

Cosas a Considerar en NoCs

David Davó Laviña

21 de enero de 2024



Objetivos de nuestra NoC

El objetivo es conectar distintas unidades funcionales dentro de un procesador, por lo que:

- Cuanto menos área ocupen los encaminadores, mejor
- Cuanto menos energía consuman los encaminadores, mejor
- Intentar minimizar la latencia (TODO: Buscar la latencia de la conexión actual del Swerv)
- Intentar mantener el ancho de banda (TODO: Buscar el ancho de banda efectivo necesario en el actual Swerv)
- ¿Necesitamos corrección/detección de errores?



1 Enlaces

- Sentido de transmisión
- Segmentación

2 Direccionamiento

- Broadcasting y Multicasting:w

3 Encaminamiento

- Switching
- Determinista o adaptativo
- Desde la fuente o desde los nodos

4 Características de la red

- Topología
- Dentro de los nodos



Sentido de transmisión

Half-duplex

- **Ventaja:** Usamos la mitad de cable
- Necesario crear algún tipo de arbitraje (más área gastada)
- Pueden producirse colisiones
- Mayor latencia y menor throughput en general
- El objetivo de usar una NoC es quitarnos el medio compartido

Full-duplex

- En el silicio, los cables son lo más barato y fácil de implementar
- Al no ser necesario arbitraje, nos ahorramos área (y tiempo diseñando)

Conclusión

Usaremos full-duplex



- Podríamos segmentar los enlaces para aumentar el throughput reduciendo el tiempo de ciclo

Conclusión

Dependiendo del ciclo de reloj objetivo y la latencia de los enlaces implementados, se segmentarán o no los enlaces.



¿Necesitamos **broadcasting**? ¿Multicasting?

- Depende de para qué usemos la red
- Dependiendo del direccionamiento usado, el broadcasting no debería ser muy difícil de implementar si es necesario
- Para multicasting, es necesario ampliar el espacio de direcciones considerablemente
- (En broadcasting, solo es necesario una dirección)



Tipo de Switching I

- **Store-and-forward**

- No proporciona a penas ventajas en nuestro contexto
- Hay que almacenar los paquetes enteros, por lo que necesita demasiada área

- **Cut-through**

- Sigue necesitando búfers para almacenar el paquete cuando se produce una contención
- Menos latencia
- Degrada a store-and-forward en routing no adaptativo

- **Wormhole**

- Se parten los paquetes en *flits*. Se manda un flit cabecera que reserva el camino, y uno cola que lo libera. Solo hay que guardar el flit (no todo el paquete) en caso de contención.
- Se puede mitigar la contención (y evitar deadlocks) con canales virtuales



Tipo de Switching II

- **Conmutación de circuitos**

- Se establece un circuito antes de comenzar la transmisión (más latencia que wormhole)
- El circuito está reservado, por lo que se aprovecha el ancho de banda al máximo en esa conexión (pero no lo pueden usar otras conexiones)

Conclusión

Usamos wormhole (o conmutación de circuitos si vemos que no funciona)



¿Determinista o adaptativo?

Determinista

- Fácil de implementar (Ej: DOR)
- No hay **livelocks**
- Para solucionar el **starvation** podemos usar RR
- Pueden evitarse deadlocks con algún esquema concreto

Adaptativo

- Necesitamos almacenar de alguna manera el estado de la red → buffers → más área
- Necesitamos comunicar el estado de la red → gossiping protocols
- Hay que tratar los **livelocks**

Conclusión

Usar un algoritmo determinista, aunque puede merecer la pena buscar un algoritmo adaptativo sencillo



¿Encaminamiento en la fuente o en los nodos?

En la fuente

- La interfaz de cada elemento necesita conocer la red
- (Y su estado en el caso de encaminamiento adaptativo)
- Buena idea si quieres adaptativo con rúters ligeros (pero NIC grandes)

En cada nodos

- Cada rúter necesita conocer la red
- (Y su estado en el caso de encaminamiento adaptativo)
- Con encaminamiento determinista puede calcularse el siguiente salto con una operación aritmética combinacional o una pequeña ROM

Conclusión

Encaminamiento en los nodos



- Toro 2D y cada nodo tiene 5 enlaces: $NSEO + local$
- Malla/Toro 2D y los nodos tienen 4 enlaces
 - En malla, los de los márgenes conectan con dispositivos
 - En toro, algunos conectan con dispositivos y tres direcciones, y otros con NSEO (sin disp)

Conclusión

TBD



Contención en los nodos







- **No bloqueante** → crossbar 4x4 o 5x5
- **Bloqueante** → red MIN u otra cosa (menos transistores)

¿Y el arbitraje?

Si dos o más entradas distintas quieren ir a la misma salida, podemos usar round robin.





Referencias I

-  Krste Asanović y David A Patterson. *Instruction Sets Should Be Free: The Case For RISC-V*. Inf. téc. 2014. URL:
<http://www.eecs.berkeley.edu/Pubs/TechRpts/2014/EECS-2014-146.html>.
-  John L. Henessy y David A. Patterson. «Interconnection Networks». En: *Computer architecture: a quantitative approach*. 6.^a ed. ISBN: 9780128119068.
-  Axel. Jantsch y Hannu. Tenhunen. *Networks on chip*. Kluwer Academic Publishers, 2003. ISBN: 0306487276.
-  Giovanni De Micheli y Luca Benini. *Networks on Chips. Technology and Tools*. 1st. Elsevier, 2006. ISBN: 978-0-12-370521-1.
-  Mark Hayden y Ken Birman. «Probabilistic Broadcast». En: (sep. de 1996).
-  Tony Chen y David A Patterson. *RISC-V Geneology*. Inf. téc. 2016. URL:
<http://www.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-6.html>.



Referencias II

-  Maryam Kamali y col. «Towards correct and reusable Network-on-Chip architectures». En: *Modeling and Simulation of Computer Networks and Systems: Methodologies and Applications* (abr. de 2015), págs. 357-392. DOI: 10.1016/B978-0-12-800887-4.00012-2.
-  Zhipeng Zhao y James C. Hoe. «Using Vivado-HLS for Structural Design: a NoC Case Study». En: (oct. de 2017). URL: <http://arxiv.org/abs/1710.10290>.

