla. Los apartados a) y b) muestran la forma más simple de un *deadlock*. El apartado c) muestra una dependencia cíclica compleja, la cual está formada por 6 giros y tránsito en direcciones opuestas. d) Giros prohibidos por el algoritmo XY para prevenir la formación de dependencias cíclicas.

por el modelo basado en giros es el prohibir un subconjunto de giros posibles en la red de manera que se elimine toda posibilidad de creación de referencias cíclicas.

La estrategia de prohibición de virajes no es única de los algoritmos desarrollados a partir del modelo basado en giros. El algoritmo XY<sup>8</sup> por ejemplo prohíbe los cuatro giros exhibidos en la figura 3.5 d) con la finalidad de evitar la creación de dependencias cíclicas. El modelo basado en giros tiene como fundamento un conjunto de pasos para la creación de nuevos algoritmos de ruteo. A continuación se lista un subconjunto de estos pasos, los cuales resultan de particular interés para topologías tipo malla:

1. Clasificar cada canal de la red de acuerdo a la dimensión a la cual pertenece. Por ejemplo, en una malla 2d podemos clasificar los canales en las dimensiones  $x$  o  $y$ .
2. Determinar los giros posibles en la red. Para mallas existen 8 giros distintos.
3. Identificar todos las relaciones cíclicas que se pueden formar en la red, es importante verificar la existencia de ciclos formados a partir de combinaciones complejas de giros.
4. Prohibir un giro de cada ciclo para evitar su formación.

Si la red en-chip permite la inclusión de giros de  $0^\circ$  y  $180^\circ$ , estos se agregan después del punto 4 de la lista anterior y bajo la condición que la inclusión no genere nuevas dependencias que formen un ciclo.

#### GRADO DE ADAPTABILIDAD

Como se mencionó al inicio de esta sección, los algoritmos generados a partir del modelo basado en giros son parcialmente adaptativos. Cada algoritmo puede presentar diferentes grados de adaptabilidad, este último concepto se expresa mediante la ecuación:

$$S_f = \frac{(\delta x + \delta y)!}{\delta x! \delta y!} \quad (3.12)$$

Donde  $S_f$  representa el numero de caminos cortos entre los nodos origen ( $o_x, o_y$ ) y destino ( $d_x, d_y$ ) ofrecidos por un algoritmo completamente adaptativo,  $\delta x$  representa la diferencia  $|d_x - o_x|$ , y finalmente  $\delta y = |d_y - o_y|$ . Los algoritmos presentados en las siguientes subsecciones no pueden ofrecer el número máximo de caminos cortos en todas las direcciones de la red, por este motivo son considerados parcialmente adaptativos.

#### ALGORITMO WEST FIRST MINIMAL

El algoritmo *west first minimal* prohíbe los giros  $Y_{neg} \rightarrow X_{neg}$  y  $Y_{pos} \rightarrow X_{neg}$ , cada uno de estos giros rompe la posibilidad de crear una relación cíclica. Bajo west first, todos los paquetes deben de desplazarse en primer lugar en la dirección  $X_{neg}$  en caso de ser necesario, ya que los giros para tomar esta dirección se encuentran prohibidos. Una vez terminado este desplazamiento inicial, el paquete puede ser encaminado de manera adaptativa hasta alcanzar su destino. El algoritmo 1 ofrece un pseudocódigo para el encaminamiento tipo *west first*

*minimal*, esta descripción es capaz de manejar un paquete desde cualquier dirección inicial.

La ecuación 3.13 modela el grado de adaptabilidad para el algoritmo west first. De acuerdo a esta ecuación, el área de la red que tiene capacidad de encaminar paquetes de manera adaptativa abarca todos los nodos que satisfacen la condición  $d_x \geq o_x$ . Esta condición resulta valiosa si se conoce el patrón de tráfico que se espera en la NoC.

$$S_{west-first} = \begin{cases} \frac{(x+y)!}{x! y!}, & \text{si } d_x \geq o_x \\ 1, & \text{en otros casos} \end{cases} \quad (3.13)$$

La figura 3.6 muestra 4 rutas calculadas mediante el algoritmo west first, 3 de estas rutas presentan bloqueos a través de la red mostrando la capacidad de adaptabilidad del encaminamiento de paquetes. El caso particular donde el paquete inicia en la esquina inferior derecha de la red ejemplifica el peor de los casos para este algoritmo, ya que no se satisface la condición  $d_x \geq o_x$ . West first se implementa como un algoritmo de ruteo distribuido, por lo que cada nodo de la red toma sus propias decisiones de encaminamiento tomando en cuenta solo la información de tráfico de sus vecinos inmediatos. Este último mecanismo permite tomar rutas alternas inclusive cuando el camino inicial fuese el peor de los casos.

Data: dirección de nodo actual ( $localX, localY$ )  
dirección de nodo destino ( $destX, destY$ )

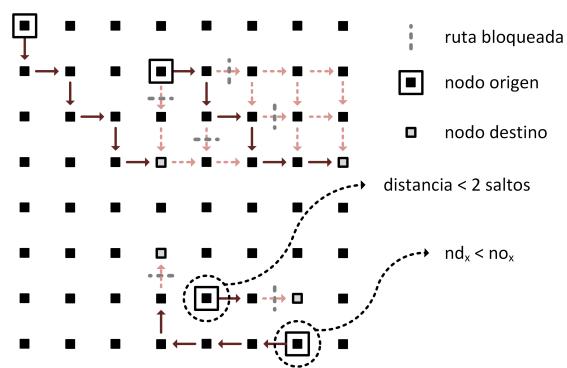
Result: canales de salida válidos ( $canales$ )

begin

```
    offsetX = destX - localX;
    offsetY = destY - localY;
    if  $offsetX < o$  then
        | canales := X-;
    end
    if  $offsetX > o \text{ and } offsetY < o$  then
        | canales := (X+, Y-);
    end
    if  $offsetX > o \text{ and } offsetY > o$  then
        | canales := (X+, Y+);
    end
    if  $offsetX > o \text{ and } offsetY = o$  then
        | canales := X+;
    end
    if  $offsetX = o \text{ and } offsetY < o$  then
        | canales := Y-;
    end
    if  $offsetX = o \text{ and } offsetY > o$  then
        | canales := Y+;
    end
    if  $offsetX = o \text{ and } offsetY = o$  then
        | canales := elemento de procesamiento;
    end
```

end

Algoritmo I: *West-First minimal*



**Figura 3.6:** Rutas planificadas por medio del algoritmo west-first minimal. Las rutas creadas por medio de este algoritmo permite sortear bloqueos gracias a su capacidad de adaptabilidad. La ruta con inicio en la esquina inferior derecha muestra los cambios de adaptabilidad de la ruta conforme es reevaluada en cada nodo de la red.

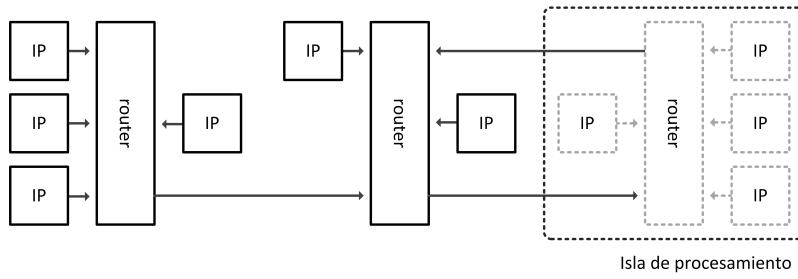
# 4

## Trabajo relacionado

Dentro de este capítulo se presenta un resumen de diseños de redes en chip específicas para dispositivos reconfigurables. Cada trabajo aborda el tema de optimización de diseño tomando ventaja de alguna de las características particulares para dispositivos FPGA.

### REDES EN-CHIP EN FPGA

La universidad Brigham Young en conjunto con la corporación Ricon Research desarrollaron una arquitectura exclusiva para dispositivos reconfigurables de nombre PNOC<sup>26,27</sup>. La arquitectura se caracteriza por el uso del concepto de isla de procesamiento, cada isla está formada por un conjunto de bloques propietarios funcionales y un router para la interconexión

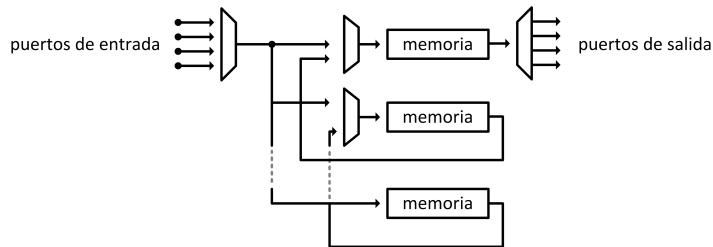


**Figura 4.1:** Diagrama a bloques de red PNoC. La comunicación entre islas permite escalar el número de unidades funcionales del sistema. Además, el uso de múltiples isla permite la distribución de la demanda de recursos en diferentes áreas del sistema.

de todos ellos. El router de pNoC utiliza conmutación de circuitos para crear enlaces de comunicación entre sus miembros, además, utiliza tablas de encaminamiento para todos los cálculos de ruta. La figura 4.1 muestra la organización de una red pNoC.

El uso de tablas de ruteo permite la implementación de un sistema sencillo de transmisión masiva (broadcast) para todos los miembros de la red. En cuanto a escalabilidad, pNoC permite el uso de un puerto de salida para la comunicación con otras islas de procesamiento, permitiendo la interconexión de múltiples islas. La interacción entre unidades funcionales y estructura de interconexión se lleva a cabo por medio de estructuras FIFO, permitiendo el uso de diferentes frecuencias de operación entre miembros de la red.

Jovanovic *et al.* presenta la arquitectura denominada CUNOC<sup>38</sup>. El uso de una topología tipo malla y una estrategia de conmutación de paquetes define el comportamiento para esta red. La cualidad principal de este trabajo es el uso de routers denominados *communication unit*. Este último busca reducir el número de recursos necesarios para su implementación, en específico los requerimientos de elementos de memoria, eliminando de cada router cualquier forma de medio de almacenamiento temporal en sus puertos de salida. En lugar de unidades de almacenamiento distribuidas en cada puerto de salida, se ofrece una unidad de almacenamiento



**Figura 4.2:** La estructura central de almacenamiento para los paquetes entrantes a un router CuNoC representa una disminución en los recursos necesarios para su implementación, sin embargo, crea un cuello de botella para el envío rápido de información hacia su siguiente salto en la red.

miento centralizada para todos los paquetes en transito a través del encaminador.

todos los paquetes entrantes al encaminador de la red CUNOC deben de ser almacenados en el espacio de memoria común antes de poder ser propagados a una salida del router. El proceso de liberación de un paquete a la red requiere de 2 ciclos de reloj adicionales a la latencia generada por el tráfico de paquetes dentro del encaminador. La figura 4.2 muestra el esquema de almacenamiento propuesto por CUNOC. Para los cálculos de trayectoria a través de la red se utiliza una versión modificada del algoritmo *xy*, la cual permite la inclusión del tráfico presente en los vecinos de cada nodo de la red para la toma de decisión de la ruta a seguir.

Quark<sup>49,52,48,51</sup> es una red en-chip que propone una topología *spidergon*<sup>12,50</sup> modificada. En una topología spidergon tradicional cada nodo de la red presenta canales de comunicación con sus vecinos izquierdo, derecho y diagonalmente opuesto. En particular, el canal diagonal presenta cargas de trabajo mayores que sus contrapartes laterales debido a ser el único camino para alcanzar los nodos opuesto de la red. Para reducir la carga en este enlace Quark propone el anexo de un nuevo canal diagonal para incrementar el ancho de banda disponible.

El algoritmo de encaminamiento propuesto para Quark divide a los nodos miembro de la red en 4 cuadrantes, si el destino de una transferencia se encuentra en los cuadrantes vecinos

al nodo, la red utilizará de manera constante los canales izquierdo o derecho para enviar el paquete. En caso que el destino se encuentre en los cuadrantes opuestos al nodo origen, la red utiliza el canal diagonal. El algoritmo mencionado anteriormente es determinístico, por lo que la red siempre utiliza las misma ruta para la transferencia de paquetes entre dos nodos.

Moraes *et al.*<sup>33</sup> presenta la arquitectura HERMES. La arquitectura se caracteriza por una topología tipo malla irregular, ya que existen dos modelos de router. Los nodos al centro de la red presentan 5 canales de comunicación mientras que los nodos a la periferia sólo ofrecen 4 puntos de interconexión. *HERMES* se caracteriza por la reducción del consumo de recursos mediante el uso de una sola unidad de cálculo de ruta para todos los puertos del router. El proceso de arbitraje, cálculo de ruta y reenvío de paquete toma a un nodo de *HERMES* dos ciclos de reloj, sin embargo, solo puede atender una sola petición a la vez. El uso de multiplexado en tiempo de la unidad de cálculo de ruta impone una fuerte restricción en rendimiento para esta red.

Janarthanan *et al.* presentan la arquitectura MOCRES<sup>37</sup>, una red en-chip específica para dispositivos reconfigurables. El diseño sobresale por su propuesta de integración de canales virtuales, de longitud variable, mediante el uso de bloques rígidos de memoria BRAM. El uso de bloques propietarios de memoria limitan la implementación del sistema a dispositivos fabricados por Xilinx. Los autores del trabajo resalta la disminución en el consumo de elementos reconfigurables al utilizar bloques BRAM como medio de almacenamiento temporal. *MOCRES* utiliza el algoritmo determinístico *xy* para la planificación de rutas de transmisión de paquetes. La arquitectura de este trabajo deja pasar la oportunidad del uso de algoritmos de encañamiento adaptativos aun cuando se cuenta con canales virtuales.

socwire<sup>36,35</sup>, una red en-chip diseñada para dispositivos reconfigurables en aplicaciones es-

paciales, destaca por su compatibilidad con el protocolo spacewire<sup>58,59,62,35,45</sup> utilizado por la agencia espacial Europea. Una red Socwire está formada por routers capaces de ofrecer servicio de comunicación a un máximo de 32 unidades funcionales. Otro punto destacable de este desarrollo es la habilidad de tomar ventaja de la capacidad de reconfiguración parcial<sup>20</sup> de los dispositivos FPGA para permitir el intercambio de unidades funcionales del sistema sin perturbar su operación.

Los puertos del router con compatibilidad para el protocolo Socwire descomponen los mensajes a un formato de trama intermedio que el router utiliza para la entrega de mensajes entre unidades funcionales. De igual forma, una trama de datos que abandona el encaminador a través de un puerto con compatibilidad Socwire, es transformada del formato intermedio del router a un mensaje del protocolo anteriormente mencionado. Los encaminadores ofrecen servicios de calidad de servicio como detección y corrección de errores durante la transmisión de mensajes.

Zeferino *et al.* proponen la arquitectura parametrizable SOCIN<sup>75</sup>. Este trabajo propone el diseño de un encaminador parametrizado capaz de adecuarse para su implementación en redes con topologías tipo torus y malla. Una característica destacable es el cálculo de ruta procesado en las interfaces de red de cada nodo, de manera que la lógica necesaria para implementar los encaminadores se simplifica, al solo operar como decodificadores y retransmisores de información. La técnica de conmutación de paquetes es utilizada para la administración de recursos en los routers de la red y se utiliza la técnica *wormhole* para el control de flujo de información.

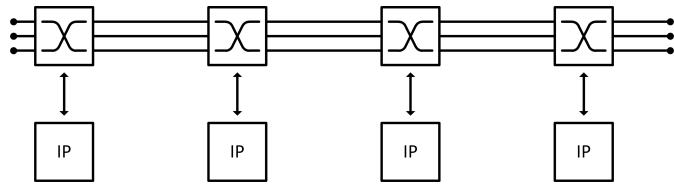
Vale la pena resaltar que esta red utiliza un modelo maestro - esclavo para efectuar las transferencias de información entre nodos. Un nodo maestro debe de iniciar una transacción para comenzar el envío de mensajes, por su parte, el nodo esclavo se limita a la recepción de infor-

macion.

ROC<sup>24,17</sup> es una red basada en una topología tipo anillo redundante. Esta arquitectura presenta un mecanismo minimalista para el manejo de paquetes a través de la red: cada paquete en tránsito es capturado por los encaminadores en su trayecto. Dentro del encaminador, se analiza la dirección destino del paquete, en caso de estar dirigido al nodo actual el encaminador retira el paquete de la red y lo transfiere a la unidad funcional, en caso contrario, el router propaga el paquete al siguiente encaminador. La interfaz de red de *ROC* se encuentra fuertemente integrada al router, al grado de que resulta imposible distinguir actividades exclusivas de la interfaz.

La interacción de encaminador con unidad funcional se lleva a cabo mediante una estructura de almacenamiento FIFO, la unidad funcional escribe datos al FIFO de router, el cual a su vez reserva el uso de un anillo para la inyección del nuevo paquete. En caso que todos los anillos se encuentren ocupados, el router almacena de manera temporal el paquete y espera por disponibilidad de canales de comunicación.

A. Ahmadinia *et al.* proponen la arquitectura RMBOC<sup>1,5</sup>, la cual consiste en un sistema híbrido entre red en-chip y una estructura de medio compartido. Los routers en una red *RMBOC* se denominan unidad de punto de cruce con elemento de procesamiento, y prestan servicio de manera exclusiva a una unidad funcional. Cada unidad de punto de cruce tiene acceso a canales de comunicación con el vecino a la izquierda y derecha del nodo. El conjunto de todos los nodos de la red forma un medio compartido segmentado<sup>64</sup>, donde cada router es responsable de recibir peticiones y bloquear recursos para la formación de un enlace entre dos nodos. *RMBOC* puede analizarse como una red en-chip unidimensional con conmutación de circuitos. La figura 4.3 muestra un ensamble típico *RMBOC*.



**Figura 4.3:** RMBoC Utiliza un híbrido entre un medio compartido y una comutación de circuitos, para incrementar su rendimiento requiere de un aumento de líneas de interconexión que pueden resultar costosas para sistemas en-chip.

El manejo de contingencias durante una transmisión de *RMBoC* consiste en el envío de una señal de bloqueo al nodo transmisor, el cual, al recibir esta señal procederá a la liberación de todos los canales de comunicación reservados y esperará un tiempo de guarda para reintentar la transacción deseada.

*DYNOC*<sup>6</sup> es una red en-chip con topología tipo malla que sobresale por el uso del algoritmo de encaminamiento denominado *s-XY*<sup>5</sup>, el cual tiene la capacidad de cambiar de dirección en caso de encontrar una ruta con bloqueo y regresar al camino original una vez sorteado el obstáculo en su ruta original. En su trabajo Bobda *et al.* proponen una red *DYNOC* saturada con encaminadores, de esta manera proveen de un mayor número de rutas alternas para el desvíos de transmisiones en caso de bloqueo.

Pionteck *et al.*<sup>61</sup> desarrollaron la arquitectura de red en-chip denominada *conochi*, su principal característica es el estar diseñada para ejecutar operaciones de reconfiguración dinámica a nivel de unidades funcionales y encaminadores. El mecanismo de manejo de ruta de esta red está basado en tablas, cada tabla se encuentra alojada en un bloque de memoria rígido, el cual puede agregar o modificar entradas en base a comandos de red. Esta capacidad de edición de las tablas de encaminamiento permite a *conochi* el agregar nuevas unidades funcionales al sistema, además de permitir la salida de otras sin crear caminos inválidos para la transferencia de paquetes. La tabla 4.1 presenta un resumen de los trabajos expuestos.

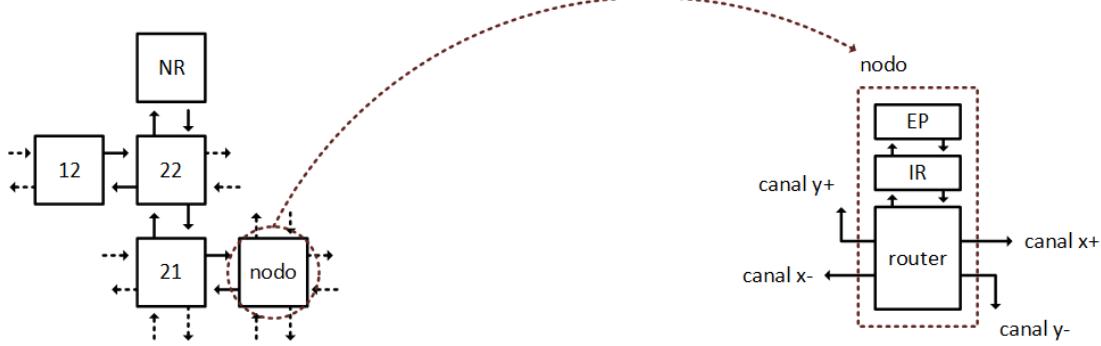
NoC	Topología	Relacion con EP	Algoritmo de calculo de ruta	Tipo de commutacion	frecuencia de operación	Ocupación (Slices)	Ancho de palabra de datos (bits)	Ancho de banda
PnoC	crossbar	red indirecta	tablas de ruteo	commutación de circuitos	126 MHz	1223	8	NP
CuNoC	malla	red indirecta	DOR XY	Store & forward	241 MHz	128	32	NP
Quark	spidergon	red directa	calculo de cuadrante	wormhole	NP	1453	32	NP
Hermes	malla	red directa	DOR XY	wormhole	25 MHz	316	10	500 Mbps
MoCReS	malla	red directa	DOR XY	virtual cut-through	357 MHz	282	8	2,85 Gbps
SoCWire	múltiple	red directa	XY modificado	wormhole	180 MHz	1270	32	2800 Mbps
SoCIN	malla	red directa	DOR XY	wormhole	49 MHz	2100	32	NP
RoC	anillo	red directa	NP	NP	138 MHz	144	32	552 Mbps
RMBoC	bus dinámico	red indirecta	a medida	circuit switched	94 MHz	5084	32	NP
DyNoC	malla irregular	red indirecta	variante DOR XY	NP	391 MHz	1689	32	NP
CoNoChi	malla irregular	red indirecta	tablas de ruteo	virtual cut-through	88 MHz	493	32	NP

Tabla 4.1: Tabla comparativa de redes en chip para dispositivos reconfigurables. NP - Dato no proporcionado en referencia

# 5

## Nodos

El capítulo 5 describe la arquitectura de los miembros de la red propuesta. El contenido se aborda en el siguiente orden: en primer lugar se presenta el formato de paquete y el *handshake* entre nodos para el intercambio de información. A continuación se describe la arquitectura del router y su correspondiente interfaz de red para la interconexión con elementos de procesamiento. La siguiente sección presenta la arquitectura del nodo *rebotador*, un elemento único de este trabajo. Finalmente se presenta un caso de estudio, donde un router es conectado a un núcleo de cifrado de algoritmo DES<sup>54</sup>. La figura 5.1 muestra un diagrama a alto nivel de los módulos que integran a un acelerador de hardware basado en NOC. .



**Figura 5.1:** Cada nodo de red es una isla de procesamiento en el acelerador. Las líneas de interconexión entre nodos se denominan canales. EP: Elemento de procesamiento, IR: Interfaz de red y NR: Nodo Rebotador.

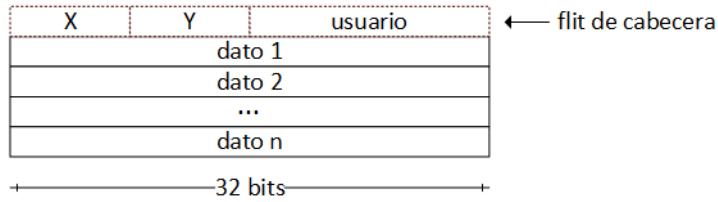
#### PROTOCOLO DE COMUNICACION

El protocolo de comunicaciones esta formado por el formato de paquete y la secuencia de control entre routers. El formato de paquete determina el tamaño de la carga útil de información a ser transmitida, ademas de establecer campos de control para el calculo de ruta de un paquete y su posterior procesamiento en el elemento funcional de un nodo.

La secuencia de control entre routers determina el intercambio de señales que indica a un nodo la llegada de un nuevo paquete a uno de sus puestos de entrada. Ademas de proporcionar el servicio anterior, la secuencia de control permite el envío/recepción de créditos entre routers para evitar la perdida de paquetes debido a la falta de espacio de almacenamiento temporal en un encaminador.

#### FORMATO DE PAQUETE

La infraestructura para el desarrollo de aceleradores de este trabajo implementa elementos de procesamiento homogéneos, con bloques de datos de trabajo de longitud fija para todos ellos. El uso de paquetes de datos de trabajo homogéneos simplifica los mecanismos de control



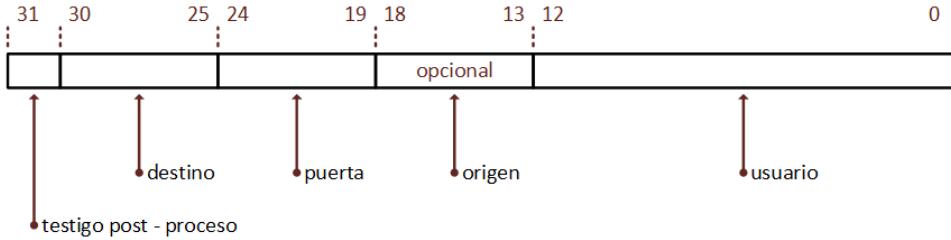
**Figura 5.2:** No existe una restricción en el número de flits para el transporte de información dentro de un paquete, sin embargo, el incremento de estas unidades de transporte acarrea consigo una mayor congestión en la red además de un incremento en la cantidad de almacenamiento temporal requerido por cada uno de los puertos del encaminador. El flit de cabecera contiene la información necesaria para permitir que un paquete sea entregado a su destino.

requeridos para la decodificación de estos, eliminando la necesidad de campos destinados al control de flujo de datagramas de longitud variable.

Todos los paquetes están formados por un flit de cabecera seguido de cualquier número de flits para el transporte de datos. El número de flits que conforman un paquete está determinado por la aplicación, y debe de especificarse de manera previa a la síntesis del acelerador. El tamaño de paquete definido en tiempo pre-síntesis hace innecesario la inclusión de un flit de final de paquete.

El contenido de los flits de datos es transparente para los nodos del sistema, los routers e interfaces de red no tienen mecanismos para la discriminación de campos o formatos dentro de ellos. El flit de cabecera es un caso particular, en el, el número y longitud de campos es fijo y tienen un profundo impacto en el funcionamiento de la lógica de los routers. La figura 5.3 muestra la distribución de campos a lo largo de un flit de cabecera.

Una vez que un paquete a sido aceptado por un elemento de procesamiento, el campo *testigo post-proceso* es activado para indicar que el paquete está en busca de una puerta de salida de la red, y no requiere competir por el uso de un elemento de procesamiento. Los campos *destino*, *puerta* y *origen* almacenan una dirección de nodo formada por una dupla  $\{x,y\}$ . Estos



**Figura 5.3:** Formato de flit de cabecera. El uso del campo *origen* permite la implementación de algoritmos de encaminamiento basados en el modelo *odd-even*. En caso de omitir el uso del campo *origen* durante el proceso de planificación de ruta, su espacio puede anexarse al campo *usuario* para ampliar su longitud.

campos tienen una longitud fija de 6 bits. La distribución de bits de los campos para la representación de direcciones puede variar dependiendo de la geometría de la red, por ejemplo, se puede asignar un mayor número de bits a la dimensión  $x$  en caso que se desee una geometría rectangular en lugar de una cuadrada. El uso de 6 bits como espacio de direcciones permite tener en la red un máximo de 64 nodos. Es posible extender el espacio de direccionamiento sacrificando capacidad de almacenamiento de los campos *usuario* y *origen* para extender de manera efectiva las direcciones que pueden representarse por medio de los campos destino y puerta.

El campo *usuario* ofrece la posibilidad de implementar mecanismos de *calidad de servicio (QoS)*. Un ejemplo del uso de este campo es la verificación de recepción de todos los paquetes liberados a la red para su procesamiento. En el escenario anterior, el campo *usuario* es utilizado para incluir un identificador a cada paquete, mediante este número se puede corroborar que todos los paquetes liberados para procesamiento en la red han sido recibidos en la puerta de salida correcta. Los servicios *QoS* pueden implementarse directamente en hardware o software dependiendo de la interfaz para la extracción de datos del acelerador.

## PROTOCOLO DE ENLACE

El protocolo de enlace entre dos routers asegura la interpretación correcta de una solicitud de envío de paquete, así como la coordinación entre ambos elementos para evitar la perdida de datos debido a una insuficiencia de espacio de almacenamiento en el nodo destino.

El *handshake* entre dos routers se lleva a cabo mediante 4 señales de control:

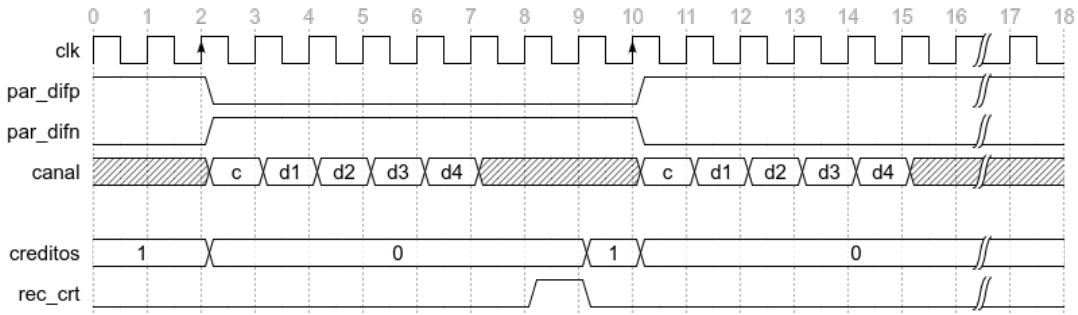
- par\_disp - señal par diferencial positivo (origen).
- par\_disn - señal par diferencial negativo (origen).
- rec\_crt - señal de retorno de créditos (origen).
- env\_crt - señal de retorno de créditos (destino).

El protocolo implementa un par de señales diferenciales dedicadas para el control de transferencia de paquetes, un cambio de estado en el par diferencial indica al router receptor la llegada de un nuevo tren de flits. El uso de un par diferencial dedicado en lugar de un campo de identificación en el flit de cabecera reduce la posibilidad de la creación de paquetes espurios resultado de la corrupción de datos. El sistema de señalización permite la integración de servicios como transferencias tipo *broadcast* o servicios QoS. La figura 5.4 muestra la transmisión de dos paquetes entre encaminadores.

El protocolo se encuentra fuertemente ligado al control de enlace basado en créditos\*, ya que la transmisión de paquetes se ve detenida cuando el router destino ha agotado todos sus espacios de almacenamiento temporal como se muestra en el ciclo 2 de la figura 5.4. Durante el flanco positivo del octavo ciclo de operación, el router destino libera un espacio de almacenamiento y envía un crédito al router origen mediante la señal de paso de crédito (rec\_crt). Un

---

\*Capítulo 3 sección - Control a nivel de enlace basado en créditos



**Figura 5.4:** El protocolo de comunicación involucra un intercambio de señales (*handshake*) de dos pasos. La transmisión inicia con la inversión de la señales del par diferencial (ciclo 2), el intercambio finaliza con la aserción de la señal *rec\_crt* la cual indica que el router destino tiene espacio disponible para la recepción de un paquete adicional.

ciclo de trabajo es necesario para que el router origen procese la recepción del crédito y reinicie el envío de paquetes como se muestra en la figura 5.4 durante el décimo ciclo. El protocolo de comunicación no implementa un medio de corrección de errores de transmisión en la capa física o de enlace del encaminador, esta tarea se delega a capas superiores para mantener un nivel bajo de complejidad en los routers de la red.

#### ARQUITECTURA DEL ROUTER

El router está diseñado para operar bajo el esquema de conmutación de paquetes utilizando la técnica *virtual cut-through*, el control de flujo se basa en el uso de *créditos*. El diseño permite definir de manera previa a la síntesis el ancho de canales de entrada/salida y la capacidad de almacenamiento temporal(*buffers*) para la retención de paquetes. El router puede ser utilizado para formar redes con topología torus o malla.

De manera interna el módulo está organizado en dos particiones lógicas, una dedicada al transporte de datos (*camino de datos*) y una segunda partición para el control de la dirección en la que la información fluye a través del módulo (*camino de control*). La tabla 5.1 muestra

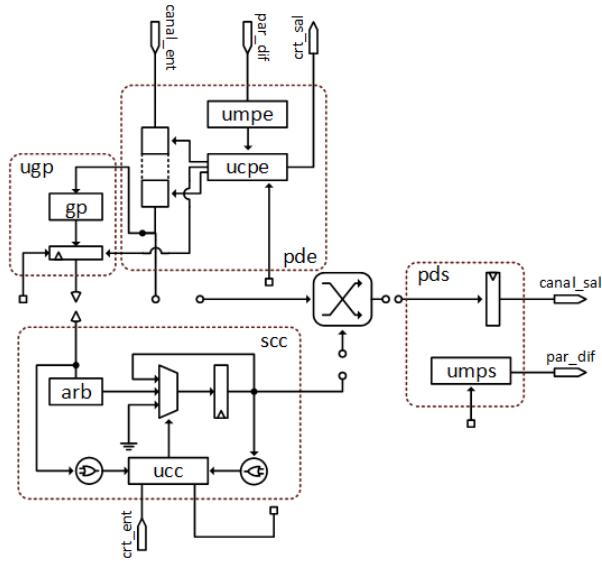
Plano lógico	Unidad	Descripción
Datos	Puerto de entrada	Unidad de recepción de paquetes. De manera interna se encarga de el manejo del protocolo de comunicación así como del almacenamiento temporal de paquetes en transito. Este modulo tiene comunicación directa con unidad de generación de peticiones para disparar el proceso de solicitud de recursos para la salida de información.
Datos	Crossbar	Lógica para la interconexión de entradas y salidas del router.
Datos	Puerto de salida	Unidad de manejo de protocolo de comunicación así como de recepción de créditos desde routers vecinos.
Control	Unidad de generación de peticiones	Modulo encargado del calculo de puertos de salida productivos para la propagación de un paquete. Ademas del calculo de ruta, este modulo se encarga de mantener la solicitud de recursos activa hasta que un arbitro otorgue recursos a su petición.
Control	Segmento control de crossbar	Lógica de control para la configuración del crossbar.

Tabla 5.1: Unidades principales del encaminador.

las principales unidades de la arquitectura.

Es importante definir los términos *canal* y *puerto*, ya que se utilizan para la descripción del encaminador. El término *canal* se utiliza para referirse al medio físico que enlaza dos routers vecinos. El término *puerto* está asociado a los modulo que se encargan de la recepción y decodificación de los paquetes entrantes a un router.

Internamente el router presenta una estructura uniforme, donde cada canal entrante al modulo esta conectado a un puerto de entrada que se encarga de la recepción de paquetes, los puertos de entrada proporcionan los campos de dirección a una unidad generadora de peticiones la cual solicita el uso de un puerto de salida para cada paquete en transito. La admi-



**Figura 5.5:** Diagrama general de un segmento de los caminos de control y datos entre un puerto de salida y uno de entrada. Glosario: UMPE - Unidad de Manejo de Protocolo de Entrada. UCPE - Unidad de Control de Puerto de Entrada. UCC - Unidad de Control de Crossbar. UMPS - Unidad de Manejo de Protocolo de Salida. SCC - Segmento de Control de Crossbar. UGP - Unidad de Generación de Peticiones. PDE - Puerto de Entrada. PDS - Puerto de Salida.

nistración de puertos de salida se lleva a cabo de manera distribuida, donde existe un segmento de control de crossbar encargado de recibir peticiones para el uso de un puerto de salida específico, arbitrar entre todas las peticiones activas y seleccionar una de ellas para otorgar un enlace entre el puerto de entrada solicitante y el puerto de salida solicitado. La conexión entre ambos puertos se lleva a cabo mediante el crossbar del router. La figura 5.5 muestra un segmento perteneciente a la interconexión entre un puerto de entrada, el crossbar y un puerto de salida, este arreglo se replica para cada uno de los canales de entrada/salida del encaminador.

El plano de control del router está formada por módulos *UMPE* - Unidad de Manejo de Protocolo de entrada, *UMPS* (Unidad de manejo de protocolo de salida) y *scc* (segmento de control de crossbar). Los módulos *UMPE* (Unidad de manejo de protocolo de Entrada) y *UMPS*

(unidad de manejo de protocolo de salida) se encargan del cumplimiento de la señalización a través de pares diferenciales requerida por el protocolo de comunicación, el UMPE detecta el cambio en el par diferencial, almacena el nuevo estado de las líneas y genera un habilitador para iniciar el proceso de recepción de un paquete. Por su parte el módulo UMPS recibe una habilitador para generar un cambio válido en el par diferencial para indicar a un router vecino que se ha iniciado la transmisión de un tren de flits. Gran parte de la responsabilidad de las dos unidades mencionadas anteriormente es el mantener el correcto orden de las transiciones en el par diferencial, un desfase en el patrón de cambios derivaría en una incorrecta interpretación de paquetes entrantes.

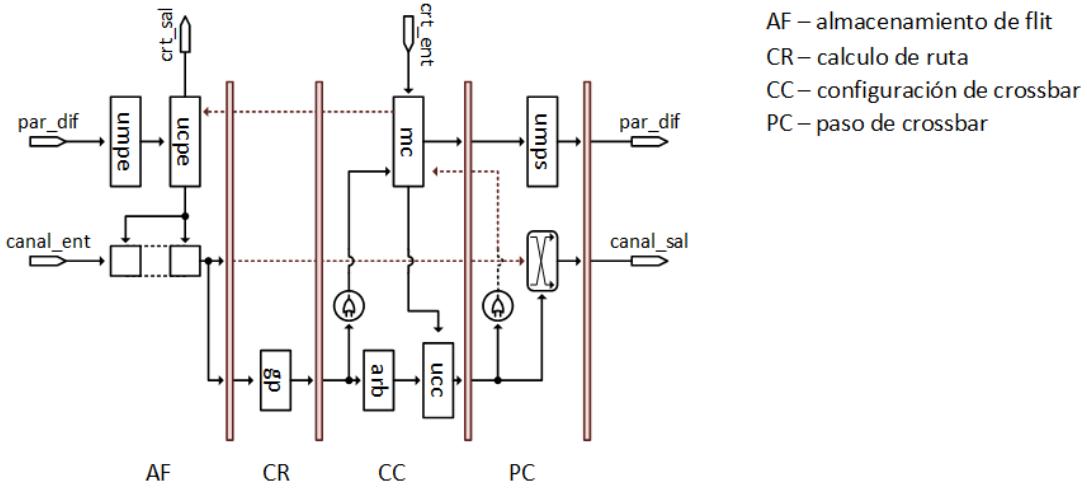
La unidad SCC integra un árbitro y un módulo de retención de resultado de arbitraje controlado por la unidad UCC (unidad de control de crossbar). Las unidades SCC se encargan de administrar un puerto de salida de manera exclusiva por lo que existe una de estas unidades por cada conexión a un vecino del router. El árbitro de la unidad SCC recibe las peticiones generadas desde los módulos UGP, como resultado del proceso de arbitraje una de las peticiones es aceptada y se procede a configurar el crossbar para crear un enlace desde el puerto de entrada ganador y el puerto de salida custodiado por la SCC. La unidad UCC se encarga de retener la configuración del crossbar el tiempo necesario para permitir el paso de todo los flits pertenecientes a un paquete. Adicionalmente, la unidad UCC integra el manejo de créditos para el puerto de salida al cual se encuentra administrando.

A partir de esta sección se utilizará de manera indiferente los términos *cola* o *buffer* para referirse a la estructura de almacenamiento temporal ligado a los puertos de entrada. De igual forma, los términos *estructura de interconexión* y *crossbar* se utilizan de manera indistinta para hacer referencia a la malla de interconexión entre puertos de entrada y salida del router.

## ETAPAS DE SEGMENTACIÓN

El camino critico a través del router se encuentra segmentado por dos etapas registros. De manera adicional, los canales de entrada y salida se encuentran registrados, fungiendo como dos etapas adicionales de segmentación. La figura 5.6 presenta las etapas de segmentación del router y las unidades que pertenecen a cada una de ellas.

- Almacenamiento de flit (AF) - Esta etapa se activa con la recepción de un nuevo paquete. La unidad UMPE actualiza el estado del par diferencial de entrada y emite un testigo a la unidad UCPE para que inicie el proceso de recepción de un nuevo paquete. La unidad UCPE habilita al buffer del puerto para la captura de los flits entrantes.
- calculo de ruta (CR)- Durante la recepción de un flit de cabecera la unidad GP calcula los puertos de salida que abonan a la entrega del paquete a su destino, ademas, genera las peticiones pertinentes. LA unidad GP mantienen el vector de peticiones a todos los puertos de salida relevantes para el paquete en transito. Una vez recibido un arbitraje favorable, esta unidad limpia el vector de peticiones actual y procede a atender peticiones de calculo de ruta pendientes en el buffer.
- configuración de crossbar (CC) - El vector de peticiones desde la unidad GP llega a los árbitros de cada puerto de salida, en caso de existir una petición, el arbitro determina el puerto de entrada que tendrá acceso al puerto de salida durante la ronda de arbitraje actual. La unidad UCC recibe el resultado del proceso de arbitraje y configura el crossbar de manera que el buffer del puerto de entrada ganador tenga un camino al puerto de salida requerido. La unidad UCC da aviso al puerto de entrada que su solicitud a sido aceptada y que el enlace con el puerto de salida se encuentra listo para la liberación del



**Figura 5.6:** El buffer dentro de un puerto de entrada funge como registro para el flit en transito durante el ciclo de reloj actual.

paquete.

- **Paso de crossbar (PC)** - Una vez configurado el crossbar, los flits son transferidos uno a uno desde el buffer de entrada al registro del puerto de salida. Durante esta etapa de segmentación la unidad UCPE se encuentra activa habilitando el proceso de lectura de datos en el buffer de entrada.

El paso de un flit a través del encaminador toma 4 ciclos de reloj. A partir del dato previo es posible determinar el tiempo de viaje redondo de un crédito entre encaminadores vecinos como se muestra a continuación:

$$\Delta_{cdt} = \Delta_{we} + \Delta_{pe} + \Delta_{wd} + \Delta_{seg} + \Delta_{ws} \quad (5.1)$$

Donde  $\Delta_{we}$  es el retardo de propagación de crédito entre routers,  $\Delta_{pe}$  es el tiempo de procesamiento requerido para asimilar un crédito entrante,  $\Delta_{wd}$  representa el retardo de propagación

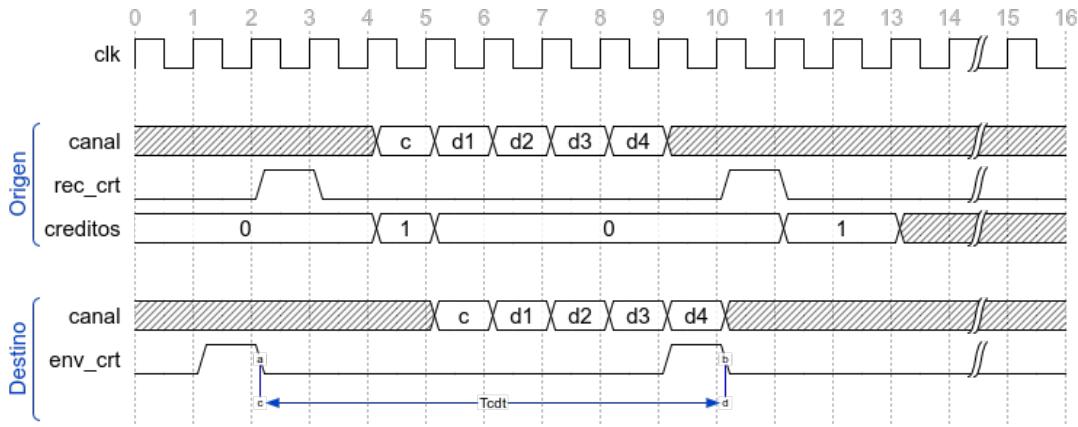


Figura 5.7: Forma de onda del intercambio de un paquete entre dos routers. Se requiere de 8 ciclos de reloj para que un crédito realice un viaje redondo entre los dos routers.

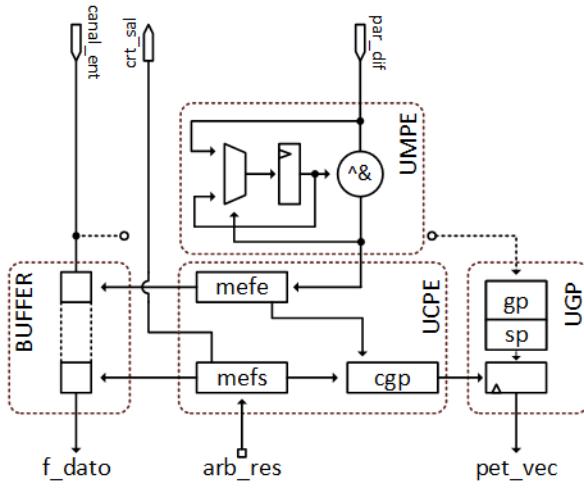
del primer flit en dirección del siguiente salto,  $\Delta_{seg}$  es la latencia para el paso de un flit a través de un router y  $\Delta_{ws}$  es el retardo de propagación de un crédito retornando al router origen. La figura 5.7 muestra la secuencia de la ecuación 5.1. Un  $\Delta_{cdt} = 8$  ciclos de reloj requiere de al menos  $2F$  espacios de almacenamiento<sup>\*\*</sup> para mantener el máximo desempeño de manera constante entre comutadores vecinos.

#### PUERTO DE ENTRADA

El puerto de entrada (figura 5.8) extiende su operación durante dos etapas de segmentación del router. Durante la etapa AF, la unidad UMPE detecta el cambio en el par diferencial, a partir de este evento se generan dos acciones, por una parte la unidad actualiza su banco de estado interno mientras se genera una señal para la unidad UCPE que a su vez iniciara con la captura de flits he indicara que es necesario iniciar el proceso de solicitud de recursos a las unidades SCC. Durante la etapa CR la unidad GP obtiene un vector de peticiones resultante del calculo

---

<sup>\*\*</sup>Consultar capítulo 3 para mas detalles en el calculo de capacidad de buffer.



**Figura 5.8:** Todos los puertos de entrada del router son estructuras homogéneas, de manera que el IP es reutilizable y presenta el mismo comportamiento sin importar al canal que se encuentre asignado.

de ruta para el paquete en transito, al final de esta etapa el vector se registra y se propagan las peticiones a las unidades SCC correspondientes

La unidad UCPE esta formada internamente por dos maquinas de estado finito: Mefe (Maquina de Estado Finito de Entrada) y Mefs (maquina de Estado de salida). Mefe genera las señales de control de escritura para el buffer, y en conjunto con la unidad CGP (control de Generación de peticiones), determina el origen de los campos de dirección para el calculo de ruta y si el resultado de este ultimo proceso debe ser propagado a las unidades SCC. La figura 5.9 a) muestra el modelado de la maquina de estados Mefe.

La unidad CGP lleva un registro interno del numero de paquetes almacenados en el buffer del puerto, ademas de conocer estado de las unidades Mefe y Mefs. A partir de los datos anteriores, la unidad indica a la UGP (Unidad Generadora de peticiones) si debe iniciar un nuevo proceso de generación de peticiones. La tabla 5.2 presenta el árbol de decisiones de la unidad CGP.

MEFE actual	MEFE siguiente	MEFS actual	MEFS siguiente	Paquetes en buffer	habilitador UGP
NEW	PUSH	IDLE	IDLE	o	Activo
X	X	ACK	PULL	>o	Activo
X	X	PULL	PULL	>o	Activo
X	X	X	X	X	Inactivo

**Tabla 5.2:** Solo los estados definidos como 'activo' dispararan la propagación de peticiones a las unidades SCC del router. Los casos no definidos en la tabla mantienen deshabilitado el registro de la unidad CGP.

La maquinada de estado finito MEFS ejerce control sobre las tareas de lectura de datos en el buffer así como el retorno de créditos al router vecino inmediato. La maquina de estados esta modelada por el diagrama que se muestra en el apartado b) de la figura 5.9. Es importante notar que el regreso de un crédito se puede llevar a cabo al momento de la recepción de una resolución favorable por parte del arbitro de cualquier unidad scc, inclusive si se encuentra en medio de la captura de un paquete entrante. La liberación del crédito no corre el riesgo de un desbordamiento en el buffer de entrada como resultado del uso de la técnica virtual cut through, donde el espacio para la recepción de un paquete esta asegurado en el siguiente router, en otras palabras una vez que la transmisión de un paquete ha iniciado se tiene la certeza que todos los flits pertenecientes al mismo serán desalojados del buffer del puerto de entrada, habilitando el espacio para la recepción de un paquete mas.

Dentro de la unidad UGP se lleva el proceso de calculo de ruta, este ultimo ofrece como resultado un vector de peticiones para los puertos de salida que resultan productivos para el avance del paquete en transito. El algoritmo *west-first minimal* ofrece múltiples rutas al ser un algoritmo parcialmente adaptativo. La generación de peticiones simultaneas puede acarrear inconsistencias en la operación de un router, por ejemplo: dos unidades scc reciben una petición de una unidad UGP, ambas unidades scc aceptan la petición y crean un enlace entre el puerto de entrada y sus correspondientes puertos de salida, como resultado el paquete en

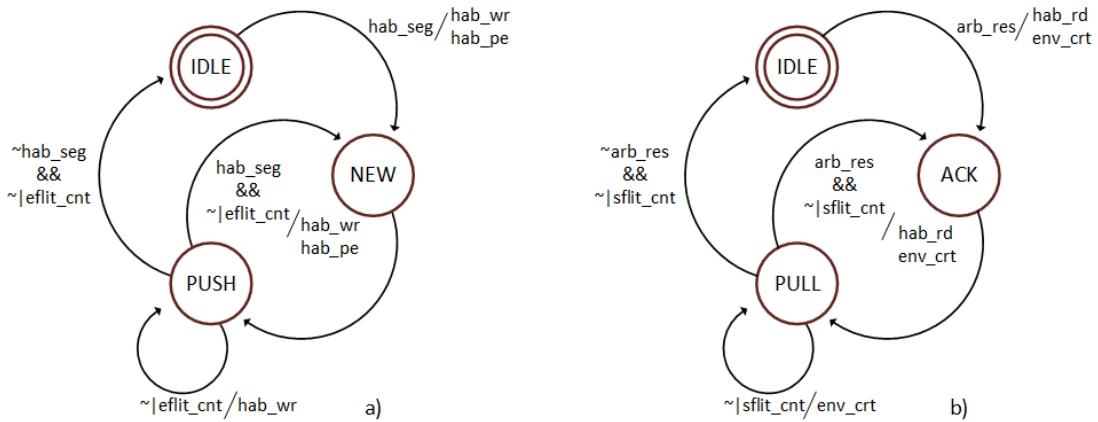
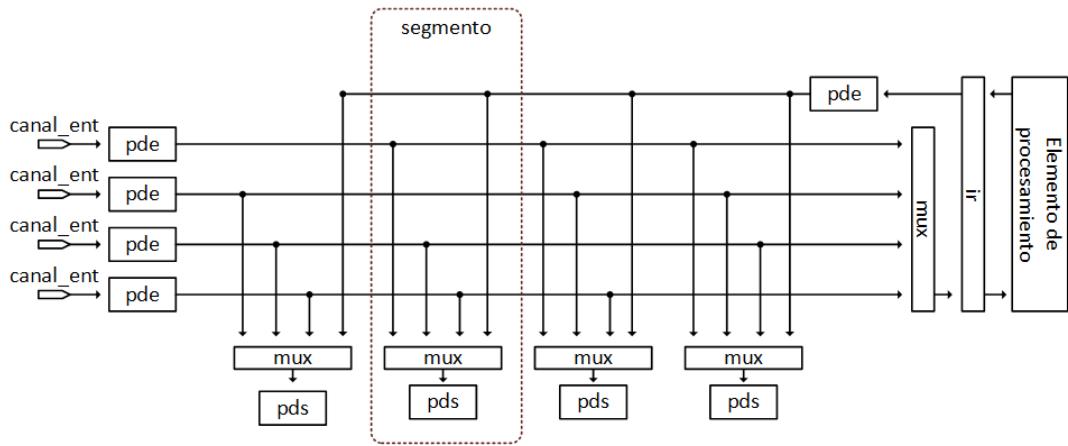


Figura 5.9: a) Maquina de estados finitos de entrada (MEFE). b) Maquina de estados finitos de salida (MEFS).

el buffer es transmitido a través de los puertos de salida asignados, creando un duplicado del paquete. La unida UGP puede agregar módulos para el manejo de casos particulares como el anteriormente descrito, en el escenario anterior un modulo SP (selector de peticiones) se encarga de seleccionar solo una petición del vector generado en base a un esquema de prioridad fija y el estado de las unidades SCC del router.

El router descrito en este trabajo implementa un modulo SP, el cual trabaja aplicando un proceso de filtrado de dos etapas. Durante la primera etapa se reciben todas las peticiones generadas por el modulo GP, solo las peticiones a unidades SCC que se encuentren disponibles en este momento son enviadas a la segunda etapa de filtrado. La segunda etapa consiste en la aplicación de un esquema de prioridad fijo que implementa el siguiente orden:

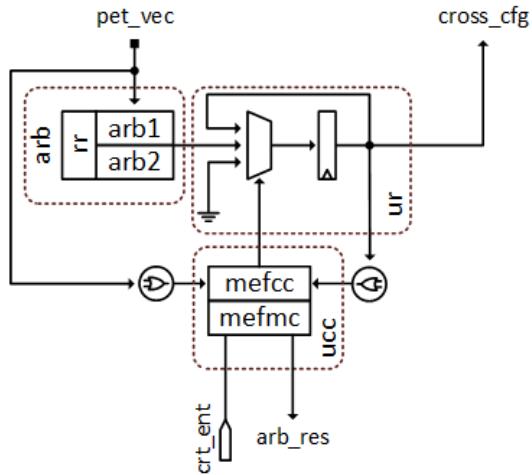
1. Elemento de procesamiento
2. Puerto 'x+' 3. Puerto 'Y+' 4. Puerto 'x-' 5. Puerto 'Y-'



**Figura 5.10:** Cada unidad SCC genera el vector de control para cada uno de los segmentos del crossbar. Los segmentos están configurados para mantener sus líneas de salida en cero mientras no se encuentren paquetes en transito, la inclusión de una un valor constante no repercute en el numero de entradas al multiplexor, solo en la señales de control del mismo.

#### CROSSBAR

El medio de interconexión entre puertos está implementado mediante una estructura tipo *crossbar*, la cual consiste en una cascada de multiplexores independientes para la conexión de varios puertos de entrada con un puerto de salida. Los algoritmos de encaminamiento WFM (west Firts Minimal) y DOR XY (dimensional order routing XY) no permiten el reenvío de un paquete en la misma dirección por la cual ingreso al router, por lo que cada segmento del crossbar no requiere proporcionar un servicio completo de interconexión. Dado que  $\delta$  tiene un valor de 4 para los router tipo malla, y se cuenta con un canal de salida para el elemento de procesamiento de un nodo, cada segmento del crossbar requiere de un multiplexor de 4 entradas y una salida. Un multiplexor es implementado en un dispositivo reconfigurable mediante el uso de LUTs (Look up tables), en la mayoría de los casos dichos elementos cuentan con un 6 entradas y dos salidas, permitiendo la implementación de multiplexores en combinaciones de



**Figura 5.11:** Las unidades SCC son responsables de mantener el control del numero de créditos disponibles en el router vecino inmediato, por lo cual reciben de manera directa la señal de recepción de créditos. El tiempo de proceso de un crédito es igual a un ciclo de reloj debido al cambio de estado de la maquina de estados finito que se encarga de administrar el proceso. La señal 'pet\_vec' representa el vector formado por todas las peticiones enviadas a la unidad SCC desde los puertos de entrada ligados a ella.

4x1 o 2x2. La restricción en la implementación de multiplexores requiere que cada segmento del croosbar sea sintetizado como una estructura tipo cascada en lugar de una jerarquía plana con un solo nivel. La figura 5.10 muestra un diagrama a bloques de la infraestructura de interconexión interna del router.

#### SEGMENTO DE CONTROL DE CROSSBAR

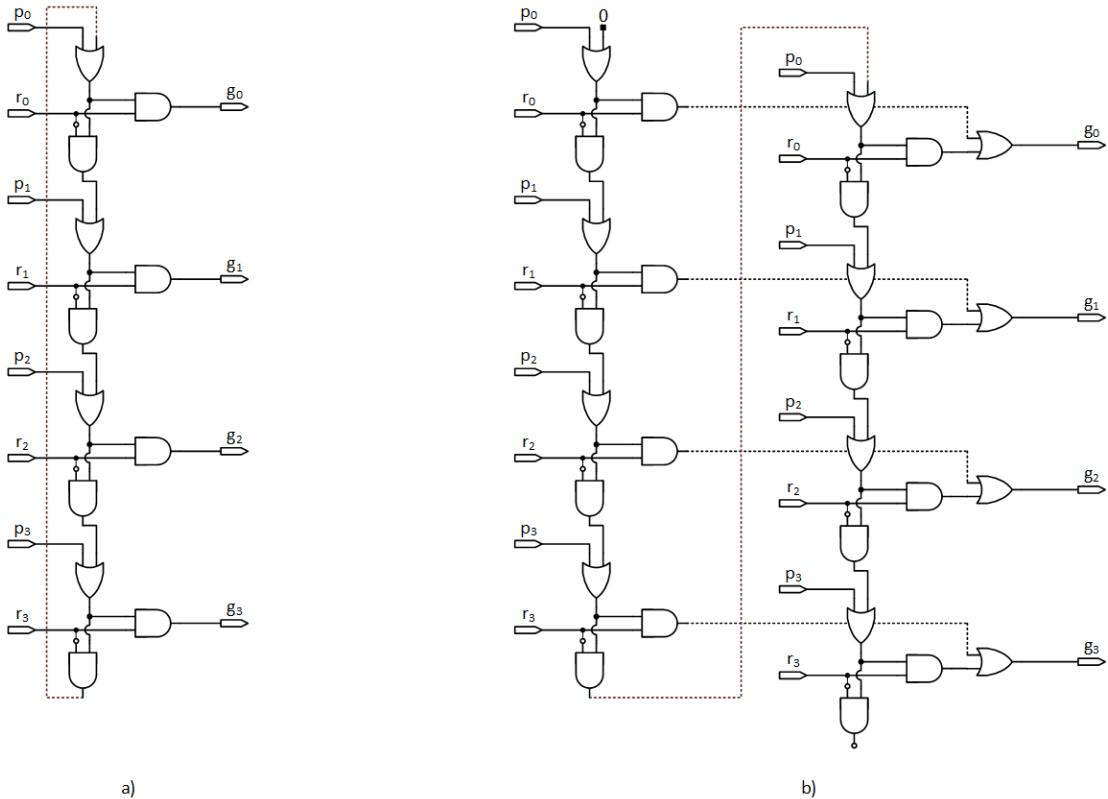
cada unidad scc se encarga de el arbitraje y retención de enlaces entre un puerto de entrada y uno de salida. Internamente se encuentra formado por un par de árbitros en cascada (ARB1 Y ARB2) y un generador de prioridad round robin (RR), una Unidad de retención (UR), una unidad de control de corssbar (UCC) formada por dos maquinas de estado finito: MEFCC (Maquina de Estado Finito de Control de crossbar) y MEFMC (Maquina de Estado Finito para

manejo de créditos).

El esquema convencional de un arbitro de prioridad variable se muestra en la figura 5.12 a). Este modelo de arbitro no es funcional dentro de dispositivos reconfigurables debido a la meta estabilidad en la señal de acarreo que forma una relación cíclica a lo largo del arbitro. Para superar la restricción del ciclo combinacional, se implemento el diseño de arbitro en cascada de la figura 5.12 b). En este, se duplica el hardware para iniciar una cadena de acarreo que supone un estado bajo a su entrada, la salida de acarreo del primer arbitro es utilizada por el segundo arbitro del sistema en cascada. si el valor de acarreo de salida para el primer arbitro del arreglo es 0, ambos árbitros coincidirán en la resolución del proceso de arbitraje, en caso contrario, el resultado del segundo arbitro producirá la salida correcta del proceso de arbitraje. El vector de prioridad  $P_x$  es generado por medio del algoritmo round-robin.

La salida del arbitro es un vector utilizando codificación 'binario natural' para representar el puerto de entrada al cual se le ha otorgado la concesión del uso del puerto de salida. El uso de binario natural reduce el numero de bits necesarios para representar el resultado del proceso de arbitraje, ademas permite utilizar el vector directamente para la configuración de los multiplexores que conforman un segmento del crossbar.

La unidad UCC en conjunto con el modulo UR administra el vector de configuración que se inyectara al crossbar. El modulo UR permite seleccionar entre el registro del vector generado por el arbitro, retener la configuración actual o enviar la señal cross\_cfg a estado bajo. La decisión anterior recae en la unidad MEFCC, la cual decide si un nuevo proceso de arbitraje es necesario tomando en cuenta los siguientes parámetros: solicitudes existentes desde puertos de entrada (pet\_pen) y disponibilidad de créditos (crd\_disp). La MEFCC inicia un proceso de arbitraje siempre que el puerto de salida se encuentre en reposo, se cuente con créditos dis-



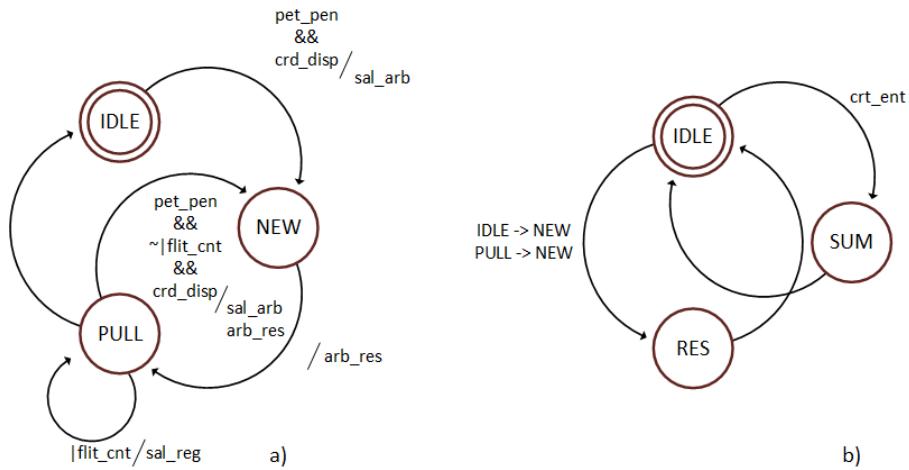
**Figura 5.12:** a) Implementación tradicional de arbitro con sistema de prioridad dinámica. b) Implementación de arbitro para dispositivos reconfigurables. La eliminación del ciclo combinacional en la cadena de acarreo del arbitro es necesario para su porte a dispositivos FPGA.

ponibles y exista una petición activa. Una vez obtenido el resultado de la unidad ARB, este permanecerá almacenado en el registro de la unidad UR durante el tiempo necesario para la transferencia de los flits de un paquete desde el buffer hasta el puerto de salida. En caso de no existir peticiones activas, la unidad MEFCC mantiene el vector de configuración para el crossbar en estado bajo.

El manejo de créditos se lleva a cabo mediante una máquina de estados en la unidad MEFMC, la cual responde a dos eventos: al momento del inicio de la transferencia de un paquete en dirección al router vecino, la unidad MEFMC disminuye el registro de créditos disponibles en una unidad, ya que el buffer del siguiente encaminador se encuentra almacenando el paquete enviado. Al recibir la liberación de un crédito por parte del siguiente encaminador, la unidad MEFMC incrementa el registro de créditos. Vale la pena mencionar que la liberación de créditos no se lleva a cabo en la unidad MEFMC, es tarea de la unidad MEFS la liberación de un crédito al router anterior como respuesta al inicio de un proceso de arbitraje en la unidad SCC. La figura 5.13 presenta el modelado de ambas máquinas de estados de las unidades MEFCC y MEFMC.

#### PUERTO DE SALIDA

El puerto de salida consta de un registro para almacenar de manera temporal un flit en tránsito al siguiente encaminador y un módulo UMPS. Este último módulo opera de manera similar al módulo UMPE del puerto de entrada, su tarea es llevar registro del par diferencial el puerto de entrada y llevar a cabo el cambio de estado cuando se inicie la transmisión de un paquete.



**Figura 5.13:** a) Maquina de estado finito de la unidad MEFCC. b) Maquina de estado finito para la unidad MEFMC.

#### ARQUITECTURA DE INTERFAZ DE RED

Los paquetes de información en tránsito a través de la red utilizan el formato descrito en la sección 5.1.1. El formato fue diseñado en orden de simplificar el hardware necesario para la decodificación y transporte de información tomando en cuenta las restricciones físicas que impone el ancho de canal entre routers. La información contenida en el paquete debe ser preprocesada antes de ser injectada a un elemento funcional de un nodo, y de igual forma el resultado entregado por una unidad funcional debe de ser encapsulado junto con los campos de control para formar un paquete.

La tarea de empaquetado/desempaquetado de información se lleva a cabo en las interfaces de red de cada nodo. La figura 5.14 muestra el diagrama a bloques general de una interfaz de red. El encaminador del nodo interactúa con la interfaz de red como si se tratase de un encaminador vecino. La transferencia de información entre la interfaz de red y el encaminador se basa en el intercambio de paquetes y la entrega/recepción de créditos para mantener el

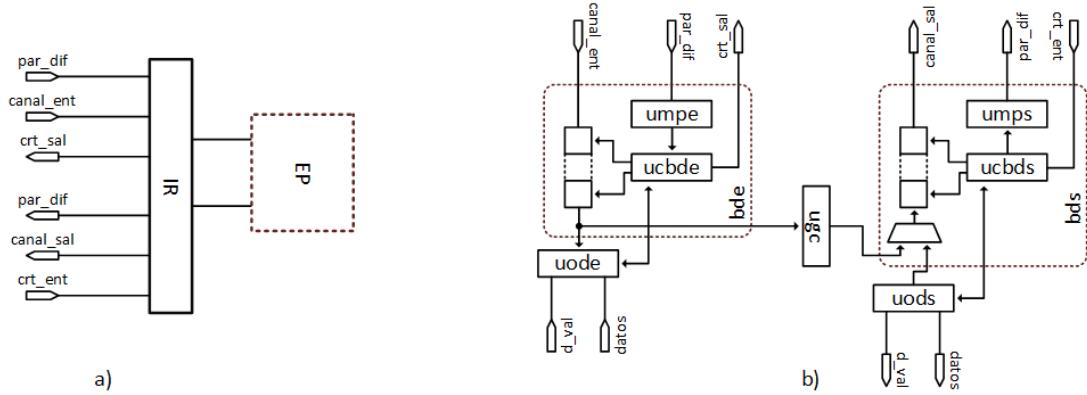


Figura 5.14: Interfaz de red. a) la interfaz de red se conecta al encaminador como un vecino más de la red. b) Las unidades UCBDE y UCBDS son versiones modificadas de la unidad UCPE del puerto de entrada.

control sobre el flujo de información.

El elemento de procesamiento define las señales de control y datos involucrados en la interacción con la interfaz de red. El área de intercambio entre interfaz y elemento de procesamiento representa un desafío para la versatilidad de la red, ya que diferentes elementos de procesamiento requerirán interfaces de red a medida. Para facilitar la tarea de diseño de interfaces se ha dividido internamente el módulo en un bloque de entrada (BDE) y un bloque de salida (BDS) como se muestra en la figura 5.14 b).

La áreas de interacción con el encaminador no requieren un análisis profundo ya que utilizan la misma filosofía de envío/recepción de paquetes entre routers, y de hecho, reutilizan los mecanismos de manejo de paquetes utilizados por los módulos UMPE, UMPs, MEFE y MEFS de los puertos de entrada y unidades SCC del router.

EL bloque de entrada de la interfaz provee los datos de trabajo para el elemento de procesamiento, además de al menos una señal de control para indicar que los datos a la entrada del elemento de procesamiento son válidos y puede iniciar la captura del dato. Es trabajo del

bloque de entrada es el reorganizar los flits de datos de un paquete de manera que se presenten en el formato nativo de trabajo del elemento de procesamiento, además de cumplir con cualquier protocolo de transferencia de información impuesto por este último.

Por su parte, el bloque de salida espera la finalización del trabajo llevado a cabo por el elemento de procesamiento. Una señal de control o una transacción de un protocolo de comunicación (*handshake*) indica al bloque de salida que los datos son válidos y puede iniciar la captura de resultados. Los datos transmitidos por el elemento de procesamiento pueden encontrarse en tamaños de palabra diferentes a los de un flit, por lo que es tarea del bloque de salida reorganizar la información de manera que puedan empaquetarse previo a su liberación a la red.

El flit de cabecera no se envía al elemento de procesamiento, todo el trabajo de re-codificación de la información de la cabecera se lleva a cabo dentro de la interfaz de red en la Unidad de Generación de Cabecera (UGC). De manera ordinaria, el tratamiento que se le da a un flit de cabecera consiste en asignar el estado alto a su campo *testigo post-proceso* y el intercambio de los campos *destino* y *puerta*. Los cambios al flit de cabecera ocasionan que el paquete viajando con los resultados no compita nuevamente por el uso de un elemento de procesamiento en ningún otro nodo de la red, además, su nueva dirección destino es una de las puertas de salida de la red.

Las unidades UCBDE y UCBDS son versiones modificadas de la unidad UCPE del puerto de entrada. Estas unidad agrega una señal gatillo para indicar que los datos en el buffer son válidos o para indicar que los datos provenientes del elemento de procesamiento se encuentran listos para su transmisión a la red. Las unidades UODE (Unidad de ordenamiento de datos de entrada) y UODS (Unidad de ordenamiento de datos de salida) son unidades a medida para

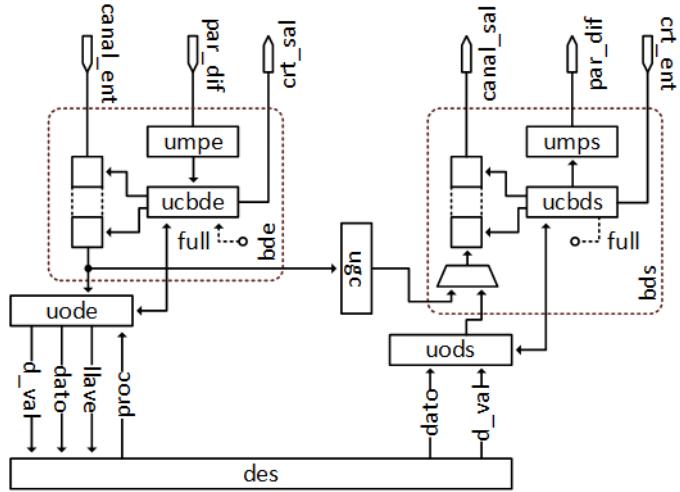
cada elemento de procesamiento que se conecte a un nodo de red.

#### CASO DE ESTUDIO: ENCRIPCIÓN DES

EN esta sección se presenta el desarrollo de una interfaz de red para un bloque encriptador basado en el algoritmo DES<sup>54</sup>. El encriptador trabaja con dos bloques de 64 bits de longitud, uno de ellos contiene el conjunto de datos a encriptar, mientras el segundo bloque acarrea la llave de cifrado. El estándar de encriptación DES especifica el uso de claves de longitud de 56 bits, sin embargo, es una práctica común el agregar 1 bit de paridad por cada 7 bits que conforma la clave. El bit de paridad sirve para verificar la integridad de la clave de encriptación. El elemento de encriptación recibe la frase original y la clave de cifrado en forma de una carga paralela como se muestra en la figura 5.15. La señal *d\_val* se utiliza para indicar al encriptador que los datos en sus puertos de entrada son válidos y puede iniciar con su tarea.

Como resultado, el módulo DES regresa una palabra de 64 bits conteniendo la información cifrada. Se utiliza una segunda señal *d\_val* para indicar a la interfaz de red que los datos en el puerto de salida del encriptador son válidos para su captura. Para complementar el control de flujo de información entre la interfaz de red y el encriptador se agrega la señal de control *proc*, que funge como indicador de un retorno de crédito desde el núcleo DES. Cada vez que el núcleo de encriptación libera el resultado de su procesamiento al modulo UODE, un nuevo paquete desde el bloque BDE pasa al modulo UODE, la señal *proc* deja saber al bloque BDE que puede recibir un nuevo paquete desde la red ya que el dato almacenado en la unidad UODE ha sido consumido.

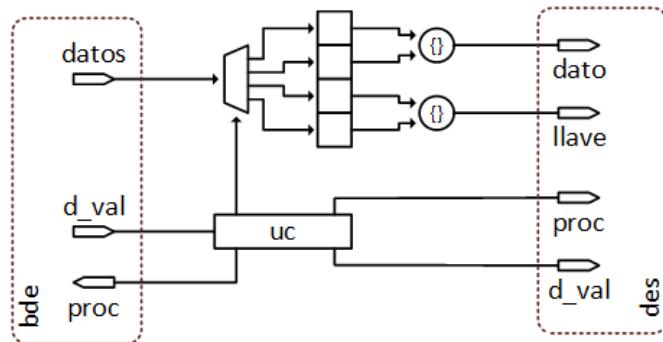
Más a detalle, la unidad UODE está compuesta por un banco de registros y una unidad de control. El encriptador requiere que se alimente con los datos de trabajo de forma paralela,



**Figura 5.15:** Las líneas de comunicación entre interfaz y elemento de procesamiento están definidas para acoplarse al formato utilizado por el elemento funcional. En este ejemplo se favorece una interfaz con de carga paralela de datos.

por lo tanto, cada registro del banco tiene un puerto de salida dedicado. El bus de datos de entrada del encriptador consta de dos palabras de 64 bits, por lo cual, suponiendo flits de 32 bits, se deduce que cada paquete de la red debe de estar formado por un flit de cabecera y 4 flits de datos.

El trabajo de la unidad **UODE**, mostrado en la figura 5.16, inicia con el disparo de la señal **d\_val**, esta dispara el ciclo de captura de flits desde el modulo **BDE**. El tamaño del paquete (5 flits) se especificó de manera previa a la síntesis del acelerador, por lo que la unidad de control (**UC**) está programada de antemano para capturar el número de flits del paquete, el flit de cabecera no es procesado por el núcleo **DES** por lo que el modulo **BDE** no lo almacena. Con la captura de todos los flits de datos de un paquete, la unidad de control propaga la señal **d\_val** para indicar al encriptador que un conjunto de datos está disponible para ser consumidos en una nueva ronda de cifrado.



**Figura 5.16:** Las líneas de comunicación entre interfaz y elemento de procesamiento están definidas para acoplarse al formato utilizado por el elemento funcional. En este ejemplo se favorece una interfaz con carga de datos paralela.

La aserción de la señal *proc* por parte del núcleo indica que los datos disponibles en los puertos de salida a han sido consumidos, la unidad de control retransmite la señal *proc* al modulo *BDE* para indicar que un nuevo paquete de flits de datos puede ser enviado al modulo *UODE*. El envío de un nuevo paquete puede ser inhibido por la unidad *BDS*, en este escenario el bloque de salida tiene paquetes en cola esperando a ser transmitidos a la red, por lo que su buffer no cuenta con capacidad para la recepción de un nuevo resultado por parte del núcleo. La aserción de la señal *full* inhibe la transmisión de un nuevo paquete a la unidad *UODE*.

El bloque de salida de la interfaz de red está formado por un par de registros para contener el resultado del proceso de cifrado. La unidad de control espera la confirmación por parte del núcleo del final de la ronda actual, una vez recibida la señal *d\_val* la unidad *UODS* captura el dato y retransmite la señal *d\_val* a la unidad *BDS*. La unidad *DBS* indica el inicio de la transmisión de datos por medio de la señal *d\_acp*. La espera a la confirmación por parte de la unidad *BDS* es necesaria ya que se requiere el ordenamiento del paquete en el buffer de transmisión, la unidad *BDS* inserta la cabecera proporcionada por la unidad *UGC* y posteriormente solicita el envío del dato recién capturado a la unidad *UODS*. El diagrama de la Unidad de ordenamiento

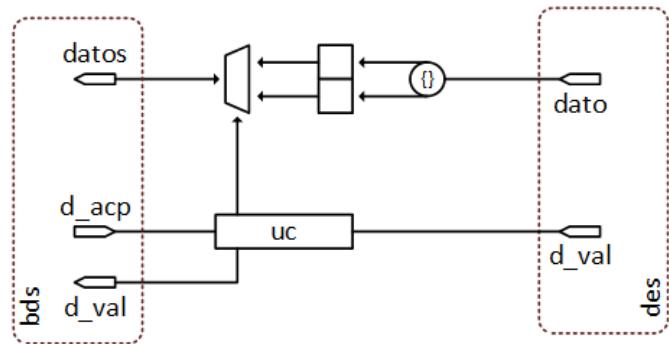


Figura 5.17: Unidad UODS.

de salida se presenta en la figura 5.17. La unidad UODS proporciona dos flit, haciendo un total de 3 flits contando con la cabecera del paquete, los flits restantes son generados con una carga útil vacía pero son necesarios para mantener el tamaño fijo de paquete de 5 flits.

# 6

## Acelerador multi-núcleo

El concepto de sistemas basados en múltiples núcleos de procesamiento con redes en chip como medio de interconexión se denomina MPNOC (*multiple processor network on-chip*). El uso de sistemas MPNOC como aceleradores de tareas en hardware enfrenta al desarrollador con un conjunto de características de diseño que pueden penalizar el rendimiento general del sistema, algunas de ellas se mencionan a continuación:

- En redes en-chip tradicionales, los puntos de inyección de paquetes de información se encuentran distribuidos a lo largo de todos los nodos que pertenecen al borde del sistema. La presencia de múltiples puntos de entrada no resulta en un mayor rendimiento debido a que una vez que la red alcanza su punto de saturación ya no le es posible el

procesar y despachar paquetes a un mayor ritmo.

- El uso de un esquema de direccionamiento estricto para la transferencia de información a través de la red puede causar el incremento de latencia en el transporte de paquetes. El uso de direcciones específicas para el envío de datos requiere la implementación de esquemas complejos para el cálculo de direcciones de red, además de propiciar puntos de saturación si dicho esquema no distribuye de manera equitativa el tráfico entre todos los nodos.
- Una selección pobre de arquitectura de procesamiento para las unidades funcionales del acelerador puede impactar de manera negativa el rendimiento del sistema. Por ejemplo, la aceleración de rutinas con múltiples condicionales verá castigado el número de nodos que la red será capaz de albergar debido al alto consumo de recursos reconfigurables. Esta penalización en área está asociada a las múltiples unidades funcionales requeridas para ejecutar cada caso de las condicionales de la aplicación.

El presente capítulo describe propuestas para mitigar la degradación de rendimiento debida a los puntos mencionados anteriormente. El resto del contenido de esta sección se desarrolla en el siguiente orden: El apartado 6.1 describe las principales diferencias entre aceleradores basados en redes en chip tradicionales con respecto al presentado en este trabajo, además, se describe la arquitectura del acelerador desde el punto de vista de sistema. Los módulos particulares para el acelerador propuesto se describen en la sección 6.2. La sección 6.3 muestra la infraestructura desarrollada para la verificación del acelerador, así como los mecanismos básicos para someter al sistema a pruebas de desempeño relacionadas con latencias de transmisión y flujo de paquetes. En la sección 6.4 cierra el capítulo con una descripción de la metodolo-

gía utilizada para llevar a cabo pruebas de rendimiento sobre la infraestructura descrita en la sección anterior.

## CONCEPTO

De manera tradicional los aceleradores basados en la metodología de diseño MPNOC, implementan esquemas estáticos para la asignación de paquetes de datos a elementos de procesamiento, es decir, cada paquete de datos contiene la dirección de la unidad funcional en la cual se llevará a cabo el procesamiento de su información. Bajo el esquema anterior ningún otro elemento en el camino podrá llevar a cabo trabajo sobre los datos que porta el paquete en tránsito. El uso de esta metodología de distribución de datos facilita el análisis del comportamiento de la red bajo diferentes cargas de trabajo, inclusive cuando se utilizan algoritmos de planificación de ruta parcialmente adaptativos. Aunado a la facilidad de análisis, el uso de estos tipos de esquema habilita la inclusión de elementos heterogéneos de procesamiento, ya que se puede asignar paquetes de datos a unidades funcionales específicas.

Los aceleradores con núcleos heterogéneos presentan un rendimiento superior debido a su alto nivel de especialización. Los esquemas estáticos de asignación de trabajo funcionan de manera excelsa en estos sistemas debido al patrón determinístico de comunicación que se presenta entre bloques. El alto rendimiento erogado por aceleradores heterogéneos trae consigo una falta de flexibilidad en su estructura de interconexión, haciendo prácticamente imposible su reutilización para otras tareas.

El uso de aceleradores a base de núcleos homogéneos de procesamiento busca la aceleración de tareas afines al set de instrucciones presente en sus unidades funcionales, ejecutando una tarea idéntica en todos sus nodos de red. El uso de esquemas estáticos para la asignación de

trabajo no resulta tan atractivo en redes homogéneas donde todos los núcleos ejecutan una misma operación, y sus patrones de tráfico no se encuentran definidos de manera estricta.

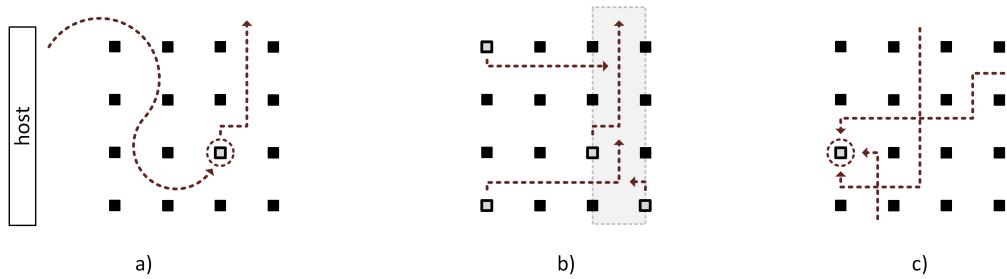
La figura 6.1 muestra algunos de los escenarios donde un acelerador con núcleos homogéneos ve comprometido su rendimiento: para promover un balance de carga a través de la red es necesario el uso de soluciones complejas para la asignación de paquetes a sus núcleos de procesamiento. El uso de algoritmos de menor grado de complejidad para la asignación de cargas de trabajo puede derivar un saturación de nodos o desbalance en el uso de canales de comunicación. Este escenario se representa en la figura 6.1 a).

El uso de un conjunto reducido de direcciones estáticas de salida genera congestiones en los canales de transmisión utilizados para el envío de resultados fuera del acelerador, este problema se ve agravado cuando los paquetes egresando de la red deben de competir con un gran número de paquetes entrantes como se muestra en la figura 6.1 b). El tráfico dentro del acelerador se ve comprometido cuando un nodo recibe una carga de trabajo desbalanceada con respecto a sus pares. Una distribución de trabajo desbalanceada no solo afecta de manera local a un nodo de la red, sino al tráfico en curso en su vecindario como se muestra en la figura 6.1 c).

La figura 6.2 muestra la estructura propuesta para el acelerador conformado por múltiples núcleos de procesamiento. A grandes rasgos, este se encuentra compuesto por una red en-chip con topología malla, los nodos que pueblan la red pueden clasificarse dentro de 3 categorías: nodos frontera, nodos terminal y nodos centrales.

Cada clase de nodo se distingue por su capacidad de procesamiento y de direccionamiento de paquetes, a continuación se presenta una descripción de cada uno de ellos:

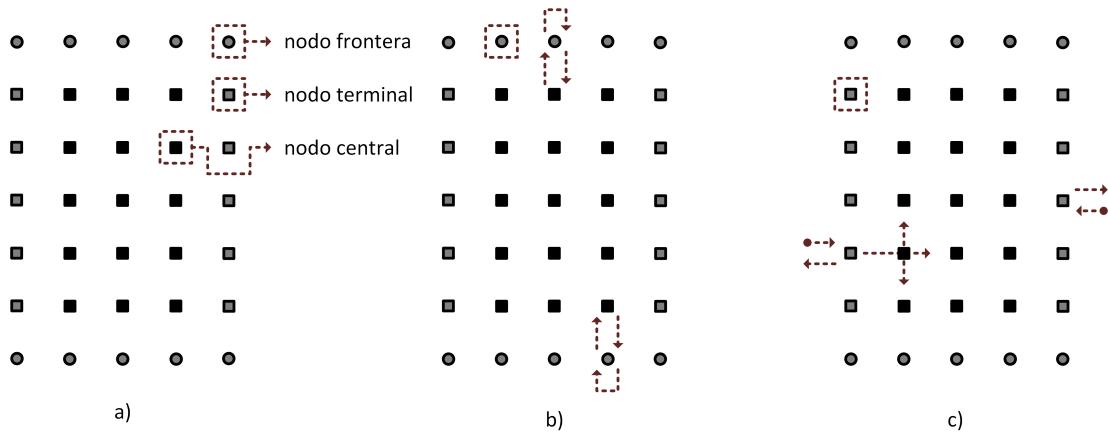
- *nodo terminal*: punto base para la inserción y captura de paquetes procesados por la



**Figura 6.1:** Características en aceleradores convencionales. a) El procesamiento de información se lleva a cabo en un nodo específico de la red. b) Cada paquete portando el resultado de un proceso se dirige a una puerta de salida preestablecida, compitiendo con los recursos de transporte con los paquetes de recién ingreso al sistema. c) El uso de políticas desbalanceadas para la asignación de cargas de trabajo de un nodo puede saturar rápidamente varios canales de comunicación en el acelerador.

red. Todo paquete que desee abandonar el acelerador, una vez que ha sido procesado, debe de dirigirse a uno de estos nodos para su extracción. Las terminales del acelerador deben estar colocadas en la periferia de la red, sin importar en cuál de sus frentes. Los nodos terminal cuentan con una unidad funcional, por lo que participan de manera activa en el procesamiento de paquetes. A nivel de micro-arquitectura los nodos terminal son equivalentes a un nodo central, sin embargo la implementación de su algoritmo de planificación de ruta difiere con estos últimos.

- *nodo central:* Elementos estándar de procesamiento. El interior de la red está formada por nodos centrales, los cuales cuentan con un elemento de procesamiento y la capacidad de encaminar paquetes a cualquiera de sus vecinos. El algoritmo de planificación de ruta de estos nodos no presenta alteración alguna, por lo que no tienen la capacidad de filtrar ningún tipo de paquete en tránsito. Todos los paquetes de nuevo ingreso son entregados a los nodos centrales a manos de los nodos frontera o terminal como se muestra en la figura 6.2 c).



**Figura 6.2:** Arquitectura propuesta. a) el acelerador está compuesto por una granja de núcleos comunicados por medio de una red en-chip, se hace uso de 3 tipos de nodo: Nodos terminales para la inyección y/o recepción de paquetes de la red, nodos frontera para re circular paquetes que no han sido procesados por una unidad funcional, y nodos centrales, los cuales procesan y envían los resultados fuera del acelerador. b) los nodos frontera solo tienen acceso a la red mediante un canal de comunicación. c) Los nodos terminal comparten micro-arquitectura con los nodos centrales, sin embargo, solo un nodo terminal es capaz de recibir y enviar paquetes fuera de la red.

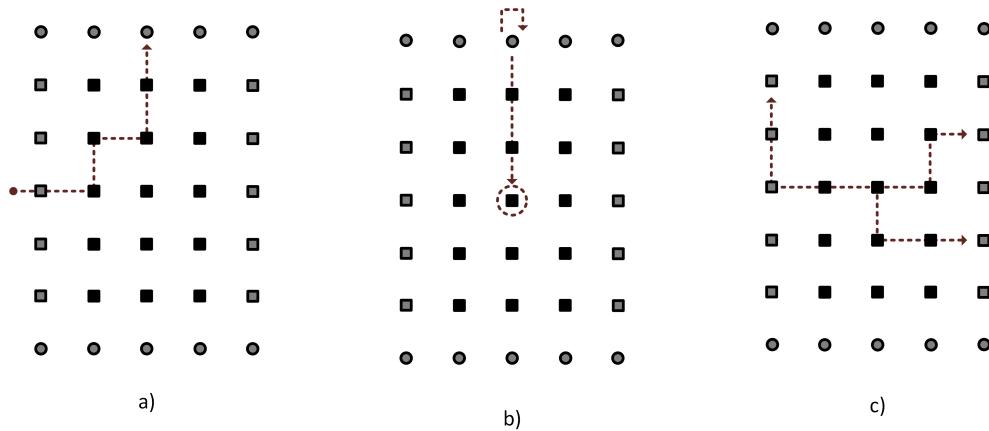
- *nodo frontera:* Los nodos frontera no contienen un elemento de procesamiento, por lo que no se consideran parte del área funcional de la red. Además de su falta de unidad funcional, su conectividad con la red se limita a un punto de captura e inyección de datos. Su tarea es la reinserción de paquetes que han alcanzado alguno de los límites de la red sin haber visitado una unidad funcional durante su trayecto, este comportamiento se muestra en la figura 6.2 b).

El procesamiento de un paquete en la arquitectura propuesta sigue el flujo de ejecución mostrado en la figura 6.3 y que se describe a continuación: un paquete ingresa a través de un nodo terminal en dirección a un nodo destino. La cabecera de cada paquete contiene una dirección destino y una dirección de puerta de salida, los campos que conforman la cabecera de un paquete puede encontrarse en el capítulo 5 dentro de la sección 5.1.1. El paquete en tránsito

busca llegar a la dirección destino asignada en su flit de cabecera, todas las direcciones destino corresponden a nodos frontera. Durante su avance a través de la red, el paquete busca un nodo cuya unidad funcional se encuentre disponible para el procesamiento de la información que acarrea consigo. En caso de llegar a un nodo atendiendo una petición previa, el paquete seguirá adelante como se muestra en la figura 6.3 a). Al alcanzar un nodo frontera, el paquete ingresa para la asignación de un nuevo nodo destino y su posterior reenvío a la red en búsqueda de una unidad de procesamiento disponible. Al encontrar un nodo disponible para el procesamiento de información, el paquete lleva a cabo una solicitud de ingreso. Una vez finalizado el procesamiento de información dentro de la unidad funcional, la cabecera del paquete cambia su dirección destino por la dirección del nodo terminal donde está programada su salida de la red.

El uso de una estrategia *el primero en llegar, el primero en ser atendido* para la asignación de recursos de procesamiento libera la presión sobre nodos específicos de la red, permitiendo un continuo flujo de paquetes a través del sistema. Para implementar la estrategia descrita anteriormente es necesario el uso de un nuevo esquema de planificación de ruta a través de la red. Como se mencionó en el capítulo 2, dentro de la sección 2.3, los algoritmos de planificación de ruta tradicionales requieren de un destino fijo para el manejo de paquetes, una vez alcanzado dicho destino el paquete lleva a cabo una solicitud de ingreso al elemento de procesamiento del nodo actual. El acelerador basado en múltiples núcleos de este trabajo agrega como primera opción de ruta el elemento de procesamiento de cualquier nodo visitado, en caso de no estar disponible, el algoritmo se comportara de manera tradicional y solicitará el uso del puerto de salida que lo aproxime a su destino.

El algoritmo de planificación de ruta requiere de un mecanismo externo para mantener la



**Figura 6.3:** Tránsito de un paquete a través del acelerador. a) en primera instancia un paquete ingresa a través de un nodo terminal. El paquete buscará la primera unidad funcional disponible para el procesamiento de información. b) en caso de alcanzar un nodo frontera sin haber procesado su información, el paquete cambia su dirección y continúa la búsqueda de una unidad funcional disponible. c) Una vez terminado el procesamiento de información de un paquete, este reingresa a la red en búsqueda de un nodo terminal para su egreso del sistema.

circulación de paquetes de manera continua a través de todos los nodos de la red, este mecanismo externo se implementa por medio de los nodos frontera del acelerador. Los nodos frontera no pueden llevar a cabo ninguna tarea de procesamiento y su micro arquitectura difiere en gran medida con la de los demás módulos de la red. Internamente los nodos frontera no implementan mecanismo de arbitraje entre puerto debido a que todo paquete ingresando solo tiene la posibilidad de egresar por un solo canal. El mecanismo de control de flujo si se encuentra implementado en estos nodos para mantener el protocolo de intercambio de información a nivel de capa física con los demás bloques del acelerador.

La estrategia de distribución de carga de trabajo, en conjunto con el medio de reinserción de paquetes no procesados a la red, forman el mecanismo híbrido en el cual se basa el acelerador presentado en este trabajo.

## MÓDULOS PROPIOS DEL ACELERADOR MULTI-NÚCLEO

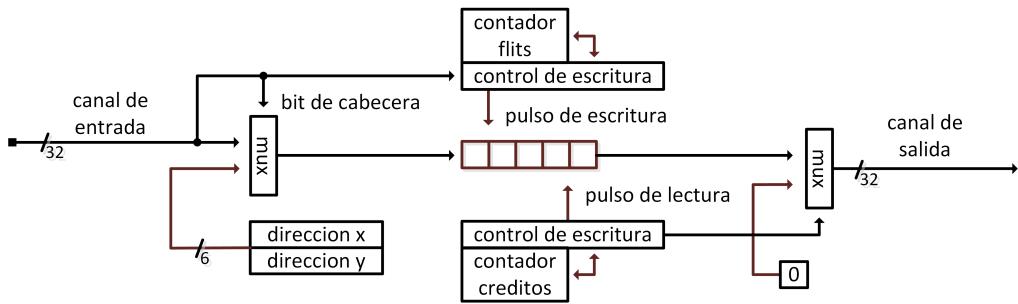
El sistema de navegación de paquetes a través de la red está dividido en dos partes: en primer lugar se encuentra el algoritmo modificado de encaminamiento, el cual se encuentra distribuido a través de todos los nodos del sistema. Cualquier algoritmo de encaminamiento que no requiera de canales virtuales y sea viable para trabajar sobre redes de topología malla puede adecuarse para trabajar en la arquitectura propuesta.

En segundo lugar, se encuentra el mecanismo de inserción de paquetes a la red, el cual es implementado por medio de nodos frontera. La decisión del uso de módulos dedicados para llevar a cabo esta función se tomó en aras de mantener un diseño sencillo de los nodos de procesamiento de la red. La implementación de esta restricción extra como una característica más del algoritmo de encaminamiento aumentaría el grado de complejidad de todo el sistema.

En esta sección se presenta la arquitectura de los nodos frontera así como una descripción de su participación en el flujo de información a través del acelerador. En segundo lugar se presenta la metodología desarrollada para la modificación de algoritmos de encaminamiento pre existentes. La figura 6.5 a) muestra el esquema de etiquetado de nodos de la red, este se utilizará en los ejemplos de esta sección.

### NODO FRONTERA

La figura 6.4 muestra la arquitectura interna de nodo frontera de la red. La tarea a llevar por estos módulos es el proporcionar un punto inicial de destino para todos los paquetes ingresando a la red. Los paquetes buscan llegar a su punto destino, sin embargo, harán la solicitud de ingreso a una unidad funcional disponible en cualquiera de los nodos que se visite en su trayecto. Un paquete procesado cambia su dirección destino por una dirección de



**Figura 6.4:** Arquitectura de nodo frontera. El camino de datos de este módulo solo lleva a cabo la sustitución de la dirección destino de los paquetes en tránsito.

salida de la red.

Todos los paquetes que no encontraron en su camino un procesador disponible eventualmente alcanzan un nodo frontera. El nodo frontera al recibir el paquete modificará la dirección destino por la dirección de un nodo frontera opuesto a él y posteriormente reenviará el paquete para que continúe su búsqueda de un procesador en una dirección distinta.

Internamente el nodo frontera está formado por una estructura *FIFO* de almacenamiento temporal y dos máquinas de estado independientes que se encargan del ingreso y egreso de paquetes a la red. El control de flujo de datos dentro de un nodo frontera se lleva a cabo de mediante un contador de créditos, de esta manera se homologa el flujo de datos con todos los otros nodos de la red.

Las máquinas de estado siguen la misma lógica de captura y liberación que se presenta en los módulos MEFE y MEFS de la arquitectura del encaminador, esta se puede consultar en el capítulo 5 dentro de la sección 5.2.2. Los nodos frontera requieren de una dirección estática que se asignará a todos los paquetes rebotando a la red a través de ellos, esta dirección se determina de manera previa a la síntesis del sistema y debe de cumplir con las restricciones impuestas por el algoritmo de enrutamiento, de lo contrario existe el riesgo de crear *deadlocks* o *livelocks* en

la red.

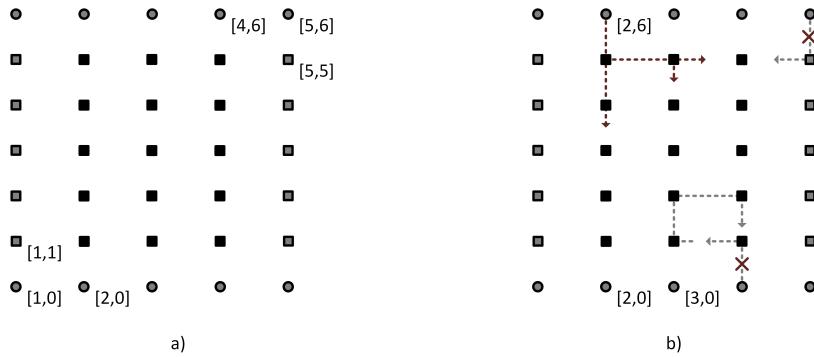
La sobre escritura de la dirección destino de un paquete se lleva a cabo durante el ingreso del flit de cabecera al *FIFO* del nodo, evitando la necesidad de lógica extra durante la salida del paquete. La figura 6.5 b) muestra un acelerador utilizando el algoritmo de encaminamiento *west first minimal*.

#### MODULO PLANIFICADOR DE RUTA

La capacidad de los paquetes para ingresar a cualquier núcleo de procesamiento de la red radica en el algoritmo de encaminamiento de información. La inserción del mecanismo de solicitud de ingreso a elementos de procesamiento en cada salto de un paquete es implementado mediante la alteración de un algoritmo pre existente, sin embargo, para mantener el continuo flujo de información se requiere la integración de nodos frontera.

A grandes rasgos, el proceso de modificación de un algoritmo de encaminamiento pre existente consiste en la inserción de un nivel jerárquico superior, en donde la solicitud de entrada a un elemento de procesamiento tenga un nivel mayor de prioridad respecto al orden propuesto por el algoritmo original. De igual forma, se debe de incluir un supresor de la modificación en el algoritmo en caso de que el paquete en circulación ostente una bandera indicando que ya a sido procesado en un nodo anterior.

El algoritmo de encaminamiento original tiene un fuerte impacto en el cálculo de las direcciones válidas a las que un nodo frontera puede redirigir su tráfico. La figura 6.5 b) muestra una red hipotética utilizando el algoritmo de encaminamiento *west first minimal*, el cual se discutió en el capítulo 3 sección 3.3.2, este algoritmo prohíbe giros en la dirección *x*-, por lo que paquetes saliendo de los nodos frontera con dirección a nodos terminales o centrales que



**Figura 6.5:** a) Los nodos se etiquetan de acuerdo a su posición en la red. Vale la pena notar que todos los nodos centrales formarán una sub-red tipo malla al centro del acelerador. Las etiquetas para cada nodo sirven para reconocer que trabajo desempeña en el acelerador, además de facilitar la selección de instancia correcta durante el proceso de síntesis del sistema. b) Los nodos frontera deben de tomar en cuenta las restricciones impuestas por el algoritmo de encaminamiento para la selección de direcciones válidas en sus paquetes de salida.

se encuentren a la izquierda de los mismos producen el riesgo de crear deadlocks como se muestra en la misma figura. De manera general el proceso de modificación de un algoritmo de encaminamiento requiere analizar las rutas y direcciones válidas para los nodos frontera de la red.

La modificación de un algoritmo de encaminamiento pre existente requiere realizar un cambio en la estructura jerárquica de selección de puerto de salida. Dentro de la estructura de selección de puerto, se inserta en el nivel superior la solicitud de ingreso a el elemento de procesamiento si y sólo si el paquete contiene la información original en sus flits de datos. En caso de que el campo de *testigo de post - proceso* presente un estado alto, esta solicitud se omite y se continúa con el árbol de selección de siguiente salto para el paquete. El código 2 muestra la modificación pertinente para el algoritmo *west first minimal* que se mostró en el capítulo 3 dentro de la sección 3.3.2.

El código 2 es utilizado de manera exclusiva en nodos centrales de la red, ya que no cuenta

Data: dirección de nodo actual (*localX*, *localY*)  
 Data: dirección de nodo destino (*destX*, *destY*)  
 Data: campo de testigo de procesamiento  
 Result: canales de salida válidos (*canales*)  
 begin  
     offsetX = destX - localX;  
     offsetY = destY - localY;  
     if *campo de testigo de procesamiento* then  
         | canales := elemento de procesamiento;  
     end  
     if *offsetX < o* then  
         | canales := X-;  
     end  
     if *offsetX > o and offsetY < o* then  
         | canales := (X+, Y-);  
     end  
     if *offsetX > o and offsetY > o* then  
         | canales := (X+, Y+);  
     end  
     if *offsetX > o and offsetY = o* then  
         | canales := X+;  
     end  
     if *offsetX = o and offsetY < o* then  
         | canales := Y-;  
     end  
     if *offsetX = o and offsetY > o* then  
         | canales := Y+;  
     end  
 end

Algoritmo 2: *West-First minimal* para nodos centrales del acelerador.

con un mecanismo para manejar el caso en el cual la dirección local del nodo es la dirección destino del paquete. Este caso se omite debido a que el diseño de la red prohíbe se asigne un nodo central como destino de un paquete. La versión del algoritmo para nodos terminales requiere el manejo del escenario anterior, donde el nodo terminal destino retira de la red el paquete recibido.

La implementación en hardware de este algoritmo no se lleva de manera centralizada, es decir, no existe un módulo centralizado de cálculo de ruta para todos los paquetes llegando al router. En su lugar, se distribuye el cálculo de ruta entre los diferentes puertos de entrada, donde cada uno de ellos integra un segmento del algoritmo que atiende solo los casos que se pueden presentar para los paquetes ingresando desde esa dirección. La figura 6.6 en sus apartados a) y b), muestra el diagrama de los segmentos del algoritmo implementado en los puertos de entrada  $x$ - de los nodos terminal (5,3) y (1,3). se puede apreciar la diferencia del hardware generado para cada instancia. El segmento de algoritmo mostrado en el código 3 representa el segmento para el envío de paquetes fuera de la red.

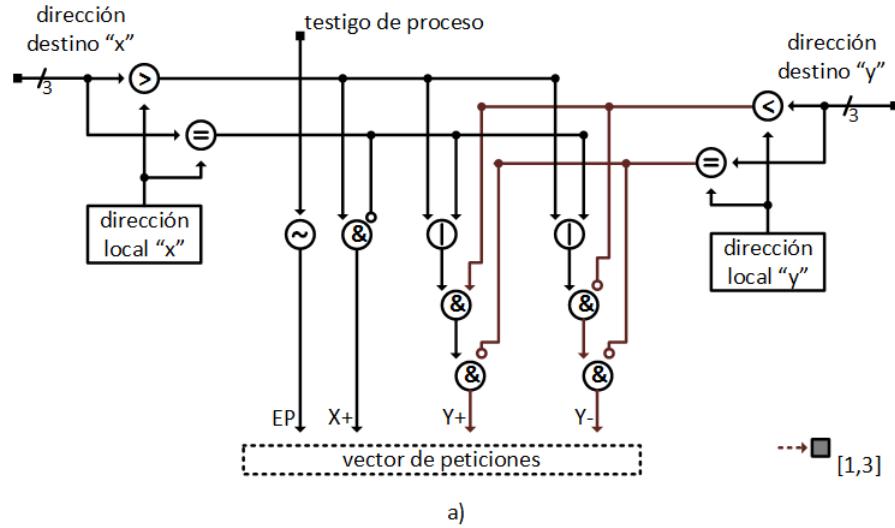
```

begin
  | if offsetX == o and offsetY == o then
  |   | canales := X;
  | end
end

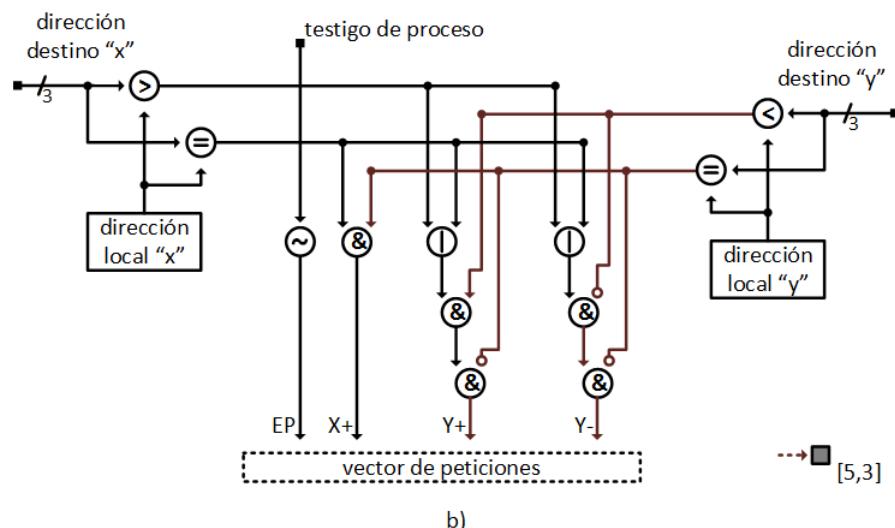
```

Algoritmo 3: Condicional exclusiva para nodos terminal del acelerador.

Es necesario recordar que el algoritmo *west first minimal* es parcialmente adaptativo, por lo que ofrece múltiples opciones de puertos de salida en caso que varias direcciones acerquen al paquete a su destino. La salida de los segmentos presentados en la figura 6.6 requieren una conexión a un módulo SP descrito en el capítulo 5 dentro de la sección 5.2.2. La adaptación de algoritmos no adaptativos conlleva el mismo procedimiento descrito anteriormente para



a)



b)

Figura 6.6: a) Módulo planificador de ruta para puerto  $x-$  del nodo  $[1,3]$ . b) Módulo planificador de ruta para puerto  $x-$  del nodo  $[5,3]$ , el nodo en cuestión maneja la salida de red a través del puerto  $x+$ .

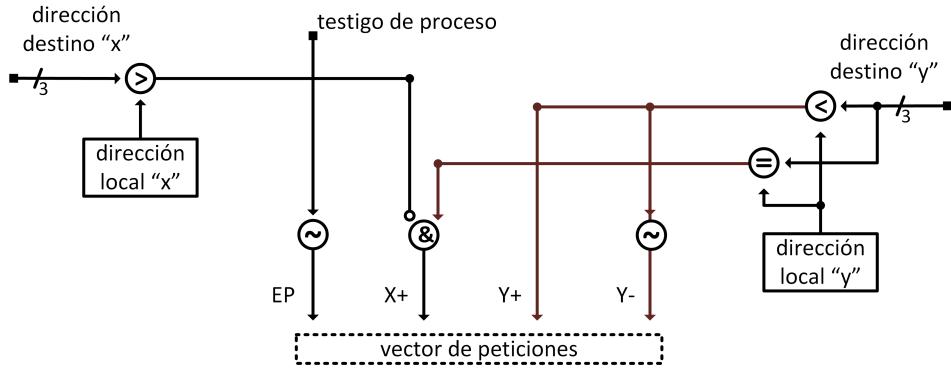


Figura 6.7: Segmento de calculo de ruta en el puerto de entrada x- mediante el uso del algoritmo XY.

su incorporación al acelerador propuesto, el código 4 muestra una versión modificada del algoritmo *XY* para el uso dentro del acelerador.

La figura 6.7 muestra el segmento de selección de ruta para el próximo salto en la red mediante el uso del algoritmo *XY*. La metodología para extender el uso de algoritmos no adaptativos sigue la misma directriz de agregar un nivel jerárquico en la toma de decisiones de la ruta a seguir.

#### ANDAMIAJE DE VALIDACIÓN

La validación del acelerador se llevó a cabo de manera progresiva, donde cada bloque se somete a pruebas funcionales. Los sub-módulos del sistema utilizan camas de prueba sencillas para su evaluación, estas se pueden encontrar en el repositorio del proyecto ([github.com/hcabrera-/lancetfish/](https://github.com/hcabrera-/lancetfish/)) dentro de los directorios *verif*.

La evaluación del acelerador a nivel sistema se llevó a cabo mediante el módulo denominado *network core*, este último está formado por el arreglo de nodos en topología tipo malla, junto a sus respectivos canales de interconexión. Cada nodo se enlaza con sus vecinos mediante un

Data: dirección de nodo actual (*localX*, *localY*)  
 Data: dirección de nodo destino (*destX*, *destY*)  
 Data: campo de testigo de procesamiento  
 Result: canales de salida válidos (*canales*)  
 begin  
     *offsetX* = *destX* - *localX*;  
     *offsetY* = *destY* - *localY*;  
     if *testigo de procesamiento* then  
         | canal := elemento de procesamiento;  
     else  
         | if *offsetX* > 0 then  
             | | canal := X+;  
         | else  
             | | if *offsetX* < 0 then  
                 | | | canal := X-;  
             | | else  
                 | | | if *offsetY* > 0 then  
                     | | | | canal := Y+;  
                 | | | else  
                     | | | | canal := Y-;  
                 | | | end  
             | | end  
         | end  
     end  
 end

Algoritmo 4: Algoritmo *XY* modificado para su uso en la arquitectura de acelerador propuesta.

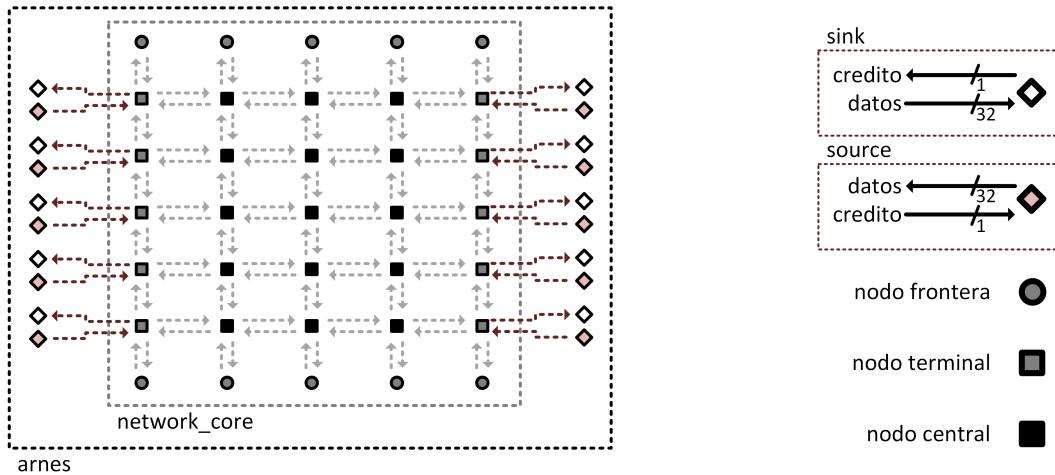
canal de transmisión y uno de recepción. Los canales están compuestos por un bus de 32 bits para la transmisión de datos, una línea de señalización para el envío o recepción de créditos y un par diferencial para el control de flujo entre nodos. El núcleo estándar de red para las pruebas funcionales está compuesto por 10 nodos terminal, 15 nodos centrales y 10 nodos frontera como se muestra en la figura 6.8.

Para la inyección y captura de paquetes se desarrollaron los módulos de verificación *source* y *sink*, ambos descritos a nivel comportamental. Los módulos de verificación ofrecen un conjunto de métodos y atributos para interactuar con la red así como para la captura de resultados. Cada canal de entrada del módulo *network core* se conecta a un módulo *source*, mientras cada canal de salida será enlazado con un módulo *sink*.

El módulo *source* provee 4 servicios principales: recepción de créditos de manera automática, este es un servicio que no requiere ser invocado y se encuentra en ejecución en segundo plano desde el inicio de la simulación. Envío de paquetes mediante la tarea *send packet*, este método se encarga de la deconstrucción de una cada dena de datos para su envío en flits, además de mantener los protocolos de control de flujo de datos utilizado por la red. La tarea *send packet* requiere se proporcione una cadena de bits que contenga la información que conforma un paquete. Los dos métodos restantes del módulo se encarga de la apertura y cierre de bitácoras de eventos para el módulo.

Es importante notar que el módulo *source* no tiene manera de generar paquetes válidos para la red, su tarea se limita al control físico del envío de datos. Para cumplir esta tarea se proporciona el módulo *packet generator*, el cual se describe más adelante en esta sección.

El módulo *sink* ofrece un servicio en segundo plano para la captura de paquetes de la red así como la devolución de créditos. Este módulo se encuentra de manera constante escuchando al



**Figura 6.8:** El núcleo de la red integra todos los nodos necesarios para su funcionamiento, sin embargo no incluye un mecanismo para la inyección de paquetes. El arnés provee de los medios para evaluar el núcleo de la red, incluyendo medios para el envío y recepción de paquetes.

canal de entrada de datos a la espera de la recepción de paquetes desde el nodo terminal ligado a él. El módulo no tiene capacidad de discriminar información, ya que solo se hace cargo de las operaciones equivalentes a la capa física de la red. Esta última sentencia aplica de igual forma para el módulo *source*. El módulo *sink* provee de métodos para el manejo de una bitácora de eventos para el puerto de recepción de paquetes.

El arnés de pruebas estándar para el acelerador está compuesto de un módulo *network core* con módulos *source* y *sink* enlazados a todos sus canales de entrada/salida respectivamente. Adicionalmente a estos últimos módulos, el arnés está encargado de la generación de la señal de reloj para todos los módulos del sistema así como una señal global de restablecimiento. El arnés es un módulo general para la ejecución de pruebas en el sistema, sobre el se crean los diferentes escenarios de evaluación para el acelerador.

Los atributos de cada uno de los módulos de validación son accesibles mediante la jerarquía

del acelerador, permitiendo el acceso a los resultados de cada operación para su valoración de manera previa a su registro en alguna bitácora del sistema o para la ejecución de patrones de evaluación específicos para ejercitar algún escenario de prueba.

Para la generación de paquetes válidos para la red se creó el módulo *packet generator*, este proporciona métodos para la generación de paquetes puramente aleatorios o con características específicas para una prueba. Los métodos para la generación de paquetes completamente aleatorios están diseñados para la evaluación de nodos de red de manera individual, mientras que la creación de paquetes a medida son preferidos para la evaluación del acelerador, permitiendo al escenario de prueba elegir las características del tráfico de la red.

## PRUEBAS

El acelerador se sometió al siguiente banco de pruebas para la obtención de un perfil de rendimiento:

- Acelerador compuesto por 25 nodos de procesamiento: 10 nodos terminales, 15 nodos centrales y 10 nodos frontera de soporte.
- La frecuencia de operación para el acelerador se fijó en 250 MHZ. Esta señal de reloj fue seleccionada para empatar velocidades de operación de trabajos similares, sin embargo, el diseño puede operar a frecuencias superiores a 300 MHZ.
- se utilizó un patrón de tráfico uniformemente distribuido. El mismo número de paquetes se inyectaron a cada una de los nodos terminales de la red, de igual forma, se distribuyó de manera equitativa el tráfico en dirección de nodos frontera y puertas de salida del acelerador.

- Todos los resultados son obtenidos con un acelerador trabajando en la región de saturación, la carga de trabajo se encuentra administrada por el mecanismo de control de flujo de la red.
- Los elementos de procesamiento en cada nodo de red requieren de 16 ciclos de reloj para obtener el resultado de su operación.
- Las corridas de evaluación consisten en la inyección de conjuntos de 250,000 paquetes por simulación. El número de paquetes de evaluación se determinó tras observar un comportamiento estable de la red bajo una saturación de canales.

El trabajo presenta los resultados de rendimiento de la red después del proceso de síntesis. Las frecuencias de operación para las pruebas de rendimiento son las obtenidas después del proceso *place & route* del acelerador a la lógica reconfigurable del **FPGA**.

# 7

## Resultados y conclusiones

La síntesis e implementación del acelerador se llevó a cabo mediante la herramienta vivado 2015.3, utilizando la estrategia por defecto para ambos procesos. El dispositivo xc7zo45ffg900-2 se fijó como objetivo para la obtención de los resultados de ocupación y rendimiento. El código rtl del proyecto no invoca elementos exclusivos para dispositivos de xilinx, por lo que el código puede ser portado a elementos de familias de Altera obteniendo resultados de ocupación similares para dispositivos con LUTs de 6 entradas.

La tabla 7.1 muestra la ocupación de una red con 25 nodos de procesamiento. Elementos de memoria distribuida son utilizados para la implementación de bancos de registros y registros de segmentación.

Recurso	Cantidad	Utilización %
LUT	25016	11.44
LUTRAM	3240	4.60
FF	21200	4.85
IO	662	

Tabla 7.1: Ocupación para acelerador con 25 nodos de procesamiento y 10 nodos frontera de soporte

recurso	router	interfaz de red	procesador	nodo de red
LUT	619	225	9	850
FF	374	330	134	838

Tabla 7.2: Recursos necesarios para la implementación de nodos incorporando un elemento de procesamiento. Los resultados de ocupación se mantienen válidos para nodos terminales o nodos centrales.

La estructura regular del acelerador permite estimar los recursos necesarios para escalar la red a un mayor número de nodos, la tabla 7.2 muestra los recursos necesarios para la implementación de un nodo de procesamiento, sea central o terminal, de manera individual. El elemento de procesamiento está formado por una cadena feistel con un número de rondas programable. La unidad funcional fue diseñada para habilitar diversos escenarios de evaluación de rendimiento del acelerador.

Dado su rol de elementos de soporte, los nodos frontera no representan un consumo significativo de recursos. La tabla 7.3 muestra los recursos necesarios para la implementación de nodos frontera. Para el acelerador en evaluación, los nodos frontera representan el 3.39 % de las LUTs requeridas por la red y el 1.17 % de flip flops del sistema.

Recurso	Estimado
LUT	85
FF	25

Tabla 7.3: Recursos necesarios para lógica de soporte del acelerador.

Trafico Demandado	Rendimiento (Gbps)
10 %	15.25
20 %	22.71
30 %	26.88
40 %	32.25
50 %	35.92
60 %	35.81
70 %	35.60
80 %	35.62
90 %	35.58
100 %	35.30

Tabla 7.4: Resultados de rendimientos del acelerador

#### RESULTADOS DE RENDIMIENTO

De manera tradicional los resultados de rendimiento para redes en chip se presentan como una relación entre el tráfico ofrecido y el tráfico servido, sin embargo, la red utilizada en este trabajo tiene como aplicación destino un acelerador, por lo que es deseable mantener la red en un estado de saturación constante. La 7.4 presenta los resultados de rendimiento bajo diversas cargas de trabajo.

Los resultados de la tabla 7.4 representan el promedio del comportamiento de 10 ejecuciones de cada simulación, utilizando un generador de números pseudo-aleatorios para inicializar los inyectores de tráfico a la red. Vale la pena resaltar que el que la red alcanzo su punto de saturación en la demanda de tráfico del 40 %, a partir de este punto la red ya no es capaz de manejar tráfico extra. Los promedios de latencia para la ejecución de cargas de trabajo sintéticas bajo diferentes escenarios de tráfico demandado se presentan en la tabla 7.5.

Latencia promedio	660
Latencia máxima	3357
Latencia mínima	200

Tabla 7.5: Latencia de tránsito de paquetes a través de la red

## CONCLUSIONES

Los resultados obtenidos de la evaluación parcial de la red muestran el potencial del trabajo como se puede apreciar en la tabla 7.6, sin embargo el desarrollo de un perfil de rendimiento bajo otros escenarios es indispensable para optimizar el sistema.

Permitir que la red distribuya el trabajo entre todos los elementos de procesamiento resulta una técnica efectiva de administración de elementos funcionales, sin embargo los puertos de salida de la red muestran promedios de trabajo de alrededor de 26 % del tiempo de ejecución, este resultado abre la puerta a la posibilidad de reducir el número de puertos de salida bajo una penalización sobre la saturación de canales de la red.

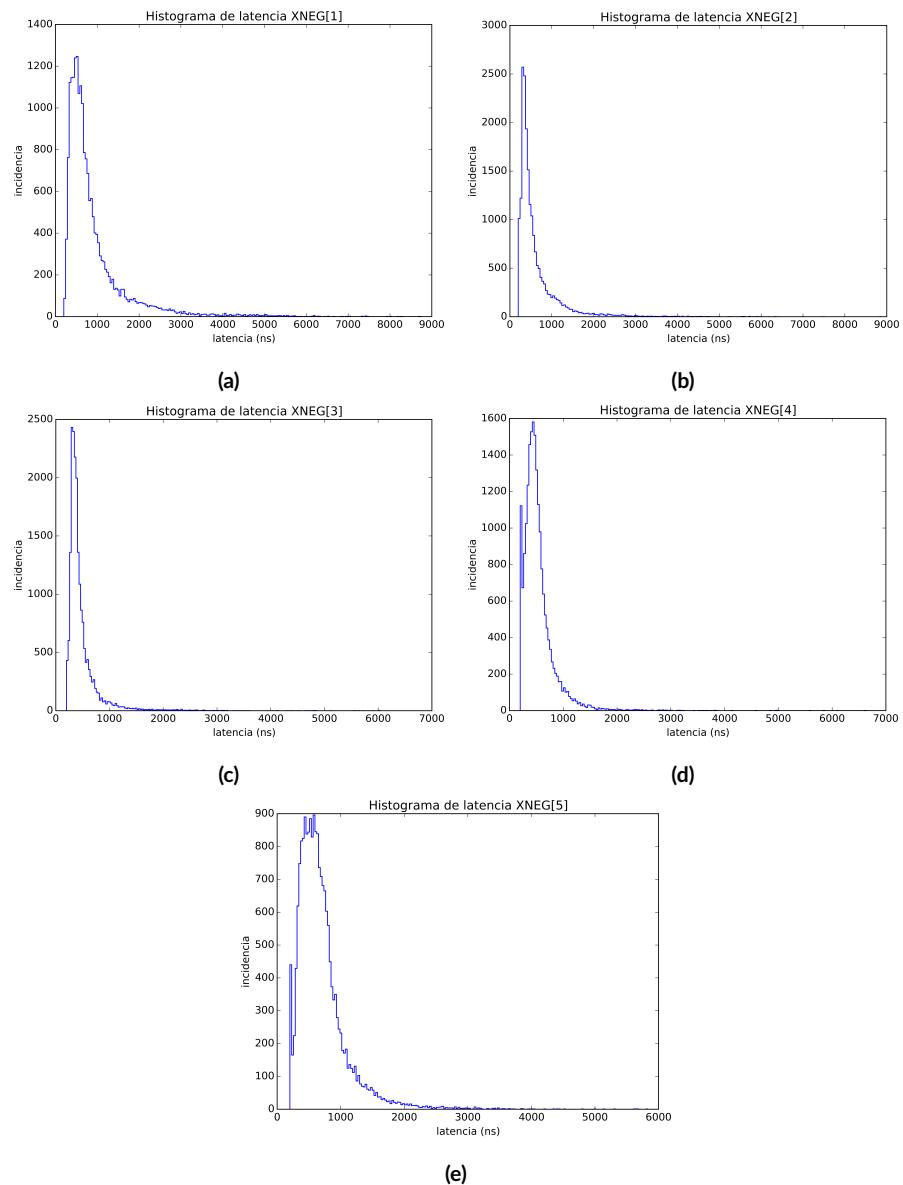
La distribución equitativa de paquetes entre nodos frontera destino y puertos de salida de la red no aparece ser el mejor patrón de tráfico para el acelerador. Las figuras 7.1 y 7.2 muestran los histogramas de latencia para cada puertos de salida, los resultados no uniformes de tiempos de salida de paquetes es evidencia de un desbalance en la transmisión de paquetes. El fenómeno de desbalance entre puertos de salida es resultado de la naturaleza parcialmente adaptativa del algoritmo *west first minimal*, el cual presenta un mayor grado de adaptabilidad para paquetes en tránsito en dirección  $x^- \rightarrow x^+$ , mientras paquetes en dirección  $x^+ \rightarrow x^-$  solo cuentan con una dirección de avance. La implementación del algoritmo de enrutamiento *xy* podría ofrecer una alternativa al algoritmo utilizado durante las pruebas de rendimiento realizadas, sin embargo el mayor número de restricciones para la toma de rutas

NoC	Topología	Relacion con EP	Algoritmo de calculo de ruta	Tipo de commutacion	frecuencia de operación	Ocupación (Slices)	Ancho de palabra de datos (bits)	Ancho de banda de banda
PnoC	crossbar	red indirecta	tablas de ruteo	commutación de circuitos	126 MHz	1223	8	NP
CuNoC	malla	red indirecta	DOR XY	Store & forward	241 MHz	128	32	NP
Quark	spidergon	red directa	calculo de cuadrante	wormhole	NP	1453	32	NP
Hermes	malla	red directa	DOR XY	wormhole	25 MHz	316	10	500 Mbps
MoCReS	malla	red directa	DOR XY	virtual cut-through	357 MHz	282	8	2.85 Gbps
SoCWire	múltiple	red directa	XY modificado	wormhole	180 MHz	1270	32	2800 Mbps
SoCIN	malla	red directa	DOR XY	wormhole	49 MHz	2100	32	NP
RoC	anillo	red directa	NP	NP	138 MHz	144	32	552 Mbps
RMBoC	bus dinámico	red indirecta	a medida	circuit switched	94 MHz	5084	32	NP
DyNoC	malla irregular	red indirecta	variante DOR XY	NP	391 MHz	1689	32	NP
Tesis	malla	red directa	WFM modificado	virtual cut-through	250 MHz	212	32	35.30 Gbps

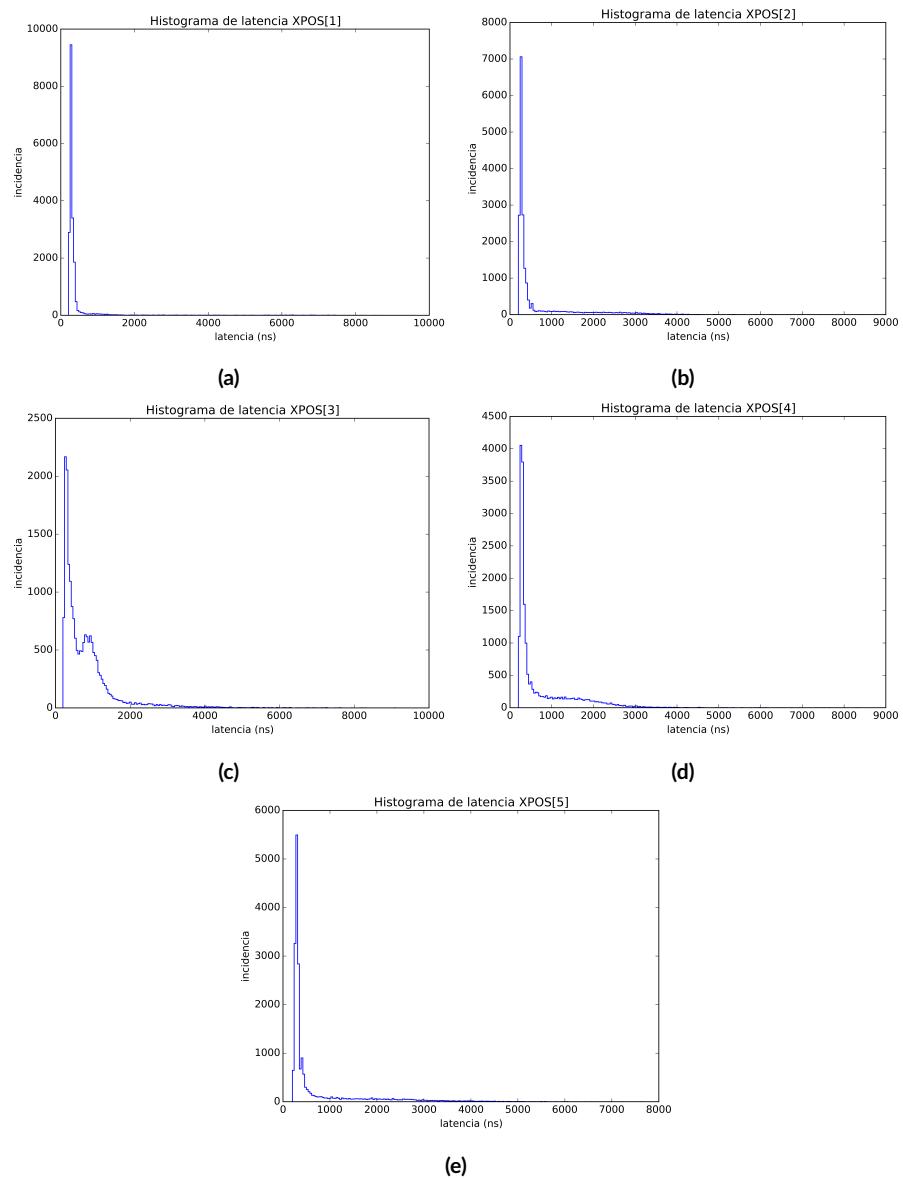
Tabla 7.6: Tabla comparativa de trabajos similares con el diseño propuesto en esta tesis. NP - Dato no proporcionado en referencia

entre nodos no lo perfila como una solución viable.

para mantener los resultados de rendimiento presentados en el capítulo 6, se requiere de una estructura de memoria capaz de alimentar de manera constante al acelerador. La capacidad de los facilitadores de datos a la red limitará su rendimiento absoluto.



**Figura 7.1:** Histogramas de latencia de recepción de paquetes para puertos en dirección x-.



**Figura 7.2:** Histogramas de latencia de recepción de paquetes para puertos en dirección x+.

# Bibliografía

- [1] Ahmadinia, A., Bobda, C., Ding, J., Majer, M., Teich, J., Fekete, S., & van der veen, J. (2005). A practical approach for circuit routing on dynamic reconfigurable devices. In *Rapid system prototyping, 2005. (RSP 2005). The 16th IEEE International workshop on* (pp. 84--90).
- [2] Bahn, J. H. & Bagherzadeh, N. (2009). A generic traffic model for on-chip interconnection networks. In *In the first International workshop on Network-on-chip Architectures.*
- [3] Benini, L. & De Micheli, G. (2002). Networks on chips: a new soc paradigm. *computer*, 35(1), 70--78.
- [4] Bjerregaard, T. & Mahadevan, S. (2006). A survey of research and practices of network-on-chip. *ACM comput. surv.*, 38(1).
- [5] Bobda, C. & Ahmadinia, A. (2005). Dynamic interconnection of reconfigurable modules on reconfigurable devices. *Design Test of computers, IEEE*, 22(5), 443--451.
- [6] Bobda, C., Ahmadinia, A., Majer, M., Teich, J., Fekete, S., & van der veen, J. (2005). Dynoc: A dynamic infrastructure for communication in dynamically reconfigurable devices. In *Field programmable Logic and Applications, 2005. International conference on* (pp. 153--158).
- [7] Boeri, F. & Auguin, M. (1993). Opsila: a vector and parallel processor. *computers, IEEE transactions on*, 42(1), 76--82.
- [8] Chawade, S. D., Gaikwad, M. A., & Patrikar, R. M. (2012). Article: Review of xy routing algorithm for network-on-chip architecture. *International journal of computer applications*, 43(21/973-93-80867-69-8), 20--23. Full text available.

- [9] chopkar, p. n. & gaikwad, m. a. (2013). Article: review of xy routing algorithm for 2d torus topology of noc architecture. *IJCA special issue on Recent Trends in Engineering technology*, RETRET, 22--26. Full text available.
- [10] cireşan, D. C., Meier, U., Masci, J., Gambardella, L. M., & schmidhuber, J. (2011). Flexible, high performance convolutional neural networks for image classification. In *proceedings of the twenty-second international joint conference on Artificial Intelligence - volume volume two*, IJCAI'11 (pp. 1237--1242).: AAAI Press.
- [11] concer, N., iamundo, s., & bononi, L. (2009). aequalized: A novel routing algorithm for the spidergon network on chip. In *Design, Automation Test in Europe conference exhibition, 2009. DATE '09*. (pp. 749--754).
- [12] coppola, M., Locatelli, R., Maruccia, G., Pieralisi, L., & scandurra, A. (2004). spidergon: a novel on-chip communication network. In *system-on-chip, 2004. proceedings. 2004 International symposium on* (pp. 15--).
- [13] corporation, I. (2014). *Intel® Quark™ SOC X1000 Datasheet*. santa clara, CA, USA: Intel corporation.
- [14] cota, É., de morais Amory, A., & Lubaszewski, M. S. (2011). *Reliability, Availability and serviceability of Networks-on-chip*. Springer science & business media.
- [15] Daemen, J. & rijmen, v. (2002). *The Design of Rijndael*. secaucus, NJ, USA: springer-verlag New York, inc.
- [16] dally, w. & towles, b. (2003). *Principles and practices of interconnection networks*. san Francisco, CA, USA: Morgan Kaufmann Publishers Inc.
- [17] Deslauriers, F., Langevin, M., Bois, G., Savaria, Y., & Paulin, P. (2006). roc: A scalable network on chip based on the token ring concept. In *Circuits and systems, 2006 IEEE North-East workshop on* (pp. 157--157).
- [18] duato, J., robles, A., sillà, F., & beivide, R. (2001). A comparison of router architectures for virtual cut-through and wormhole switching in a now environment. *J. Parallel distrib. comput.*, 61(2), 224-253.

- [19] Duato, J., Yalamanchili, S., & Lionel, N. (2002). *Interconnection Networks: An Engineering Approach*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.
- [20] Dye, D. (2012). *Partial Reconfiguration of Xilinx FPGAs Using ISE Design Suite (WP374)*. San Jose, CA, USA: Xilinx Inc.
- [21] Flynn, M. J. (1972). Some computer organizations and their effectiveness. *IEEE Trans. Comput.*, 21(9), 948--960.
- [22] Foundation, F. S. (2007). GNU General Public License v3. (gpl v3).
- [23] Glass, C. J. & Ni, L. M. (1998). The turn model for adaptive routing. In *25 Years of the International Symposia on Computer Architecture (Selected Papers)*, ISCA '98 (pp. 441--450). New York, NY, USA: ACM.
- [24] Hadjat, K., St-Pierre, F., Bois, G., Savaria, Y., Langevin, M., & Paulin, P. (2007). An fpga implementation of a scalable network-on-chip based on the token ring concept. In *Electronics, Circuits and Systems, 2007. ICECS 2007. 14th IEEE International Conference on* (pp. 995--998).
- [25] Hassen, W. & Amiri, H. (2013). Block matching algorithms for motion estimation. In *e-Learning in Industrial Electronics (ICELIE), 2013 7th IEEE International Conference on* (pp. 136--139).
- [26] Hilton, C. & Nelson, B. (2005). A flexible circuit switched noc for fpga based systems. In *Field Programmable Logic and Applications, 2005. International Conference on* (pp. 191--196).
- [27] Hilton, C. & Nelson, B. (2006). Pnoc: a flexible circuit-switched noc for fpga-based systems. *Computers and Digital Techniques, IEE Proceedings -*, 153(3), 181--188.
- [28] Hubel, D. H. & Wiesel, T. N. (1959). Receptive fields of single neurones in the cat's striate cortex. *The Journal of Physiology*, 148(3), 574--591.
- [29] Hunt, M. & Rowson, J. (1996). Blocking in a system on a chip. *spectrum, IEEE*, 33(II), 35--41.

- [30] Inc., A. (2015a). Adapteva inc. <http://www.adapteva.com/>.
- [31] Inc., A. (2015b). *Nios II Gen2 Processor Reference Guide (NII\$VIGEN2)*. San Jose, CA, USA: Altera Inc.
- [32] Inc., X. (2014a). *7 series FPGAs Configurable Logic Block User Guide (UG474)*. San Jose, CA, USA: Xilinx Inc.
- [33] Inc., X. (2014b). *Microblaze Processor Reference Guide (UG984)*. San Jose, CA, USA: Xilinx Inc.
- [34] Inc., X. (2015c). *Vivado Design Suite User Guide - Using the Vivado IDE (UG893)*. San Jose, CA, USA: Xilinx Inc.
- [35] Jaffe, P., Clifford, G., & Summers, J. (2008). Spacewire for operationally responsive space. In *Aerospace conference, 2008 IEEE* (pp. 1--5).
- [36] Jalier, C., Lattard, D., Jerraya, A. A., Sassatelli, G., Benoit, P., & Torres, L. (2010). Heterogeneous vs homogeneous mpsoc approaches for a mobile lte modem. In *Proceedings of the conference on Design, Automation and Test in Europe, DATE '10* (pp. 184--189). 3001 Leuven, Belgium, Belgium: European Design and Automation Association.
- [37] Janarthanan, A., Swaminathan, V., & Tomko, K. (2007). MOCRES: an area-efficient multi-clock on-chip network for reconfigurable systems. In *VLSI, 2007. ISVLSI '07. IEEE Computer Society Annual Symposium on* (pp. 455--456).
- [38] Jovanovic, S., Tanougast, C., Weber, S., & Bobda, C. (2007). CUNOC: A scalable dynamic noc for dynamically reconfigurable fpgas. In *Field programmable logic and applications, 2007. FPL 2007. International conference on* (pp. 753--756).
- [39] Kempf, T., Ascheid, G., & Leupers, R. (2011). *Multiprocessor systems on chip: design space exploration*. SpringerLink : Bücher. Springer New York.
- [40] Kermani, P. & Kleinrock, L. (1979). Virtual cut-through: a new computer communication switching technique. *Computer Networks*, 3, 267--286.

- [41] kim, j. & el-amawy, a. (1999). performance and architectural features of segmented multiple bus system. in *parallel processing, 1999. proceedings. 1999 international conference on* (pp. 154--161).
- [42] kwok, t.-o. & kwok, y.-k. (2008). on the design, control, and use of a reconfigurable heterogeneous multi-core system-on-a-chip. in *parallel and distributed processing, 2008. ipdps 2008. ieee international symposium on* (pp. i--ii).
- [43] li, x. & hammami, o. (2009). an automatic design flow for data parallel and pipelined signal processing applications on embedded multiprocessor with noc: Application to cryptography. *int. j. reconfig. comput.*, 2009, 5:2--5:2.
- [44] mahmud, s. (1989). communication performance in a hierarchical bus system. in *circuits and systems, 1989., ieee international symposium on* (pp. 122--125 vol.1).
- [45] mao, c., guan, y., & zhang, j. (2009). on-board spacewire router for space mission. in *information processing, 2009. apcip 2009. asia-pacific conference on*, volume 2 (pp. 525--528).
- [46] mas, g. & martin, p. (2004). network-on-chip: the intelligence is in the wire. in *computer design: vlsi in computers and processors, 2004. iccd 2004. proceedings. ieee international conference on* (pp. 174--177).
- [47] mello, a., tedesco, l., calazans, n., & moraes, f. (2005). virtual channels in networks on chip: implementation and evaluation on hermes noc. in *proceedings of the 18th annual symposium on integrated circuits and system design, sbcci '05* (pp. 178--183). new york, ny, usa: acm.
- [48] moadeli, m., maji, p., & vanderbauwhede, w. (2009a). design and implementation of the quarc network on-chip. in *parallel distributed processing, 2009. ipdps 2009. ieee international symposium on* (pp. 1--9).
- [49] moadeli, m., maji, p., & vanderbauwhede, w. (2009b). quarc: A high-efficiency network on-chip architecture. in *advanced information networking and applications, 2009. aina '09. international conference on* (pp. 98--105).

- [50] moadeli, m., shahrabi, a., vanderbauwhede, w., & ould-khaoua, m. (2007). An analytical performance model for the spidergon noc. In *Advanced information Networking and Applications, 2007. AINA '07. 21st International conference on* (pp. 1014--1021).
- [51] moadeli, m., vanderbauwhede, w., & shahrabi, a. (2008a). A performance model of communication in the quarc noc. In *parallel and distributed systems, 2008. ICPADS '08. 14th IEEE International conference on* (pp. 908--913).
- [52] moadeli, m., vanderbauwhede, w., & shahrabi, a. (2008b). Quarc: A novel network-on-chip architecture. In *parallel and distributed systems, 2008. ICPADS '08. 14th IEEE International conference on* (pp. 705--712).
- [53] Moraes, F., Calazans, N., Mello, A., Möller, L., & Ost, L. (2004). Hermes: an infrastructure for low area overhead packet-switching networks on chip. *Integration, the {VLSI} journal*, 38(1), 69 -- 93.
- [54] national institute of standards and technology (1977). *FIPS PUB 46: Data Encryption standard*. National Institute of Standards and Technology.
- [55] osterloh, b., Michalik, H., Fiethe, B., & Bubenhagen, F. (2010). Architecture verification of the socwire noc approach for safe dynamic partial reconfiguration in space applications. In *Adaptive hardware and systems (AHS), 2010 NASA/ESA conference on* (pp. 1--8).
- [56] osterloh, b., Michalik, H., Fiethe, B., & Kotarowski, K. (2008). socwire: A network-on-chip approach for reconfigurable system-on-chip designs in space applications. In *Adaptive hardware and systems, 2008. AHS '08. NASA/ESA conference on* (pp. 51--56).
- [57] Palesi, M. & Daneshthalab, M. (2013). *Routing Algorithms in Networks-on-chip*. Springer publishing company, Incorporated.
- [58] Parkes, S. (2008). Spacewire for adaptive systems. In *Adaptive hardware and systems, 2008. AHS '08. NASA/ESA conference on* (pp. 77--82).
- [59] Parkes, S. & Armbruster, P. (2005). Spacewire: a spacecraft onboard network for real-time communications. In *Real time conference, 2005. 14th IEEE-NPSS* (pp. 6--10).

- [60] Perry, T. S. (1989). Intel's secret is out. *IEEE spectr.*, 26(4), 22--28.
- [61] Pionteck, T., Koch, R., & Albrecht, C. (2006). Applying partial reconfiguration to networks-on-chips. In *Field programmable logic and Applications, 2006. FPL '06. International conference on* (pp. 1-6).
- [62] Rakow, G., Schnurr, R., & Parkes, S. (2006). Spacewire protocol id: what does it mean to you? In *Aerospace conference, 2006 IEEE* (pp. 7 pp.--).
- [63] Rincon, A., Cherichetti, G., Monzel, J., Stauffer, D., & Trick, M. (1997). Core design and system-on-a-chip integration. *Design Test of Computers, IEEE*, 14(4), 26--35.
- [64] Romine, P. L. (1992). *A dynamically segmented multiple bus architecture*. PhD thesis, Huntsville, AL, USA. UMI Order No. GAX92-29495.
- [65] Saban, K. (2012). *xilinx stacked silicon interconnect technology delivers breakthrough FPGA capacity, bandwidth, and power efficiency (WP380)*. San Jose, CA, USA: Xilinx Inc.
- [66] Sadawarte, Y. A., Gaikwad, M. A., & Patrikar, R. M. (2011). Comparative study of switching techniques for network-on-chip architecture. In *Proceedings of the 2011 international conference on communication, computing and security, ICCCS '11* (pp. 243--246). New York, NY, USA: ACM.
- [67] Saleh, R. (2005). An approach that will noc your socs off! *Design Test of Computers, IEEE*, 22(5), 488--488.
- [68] Shannon, C. (1949). Communication theory of secrecy systems. *Bell System Technical Journal, The*, 28(4), 656--715.
- [69] Shin, K. G. & Daniel, S. W. (1996). Analysis and implementation of hybrid switching. *IEEE Trans. Comput.*, 45(6), 684--692.
- [70] Tota, S. V., Casu, M. R., Roch, M. R., Macchiarulo, L., & Zamboni, M. (2009). A case study for noc-based homogeneous mpsoc architectures. *IEEE Trans. Very Large Scale Integr. Syst.*, 17(3), 384--388.

- [71] wiegand, t., sullivan, g., bjontegaard, g., & luthra, a. (2003). overview of the h.264/avc video coding standard. *circuits and systems for video technology, IEEE transactions on*, 13(7), 560--576.
- [72] yang, y. s., bahn, j. h., lee, s. e., & bagherzadeh, n. (2009). parallel and pipeline processing for block cipher algorithms on a network-on-chip. in *information technology: new generations, 2009. ITNG '09. sixth international conference on* (pp. 849--854).
- [73] yoon, y. j., concer, n., petracca, m., & carloni, l. (2013). virtual channels and multiple physical networks: two alternatives to improve noc performance. *computer-aided design of integrated circuits and systems, IEEE transactions on*, 32(12), 1906--1919.
- [74] yuan, r., ruan, s., & gotze, j. (2013). a practical noc design for parallel des computation. in *vlsi design, automation, and test (VLSI-DAT), 2013 international symposium on* (pp. 1-4).
- [75] zeferino, c. & susin, a. (2003). socin: a parametric and scalable network-on-chip. in *integrated circuits and systems design, 2003. SBCCI 2003. proceedings. 16th symposium on* (pp. 169--174).



HIS THESIS WAS TYPESET using  $\text{\LaTeX}$ ,  
**T**originally developed by Leslie Lamport  
and based on Donald Knuth's  $\text{\TeX}$ . The  
body text is set in 11 point Egenolff-Berner  
Garamond, a revival of Claude Garamont's  
humanist typeface. The above illustration,  
*science experiment 02*, was created by Ben  
Schlitter and released under [CC BY-NC-ND](#)  
[3.0](#). A template that can be used to for-  
mat a PhD dissertation with this look I  
feel has been released under the permis-  
sive AGPL license, and can be found on-  
line at [github.com/asm-products/Dissertate](https://github.com/asm-products/Dissertate)  
or from its lead author, Jordan Suchow, at  
[suchow@post.harvard.edu](mailto:suchow@post.harvard.edu).