Explorando com Meta-Heurísticas o Espaço de Projeto de Arquiteturas Reconfiguráveis de Grão Grosso[1]

Tiago Teixeira, Ricardo Ferreira
Universidade Federal de Viçosa
tiago.teixeira@ufv.br, ricardo@ufv.br

Joao M. P. Cardoso
Faculdade de Engenharia
Universidade do Porto
jmpc@acm.org

Abstract

Arquiteturas reconfiguráveis de grão grosso são uma alternativa para reduzir o tempo de projeto, a complexidade do posicionamento e roteamento, o tempo de configuração e a memória de configuração para projeto de sistemas embarcados com demanda de alto desempenho e baixo consumo de energia. Porém o espaço de projeto é amplo e necessita de ferramentas flexíveis para sua exploração. Este trabalho propõe uma ferramenta de exploração automática do espaço de projeto baseada em heurísticas (Algoritmos Genéticos, Simulated Annealing, Path Relinking, Escalonamento ALAP e ASAP) para as arquiteturas reconfiguráveis em arranjos com padrões regulares e escaláveis de interconexão.

1. Introdução

O mercado de sistemas embarcados demanda dispositivos móveis com performance e flexibilidade para aplicações com multimídia e transmissão e recepção de dados. Estas aplicações utilizam algoritmos de processamento de sinais que têm um alto grau de paralelismo. Arquiteturas baseadas em fluxo de dados tem se mostrado soluções eficientes para capturar o paralelismo das aplicações [1][2]. A abordagem baseada em fluxo de dados, descentraliza o controle, evita o uso de conexões globais [2], garantindo a escalabilidade da solução. Em um estudo recente, Patterson destaca a importância da escalabilidade das arquiteturas e a flexibilidade das arquiteturas reconfiguráveis [3]. A lei de Moore continua válida e a barreira de um trilhão de transistores está próxima [4]. Porém, novas metodologias e arquiteturas necessitam ser criadas para explorar o potencial da atual tecnologia de sílicio.

Arquiteturas Reconfiguráveis [5] vem surgindo como uma solução atrativa em termos de capacidade, desempenho, baixo consumo de energia e flexibilidade pelas possibilidades de reconfiguração em tempo de execução e recursos de multi-granularidade. Devido a alta complexidade de projeto, a solução adotada pela indústria vem sendo os processadores múltiplos núcleos [9][10][11]. Ao contrário dos processadores com poucos núcleos complexos, os reconfiguráveis são simples e escaláveis.

Arquiteturas reconfiguráveis de grão grosso fornecem como uma solução mais escalável que arquiteturas de grão fino (ex.: FPGA), e se tornaram cada vez mais importantes nos últimos anos [5] [7] [8]. Esta abordagem reduz o tempo e a complexidade de configuração, bem como problemas de tamanho, posicionamento e roteamento, possibilitando uma otimização do tempo de projeto e síntese. No entanto, muitas arquiteturas parecem ser concebidas sem fortes evidências do porque algumas decisões de projeto foram tomadas. Nossa abordagem propõe um ambiente para gerar e avaliar o espaço de projeto de arquiteturas reconfiguráveis em arranjos com padrões regulares de interconexão. Estes padrões locais são escaláveis, o que é fundamental para o aproveitamento das novas tecnologias. Um dos problemas principais de uma arquitetura paralela é a rede de comunicação entre as diversas unidades. Este trabalho aborda as arquiteturas reconfiguráveis e suas redes de conexões.

A próxima seção apresenta os trabalhos correlatos na área de computação reconfigurável para sistemas embarcados. A terceira seção apresenta um ambiente de geração e exploração de

1 Trabalho desenvolvido com financiamento Grices/CNPq

arquiteturas. A quarta seção apresenta os resultados experimentais de geração automática de arquiteturas para aplicações multimídia. Finalmente, na quinta seção serão apresentadas as conclusões deste trabalho.

2. Trabalhos Correlatos

Uma arquitetura comercial da PACT é apresentada em [12]. O usuário pode especificar o número de barramentos, o tamanho das palavras, o número de células, etc. Porém, esse tipo de abordagem é dependente da arquitetura. Os trabalhos apresentados em [1],[13],[14],[15] foram algumas das poucas exceções a explorar as características das arquiteturas.

Por exemplo, o KressArray Xplorer [16] é capaz de gerar sugestões de projetos usando um mecanismo de inferência por lógica difusa (fuzzy).

Outro exemplo é o ambiente ADRES (*Architecture for Dynamically Reconfigurable Embedded Systems*) [1] que apresenta uma abordagem que permite explorar alguns padrões de conexão, distribuição de registros, latência da arquitetura e arranjos heterogêneos.

Apesar de existirem diversas arquiteturas propostas na literatura, para muitas delas não se apresentam justificativas aprofundadas de como suas características foram selecionadas. Uma característica fundamental a ser estudada é o formato dos padrões de conexão dos elementos de processamento. Atualmente, o atraso das conexões vem sendo dominante em relação ao atraso das unidades de cálculo. Além disso, a complexidade de geração das conexões dificulta a estimação do atraso e da energia consumida nas etapas iniciais de alguns projetos. Para tentar achar respostas para alguns desses problemas propomos um ambiente de geração de arquiteturas que será detalhado na próxima seção.

3. Ambiente de Geração das Arquiteturas

Arquiteturas reconfiguráveis de grão grosso variam muito no número e nas características dos elementos computacionais de processamento e roteamento. Como foi anteriormente referido, este trabalho propõe uma ferramenta para explorar o espaço de projeto dessas arquiteturas reconfiguráveis para encontrar, de forma automática, a melhor arquitetura para um dado conjunto de aplicações. A ferramenta é flexível, usando heurísticas e algoritmos evolutivos de otimização na automatização do processo de busca.

Este trabalho não aborda a geração dos grafos de fluxo de dados para os núcleos de código que dominam o tempo de execução das aplicações. No ámbito deste trabalho supõe-se um conjunto de aplicações previamente compilado, onde para cada aplicação, um grafo do fluxo de dados (Dataflow) foi extraído usando técnicas propostas em [2]. O grafo é descrito em um formato XML, e pode ser acoplado a outras ferramentas de extração.

A Fig. 1 mostra um diagrama de blocos da ferramenta. A ferramenta possui dois fluxos principais: baseado em arquitetura e baseado em aplicações.

O fluxo baseado em arquitetura, supõe um conjunto inicial de arquiteturas propostas pelo usuário ou geradas por outras ferramentas. O formato de entrada é um grafo e permite modelar qualquer tipo de arquitetura. A ferramenta faz uso dos algoritmos de otimização: Algoritmo Genético, o Path Relinking e Simulated Annealing. Os algoritmos buscam a melhor solução para população inicial de arquiteturas. Para avaliar uma arquitetura, o conjunto de aplicações é mapeado na arquitetura, e é usado um criterio de custo (neste trabalho utilizamos o total de interconexões utilizadas como critério de custo). Os algoritmos de otimização irão modificar de forma automática este conjunto, e a cada passo, as arquiteturas são reavaliadas, retirando as menos adequadas, e criando novas arquiteturas até encontrar um subconjunto ou uma arquitetura mais adequada para mapear as aplicações.

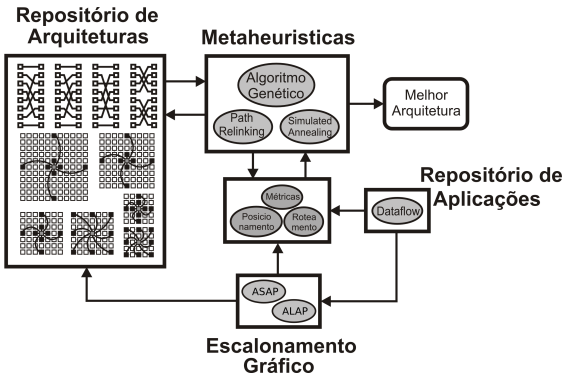

Figura 1. Ambiente de geração de arquiteturas

Para a abordagem baseada nas aplicações, apresentamos uma heuristíca simples para criar uma arquitetura a partir do conjunto de aplicações. Usando os algoritmos de escalonamento ASAP (*As Soon As Possible*) e ALAP (*As Later As Possible*) é gerado um vetor de distâncias entre os vértices dos grafos de fluxos de dados. A partir desde vetor, a ferramenta constroí uma arquitetura.

3.1. Baseado nas Arquiteturas

Nesta seção iremos detalhar o fluxo de geração baseado em arquiteturas. Para este trabalho

consideramos arquiteturas matriciais em 2D. No entanto, o nosso modelo de arquitetura é um grafo e pode ser utilizado na representação de arquiteturas n-dimensionais. Cada arquitetura é composta por um conjunto de Elementos de Processamento (EP), e.g., ULA, MUX, contadores, células de memória, etc. e um padrão de conexão. O padrão de conexão é composto por um conjunto de conexões diretas entre dois EPs. Vamos considerar que todos os EPs dispõem de um número exclusivo de identificação (ID). Este número pode ser um endereço da linha e da coluna na matriz. O conjunto de arquiteturas utilizado para construir a população inicial consistirá em uma seleção das arquiteturas mais utilizadas [5] [8] e arquiteturas com padrões aleatórios. Além disso, propomos a utilização de um conjunto de topologias baseadas em operações de bits.

As arquiteturas mais utilizadas são as compostas por padrões de interconexão local tais como as: em grade, hexagonal, octal e 1-hop como ilustrado na Fig. 2. A arquitetura 1-hop, por exemplo, conecta aos vizinhos de linha e coluna adjacentes, como também aos vizinhos com distância de 2 linhas ou colunas (fig 2). Já a arquitetura octal conecta aos oito vizinhos em linha e coluna, como também na diagonal. O conjunto das arquiteturas aleatórias é obtido pela geração aleatória de conexões locais entre cada EP. Já o conjunto das arquiteturas baseadas em operações de bits é criado usando padrões locais agregados a padrões baseados em bits tais como: *perfect shuffler*, *inverse perfect shuffler*, *bit reverse*, *butterfly*, *baseline*, *cube*, etc.

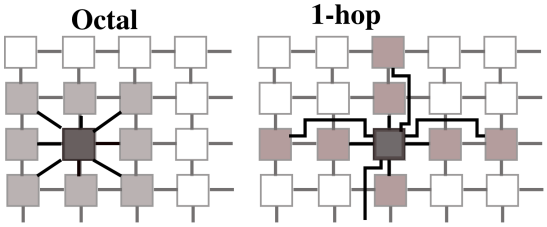

Figura 2. Padrão Octal e Padrão 1-hop.

3.1.1 Heuristíca de Geração de Arquiteturas

O ambiente proposto neste trabalho é flexível, e permite a incorporação de várias heuristícas. Como estudo de caso, três heurísticas foram implementadas: Algoritmos Genéticos, *Simulated Annealing* e *Path Relinking*, que tanto podem ser utilizadas de forma isolada, como também podem ser combinadas. A opção do Algoritmo Genético foi implementada para dois modelos diferentes. No primeiro modelo, a arquitetura é descrita pelo conjunto de nodos que compõem o grafo da arquitetura, e permite arquiteturas com padrões heterogêneos. No segundo modelo, a arquitetura é descrita pelo padrão local de conexão de um nodo que é replicado para os outros nodos, ou seja, um padrão homogêneo. As heurísticas de *Path Relinking* e *Simulated Annealing* podem ser utilizadas para se tentar melhorar os resultados obtidos tanto pelas variações do Algoritmo Genético utilizado, ou na melhoria de alguma outra arquitetura gerada por qualquer outro algoritmo. A inclusão de novas heurísticas é facilitada devido a abordagem orientada a objetos e a utilização de formatos intermediários em XML.

3.1.2 Genético com Conjunto de Vértices

O algoritmo genético, parte um de conjunto de arquiteturas, que são os indivíduos da população, e a cada iteração seleciona, com uma determinada probabilidade, os melhores para gerar novos indivíduos que serão incluídos na próxima geração, e seleciona também, com uma determinada probabilidade, os piores para serem excluídos da próxima geração. O modelo de representação do indivíduo e a operação de crossover, responsável pela geração de novos indivíduos tem influência direta nos resultados.

Inicialmente, foi incluído na ferramenta proposta neste artigo, uma modelagem baseada no conjunto de nodos da arquitetura, como mostra a Fig. 3, onde são ilustradas duas arquiteturas A e B.

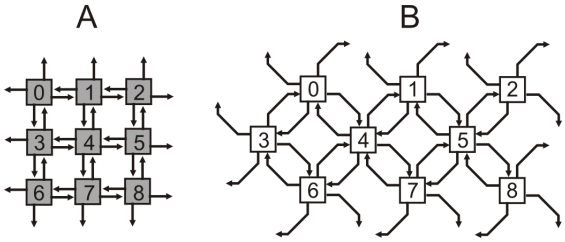

Figura 3. Duas Arquiteturas com dois padrões diferentes de conexão.

Cada arquitetura é representada por um vetor com N elementos, onde N é o número de nodos da arquitetura. Para gerar uma nova arquitetura, a operação de crossover, seleciona de forma aleatória um ponto de corte no vetor. Duas novas arquiteturas são geradas com a fusão das arquiteturas A e B, como ilustra a Fig. 5, onde são geradas as arquiteturas C e D. A arquitetura C é formada pelos 3 primeiros elementos da arquitetura B e o restante pelos elementos da arquitetura A. A arquiteura D, por sua vez, tem os 3 primeiros elementos provenientes da arquitetura A e o restante de B.

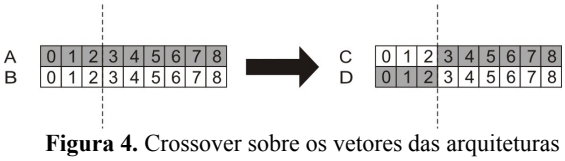

Figura 4. Crossover sobre os vetores das arquiteturas A e B para geração dos vetores das arquiteturas C e D.

A Fig. 5 apresenta as arquiteturas C e D resultantes do *crossover* ilustrado na Fig. 4, onde o ponto de corte foi o terceiro elemento do vetor. Como podemos verificar, à medida que a população vai evoluindo, podemos gerar arquiteturas que tenham diversos padrões diferentes, resultantes da combinação de várias arquiteturas.

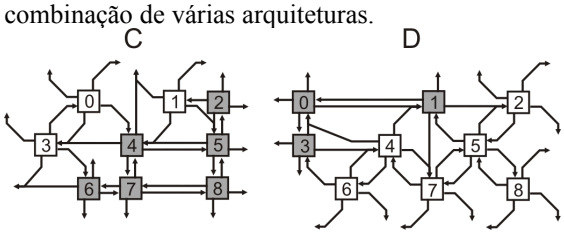

Figura 5. Arquiteturas resultantes do Crossover

3.1.3 Genético com Conjunto local de Interconexões

Para garantir a escalabilidade das arquiteturas e gerar padrões homogêneos, o segundo modelo é baseado no conjunto local de ligação de um nodo. Suponha um nodo x, na linha i e coluna j do arranjo 2D. Um padrão local é uma função da localização do nodo. Por exemplo, o padrão (i+1, j), irá gerar uma ligação do elemento i, j ao elemento i+1, j. Este formato permite também uma função como F(i), G(j), onde é aplicada uma função para i e j. Por exemplo, F(i)=i+3 e G(j) = rotação_a_esquerda(j), onde i é incrementado em 3 e j sofre uma rotação de bits à esquerda. As figuras 6, 7 e 8 ilustram as arquiteturas A, B, C e D, descritas anteriormente, porém usando o modelo de interconexões locais. Podemos observar que o vetor de descrição da arquitetura é simples, possui apenas 4 entradas. Todos os nodos são homogêneos. O *crossover* atua da mesma forma, i.e., escolhe um ponto no vetor e realiza a combinação das arquiteturas.

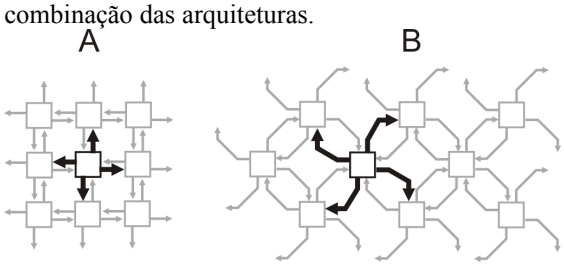

Figura 6. Arquitetura com padrão local de ligações

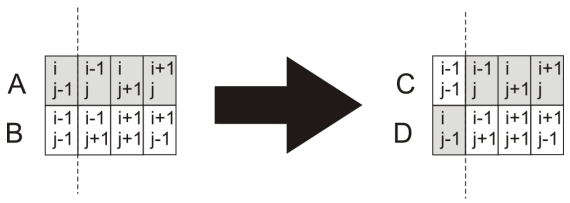

Figura 7. Representação dos vetores e crossover para modelo local de interconexões

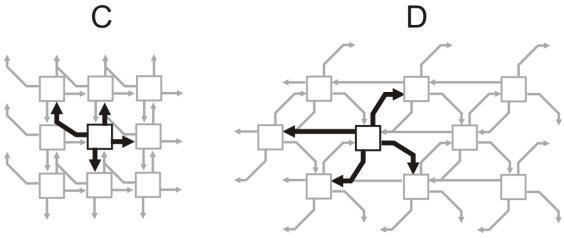

Figura 8. Arquiteturas resultantes com padrão regular

3.1.4 Path Relinking

O algoritmo de *Path Relinking* consiste na transformação de um indivíduo S1 em um indivíduo S2. O processo de transformação transcorre de forma aleatória onde os indivíduos gerados com maior número de características semelhantes ao indivíduo alvo são selecionados para uma próxima fase, até que um indivíduo S* seja igual ao indivíduo S2. A Figura 4 representa essa transformação.

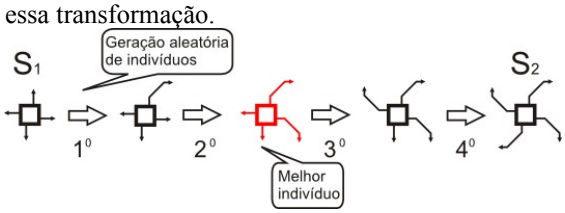

Figura 9. Processo de transformação do indivíduo S1 em S2

O *Path Relinking* pode ser aplicado aos melhores indivíduos gerados pelo algoritmo genético, como também pode ser aplicado a uma população de indivíduos sem utilizar o algoritmo genético. Dados dois indivíduos, o *Path Relinking* irá selecionar o melhor indivíduo gerado no processo de transformação do S1 em S2, ilustrado na Fig. 9.

3.1.5 Simulated Annealing

O algoritmo *Simulated Annealing* (SA) foi implementado na ferramenta proposta. O SA pode ser aplicado a um indivíduo isolado, ou ao melhor indivíduo gerado pelo algoritmo genético, *path relinking* ou combinação de ambos. O SA realiza pertubações em um indivíduo com uma determinada probabilidade de aceitar indivíduos com um custo maior, visando escapar de mínimos locais. O SA possui uma função de resfriamento, que a cada interação reduz a probabilidade de aceitar indivíduos que piorem a solução.

3.1.6 Mapeamento

A função de custo utilizada neste trabalho para avaliação das arquiteturas é o mapeamento de um conjunto de aplicações na arquitetura. O mapeamento é realizado em dois passos. No

primeiro passo é realizado o posicionamento. Neste, os nodos do Dataflow (Fig. 10a) que representa a aplicação são posicionados sobre os nodos da arquitetura. A Fig. 10b mostra dois exemplos de arquiteturas. A Fig. 11a mostra o Dataflow da Fig. 10a posicionado nas duas arquiteturas diferentes. O segundo passo é o processo de roteamento, onde os nodos que possuem uma conexão no Dataflow têm que ser ligados através da estrutura de interconexão oferecida pela arquitetura (Fig. 10b). Na primeira arquitetura todas as ligações são diretas, ou seja, tem custo 1, exceto as ligações entre C1→M0, C2→C3, S0→Y. Estas três ligações tem que ser encaminhados por vários EPs. Por exemplo, para conectar C1 em M0 na arquitetura 1 (veja Fig. 10 e 11a), é necessário passar pelo EP X ou C2. Esta ligação terá o custo mínimo de 2 segmentos, C1→X e X→M0, por exemplo. O número médio de segmentos para o posicionamento e roteamento para cada arco do Dataflow na arquitetura é: *total de segmentos utilizados para o roteamento / total arcos do Dataflow*. Para a arquitetura 1, que possui a maioria dos arcos com custo 1, e apenas 2 com o custo de 2 segmentos e 1 com o custo de 5 segmentos, o custo médio será 1,4667. Para a segunda arquitetura, o número médio de segmentos utilizados pelo roteamento é 1,5333. O número médio de segmentos ideal seria 1, onde cada arco do Dataflow seria mapeado diretamente em um segmento da arquitetura.

Neste trabalho utilizamos um algoritmo de posicionamento polinomial [17] e o roteamento utiliza o algoritmo PathFinder [18].

As implementações permitem trabalhar com um modelo de grafo para arquitetura podendo ser estendidas para arranjos em 3D ou outras arquiteturas.

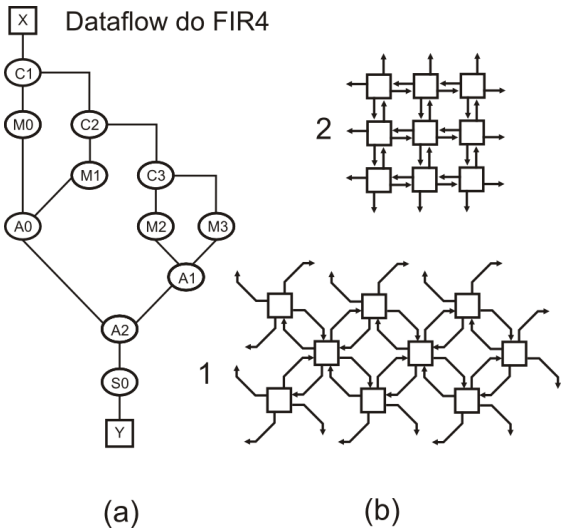

Figura 10. Exemplo de Dataflow e arquiteturas: (a) Dataflow; (b) 2 arquiteturas

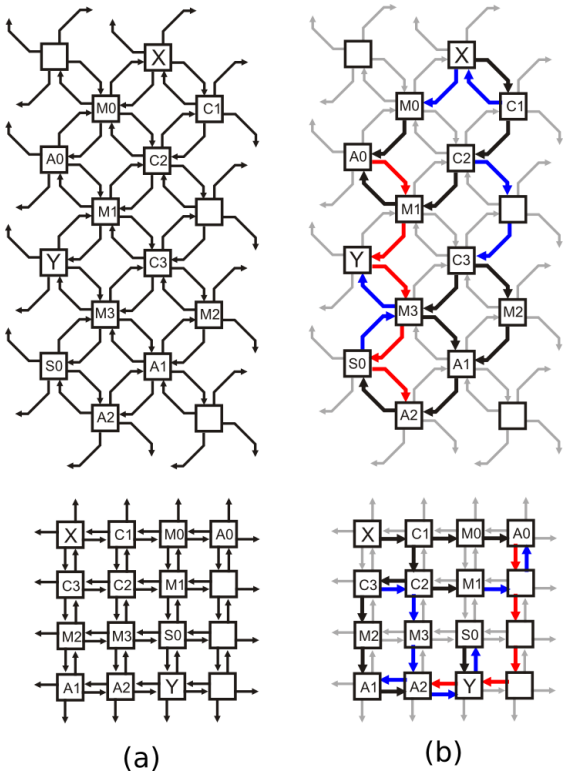

Figura 11. Mapeamento em duas arquiteturas: (a) Posicionamento; (b) Roteamento

3.2. Baseado nos grafos das Aplicações

A ferramenta proposta neste trabalho pode também gerar uma arquitetura diretamente dos grafos das aplicações. A abordagem proposta extrair padrões de conexões dos grafos de fluxo de dados. Neste artigo são apresentadas duas técnicas baseadas em algoritmos de escalonamento.

Uma arquitetura tem que fornecer uma conectividade suficiente para que o mapeamento possa ser possível para todo o conjunto de aplicações alvo. Além disso, a arquitetura deve permitir o mapeamento para outras aplicações semelhantes que não foram usadas como métrica no processo de geração.

Os algoritmos de escalonamento ASAP (*As Soon As Possible*) e ALAP (*As Later As Possible*) são utilizados para gerar uma métrica de distância para os grafos de fluxo de dados. Em um grafo, todas as conexões são diretas entre os nodos. Para criar um conceito de distância, o número ASAP ou ALAP é usado. Suponha o grafo de fluxo de dados para a aplicação FIR4 na Fig. 10a. A Fig. 12 mostra o diagrama ASAP para o grafo. Os nodos são ranqueados em níveis. Para posicionar os nodos, o grafo é percorrido das entradas em direção às saídas, sendo os nodos escalonados o mais cedo possível, ou seja, assim que todos os seus predecessores são percorridos, o nodo pode ser escalonado no próximo nível.

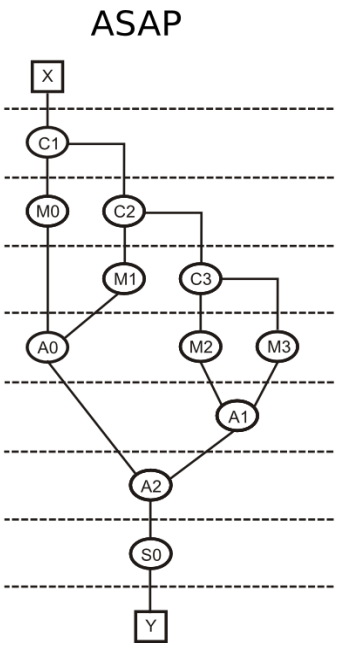

Figura 12. Escalonamento ASAP para o grafo de fluxo de dados do FIR4

A partir do grafo ASAP, podemos observar que a maioria dos nodos tem distância 1. Porém, existem nodos com distâncias maiores na métrica ASAP. Por exemplo, o nodo M0 está a uma distância 2 do nodo A0. O grafo possui 15 arestas, das quais 13 tem distância 1, e apenas 2 tem distância 2. Um vetor com a distribuição das distâncias é construído. Neste caso, 87% tem distância 1 e 13% tem distância 2.

Todo o conjunto de aplicações é percorrido. Um vetor global acumula todas as ligações e uma distribuição de distâncias é gerada. A partir do vetor final, uma arquitetura é construída com um padrão de conexão local. Suponha uma arquitetura com 6 ligações locais e suponha que a distribuição final seja 50% para distância 1, 30% para distância 2, e 20% para distância 3. A arquitetura terá 3 ligações diretas com distância 1, 2 com duas ligações diretas a uma distância 2, e 1 ligação direta com distância 3. Por exemplo, no grid o EP da linha i e coluna j, conecta-se a linha i+2 e coluna j, ou seja distância 2 em linha no grid. No processo de geração, as direções no grid são escolhidas de forma aleatória.

4. Resultados Experimentais

Para avaliar o desempenho do ambiente, são usados vários grafos de fluxo de dados que representam núcleos fundamentais de benchmarks de processamento de sinal e grafos gerados pseudo-aleatoriamente com alguma representatividade em termos de aplicações reais. Três conjuntos de grafos são utilizados O primeiro foi obtido utilizando a abordagem de compilação descrita em [19]. O segundo utiliza grafos extraídos utilizando a aplicação MediaiBench, disponíveis em [20]. Finalmente, o terceiro foi obtido com o gerador de grafos aleatórios [21].

As duas abordagens de geração ferramenta foram utilizadas: baseada nas arquiteturas e baseada nas aplicações. Um sub-conjunto de 4 benchmarks foi usado com padrão para gerar as arquiteturas. A tabela 1 mostra o resultado do comprimento médio das ligações para cada técnica. A coluna “Fir16” representa os resultados de mapeamento de um filtro FIR com 16 coeficientes. As colunas “feed” e “inter” apresentam os resultados para os benchmark Feedback_point e Interpolate do MiBench incluídos em [20]. A coluna “graph50” apresenta um grafo gerado pela ferramenta [21].

As linhas representam as técnicas utilizadas. Como o espectro de opções da ferramenta de geração é amplo, mostraremos apenas 7 opções. Para termos uma referência de comparação, usaremos a arquitetura 1-hop (Fig. 2), que está na linha 0_1_hop da tabela 1, pois esta arquitetura foi apontada por outros autores de ferramentas de exploração com uma das mais adequadas [1],[8].

A linha GCV mostra os resultados para a técnica baseda em genético com conjunto de vértices, apresentada na seção 3.1.2. Esta técnica pode gerar bons resultados, mas as arquiteturas são irregulares dificultando a escalabilidade.

As linhas GA (*Genetic Algorithm*), PR (*Path Relinking*) e SA (*Simulated Annealing*) no GA, representam as técnicas de genético com padrões regulares de interconexão da seção 3.1.3, a técnica *Path Relinking* da seção 3.1.4, e a técnica de *simulated annealing* aplicada sobre o melhor resultado do genético das seções 3.1.5 e 3.1.3. O PR apresenta os piores resultados quando usado isoladamente, pois parte de um pequeno subconjunto de arquiteturas. O GA apresenta os melhores resultados.

Entretanto, se o conjunto de aplicações e a população for muito amplo, o tempo de CPU requerido pelo GA é significativamente maior que o do PR. Já o SA aplicado no GA, não melhora muito para os 4 casos, pois o critério de avaliação de uma arquitetura usa a média do mapeamento das 4 aplicações da tabela 1.

As linhas ALAP, SA do ALAP, e SA do ASAP, geram os resultados da técnica baseada em ALAP isoladamente, e da técnica de *Simulated Annealing* aplicada sobre os resultados do ASAP e do ALAP, descritos na seção 3.2. Apesar da simplicidade das técnicas baseadas nos escalonamentos ASAP e ALAP, os resultados em custo são bem próximos dos melhores resultados gerados por todas as técnicas. Além disso, o tempo de execução é muito pequeno, devido à baixa complexidade da técnica.

	Benchmarks			
Arquiteturas	Fir16	Feed	Inter	Graph50
0_1_hop	1,30	1,41	2,11	1,57
GCV	1,46	1,37	**1,46**	1,55
GA	**1,27**	1,41	1,81	1,44
PR	-	1,55	2,04	1,81
SA de GA	1,32	1,43	1,72	1,54
ALAP	1,32	1,33	1,63	1,47
SA de ALAP	1,38	**1,31**	1,68	**1,43**
SA de ASAP	1,49	1,41	1,70	1,47

Tabela 1: Avaliação das heurísticas usando um sub-conjunto dos benchmarks

De seguida elaborou-se um estudo para avaliar se uma arquitetura gerada para um sub-conjunto de aplicações apresenta bons resultados para outras aplicações que não foram avaliadas durante o processo de seleção de uma arquitetura ótima. Um conjunto amplo de 30 benchmarks foi mapeado nas arquiteturas resultantes do subconjunto de 4 aplicações ilustrado na tabela 1. O conjunto de benchmarks é composto pelos seguintes benchmarks: invert_matrix_general, graph200_2x2, SNN, FilterRGB+Paeth+FDCT, Fir64, TreeFir64, idctcol, jpeg_idct_ifast, jpeg_fdct_islow, smooth_color_z_triangle, FDCTa, graph100_2x2, FDCT, matmul, FilterRGB+Paeth, interpolate_aux, write_bmp_header, cosine2, graph50_2x2, collapse_pyr, cosine1, FilterRGB, h2v2_smooth_downsample, ewf, feedback_points, fir2, Conf10, arf, motion_vectors, e Conf11.

Para facilitar a visualização dos resultados, são apresentados três gráficos, onde cada gráfico apresenta duas técnicas da tabela 1, juntamente com arquitetura 1-hop que é usada como referência.

A Fig. 13 ilustra a comparação entre o algoritmo genético (GA) para padrões regulares e o *path relinking* (PR), ambos tendo com referência a solução com a arquitetura 1-hop. Podemos observar que o PR melhora os resultados da arquitetura 1-hop. A arquitectura gerada pelo GA apresenta uma melhoria significativa, em média 13% melhor que a arquitectura 1-hop.

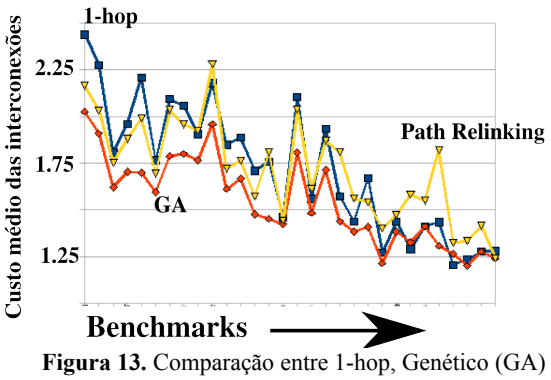

Figura 13. Comparação entre 1-hop, Genético (GA) e Path Relinking (PR)

A Fig. 14 apresenta a comparação do SA, GA, e da técnica baseada nas aplicações com o escalonamento ALAP. A técnica ALAP apresenta os melhores resultados. Em média, o custo da 1-hop é reduzido em 20%. Nesta figura podemos ver que a ferramenta de exploração possibilita gerar novas arquiteturas de forma automática, que apresentam resultados significativamente melhores que as arquiteturas apontadas por outros autores [1][8], sem uma exploração amplo do espaço de projetos.

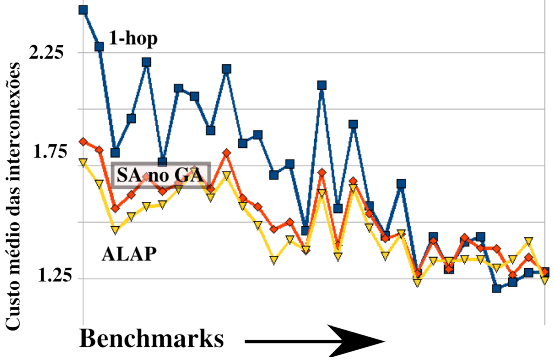

Figura 14. Comparação entre 1-hop, Simulated Annealing sobre Genético (SA no GA) e ALAP.

A Fig. 15 ilustra a saída gráfica da ferramenta para visualização das melhores arquiteturas, neste caso é mostrado o padrão gerado pelo ALAP. A Fig. 16 mostra uma tela da ferramenta, que além dos uso de XML para comunicação com outras ferramentas, possui uma interface gráfica que facilita a escolha das combinações de técnicas que podem ser utilizadas no processo de geração.

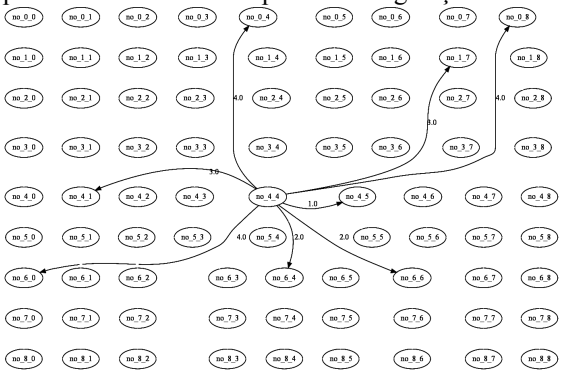

Figura 15. Padrão gerado para escalonamento ALAP

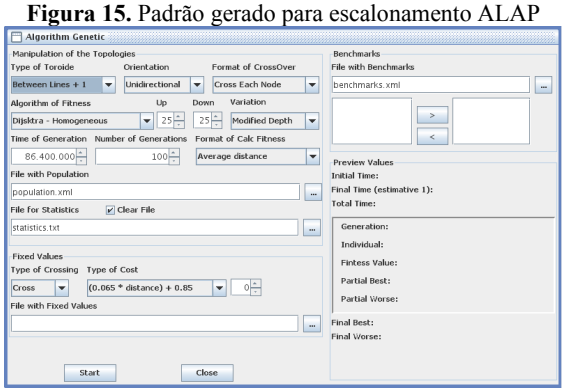

Figura 16. Tela Gráfica da Ferramenta

5. Conclusão

Neste trabalho apresentamos uma ferramenta para exploração automática do espaço de projeto de arquiteturas com múltiplas unidades de processamento. O foco do trabalho são as arquiteturas reconfiguráveis de grão grosso, onde cada unidade representa o equivalente a uma unidade lógica aritmética.

A busca da melhor arquitetura é feita na exploração do espaço de projeto das interconexões. As interconexões são um ponto chave nas arquiteturas com múltiplas unidades e devem respeitar a escalabilidade acima de tudo, uma vez que a tecnologia permite dobrar rapidamente o número de unidades.

Como ferramenta de busca usamos *Algoritmos Genéticos*, *Simulated Annealing* e *Path Relinking*. Foram utilizadas duas abordagens de geração de topologias. Primeiro, o uso dos algoritmos evolutivos para seleção automática de uma arquitetura a partir de um conjunto inicial de arquiteturas. A seleção leva em conta o custo do mapeamento de um conjunto de aplicações nas arquiteturas alvo. A segunda abordagem derivou a arquitetura diretamente das aplicações, usando algoritmos de escalonamento como métrica.

Referências

[1] Mei, B., Lambrechts, A., Verkest, D., Mignolet, J.Y., Lauwereins, R.: Architecture exploration for a reconfigurable architecture template. IEEE Des. Test 22(2) (2005) 90–101

[2] Budiu, M., Artigas, P., Goldstein, S.: Dataflow: A complement to superscalar. Performance Analysis of Systems and Software, 2005. ISPASS 2005. IEEE International Symposium on (2005) 177–186

[3] Patterson, D.A.: New directions for cacm? Commun. ACM 49(1) (2006) 33–35

[4] Robinett, W., Snider, G.S., Kuekes, P.J., Williams, R.S.: Computing with a trillion crummy components. Commun. ACM 50(9) (2007) 35–39

[5] Bossuet, L., Gogniat, G., Philippe, J.L.: Fast design space exploration method for reconfigurable architectures. In: Engineering of Reconfigurable Systems and Algorithms. (2003) 65–71

[6] Arvind, Asanovic, K., Chiou, D., Hoe, J.C., Kozyrakis, C., Lu, S.L., Oskin, M., Patterson, D., Rabaey, J., Wawrzynek, J.: Ramp: Research accelerator for multiple processors - a community vision for a shared experimental parallel hw/sw platform. Technical Report UCB/CSD-05-1412, EECS Department, University of California, Berkeley (Sep 2005)

[7] Hartenstein, R.: A decade of reconfigurable computing: a visionary retrospective. In: DATE '01: Proceedings of the conference on Design, automation and test in Europe, Piscataway, NJ, USA, IEEE Press (2001) 642–649

[8] Bansal, N., Gupta, S., Dutt, N., Nicolau, A., Gupta, R.: Network topology exploration of mesh-based coarse-grain reconfigurable architectures. In: DATE '04: Proceedings of the conference on Design, automation and test in Europe, Washington, DC, USA, IEEE Computer Society (2004) 10474

[9] Intel: Intel's first microprocessor-the intel 4004. Technical report, Intel (April 2008)

[10] Intel: Moore's law: Made real by intel innovation. Technical report, Intel (April 2008)

[11] Intel: From a few cores to many: A tera-scale computing research overview. Technical report, Intel (April 2008)

[12] Baumgarte, V., Ehlers, G., May, F., Nückel, A., Vorbach, M., Weinhardt, M.: Pact xpp-a self-reconfigurable data processing architecture. J. Supercomput. 26(2) (2003) 167–184

[13] Hartenstein, R.W., Herz, M., Hoffmann, T., Nageldinger, U.: Generation of design suggestions for coarse-grain reconfigurable architectures. In: FPL '00: Proceedings of the The Roadmap to Reconfigurable Computing, 10th International Workshop on Field-Programmable Logic and Applications, London, UK, Springer-Verlag (2000) 389–399

[14] Hartenstein, R., Herz, M., Hoffmann, T., Nageldinger, U.: Kressarray xplorer: a new cad environment to optimize reconfigurable datapath array. In: ASP-DAC'00: Proceedings of the 2000 conference on Asia South Pacific design automation, New York, NY, USA, ACM Press (2000) 163–168.

[15] Nageldinger, U.: Coarse-Grained Reconfigurable Architecture Design Space Exploration. PhD thesis, University of Kaiserslautern CS Department Informatik (2001)

[16] Hartenstein, R., Kress, R., Reinig, H.: A dynamically reconfigurable wavefront array architecture for evaluation of expressions. Application Specific Array Processors, 1994. Proceedings., International Conference on (1994) 404–414

[17] R. Ferreira, A. Garcia, T. Teixeira, and J. Cardoso, "A polynomial placement algorithm for data driven coarse-grained reconfigurable architectures," in IEEE Computer Society Annual Symposium on VLSI, 2007.

[18] Ebeling, C., McMurchie, L., Hauck, S.A., Burns, S.: Placement and routing tools for the triptych fpga. IEEE Trans. Very Large Scale Integr. Syst. 3(4) (1995) 473–482

[19] Budiu, M., Goldstein, S.C.: Compiling application-specific hardware. In: FPL '02: Proceedings of the Reconfigurable Computing Is Going Mainstream, 12th International Conference on Field-Programmable Logic and Applications, London, UK, Springer-Verlag (2002) 853–863

[20] "Expressdfg - instruction scheduling becnhmarks,http://express.ece.ucsb.edu/benchmark/, último acesso em 24 novembro 2008."

[21] Dick, R. P., Rhodes, D. L., and Wolf, W. 1998. TGFF: task graphs for free. In *Proceedings of the 6th international Workshop on Hardware/Software Codesign* (Seattle, Washington, United States, March 15 - 18, 1998). International Conference on Hardware Software Codesign. IEEE Computer Society, Washington, DC, 97-101.