# Diseño e Implementación de un Simulador de Arquitectura Network-on-Chip (NoC)

Sr. Edwin Tufiño, Ing. Plabo Ramos, Ing. Darwin Aguilar
Departamento de Eléctrica y Electrónica
Escuela Politécnica del Ejército (ESPE)
Sangolquí, Ecuador
E-mail: edwin.t@outlook.com

***Resumen*— En la actualidad, los System-On-Chips (SoCs) han evolucionado considerablemente en términos de rendimiento, confiabilidad e integración. La integración ha generado un crecimiento notable del número de Bloques de propiedad Intelectual (IP-cores) en el chip. Desafortunadamente, el crecimiento de los IP-cores ha causado problemas entre la interconectividad de los mismos. Para resolver este problema, un nuevo paradigma ha sido introducido: los Network-On-Chip ((NoC). El objetivo de este trabajo es realizar una investigación de los parámetros que componen a las arquitecturas de las Network-On-Chip (NoC), de igual manera, indagar y analizar las herramientas de NoC existentes y, en base a una de ellas, realizar pruebas de la arquitectura NoC propuesta.**

***Palabras Claves: System-on-Chip (SoC), IP-Core (Bloque de Propiedad Intelectual), Networks-on-Chip (NoC).***

## I. INTRODUCCIÓN

En la actualidad, los sistemas embebidos han llegado a escalas nanométricas, proporcionando la integración de varios sistemas en un solo chip llamado "System On Chip" (SoC). Estos sistemas intra-chip por su alta complejidad computacional contienen varios módulos internos con un grado de comunicación importante.

El aumento de las conexiones dentro de un Circuito Integrado (IC) está llegando a niveles en los cuales las soluciones tradicionales, como cables dedicados y buses de datos, no son viables principalmente por el incremento en la disipación de potencia y el espacio físico que éstos ocupan.

Además, se está trabajando en espacios de 40nm, 35nm y, recientemente, en 28nm con frecuencias de reloj mayores y consumos de energía menores, por lo tanto, se tendrán billones de transistores y con ello billones de conexiones. De allí, la importancia de encontrar una nueva alternativa de interconexión que supere los problemas de disipación de potencia y proporcione una mejora en el throughput reduciendo la latencia y pérdida de paquetes. La tendencia actual es emplear las Networks-On-Chip (NoC) para solventar la comunicación intra-chip. [1]

El paradigma NoC es de suma importancia porque permite a los ingenieros de diseño trabajar en sus IP-cores, sin tener que preocuparse de los problemas de interconectividad de los mismos.

Antes de analizar los simuladores existentes y poner a prueba una nueva propuesta de arquitectura, se va a dar a conocer los parámetros que conforman las NoC al igual que las características que sus arquitecturas poseen, para tener una mejor visión

## II. NETWORK-ON-CHIP

Las NoC surgen como una solución de las limitaciones en las arquitecturas de comunicación existentes, tales como: eficiencia energética, escalabilidad del ancho de banda con respecto a las soluciones tradicionales como son las de buses de datos.

Existen varios tipos de redes en los chips, los cuales pueden ser buses de datos, off-chip networks y on-chip networks.
Primero, hay que comprender estos diferentes tipos de redes en chips para así poder analizar las ventajas que presenta una NoC.

Los buses de datos son simplemente canales que se interconectan entre dos unidades impidiendo que exista una conexión distinta a la de las dos unidades conectadas.
Las network off-chip permiten, en algunos casos, múltiples transferencias de datos por el mismo enlace que también permite que las unidades sean conectadas o desconectadas de la red sin previa notificación. Esto produce que la topología cambie al momento que una unidad sea desconectada, lo cual afectará al desempeño de la red.

Las Network On-Chip residen en un ambiente controlado, una vez creada la red, ningún elemento puede ser eliminado o aumentado a ésta y todos los parámetros que se deseen controlar se los debe realizar antes de la implementación.[2]

### *A. Arquitectura NoC*

Las arquitecturas de las NoC están compuestas de los mecanismos de comunicación, el modo de conmutación y los algoritmos de ruteo.[3]

- Los mecanismos de comunicación específica como los paquetes pasan a través de la red.

  Dos mecanismos de comunicación son la comunicación de paquetes y la comunicación de circuitos.

  En la comunicación de circuitos se establece un camino entre el origen y el destino antes de comenzar a transmitir los datos, una vez establecido el camino se comienzan a transmitir los datos y cualquier otra comunicación que se desee realizar es denegada.

  En la comunicación de paquetes se transmiten los datos sin necesidad de establecer un camino de comunicación. La comunicación de paquetes necesita definir el modo de conmutación.

- Los modos más importantes de conmutación son almacenamiento-envío, virtual cut-through y wormhole.

  En almacenamiento-envío, el conmutador no transmite hasta que todo el paquete sea recibido, lo que genera latencia en la conmutación.

  En virtual cut-through el conmutador guarda los datos en un buffer y puede enviar el paquete si el siguiente conmutador está listo para aceptar a éste, lo cual reduce la latencia de conmutación.

  El wormhole es una variante del virtual cut-through, puesto que reduce el tamaño de los buffers; esto se debe a que los paquetes son transmitidos en unidades llamadas flits (unidades de control de flujo) que son las unidades más pequeñas para el control de flujo. La cabecera del flit es la única que posee la ruta y los demás paquetes de flits que llevan los datos siguen el camino determinado por esta cabecera.

- Los algoritmos de ruteo definen el camino a tomar por los paquetes entre el origen y el destino. De acuerdo a dónde se toman las decisiones para enrutar, se puede clasificarlos en dos grupos: los algortimos de ruteo de la fuente y los distribuidos; los de fuente son aquellos cuyo camino a seguir, desde el origen al destino, está determinado por un solo conmutador (fuente); en cambio, en el distribuido, cada vez que un paquete llega a un conmutador éste decide qué camino tomar.
  De acuerdo a cómo el camino es seleccionado, se los puede clasificar en determinado y adaptativo. En el determinado, el camino a tomar está delimitado por el origen y el destino; en cambio en el adaptativo, el camino se determina de acuerdo a la congestión que tenga la red.

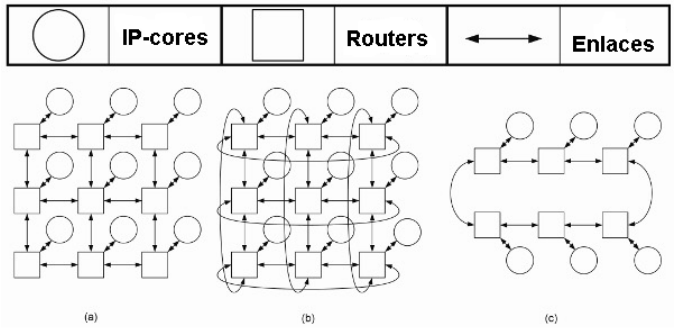
  También es conocido que el algoritmo de ruteo predominante es el XY. La única falencia que este tiene es la generación de puntos muertos.

*B. Topología*

La topología de una red es la configuración geométrica de un nivel lógico utilizado para conectar diferentes dispositivos de red. Existen diferentes tipos de topologías, desde las más sencillas como una conexión punto a punto y otras complejas como una distribución jerárquica.

El aspecto principal para determinar qué topología se debe seleccionar es la de saber las características que va a tener el tráfico que atraviesa la red. Por lo tanto, se debe realizar un estudio previo para determinar qué topología será la correcta para la red antes de realizar la implementación de la misma.

La figura 1 muestra algunos tipos de topología existentes.
a) mesh , b) torus , c) anillo




*C. Stack NoC*

Las 7 capas del modelo OSI (Open System Interconnect) fueron desarrolladas para un propósito general de las redes.

Cabe recalcar que las 7 capas del modelo no son estándares para la creación de una red, pero sí nos dan las pautas necesarias para lograr comprender el comportamiento de la misma; un ejemplo sería las NoC puesto que, a diferencia de una red de propósito general (internet) donde no se conocen los números de nodos existentes y está en constante crecimiento, en las NoC se conocen todos los parámetros de la arquitectura como el número de nodos existentes, la topología y sus algoritmos de enrutamiento.

Con este conocimiento adicional de la infraestructura de la red, en el chip se puede omitir un grupo de capas del modelo OSI; de igual manera, al reducir el número de capas del modelo, se reduce la latencia en la red.

Las capas del modelo OSI que se utilizan en una NoC son las siguientes:

- La capa física: Determina el número de cables y su longitud entre las conexiones de los conmutadores y de los recursos.

- La capa de enlace: Es la que determina qué protocolos se van a utilizar para la comunicación entre conmutadores y conmutadores-recursos. Esta capa está formada por celdas que están compuestas de cables tanto de transmisión como de control.

- La capa de red: Determina cómo un paquete viaja a través de la red, desde una dirección de origen hacia una dirección de destino.

- La capa de transporte: Es encargada de unir los paquetes de la capa de red para así formar el mensaje.

.

## III. SIMULADORES NOC

Los SoCs están formados por una complicada arquitectura la cual está compuesta de numerosos elementos de procesamiento (IP-Cores). Por lo tanto, esto requiere un diseño óptimo de la NoC para asegurar un correcto manejo interno de los datos.

Para facilitar el desarrollo de sistemas embebidos que contengan una red NoC, algunas herramientas de simulación dedicadas han sido propuestas.

Las herramientas NoC dependen del propósito para el cual fueron creadas. Podemos distinguir dos grandes clases de herramientas como son los sintetizadores o compiladores y los simuladores.
Los sintetizadores se refieren a la calidad con que se generan las arquitecturas (espacio, energía consumida, material) y el nivel de abstracción para modelar las NoC.

En cambio, en los simuladores, hay dos criterios que se toman en cuenta: estimado de la energía disipada y el rendimiento computacional (troughput, latencia). Una estimación de estas características en las NoC es de suma importancia a la hora de diseñar.
A continuación se mostrará una lista tanto de los sintetizadores como de los simuladores de NoC. Cabe recalcar que esta lista no muestra todos las herramientas existentes.[4]

### *A. NS-2*

NS-2 fue originalmente desarrollado para simular redes computacionales. Sin embargo, como las NoC comparten muchas de las características con las redes computacionales, NS-2 fue ampliamente utilizado por muchos investigadores acerca de las NoC. Varios estudios que se han realizado utilizando la herramienta de simulación NS-2 la convierte en un simulador fiable a la hora de comparar el rendimiento entre dos arquitecturas diferentes. Cabe recalcar que NS-2 es un simulador de código abierto desarrollado en C++ por lo cual los investigadores pueden compartir su información.[5]

### *B. Noxim*

Esta herramienta fue desarrollada por el equipo computacional de la Universidad de Catania. Está desarrollado en SystemC. Permite al usuario evaluar una arquitectura 2D Mesh variando ciertos parámetros. Noxim permite evaluar a la NoC en términos de troughput, latencia y el poder consumido.[6]

### *C. Sunfloor*

Sunfloor es una herramienta de apoyo para el diseño NoC. Se puede utilizar en las primeras fases de diseño para sintetizar la más adecuada topología con los siguientes parámetros como entrada (Energía y Espacio). A partir de estos datos, Sunfloor genera unas especificaciones listas para ser traducidas a una arquitectura integral, por lo general en lenguaje SystemC. [7]

### *D. Orion*

Esta herramienta fue desarrollada por el equipo de la Universidad de Princeton en el 2003. Es un simulador dedicado exclusivamente para la estimación del poder consumido y el espacio de las arquitecturas NoC. Una de sus mejoras, comparado con otros simuladores, es que existe el soporte para probar nuevos semiconductores a través de los modelos de transistores y su capacitancia de acuerdo a cómo evolucione la industria. [8]

### *E. BookSim*

Fue desarrollada entre los años 2002-2010 en Stanford. La versión actual, BookSim 2.0, es compatible con una amplia variedad de topologías, tales como Mesh, Torus y Fat tree, ofrece diversos algoritmos de enrutamiento e incluye numerosas opciones para personalizar la micro arquitectura de la red del enrutador. [9]

### *F. Worm_Sim*

Worm_sim proporciona una solución eficiente para permitir al usuario especificar condiciones de tráfico arbitrario para la NoC. El usuario tiene el control sobre la velocidad de transmisión de cada nodo IP, el tamaño del paquete y su distribución. Aún más, permite al usuario adjuntar un archivo de rastreo para cada nodo IP individual para que el sistema, bajo simulación, pueda imitar el estado del tráfico exacto de aplicaciones reales. Fue desarrollada en el año 2005. [10]

*G. Nostrum*

NoC Nostrum Simulator Environment (NNSE) es parte del proyecto de Nostrum y contiene un simulador basado en SystemC. Una interfaz gráfica de usuario que se utiliza para seleccionar el tamaño, la topología, el enrutamiento y los patrones de tráfico. Fue desarrollada por KTH – Royal Institute of Technology entre los años 2002-2006. En base a estos parámetros de configuración los resultados de la simulación se pueden mostrar en una variedad de gráficos. [11]

*H. Nirgam*

La Universidad de Southampton la desarrolló en el año 2007. NIRGAM está basado en SystemC que es un simulador de ciclo preciso para la investigación en red en un chip (NoC). Presta un apoyo sustancial a experimentar con el diseño de NoC en términos de algoritmos de enrutamiento y nuevas propuestas de topologías. [12]

La Tabla 1 muestra la lista de simuladores encontrados, al igual que su año de desarrollo. [4]

| # | Nombre | Año | Grupo |
|---|---|---|---|
| 1 | NS-2 | 1995 | DARPA |
| 2 | Noxim | 2010 | Universidad de Catagne |
| 3 | Darsim | 2009 | MIT |
| 4 | SunFloor - 3D | 2006-09 | EPFL |
| 5 | Orion | 2003-09 | Universidad de Princeton |
| 6 | INSEE | 2005 | Universidad Basca |
| 7 | Atlas | 2005 | Universidad Federal de Brazil |
| 8 | Nocic | 2004 | Universidad de Massachussetts |
| 9 | Pestannna | 2004 | Laboratorios Phillips |
| 10 | Pirate | 2004 | Escuela Politecnica de Milan |
| 11 | Sunmap | 2004 | Universodad de Stanford |
| 12 | NoCgen | 2004 | - |
| 13 | FlexNoc | - | ARTERIS |
| 14 | Inoc | - | - |
| 15 | Chain | - | Silistix |
| 16 | BookSim | 2002-10 | Universidad de Stanford |
| 17 | Worm_sim Si | 2005 | CMU |
| 18 | NoC Simulato | 2007 | Universidad de Las Palmas de Gran Canaria |
| 19 | VNOC | 2009 | - |
| 20 | Sicosys | 2008 | Universidad de Cantabria |
| 21 | Nostrum | 2002-06 | KTH – Royal Institute of Technology |
| 22 | Nirgam | 2007 | Universidad de Southampton |
| 23 | NoCSim | 2002-06 | TAMU |
| 24 | gpNoCsim | 2006 | Bangladesh University of Engineering and Technology |

Tabla 1. Simuladores NoC encontrados

## IV. PARÁMETROS A EVALUAR

Lo que se desea evaluar en una NoC es un paso importante para lograr analizar a las arquitecturas. Existen 3 criterios generales tomados en cuenta por la comunidad de investigadores: i) El área ii) El poder consumido iii) La latencia. Existen también otros criterios tales como la pérdida de paquetes o la longitud de los cables pero no son criterios discutidos a la hora de proponer una arquitectura NoC. En el estudio de los simuladores sobre las NoC se pude encontrar que, en general, todas las herramientas se centran en los 3 criterios anteriormente mencionados.
La Tabla 2 lista los simuladores NoC que se pudieron analizar mostrando las características que cada uno facilita [4]

| # | Simulador | TR | TB | DP | AR | PIR | ES | DT | CA | CP | T | L | Disponibilidad |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | NS-2 | + | + | + | + | + | + | + | + | + | + | + | + |
| 2 | Noxim | + | + | + | + | + | + | + | - | + | + | + | + |
| 3 | Darsim | + | + | + | + | + | + | + | - | - | + | + | - |
| 4 | SunFloor -3D | + | - | - | - | - | - | - | - | + | + | + | - |
| 5 | Orion | - | + | - | - | - | - | - | + | + | - | - | - |
| 6 | Atlas | + | - | + | + | + | - | - | - | - | + | + | + |
| 7 | Pirate | + | + | - | - | - | - | - | + | + | - | - | - |
| 8 | Sunmap | + | + | - | - | - | - | - | - | + | + | + | - |
| 9 | uSpider | + | + | - | - | - | - | - | - | - | + | + | - |
| 10 | NoCgen | - | + | - | - | - | - | - | - | - | + | + | - |
| 11 | FlexNoc | + | + | + | + | + | + | + | + | + | + | + | commercial |
| 12 | iNoc | + | + | + | + | + | + | + | + | + | + | + | commercial |
| 13 | Chai Tool | + | + | + | + | + | + | + | + | + | + | + | commercial |

| | |
|---|---|
| TR= Tamaño de Red | ES= Estrategia de Selección |
| TB= Tamaño del Buffer | DT= Distribucion de Trafico |
| DP= Distribucion de Paquetes | CA= Consumo de Area |
| AR= Algoritmo de Ruteo | CP= Consumo de Poder |
| PIR= Packet Injection Ratio | T= Throughput |
| | L= Latency |

Tabla 2. Simuladores y sus características

## V. SIMULADOR BASE - NOXIM

Luego de haber analizado las ventajas y desventajas de los simuladores existentes en el mercado, al igual que los parámetros que éstos nos permiten controlar, los resultados que entregan y el código de programación que utilizan, se ha optado por seleccionar al simulador Noxim como el simulador base para el propósito del proyecto, debido a que es un simulador muy completo porque permite manipular varios parámetros de entrada y nos entrega 3 de los 4 criterios a tomar en cuenta a la hora de analizar una arquitectura NoC. Además, es de código abierto y puede ser descargado bajo los términos de la licencia GPL.

Cabe recalcar que su lenguaje de programación es SystemC, el cual es frecuentemente descrito como un lenguaje de descripción de hardware como son VHDL y Verilog. Ambos lenguajes de programación son utilizados en el desarrollo de los SoC por lo tanto, Noxim al utilizar System C como base, arroja resultados semejantes a los reales.

### A. *Noxim*

Noxim es un simulador de NoC desarrollado en la Universidad de Catania (Italia), en el lenguaje de programación llamado SystemC que es un lenguaje descriptor del sistema basado en C++ y puede ser descargado de SourceForge bajo los términos de la licencia GPL.

Como ya se ha mencionado, Noxim se apoya en SystemC para modelar una red de interconexión de la forma más real posible. SystemC es un lenguaje de descripción de hardware similar a VHDL y Verilog y su principal característica radica en el modelado de sistemas a nivel de comportamiento. Los objetos descritos bajo este lenguaje son capaces de comunicarse durante una simulación de tiempo real utilizando señales de cualquier tipo. Un sistema en SystemC está formado por un conjunto de módulos que describen cierta funcionalidad y se comunican con otros módulos a través de canales o eventos.

Noxim tiene una interfaz de comandos de línea para definir varios parámetros de la NoC. En particular, el usuario puede modificar el tamaño de la red, el tamaño del buffer, el tamaño del paquete a ser distribuido, el algoritmo de ruteo, el modo de selección, la velocidad de paquetes a ser inyectados al sistema y la distribución del tráfico.
El simulador permite evaluar a la NoC en términos de troughput, retraso y el poder consumido. Estos resultados son entregados al usuario en términos de promedio y unitarios.
Siendo más precisos, al usuario le está permitido obtener las diferentes evaluaciones de las métricas de la NoC incluyendo el total número de paquetes/flits recibidos, el promedio global del troughput, el máximo y mínimo retraso global, la energía total consumida, y el retraso/troughput/energía por cada comunicación realizada. [6]

## VI. DESARROLLO

Se desea comparar el rendimiento en cuanto a latencia, troughtput y energía consumida de diferentes arquitecturas NoC y de una nueva propuesta.

El Simulador base Noxim nos entrega los promedios en cuanto a latencia, troughtput y energía consumida de solamente una arquitectura conformada por una red Mesh (Figura 2) y su algoritmo de ruteo.

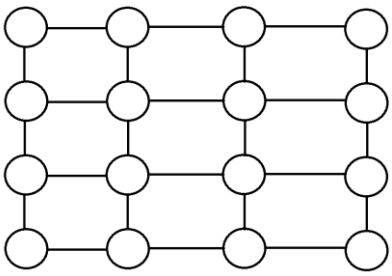


Figura 2. Topología Mesh

Se modificó el código de Noxim para que se pueda realizar la simulación de 2 arquitecturas más. La primera arquitectura posee una red Torus (Figura3) que puede ser probada con dos algoritmos de ruteo: el algoritmo de ruteo de borde que especifica que si su destino se encuentra en el lado opuesto del origen el paquete será enviado por los bordes; y el algoritmo de ruteo completo, que primero calcula la mejor ruta entre el destino y el origen para luego proceder al despacho del paquete.

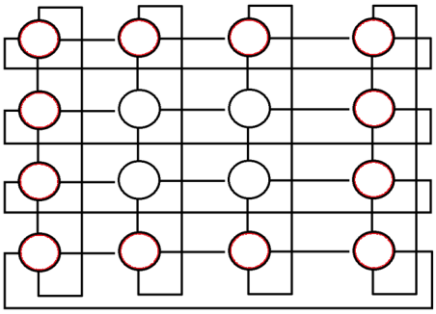


Figura 3. Topología Torus

La segunda arquitectura es una propuesta nueva que posee una red Torus modificada a la que llamaremos TorusS y un algoritmo de ruteo completo encargado de encontrar la mejor ruta entre el origen y el destino (Figura 4).

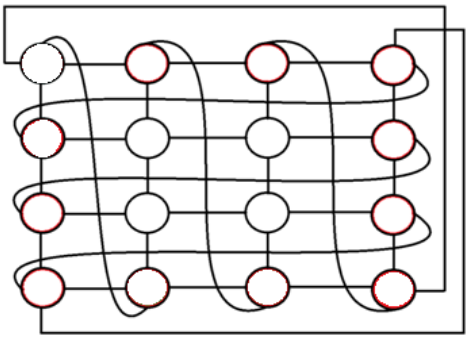


Figura 4. Topología TorusS (nueva propuesta)

## VII. RESULTADOS

La siguiente Tabla 3 muestra todos los parámetros tomados en cuenta para realizar la comparación entre las arquitecturas.

| Topología | Mesh | Torus | Torus | TorusS |
|---|---|---|---|---|
| Dimensión Matriz | 2 a 9 | 2 a 9 | 2 a 9 | 2 a 9 |
| Número de Simulaciones | 1000 | 1000 | 1000 | 1000 |
| Número de Repeticiones | 20 | 20 | 20 | 20 |
| Numero de Flits x Paquete | 3 | 3 | 3 | 3 |
| Tamaño del Buffer | 30 | 30 | 30 | 30 |
| Algoritmo de ruteo | XY | Borde | Completo | Completo |
| Simulación | Parcialmente Controlada | | | |

Tabla 3. Parámetros de Simulación

El primer parámetro a analizar entre las arquitecturas va a ser la latencia Grafico 1.

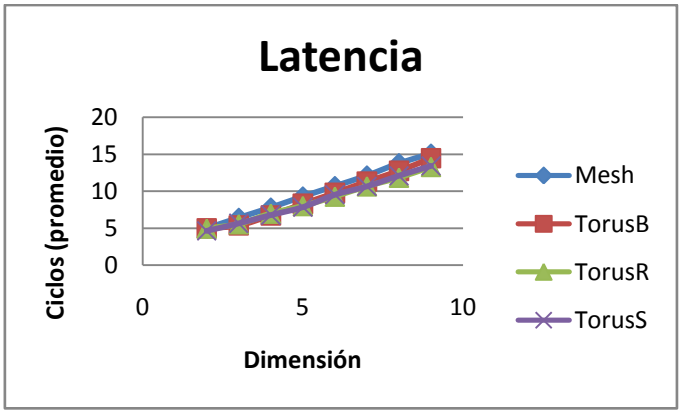


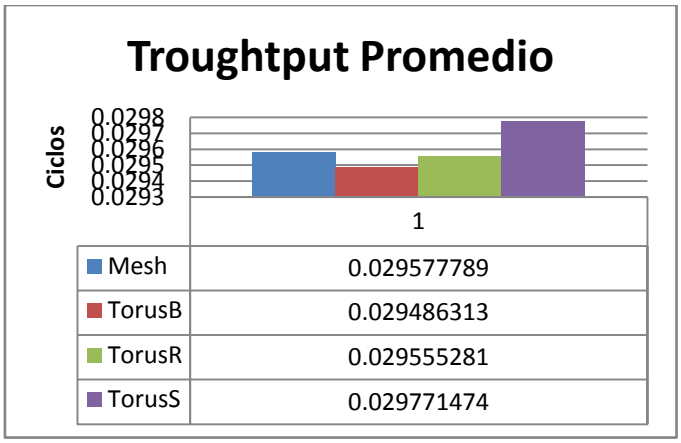

Grafico 1. Latencia promedio (Ciclos / Dimensión)

Como se puede observar claramente tanto la arquitectura TorusB, TorusR y TorusS son superiores a la arquitectura Mesh, puesto que presentan una latencia menor a la hora de despachar los paquetes desde el origen al destino.

Para dimensiones de 6x6 hasta 9x9 se puede determinar que la red TorusR es ligeramente superior a la arquitectura TorusS y superior a las demás arquitecturas.

Como se observa, la dimensión de la matriz tiene un papel importante para determinar qué arquitectura es la mejor. Se puede ver que a una dimensión 2x2 la arquitectura TorusS es ligeramente superior a las demás arquitecturas por la interconexión que presenta la topología de esta arquitectura. En cuanto la dimensión sube a 3x3, la arquitectura que presenta una mejor respuesta es la TorusB. A pesar de que la topología de la TorusB y la TorusR son la misma, no muestran un comportamiento idéntico puesto que quien determina la velocidad del despacho de los paquetes es el algoritmo de ruteo, por lo tanto, a una dimensión 3x3 la TorusB es ligeramente superior a las demás arquitecturas.

El siguiente análisis que se va a realizar es con respecto al troughtput de las arquitecturas Gráfica 2




.Grafica 2. Troughtput Promedio

El troughtput tiene que ver con la velocidad interna del sistema para el despacho de los paquetes, como se puede observar a pesar de estemos ejecutando una simulación en un mismo sistema el factor que determina que arquitectura presenta el menor troughtput es el algoritmo de ruteo, puesto a que genera un retraso en la velocidad del despacho de los paquetes, por lo tanto era de esperarse que la arquitectura TorusS presentará un troughtput mayor a las demás arquitecturas.

Finalmente, el análisis de la energía consumida nos ayudará a determinar cuándo y por qué una arquitectura es superior o no a otra Gráfica 3.

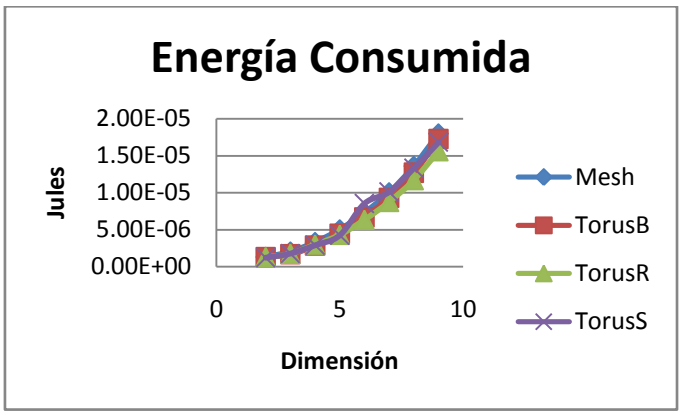



Grafico 3. Energía Consumida (Jules / Dimensión)

La energía consumida en Noxim se la calcula de acuerdo al número de saltos que debe realizar un paquete desde su origen hasta su destino y cuánto tiempo pasa el paquete almacenado hasta ser despachado.

A pesar de ser directamente proporcional la energía con la latencia, se puede observar que la arquitectura TorusS no lo es, esto se debe a que a partir de dimensiones mayores a 6x6 los cálculos que debe realizar el algortimo de enrutamiento son mayores, por lo tanto el paquete pasa más tiempo almacenado, antes de ser despachado lo cual genera un mayor consumo de energía.

## VIII. Conclusión

En conclusión la arquitectura que se desee implementar depende directamente del tamaño de la matriz de la NoC, en tamaños entre 2x2 hasta 5x5 se tiene una respuesta favorable para las arquitecturas TorusR y TorusS, siendo TorusS ligeramente superior a TorusR.

Para dimensiones superiores a 6x6 TorusS presenta un consumo de energía mucho mayor en comparación a las demás arquitecturas lo cual viene a ser un problema a la hora de la implementación, TorusR sin embargo, es superior a las demás arquitecturas en matrices superiores a 6x6.

También antes de implementar una arquitectura, se debe tomar en cuenta que los factores (latencia, troughtput, energía) son críticos para las aplicaciones que van a usar esta arquitectura. Realizar una simulación nos ayuda a tener una idea clara de la respuesta que vamos a obtener y si se encuentra dentro de nuestros requerimientos.

## Referencias


[1] Tsai, W.-C., Lan, Y.-C., Hu, Y.-H., & Chen, S.-J. (2012). *Networks on chips: Structure anda Desing Methodologies.* Wisconsin-Madison: Electrical and Computer Engineering.

[2] Keidar, I., & Cidon, I. (2010). *Zooming in on* . Springer-Verlag Berlin: Proceedings of the 16th international conference on Structural Information and Communication Complexity.

[3] Kumar, S., Jantsch, A., & Soininen, J. (2000). *Network-on-chip architecture and design methodology*. Proceeding of the Internacional Symposium on Very Large Scale Integration.

[4] Achballah, A. B., & Saoud, S. B. (2011). A Survey of Network-On-Chip Tools. *National Institute of Apllied Sciences and Technology, Dept. of Electrical Engineering*, Vol. No.2.

[5] *NS-2 website*. Available: nsnam.isi.edu/nsnam/index.php/Main_Page

[6] *Noxim website*. Available: www.noxim.org

[7] SunFloor3D. (08 de 11 de 2012). *SunFloor 3D: A Tool for Networks on Chip Topology Synthesis for 3D Systems on Chips*. Recuperado el 15 de 07 de 2013, de SunFloor 3D: A Tool for Networks on Chip Topology Synthesis for 3D Systems on Chips: http://infoscience.epfl.ch/record/150054

[8] Orion tool website. Available: www.princeton.edu/~peh/orion.html

[9] Simulator, B. I. (12 de 02 de 2013). *http://nocs.stanford.edu/cgi-bin/trac.cgi/wiki/Resources/BookSim*. Recuperado el 10 de 07 de 2013, de http://nocs.stanford.edu/cgi-bin/trac.cgi/wiki/Resources/BookSim: http://nocs.stanford.edu/cgi-bin/trac.cgi/wiki/Resources/BookSim

[10] ENGINEERING, E. &. (15 de 01 de 2013). *http://www.ece.cmu.edu/~sld/wiki/doku.php?id=shared:worm_sim*. Recuperado el 18 de 07 de 2013, de http://www.ece.cmu.edu/~sld/wiki/doku.php?id=shared:worm_sim: http://www.ece.cmu.edu/~sld/wiki/doku.php?id=shared:worm_sim

[11] NIRGAM. (04 de 08 de 2012). *http://nirgam.ecs.soton.ac.uk/*. Recuperado el 13 de 07 de 2013, de http://nirgam.ecs.soton.ac.uk/: http://nirgam.ecs.soton.ac.uk/

[12] Environment, N. T. (27 de 10 de 2009). *http://www.ict.kth.se/nostrum/NNSE/*. Recuperado el 13 de 07 de 2013, de http://www.ict.kth.se/nostrum/NNSE/: http://www.ict.kth.se/nostrum/NNSE/