Metodologia de Projecto de SoC Configuráveis Baseados em Redes Intra-Chip

Mário P. Véstias
ISEL-DEETC
mvestias@deetc.isel.ipl.pt

O aumento constante da densidade de integração e do desempenho dos circuitos integrados reconfiguráveis permitiram a implementação num único integrado de sistemas capazes de suportar os requisitos bastante exigentes dos sistemas embebidos nas áreas de multimédia e de telecomunicações.
Além disso, a flexibilidade destes sistemas permite a utilização de metodologias de projecto baseadas em plataforma com reutilização de IP, com a consequente redução do tempo para o mercado.
Neste trabalho, é proposta uma metodologia de co-síntese para o projecto de plataformas SoC configuráveis com uma arquitectura de rede intra-chip para execução de sistemas embebidos predominantemente de processamento de dados.
A metodologia foi testada com o projecto de um algoritmo de compressão de imagem em FPGA que atingiu uma taxa de processamento de 800Mbps. Os resultados são bastante promissores, uma vez que foi possível integrar facilmente vários núcleos IP num único circuito reconfigurável e obter soluções de elevada qualidade.

Introdução

O hardware reconfigurável tem sofrido um aumento constante da frequência de trabalho e da capacidade lógica, permitindo a implementação de sistemas complexos mais rápidos num único integrado, tornando-os numa solução competitiva para sistemas embebidos. O projecto destes integrados requer novas arquitecturas e metodologias que melhorem a produtividade e que reduzam a complexidade de projecto. Uma abordagem possível consiste em reutilizar blocos hardware e software. Gajski et.al. [1] propuseram uma metodologia de projecto de sistemas embebidos centrada em IP e Vahid et al. [2] uma outra baseada em plataforma que permite não só a reutilização de componentes, mas também a reutilização de arquitecturas e topologias.

Há várias propostas de arquitecturas para plataformas hardware direccionadas para os SoC futuros, com particular ênfase em garantir estruturas de comunicação eficientes para a ligação de vários recursos num único integrado, como a estrutura de redes intra-chip (NoC - *Network-on-Chip*) [3], [4], [5]. A NoC surgiu como um novo paradigma de interligação capaz de integrar muitos núcleos com uma largura de banda elevada.

Existem vários trabalhos com contributos na área das NoC em reconfiguráveis. Marescaux [6] implementou um toro bidireccional numa FPGA Virtex que suporta até 320Mbits/s a 40MHz com qualidade de serviço (QoS). O HERMES [7] é uma topologia em malha implementada numa FPGA VirtexII que suporta até 500 Mbits/s a 25MHz sem QoS.

Para implementar arquitecturas baseadas em NoC, são necessárias várias ferramentas de co-síntese para escolher a melhor configuração de plataforma e para mapear aplicações sobre a arquitectura NoC. Existe muito trabalho em co-síntese de arquitecturas baseadas em barramentos [8], [9], [10]. Como tal, podem-se adaptar muitas destas técnicas e ideias no desenvolvimento de NoC. Uma vez que o projecto de NoC é uma nova área de investigação, existem apenas algumas abordagens de mapeamento e escalonamento [11] e [12].

Neste artigo, propõe-se uma metodologia de co-síntese para o desenvolvimento de SoC baseados numa infra-estrutura NoC configurável para a execução de aplicações dominadas por fluxo de dados, como são as aplicações multimédia e de telecomunicações. Um exemplo multimédia foi simulado e implementado numa FPGA Virtex II XC2V6000.

Plataforma SoC Configurável

A plataforma SoC configurável consiste numa matriz de blocos interligados por uma NoC. A NoC consiste numa matriz de *routers* (R) interligados entre si, em que um *router* está ligado no máximo a quatro vizinhos e a um núcleo IP local. De entre as muitas topologias de interligação, optou-se pela topologia em malha, pois uma rede 2D encaixa naturalmente num integrado FPGA 2D (ver figura 1).

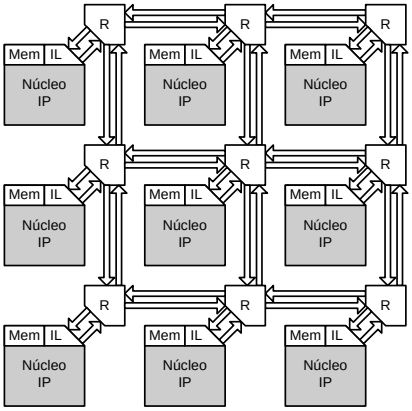

Figura 1. Arquitectura SoC

Um bloco consiste num núcleo IP, memória local e uma interface de ligação (IL). Um IP é um bloco hardware ou software com acesso a memória local que pode comunicar com outros IP através da NoC. A ligação entre um núcleo e o respectivo *router* é feita pela IL. A plataforma comunica com o exterior através de núcleos que implementam um determinado tipo de interface. O comportamento do processo de comunicação segue uma abordagem por camadas similar ao modelo da arquitectura de comunicação OSI.

Projecto do Router

Um *router* encaminha pacotes entre núcleos IP. Por cada pacote recebido, lê o endereço destino e envia-o para a saída correcta. O *router* implementado consiste num conjunto de portos de entrada e saída, um *buffer* centralizado, controladores de entrada e de saída, um árbitro e um bloco de encaminhamento (ver figura 2).

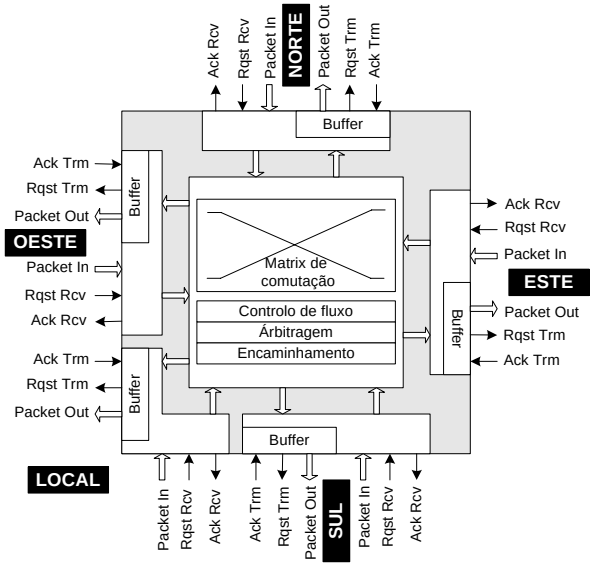

Figura 2. Arquitectura de um *router* da NoC

Um porto garante a fiabilidade da comunicação com um protocolo de controlo de fluxo ponto-a-ponto de duas vias. O controlador de entrada determina o porto de saída do pacote e gera um sinal de pedido. O con-

trolador de saída usa um esquema circular para arbitrar entre pedidos simultâneos e retorna um sinal de concessão da saída a uma das entradas com pedido activo. A estratégia de encaminhamento usada pelo controlador de entrada baseia-se no algoritmo XY [7].

Projecto da Interface Lógica

A IL consiste num controlador de entrada e outro de saída, memória partilhada para ligação ao núcleo, um porto de ligação ao *router* e memória de OS (ver figura 3).

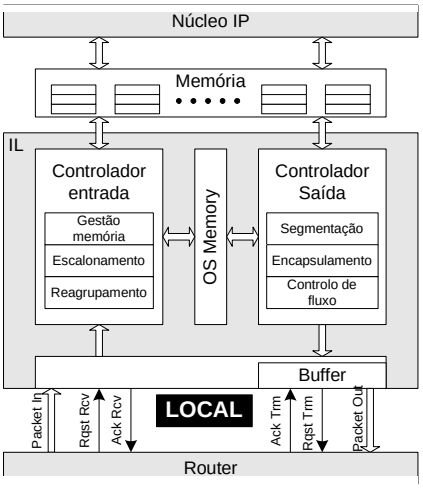

Figura 3. Arquitectura de uma interface lógica da NoC

A interface entre a IL e o *router* é idêntica à usada entre *routers*. A interface entre o núcleo e a IL é feita com memória partilhada, que pode ser RAM de duplo porto ou FIFO. O controlador de entrada recebe os dados do *router* e envia-os para a memória partilhada numa posição que depende do porto destino da tarefa.

No caso de dados segmentados em múltiplos pacotes, o controlador de entrada guarda todos os pacotes em memória e só depois indica ao núcleo que os pode processar. O controlador de saída lê os dados da memória partilhada e envia-os na forma de um ou mais pacotes. O endereço e o porto destino, bem como a prioridade do pacote, são obtidos da memória de OS. O controlador de saída também implementa um controlo de fluxo extremo-a-extremo, para garantir que não se perdem dados entre tarefas.

Avaliação do Desempenho da Plataforma

A plataforma foi analisada para determinar parâmetros de caracterização do seu desempenho (ver tabela 1):

Latência da ligação (LL) – o tempo de *transição de um pacote entre* saídas de dois *routers* vizinhos;

Latência de geração (LG) – o tempo para a IL gerar um pacote;

Latência de recepção (LR) – o tempo para a IL receber um pacote;

Largura de banda entre recursos (LBRR) – taxa de transmissão entre núcleos IP.

LL	LG	LR	LBRR
1 ciclo	4 ciclos	5 ciclos	F(frequência)/5 Pacotes/s

Tabela 1. Caracterização da NoC

A partir destes parâmetros, adoptou-se um modelo simples de determinação dos atrasos associados à comunicação de pacotes. O atraso de comunicação de um pacote entre dois núcleos separados por NR *routers*, *EdgeDelay*, é dado por $\frac{LG + LR + LL \times NR}{f(\text{frequência})}$.

Uma vez que os pacotes são armazenados nos *routers*, a transmissão de vários pacotes pode usar pipeline.

Neste caso, o atraso de transmissão de NP pacotes, *EdgeDelaypipe*, é dado por $\frac{1}{LBRR} \times (NR + NP)$.

Para além do desempenho, também se caracterização a área ocupada pelo *router* e pela IL quando sintetizados para FPGA (ver tabela 2).

Componente	**Tamanho (*slices*)**	**% XCV6000**
Router (8 bits)	189	0,56
NI	121	0,36

Tabela 2. Área em *slices* dos componentes da NoC

Um *router* consegue encaminhar 5 pacotes/ciclo de relógio a 150 MHz. Como tal, encaminha até 6 Gbps.

Metodologia de Co-síntese

A configuração da arquitectura é feita com base numa metodologia de co-síntese baseada em plataforma. A metodologia determina automaticamente uma arquitectura que execute a aplicação com optimização de desempenho sem violar qualquer restrição de projecto (ver figura 4).

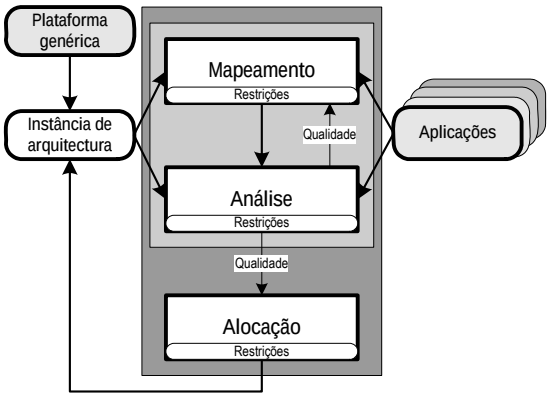

Figura 4. Fluxo de Co-síntese

Partindo de uma instância de arquitectura, a ferramenta mapeia a aplicação e analisa a qualidade da solução. Os resultados da análise, juntamente com as restrições, guiam o processo de exploração da arquitectura.

As aplicações são modeladas com um grafo de fluxo de dados iterativo. O modelo é um grafo directo cíclico G = (V, E), em que cada vértice v ∈ V representa uma tarefa atómica e um ramo e ∈ E representa uma dependência inter ou intra tarefas. Um vértice tem associado o tempo de execução, a memória de dados e de programa ocupada e a área em hardware. Um ramo tem associado o tamanho dos dados.

O passo de **alocação** é responsável por determinar a arquitectura mais apropriada a partir da arquitectura SoC genérica. Determina o tamanho da topologia e o tipo de núcleo associado a cada bloco.

Sendo um problema difícil, adoptou-se uma heurística. A partir da plataforma SoC genérica, o algoritmo cria um conjunto inicial de núcleos IP. O conjunto inicial tem de suportar a execução de todas as tarefas. A partir deste conjunto, gera uma arquitectura inicial. De seguida, mapeia a aplicação e determina a sua qualidade com a ferramenta de análise. Se a qualidade for considerada aceitável, o algoritmo pára. Caso contrário, altera o núcleo de um bloco e repete todo o processo. O processo iterativo é controlado pelo algoritmo *simulated annealing* [13] .

Para cada arquitectura é executado o algoritmo de **mapeamento** que atribui tarefas, transferências e variáveis a elementos arquitecturais (núcleo, ligações e memória) com o objectivo de maximizar a qualidade da arquitectura tendo em conta as restrições de projecto. Mesmo em pequenas instâncias, o problema de mapeamento tem complexidade exponencial, pelo que também se usou uma heurística.

O algoritmo de mapeamento usa mais uma vez o *simulated annealing* juntamente com o escalonamento por lista, tirando partido da técnica de *pipeline* para aumentar a taxa de produção de dados.

Para cada mapeamento sobre uma arquitectura, é executado o passo de **análise** para determinar a qualidade da arquitectura com base no seu desempenho e requisitos de memória. O desempenho é calculado com base no escalonamento por lista e os requisitos de memória correspondem ao número de blocos de memória interna da FPGA (BRAM) usados pelos núcleos e pela IL.

Avaliação do Projecto

Na avaliação da metodologia de projecto, desenvolveu-se um sistema de codificação de imagem sobre a plataforma proposta. O sistema foi implementado numa FPGA Virtex XC2V6000.

Codificador JPEG

O método de compressão de imagem utilizado foi o conhecido JPEG baseado no uso do DCT.

As tarefas da aplicação foram caracterizadas tendo em vista uma implementação hardware (ver tabela 3).

Tarefa	*Slices*	BRAM	Latência (ciclos)
RGB2YCbCr	204	0	64
2D-DCT	1612	1	168
Quantificador	312	1	64
Huffman	176	1	192

Tabela 3. Caracterização de tarefas

Para esta aplicação, a ferramenta de co-síntese demorou menos de cinco minutos para encontrar uma solução hardware capaz de processar dois blocos de [8×8]×24 bits em 3.8 μs (frequência dos núcleos = 100MHz, frequência do NoC = 150MHz), que é equivalente a uma capacidade de processamento de 800 Mbps. Com esta taxa de processamento, é possível comprimir imagens de acordo com os tempos da tabela 4.

Tamanho da imagem	Solução HW	Pentium 4 a 1.7GHz (s)	Aceleração
640×480	0.009 (108 fps)	0.046	5
800×600	0.015 (67 fps)	0,071	4.8
1024×768	0.024 (42 fps)	0,110	4.6

Tabela 4. Tempos de execução para vários tamanhos de imagem

O mapeamento da solução em FPGA encontra-se na figura 5.

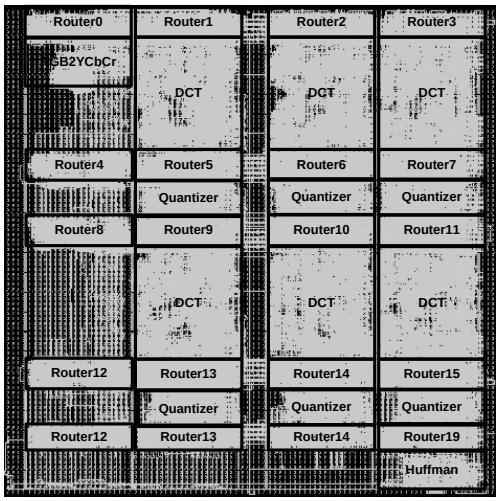

Figura 5. Implementação FPGA do JPEG

O projecto foi facilmente implementado na FPGA devido à sua regularidade e às restrições de mapeamento dos núcleos, dos *routers* e dos blocos de memória. A partir deste estudo conclui-se que:

- Os projectos podem ser implementados rápida e facilmente sem as complicações associadas aos barramentos. O mesmo projecto sobre um circuito baseado em barramento não seria capaz de atingir a mesma taxa de produção.
- Para determinados núcleos, o *router* usa mais *slices*. A área gasta na implementação dos *routers* pode-se tornar um dos problemas associados às arquitecturas NoC. Estão a ser consideradas outras soluções baseadas em memória local e *routers* partilhados.
- Vários parâmetros da NoC podem e devem ser explorados pela ferramenta de co-síntese, incluindo o tamanho das ligações, a capacidade de comutação, o algoritmo de encaminhamento e a política de arbitragem. O ajuste destes parâmetros pode conduzir à redução do tamanho dos *routers*.

Conclusões

O projecto estruturado com a plataforma SoC e os tempos de computação aceitáveis da ferramenta de co-síntese permitem o desenvolvimento rápido de arquitecturas SoC para aplicações dedicadas.

Os resultados são bastante promissores, uma vez que foi possível integrar vários núcleos com alguma facilidade num único circuito configurável e ainda assim obter soluções de elevada qualidade.

Referências

[1] D. Gajski, R. Dömer and J. Zhu, "IP-Centric Methodology and Design with the SpecC Language", in *System Level Design*, Nato Science Series 357, 1999.

[2] F. Vahid and T. Givargis, "Platform Tuning for Embedded Systems Design", in *IEEE Computer,* 34, 3.

[3] W. Dally and B. Towles, "Route Packets, Not Wires: On-Chip Interconnection Networks", in *Proc. of DAC*, 2001.

[4] A. Hemani, A. Jantsch, S. Kumar, A. Postula, J. Oberg, M. Millberg and D. Lindqvist, "Network on Chip: An Architecture for Billion Transistor Era", in *Proceedings of the IEEE NorChip Conference*, Nov. 2000.

[5] L. Benini and G. de Micheli, "Networks on Chips: a New SoC Paradigm", in *Computer*, v.35(1), Jan. 2002, pp.70-78.

[6] T. Marescaux, A. Bartic, D. Verkest, S. Vernalde, R. Lauwereins, "Interconnection Networks Enable Fine-Grain Dyanmic Multi-Tasking on FPGAs", in *Proceedings of FPL*, 2002, pp. 795-805.

[7] F. Moraes, N. Calazans, A. Mello, L. Möller, L. Ost "HERMES: an Infrastructure for Low Area Overhead Packet-Switching Networks on Chip", in *Integration*, the VLSI Journal 38, 2004, pp. 69-93.

[8] R. Dick and N. Jha, "CORDS: Hardware-Software Co-Synthesis of Reconfigurable Real-Time Distributed Embedded Systems", in *Proc. of ICCAD*, pp. 62-68, 1998.

[9] R. Szymanek and K. Kuchcinski, "Design Space Exploration in System Level Synthesis under Memory Constraints", in *Proceedings EuroMicro*, pp. 8-10, 1999.

[10] U. Shenoy, et al., "A System-Level Algorithm with Guaranteed Solution Quality", in *Proceedings of DATE*, pp. 417-422, 2000.

[11] T. Lei and S. Kumar, "Algorithms and Tools for NoC Based System Design", in *Proceedings of SBCCI*, 2003.

[12] D. Shin and J. Kim, "Power-Aware Communication Optimization for Networks-on-Chips with a Voltage Scalable links", in *Proceedings of CODES+ISSS*, pp. 170-175, 2004.

[13] S. Kirkpatrick, C. Gelatt, and M. Vecchi, "Optimization by Simulated Annealing", in *Science*, 220(4598): pp. 671-680, May 1983.