Implementación de Redes de Transputers en SOPCs

Daniel Horacio Simonelli
INCA/INTIA – Facultad de Ciencias Exactas – UNCPBA
Paraje Arroyo Seco, Campus Universitario – Tandil, Argentina
TE: +54 2293 432466 e-mail: dsimonel@exa.unicen.edu.ar

RESUMEN: *Las redes de alta velocidad constituyen una parte esencial de los sistemas de telefonía pública y privada como así también de los sistemas de comunicación entre computadoras. Desde hace ya poco más de una década se vienen usando redes soportadas en sistemas electrónicos basados en placas, chips y hasta en porciones de chips.*
Los avances registrados en los dispositivos de lógica programable permiten lograr implementaciones de estos sistemas en un chip programable (SOPCs: Systems On a Programmable Chip) aprovechando la simplicidad conceptual y las altas prestaciones de los transputers y de los routers. En este trabajo, se discute la aplicación de las ideas presentadas en [3] para la creación de arquitecturas de máquinas concurrentes basadas en transputers.

PALABRAS CLAVES: *Transputers, Routers, SOPCs, TOPCs*

1. INTRODUCCIÓN

Se presenta a continuación una sucinta descripción de los conceptos centrales a la tecnología de transputers y routers.

1.1. TRANSPUTERS

Los transputers son microcomputadoras completamente integradas en un único chip VLSI que poseen una determinada cantidad de vínculos de comunicación lo que les permite la interconexión para construir sistemas de procesamiento concurrente.
El conjunto de instrucciones del transputer es un RISC especializado en enviar y recibir mensajes por medio de estos vínculos lo que minimiza las demoras en la comunicación inter-transputers.
Los transputers se pueden conectar directamente entre sí para formar redes especializadas o se pueden interconectar a través de chips de ruteo específicos que pueden soportar una gran cantidad de mensajes con elevado throughput y baja demora [1].

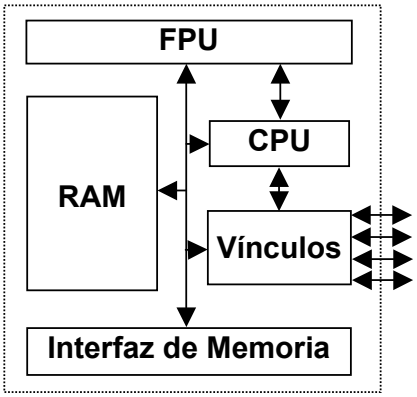

Fig. 1 - *El Transputer T800*

Por ejemplo, el INMOS T800 (Fig. 1) integra un procesador central, una unidad de punto flotante, una RAM de 4 KB más una interfaz para una memoria externa y un sistema de comunicación en un único chip [2]. Como microcomputadora es inusual en el sentido de que tiene la habilidad de comunicarse con otros transputers por medio de sus canales de comunicación lo que le permite

conectarse entre sí para construir sistemas a medida de problemas específicos. También es inusual por cuanto tiene la capacidad de ejecutar muchos procesos de software concurrentes, crear nuevos procesos rápidamente y comunicarlos dentro de un transputer y entre diferentes transputers de forma muy eficiente.
El lenguaje de programación Occam [4, 5] inspirado en el formalismo Communicating Sequential Processes (CSP) propuesto por Hoare [6, 7], permite que una aplicación sea expresada como una colección de procesos concurrentes que se comunican a través de canales. Cada canal es una conexión punto-a-punto entre dos procesos, uno que siempre toma del canal y el otro que siempre transmite por él. La comunicación siempre es sincrónica; el primer proceso listo para transmitir espera hasta que el otro también esté listo y entonces se copia el dato del proceso que transmite al proceso que recibe y ambos procesos continúan.
Cada transputer dispone de un scheduler que le permite compartir su tiempo entre varios procesos. La comunicación entre procesos dentro del mismo transputer se lleva a cabo utilizando la memoria local; la comunicación entre procesos sobre diferentes transputers se lleva a cabo utilizando un vínculo entre los dos transputers.

1.2. VÍNCULOS (CONEXIONES)

Los vínculos consisten realmente de puertos DMA serie por lo que se los puede utilizar para conectar a los transputers con periféricos a través de ASICs capaces de leer y escribir en la memoria del transputer a alta velocidad o, más habitualmente, para conectarse con otro procesador, generalmente otro transputer.
Los diferentes vínculos y el procesador disponen de accesos independientes a la memoria y están diseñados de forma tal que no necesitan sincronización para poder comunicarse a condición de que compartan la misma velocidad de transferencia de datos. De aquí que los transputers que conforman una red pueden correr a diferentes velocidades mientras se comuniquen a velocidad constante.
Los vínculos son punto a punto e involucran dos canales unidireccionales que se comunican a través de los pines *LinkIn* y *LinkOut*.

1.3. ROUTERS

Hay muchos algoritmos paralelos en los cuales el número de canales de comunicación entre procesos de diferentes transputers es mucho más grande que el número de conexiones físicas disponibles para interconectar transputers.
En estos casos se suele combinar a los transputers con dispositivos conocidos como routers que se encargan de gestionar eficientemente las redes de comunicación pues disponen de muchas conexiones (por ej. 32) lo que permite construir grandes redes de transputers con demoras mínimas.
Cada router utiliza el encabezado del paquete que llega a cada conexión para determinar que conexión utilizar para transmitir el paquete tan pronto como le es posible y, además, arbitra entre varios paquetes que deben utilizar la misma salida y los serializa (Fig. 2).

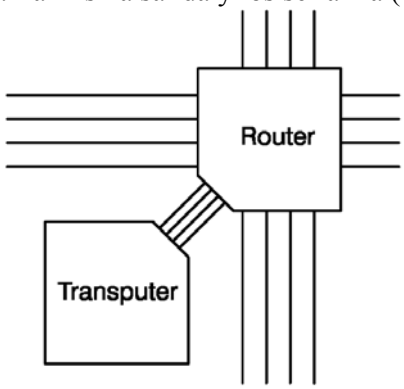

Fig. 2 – *Nodo que combina un transputer y un router*

2. TRANSPUTER ON A PROGRAMMABLE CHIP (TPOC)

Siguiendo el esquema presentado en [3], una red de transputers (RT) se puede describir como una estructura jerárquica de transputers, procesos, canales (routers) y puertos. A través del empleo de un HDL estándar para descripción a nivel de sistemas se puede obtener la generación automática (síntesis) del diseño final.

Una RT se construye a partir de un conjunto de transputers, puertos de E/S y conexiones o routers. Desde el punto de vista del HDL, la descripción del SOPC incluirá una enumeración completa de puertos de E/S, múltiples instancias de transputers (no necesariamente iguales pues los diferentes modelos son completamente compatibles), conexiones simples o routers y las interconexiones entre los transputers, o entre estos y los puertos de E/S.

En VHDL, el SOPC se puede definir de la siguiente manera:

```
LIBRARY tpoc; USE tpoc.sopc.all;
ENTITY Mi_Spoc IS
  GENERIC (pin_mapping: string:= "IN_a:23, ..., IN_m:12,"&"OUT_a:55, ..., OUT_c:60");
  PORT (-- seniales globales
           clk, clr: IN STD_LOGIC;
          -- E/S al exterior
           ....: IN     STD_LOGIC;
           ....: OUT STD_LOGIC;
           .....................................);
END ENTITY Mi_Spoc;

ARCHITECTURE arq OF Mi_Spoc IS
  SIGNAL link1: Paralelo8S;
  .....................................
  SIGNAL linkm: Serie4S;

BEGIN
   transputer_1: ENTITY  work.T800_1 PORT MAP (...);
   .....................................
   transputer_n: ENTITY  work.T425_1 PORT MAP (...);
   router_1: ENTITY work.C104_1 PORT MAP (...);
   .....................................
   router_d: ENTITY work.C104_6 PORT MAP (...);
   conexion_1: ENTITY tpoc.chpp8s1 PORT MAP (...)
   .....................................
   conexion_s: ENTITY tpoc.chpp8s1 PORT MAP (...)
END ARCHITECTURE arq;
```

Se entiende que, a este nivel el diseño es completamente estructural. Las diferentes instancias de los transputers y routers consisten de definiciones parametrizadas que están presentes en la biblioteca *work*, y las conexiones simples son objetos tpoc estándares cuya descripción se encuentra en la biblioteca *tpoc*.

Los procesos se identifican a través de su nombre, una descripción de los canales que utilizan, una enumeración de puertos de E/S y un código a ejecutar.

Como la arquitectura de los transputers fue diseñada como un motor Occam, los procesos pueden ser de cinco tipos:

- Asignación x := y + 2
- Input keyboard ? char
- Output screen ! char
- Skip SKIP -- NOP que termina
- Stop STOP -- NOP que nunca termina

Los procesos se asignan a transputers específicos por lo que es necesario proveer una transformación de la descripción del código fuente a un formato sintetizable. La solución propuesta en [3] consiste de incluir directivas de pre-compilación que pueden exportarse automáticamente desde el código fuente a las diferentes instancias de los transputers en VHDL. Otras alternativas tienden a compilar directamente el código occam en descripciones HDL sintetizables [5].

3. SÍNTESIS DEL TPOC

La definición de un TPOC involucra cuatro pasos principales:

- Instanciación de transputers
- Instanciación de routers
- Definición de conexiones
- Definición de Procesos

En la Fig. 3 se observa un ejemplo de TPOC con cuatro routers y cuatro transputers, cada uno de los cuales tiene una diferente cantidad de procesos a ejecutar. Las conexiones son manejadas por los routers y la E/S por los transputers a través de los procesos que en ellos se ejecutan.

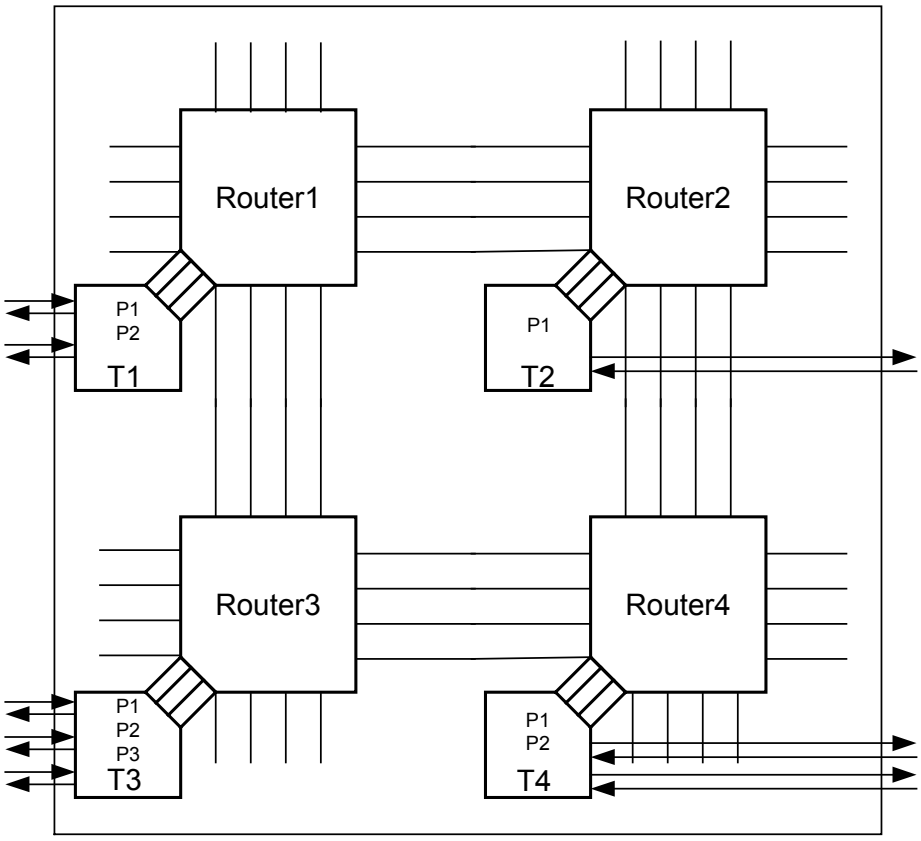

Fig. 3 – *Esquema del TOPC*

4. CONCLUSIONES

La propuesta presentada permite aprovechar la simplicidad de diseño que proveen los transputers para la implementación de sistemas cuya definición involucra procesos altamente concurrentes (redes de telefonía, procesamiento de imágenes, etc.) sumada a la potencia de los dispositivos de Field Programmable Logic (FPL).

Las características del lenguaje occam también permiten efectuar consideraciones adicionales que pueden redundar en un diseño más compacto [8, 9].

Uno de los aspectos principales que debe tenerse en cuenta es que la caracterización de los transputers y routers como núcleos IP (IP-cores) permite una estandarización que independiza a la herramienta de generación automática de la plataforma de diseño utilizada, a la vez que facilita la utilización de estructuras parcialmente homogéneas.

No sólo se está restringido a la utilización de modelos de transputers convencionales. Es posible definir un conjunto completo de transputers cuya arquitectura enfatice aspectos diferentes del conjunto de instrucciones provisto. Eventualmente, se pueden definir transputers cuyos microprocesadores utilicen un conjunto de instrucciones completamente nuevo. La generación de código de bajo nivel desde occam no presenta un problema pues ya han sido desarrollados compiladores reconfigurables que permiten generar código C desde occam a partir del cual es posible portar al assembler del transputer en cuestión [10, 11, 12].

5. REFERENCIAS

[1] D. May, W. Thompson and H. Welch, "*Networks, Routers and Transputers: Function, Performance and applications*", IOS Press Inc., ISBN 90-5199-129-0, USA, 1993.

[2] M. Homewood, D. May, D. Shepherd, "*The IMS T800 Transputer*", IEEE Micro 7, N.5, October, 1987.

[3] G. Jaquenod, M. De Giusti, H. Villagarcía, "*Towards a Field Configurable non-homogeneous Multiprocessors Architecture*", Proceedings of the 5th World Multi-conference on Systemics, Cybernetics and Informatics SCI2001, July 2001, Orlando, FL, USA.

[4] INMOS Ltd., "*occam2 reference manual*", Prentice Hall, 1998

[5] B.M. Cook, R. Peel "*Occam on Field-Programmable Gate Arrays - steps towards the Para-PC*" in "Architectures, Languages and Techniques for Concurrent Systems" (proceedings of the 22nd Technical Meeting of the World Occam and Transputer User Group) pp. 211-228, pub. IOS, Amsterdam (ISBN 90-5199-480-X), April 1999.

[6] C.A.R. Hoare, "*Communicating Sequential Processes*", Communications of the ACM 21(8): 666-677, August 1978.

[7] C.A.R. Hoare, "*Communicating Sequential Processes*", Prentice Hall International Series in Computer Science, ISBN 0-13-153271-5 (0-13-153289-8 PBK), 1985.

[8] G. Jones, "*On Guards*", Parallel Programming of Transputer Based Machines, ed. Traian Muntean, IOS, September 1987.

[9] G. Jones, "*Measuring the busyness of a transputer*", Occam user group newsletter no.12, January 1990.

[10] "*Kent Retargetable Occam Compiler (KRoC)*", disponible en http://wotug.org/kroc/

[11] "*The Southampton Portable Occam Compiler*", disponible en http://wotug.org/kroc/

[12] "*SGS-Thompson / INMOS donated compilers*", disponible en http://wotug.org/kroc/