UMA NOVA VERSÃO DO ALGORITMO ST-ROUTER

STÈPHANO MACHADO MOREIRA GONÇALVES[1]; LEOMAR SOARES DA ROSA JR.[1]; FELIPE DE SOUZA MARQUES[1]

[1]*Universidade Federal de Pelotas– {smmgoncalves, leomarjr, felipem}@inf.ufpel.edu.br*

1. INTRODUÇÃO

Com o avanço da tecnologia, os circuitos integrados tornaram-se cada vez mais complexos, exigindo o uso de ferramentas CAD para automatizar seu processo de concepção. Este processo é composto por várias etapas, sendo a síntese física uma das principais fases. A síntese física é composta pela fase de posicionamento, onde as células do circuito são posicionadas seguindo determinados critérios, e a fase de roteamento, que consiste em realizar as conexões dos componentes por meio de fios. Dessa forma, a etapa de roteamento busca os menores caminhos entre os componentes, desviando de todos obstáculos.

A história dos algoritmos de roteamento da síntese física começou com o *Maze Router*, proposto por Lee (1961). Este é o primeiro algoritmo da categoria de busca por *maze*, conhecida por obter um comprimento de fios ótimo. Em seguida, Rubin (1974) propôs uma melhoria sobre este algoritmo. Mikami et al. (1968) propôs um roteador de busca em linha, que é mais rápido que os anteriores, mas que não garante o resultado ótimo. Hetzel (1998) uniu os benefícios das duas categorias de algoritmos propondo um novo algoritmo, tão rápido quanto o de Mikami e com a garantia de encontrar o caminho ótimo. Com isso, o algoritmo de Hetzel é o estado da arte dos algoritmos de roteamento da síntese física. No entanto, Gonçalves (2014) apresentou um novo algoritmo de roteamento, chamado ST-Router, que apresentou resultados muito promissores. O algoritmo foi comparado com o algoritmo de Hetzel, em um escopo bidimensional, e apresentou um tempo de execução extremamente menor, apresentando um aumento de comprimento de fios quase insignificante. Como o roteamento na síntese física trabalha em um escopo tridimensional, e como o comprimento dos fios é um fator muito importante, o ST-Router ainda não pode ser considerado o estado da arte. Com isso, o objetivo deste trabalho é propor um série de otimizações no algoritmo ST-Router, fazendo-o garantir o resultado ótimo e mantendo o alto desempenho em relação ao algoritmo de Hetzel.

2. METODOLOGIA

O algoritmo ST-Router, apresenta algumas características que fazem que a optimalidade do resultado seja perdida. Essas características são apenas configurações, as quais podem ser ajustadas facilmente. Contudo, se esses ajustes não forem realizados, o algoritmo sofre uma grande queda de desempenho, podendo ser mais lento que o algoritmo de Hetzel. Com isso, as otimizações servem para atenuar a queda do desempenho por realizar esses ajustes.

A primeira otimização foi a respeito de segmentos colineares e segmentos nulos. Permitir a existência desses segmentos pode aumentar a complexidade do código e reduzir o desempenho. Para corrigir isso, foi modificada a função que realiza a conexão de um determinado nó *n* com um nó pai.

A segunda otimização foi no mapa de pares origem-destino necessária para evitar laços recursivos infinitos. Em primeiro lugar, foi observado que o mapa necessita apenas do ponto de destino para cada entrada. Em segundo lugar, foi observado que cada entrada do mapa deve ser não apenas o destino do caminho, mas o caminho inteiro. Na primeira versão, quando ocorria uma chamada recursiva com pontos de origem e destino iguais a uma chamada anterior, o algoritmo simplesmente retornava falha. Porém isso faz com que esse caminho seja abstraído, podendo levar o algoritmo a encontrar uma rota mais longa ao destino. Com isso, quando ocorre esta situação, a chamada recursiva retorna o caminho já conhecido e armazenado no mapa, agora chamada de *rotas*.

A expansão de nós de contorno sofreu algumas modificações. A primeira, foi remoção da condição de criação do nó de contorno, fazendo com que um novo nó de contorno sempre seja gerado como resultado da expansão. A segunda foi uma otimização de desempenho, concernente à criação de um nó comum. Na versão anterior, quando era possível criar um nó comum, este era gerado sobre o nó de contorno *n*, e possuía sua mesma direção. Agora, o nó comum é gerado apenas quando o nó *n* expande, gerando um novo nó de contorno *n'*, e sua localização é sobre o novo nó *n'*.

O algoritmo de detecção de atalhos não interrompe mais seu procedimento em certas situações, evitando que o algoritmo deixe de enxergar caminhos melhores. Além disso, alguns padrões de caminhos foram detectados, os quais evitam que o algoritmo percorra todo caminho em vão. A medida que o caminho é construído, a informação do seu padrão é deduzida, com base no padrão atual, e armazenada no último nó do caminho.

Uma das otimizações mais importantes refere-se ao critério para definir se um nó de contorno já foi visitado. Na primeira versão um nó de contorno era considerado como visitado quando existia um nó no conjunto de nós visitados que possuía a mesma direção de expansão, mesma direção de obstáculo, e o mesmo ponto. Contudo, essa abordagem permite que haja uma repetição de contorno de um obstáculo. Com isso, foi adicionada outra condição para deixar o critério de decisão de igualdade de nós de contorno mais completo e apropriado. Agora, quando o nó verificado não possui outro com as mesmas informações no conjunto de visitados, deve ser realizada outra consulta neste conjunto, mas agora buscando um nó que tenha as informações que caracterizem que ele veio de um perímetro do qual o nó verificado realizará sua futura expansão. Este nó é chamado de nó do caminho inverso. Este nó é gerado com base no nó verificado, e é realizada uma nova consulta, decidindo finalmente se o nó verificado foi visitado ou não.

A última otimização refere-se à principal causa dos problemas de desempenho. O grande problema é a sobreposição de buscas das chamadas recursivas. Quando ocorre uma chamada, uma busca é realizada, onde obstáculos são contornados. Em seguida, quando ocorre outra chamada, tanto dentro da anterior quanto fora, é possível que os mesmos obstáculos, ou parte deles, sejam interceptados e contornados. A medida que o numero de chamadas cresce o problema da sobreposição de buscas aumenta. O objetivo da última otimização é reduzir, ou remover, a sobreposição de buscas.

Considerando que o problema consiste em contornar obstáculos já contornados, a estratégia utilizada visa fazer com que o algoritmo se lembre dos contornos realizados. Para isso, cada vez que um obstáculo é contornado, a área contornada é mapeada para uma estrutura de dados global, a qual pode ser acessada por qualquer nível da recursão. Quando um nó de contorno expande, essa estrutura é consultada. Se existe algum mapeamento de contorno, a

expansão simplesmente segue o perímetro já mapeado, sem a necessidade de realizar o procedimento de expansão de nós de contorno.

3. RESULTADOS E DISCUSSÃO

Os objetivos dos experimentos são analisar o impacto de desempenho das principais otimizações e comparar o ST-Router otimizado com o algoritmo de Hetzel. Todos os experimentos foram realizados em um computador Windows 7 64 bits, processador Intel Core i5 3470 3.2/3.6 Ghz, com 8 Gb de memória RAM. O procedimento experimental consistiu em obter uma lista de redes, e rotear cada rede em uma grade com alguns obstáculos prévios. Cada rede foi decomposta em *n* redes de dois pontos, onde *n* é o número de pontos de destino da rede. Após o roteamento de uma rede, foi realizado um processo de mesclagem de todas redes de 2 pontos, e a rede resultante foi marcada na grade. Os circuitos utilizados nos experimentos fazem parte do conjunto de *benchmarks* ISCAS85 (HANSEN, 1999). Os maiores circuitos foram selecionados para os experimentos.

Para se mensurar o impacto de algumas otimizações, foram realizados experimentos que exercitam separadamente cada uma das 3 principais otimizações do algoritmo. A Tabela 1 apresenta os tempos de execução, medidos em segundos, de cada modificação, além do tempo de execução do algoritmo de Hetzel. A coluna 1 representa execuções com as 3 otimizações desativadas. As colunas 2, 3 e 4 apresentam, respectivamente, as execuções do algoritmo utilizando apenas uso de padrões de caminho, nós de caminho inverso e mapeamento de obstáculos. A coluna 5 representa todas as otimizações ativadas. A coluna “Hetzel” apresenta o tempo de execução do algoritmo de Hetzel. A última coluna apresenta o *speedup*, isto é, o ganho em desempenho, do ST-Router em relação ao algoritmo de Hetzel. Em todos os casos, as outras otimizações mencionadas na sessão anterior foram utilizadas. A otimização da coluna 2 não apresenta grande relevância quando ativada sem as outras. A da coluna 3 apresenta resultados melhores, principalmente para circuitos maiores, como o *apex6*, *C3540*, *C6288* e *vda*. O melhor resultado foi para o mapeamento de obstáculos. A média dos *speedups* de cada *benchmark* foram 1.1, 1.31, 2.89 e 4,59 para as colunas2, 3, 4 e 5, respectivamente. Apesar dessa tendência de *speedup* crescente,em alguns casos, uma otimização melhor, como as das colunas 2 e 3, apresentaram resultados piores. O caso em que isto se salienta mais é do circuito *apex6*. Quando o mapeamento de obstáculos foi ativado, seu

Tabela 1 – Tempos de execução em segundos de cada otimização

benchmark	**1**	**2**	**3**	**4**	**5**	**Hetzel**	***speedup***
9symml	2,4	2,2	1,8	1,1	0,64	64	102
alu2	7,4	7,3	8,2	2,8	2,1	126	59
apex6	46	45	38	117	91	13.014	143
C1355	11	11	11	2,4	2,5	840	333
C1908	11	11	8,8	3,1	2,5	177	72
C3540	154	133	82	44	33	1.447	44
C432	1,92	1,96	1,77	1,58	0,5	10	21
C499	81	45	37	8,8	4,1	644	158
C6288	1.045	1.037	722	312	135	3.373	25
C880	1,47	1,46	1,5	2,24	0,96	135	141
i2	1,13	1,1	1,09	1,03	0,75	4.504	6.006
vda	55	47	37	27	23	2.389	104

tempo de execução aumentou muito. Apesar de não ter sido possível investigar de forma mais aprofundada o caso, é bem provável que para este circuito o processo percorrer o perímetro não foi muito utilizado, fazendo com que o custo de criar a estrutura de mapeamento de obstáculos não tenha sido compensado.

Os resultados mostram que o ST-Router continua com uma enorme vantagem em desempenho em relação ao algoritmo de Hetzel. Além disso, para absolutamente todas as 10921 instâncias de roteamento, o caminho encontrado pelo ST-Router teve o mesmo comprimento do caminho encontrado pelo algoritmo de Hetzel.

4. CONCLUSÕES

O trabalho apresentou uma série de otimizações de desempenho no algoritmo ST-Router. Foram realizados experimentos para investigar o impacto das principais otimizações. Os experimentos mostraram que a principal otimização foi do mapa de obstáculos. Contudo, em alguns casos essa otimização foi prejudicial. Este é um ponto que pode ser investigado futuramente. Foram também realizados experimentos comparando o tempo de execução e comprimento de fios da nova versão do algoritmo com o algoritmo de Hetzel. Os experimentos mostraram que a nova versão continua extremamente mais rápida que o algoritmo comparado, enquanto que o comprimento de fio se manteve exatamente o mesmo. Apesar de não existir uma prova formal mostrando que o ST-Router é ótimo, este é no mínimo um forte indício de que isto é verdade.

Os resultados mostram que o ST-Router pode ser considerado o novo estado da arte no escopo bidimensional. No entanto, o roteamento da síntese física trabalha com 3 dimensões, assim como o algoritmo de Hetzel. Com isso, como trabalhos futuros, pretende-se estender o ST-Router para o escopo tridimensional e realizar experimentos com *benchmarks* da síntese física.

5. REFERÊNCIAS BIBLIOGRÁFICAS

GONÇALVES, S. M. M.; DA ROSA JR, L. S.; MARQUES, F. S.. A New General Purpose Line Probe Routing Algorithm. In: **INTERNATIONAL CONFERENCE ON ELETRONIC CIRCUITS AND SYSTEMS (ICECS)**, Marseille, 2014, p. 658-661

HANSEN ,M.C.; YALCIN, H.; HAYES, J. P., Unveiling the IASCAS-85 benchmarks: a case study. In: **REVERSE ENGINEERING IN DESIGN & TEST OF COMPUT**., 1999, vol. 6, p. 72-80.

HETZEL, A. A sequential detailed router for huge grid graphs. In: **DESIGN, AUTOMATION AND TEST IN EUROPE**, Paris, 1998, p. 332–338.

LEE, C. Y. An Algorithm for Path Connections and Its Applications. In: **IRE TRANSACTIONS ON ELECTRONIC COMPUTERS**, 1961. p. 346–365.

MIKAMI, K.; TABUCHI, K. A computer program for optimal routing of printed circuit connectors. In: **PROC. INT. FEDERATION FOR INFORMATION PROCESSING**, 1968. p. 1475–1478

RUBIN, F. The Lee Path Connection Algorithm. In: **IEEE TRANS. ON COMPUT.**, 1974, vol. C-23, p. 907-914.