POSICIONAMENTO E ROTEAMENTO DE ARRANJOS DE TRANSISTORES CMOS NÃO-SÉRIE-PARALELO

MAICON SCHNEIDER CARDOSO; LEOMAR SOARES DA ROSA JUNIOR; FELIPE DE SOUZA MARQUES

Universidade Federal de Pelotas – Grupo de Arquiteturas e Circuitos Integrados – {mscardoso,leomarj,felipem}@inf.ufpel.edu.br

1. INTRODUÇÃO

Recentemente, metodologias de minimização lógica em redes de transistores baseadas em grafos vêm ganhando destaque no processo de síntese lógica do fluxo *Very-Large-Scale Integration*. Tais metodologias podem gerar arranjos não-série-paralelo (NSP), topologia que apresenta, em geral, menor número de transistores em seus arranjos quando comparada com as clássicas série-paralelo (SP) (vide ilustrado na Figura 1, onde a função *f* é uma função Booleana representada tanto por uma rede SP – esquerda – quanto NSP – direita).

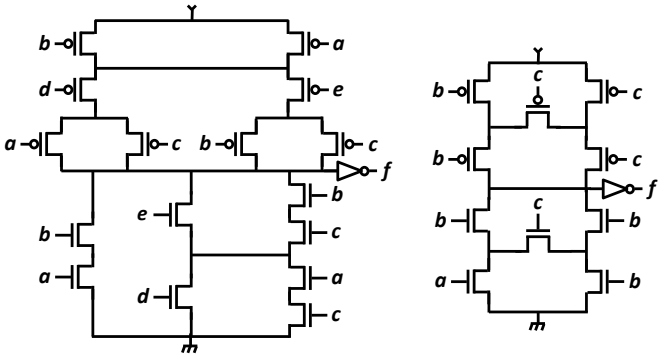

Figura 1. Arranjos SP (esq.) e NSP (dir.) que implementam a mesma função Booleana *f*.

No entanto, apesar desse claro aspecto positivo, alguns problemas são inerentes a essa topologia. Arranjos com falta de dualidade, planaridade, *gate matching* e que possuem números diferentes de transistores entre os planos lógicos constituem um desafio para os algoritmos clássicos usados na síntese física de redes série-paralelo (SP), tais como (UEHARA, 1981) (IIZUKA, 2004).

Dado a esses aspectos, faz-se necessário o desenvolvimento de metodologias específicas para o posicionamento e roteamento de redes NSP. Nesse trabalho, buscou-se prover tais métodos a fim de automatizar o processo de desenho de leiaute dessa topologia. A partir disso, estimaram-se características importantes da célula, como área, comprimento de fio usado no roteamento e número de vias, tanto para topologias NSP quanto para SP. Com a comparação direta dos leiautes NSP e SP de funções presentes num catálogo amplamente referenciado da área, intenta-se obter resultados quanto à qualidade do leiaute das células NSP relativos às SP, além de se observar os impactos que essa topologia causa no circuito integrado de maneira geral.

2. MÉTODOS PROPOSTOS

Conforme relatado anteriormente, esse trabalho busca propor uma metodologia de posicionamento e roteamento de células NSP. Relativo ao

posicionamento utilizou-se uma abordagem baseada em grafos - como proposto originalmente por (UEHARA, 1981) -, o qual buscam-se caminhos de Euler no arranjo de chaves a fim de compor o ordenamento dos *gates* dos transistores na célula. A Figura 2 (a) mostra o pseudo-código do método proposto.

Quanto ao roteamento, usou-se um algoritmo de roteamento *maze* como base. Tal método apresenta resultados ótimos em troca de alta complexidade, o que não constitui problema para o roteamento detalhado visto que o número de conexões para cada plano é pequeno, geralmente. A Figura 2 (b) mostra o pseudo-código do roteamento proposto.

```
1:  placeTransistors ( SPICE inputNetwork )
2:      PUN ← getPUNPlan ( inputNetwork )
3:      PDN ← getPDNPlan ( inputNetwork )
4:      if ( isEulerian ( PUN ) and isEulerian ( PDN ) ) then
5:          eulerPathsPUN ← findAllEulerPathsFleury ( PUN )
6:          eulerPathsPDN ← findAllEulerPathsFleury ( PDN )
7:          for each pathPUN ∈ eulerPathsPUN do
8:              for each pathPDN ∈ eulerPathsPDN do
9:                  gateSharingPercent ← gateSharingPercent +
                                        countAlignedGates ( pathPUN, pathPDN )
10:             end for
11:         end for
12:         path[0][0] ← getPUNPath ( max ( gateSharingPercent ) )
13:         path[0][1] ← getPDNPath ( max ( gateSharingPercent ) )
14:     else then
15:         subnetwork_1 ← cut ( inputNetwork )
16:         subnetwork_2 ← cut ( inputNetwork, subnetwork_1 )
17:         placedTransistors_1 ← placeTransistors ( subnetwork_1 )
18:         placedTransistors_2 ← placeTransistors ( subnetwork_2 )
19:         path[0][0] ← getPUNPath ( placeWithGap ( placedTransistors_1,
                        placedTransistors_2 ) )
20:         path[0][1] ← getPDNPath ( placeWithGap ( placedTransistors_1,
                        placedTransistors_2 ) )
21:     end if
22:     return path
23: end
```

(a)

```
1:  routeCell ( SPICE inputNetwork, EulerPath path )
2:      layout ← createLayoutPlaced ( inputNetwork, path )
3:      layout ← markViasPUNAndPDN ( inputNetwork, layout )
4:      layout ← mazeRoutePUNAndPDN (inputNetwork, layout )
5:      while ( hasInputsToConnect ) do
6:          if ( hasDifferentPlainsPath ( path ) then
7:              layout ← mazeRouteInputBetweenPUNAndPDN ( path, layout )
8:          else if ( isPossibleToRouteAbovePUN ) then
9:              layout ← mazeRouteInputAbovePUN ( path, layout )
10:         else if ( isPossibleToRouteBelowPDN ) then
11:             layout ← mazeRouteInputBelowPDN ( path, layout )
12:         else then
13:             layout ← mazeRouteInputBetweenPUNAndPDN ( path, layout )
14:         end if
15:     end while
16:     layout ← mazeRouteOutput ( path, layout )
17:     return layout
18: end
```

(b)

Figura 2. Métodos propostos. (a) Pseudo-código do posicionamento. (b) Pseudo-código do roteamento.

3. RESULTADOS E DISCUSSÃO

Os métodos propostos foram encapsulados numa ferramenta de estimativa de leiaute. A partir disso, construíram-se duas versões de cada célula do catálogo

(UFRGS, 2012) através de métodos estado-da-arte que geram redes SP (MARTINS, 2010) e NSP (POSSANI, 2015). Com isso, torna-se possível a análise de impactos em leiaute das redes NSP ante a sua versão SP, detalhado abaixo.

Primeiramente, faz-se necessário observar o número de chaves das redes SP e NSP, para que, após, verifique-se se há correlação entre os parâmetros de leiaute e a quantidade de transistores. Tal resultado pode ser visto na Figura 3, onde é claro o ganho na versão NSP (27%, em média).

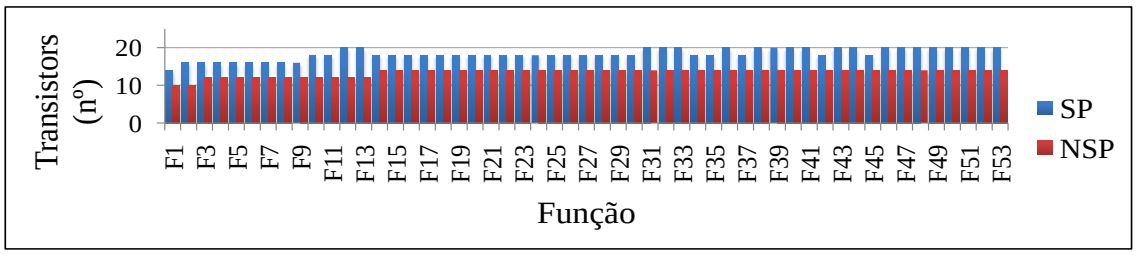

Figura 3. Quantidade de transistores presentes em cada rede SP e NSP.

Finalmente, analisa-se o leiaute das células NSP e SP. Quanto à área, a Figura 4 resume as estimativas geradas. Fica evidente o ganho obtido (36%, em média).

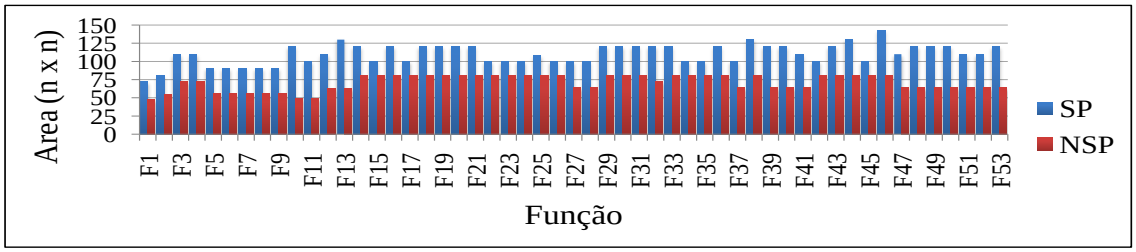

Figura 4. Estimativa de área das células SP e NSP.

Em um segundo momento, verificou-se a quantidade de fio usado no roteamento, resultado ilustrado pela Figura 5. Nesse caso, no entanto, faz-se necessário um detalhamento maior na análise, visto que há diferentes tamanhos de fio para cada camada de metal, como mostra a Tabela 1. Assim, apesar de, em geral, haver uma redução expressiva em tamanho de fio (34%), houve a necessidade de se utilizar mais camadas de metal, o que, em geral, prejudica o roteamento global e deteriora características elétricas importantes como o atraso da propagação do sinal, por exemplo.

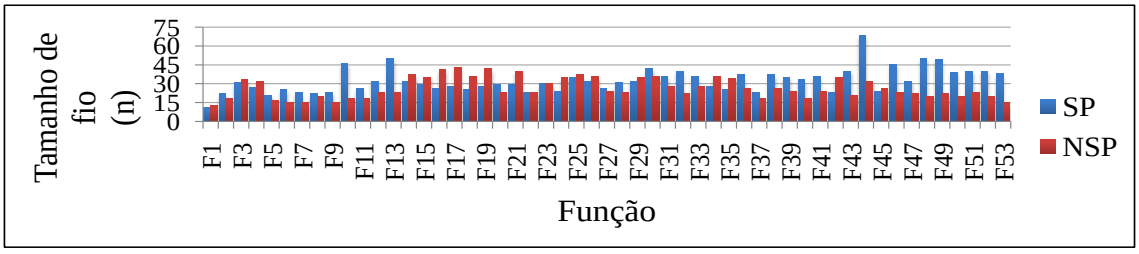

Figura 5. Tamanho total de fio nas versões SP e NSP.

Tabela 1. Tamanho de fio separado por camadas de metal.

	Poly (n)	*Metal 1 (n)*	*Metal 2 (n)*	*Metal 3 (n)*	*Total (n)*
SP	3641	2280	554	24	6499
NSP	2039	1532	584	177	4332
Dif.	-44%	-33%	+1%	+737%	-34%

Por último, analisou-se o número de vias nas células SP e NSP. Como essa característica está correlacionada com a quantidade de camadas de metal do

roteamento, algo semelhante com o observado no parágrafo anterior ocorreu: houve aumento do número de vias em camadas de metais superiores na versão NSP. A Figura 6 ilustra os resultados totais (13% de ganho da versão NSP).

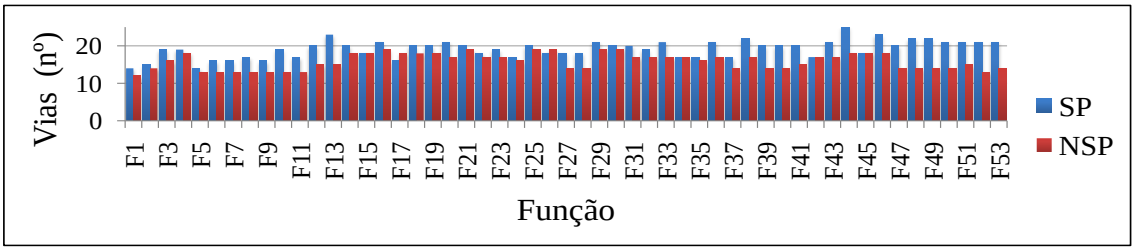

Figura 6. Número total de vias nas versões SP e NSP.

4. CONCLUSÕES E TRABALHOS FUTUROS

Nesse trabalho apresentaram-se as metodologias propostas para posicionamento e roteamento de células NSP, topologias que, em geral, apresentam redução no número de transistores.

Aplicando-se os métodos desenvolvidos a uma biblioteca de células com versões SP e NSP, geraram-se estimativas sobre aspectos importantes para o leiaute, tais como área, tamanho de fio e quantidade de vias. Observou-se que é recorrente o ganho em área (inclusive maior que o ganho no número de chaves). No caso de tamanho de fio e número de vias, a versão NSP, apesar de apresentar ganho no total, necessitou de metais de camadas superiores, o que causa maior atraso da propagação do sinal e dificulta o roteamento global do circuito integrado, por exemplo.

Como trabalhos futuros, intenta-se incorporar as metodologias propostas a ferramentas de desenho automatizado, tal como o ASTRAN (ZIESEMER, 2014), o qual gerencia as regras de projeto inerentes a cada tecnologia. Ademais, pretende-se investigar o comportamento elétrico da célula (relativo ao atraso de propagação de sinal e potência elétrica dissipada), tanto através de estimativas quanto de simulação.

5. REFERÊNCIAS BIBLIOGRÁFICAS

IIZUKA, T., IKEDA, M., ASADA, K. High Speed Layout Synthesis for Minimum-width CMOS Logic Cells via Boolean Satisfiability. **Conference on Asia South Pacific Design Automation**, p.149-154, 2004.

MARTINS, A., DA ROSA, L., RASMUSSEN, A., RIBAS, R., REIS, A. Boolean factoring with multi-objective goals. **IEEE International Conference on Computer Design**, p.229-234, 2010.

POSSANI, V., CALLEGARO, V., REIS A., RIBAS, R., MARQUES, F., DA ROSA, L. Graph-based transistor network generation method for supergate design. **IEEE Transactions on Very Large Scale Integration (VLSI) Systems**, vol. PP, no. 99, 2015.

UEHARA, T., VANCLEEMPUT, M. Optimal Layout of CMOS Functional Arrays. **IEEE Transactions on Computers**, p.305-312, 1981.

UFRGS. **Catalog of 53 Handmade Optimium Switch Networks**. Logics Lab. Porto Alegre, 2012. Acessado em 10 dez. 2014. Online. Disponível em: http://www.inf.ufrgs.br/logics/docman/53_NSP_Catalog.pdf

ZIESEMER, A., et al. Automatic layout synthesis with ASTRAN applied to asynchronous cells**. Latin American Symposium on Circuits and System**, p.1-4, 2014.