RAISE: A Detailed Routing Algorithm for Field-Programmable Gate Arrays

V. Baena-Lecuyer, M. A. Aguirre, A. Torralba, L. G. Franquelo and J. Faura*
Dpto. de Ingeniería Electrónica
Escuela Superior de Ingenieros,
Avda. Reina Mercedes s/n, Sevilla–41012 (SPAIN)
Tel.: +34 (9)5 455 68 57
FAX: +34 (9)5 455 68 49
e–mail: baena@gte.esi.us.es
*SIDSA
C/ Isaac Newton 1, Parque Tecnológico de Madrid, Tres Cantos, Madrid–28760
Tel.: +34 (9)1 803 50 52
e–mail: faura@sidsa.es

Conference Topic: FPGA Design and Applications

Abstract— **This paper describes a new detailed routing algorithm, specifically designed for those types of architectures that are found on the most recent generations of Field-Programmable Gate Arrays (FPGAs). The algorithm, called RAISE, can be applied to a broad range of optimizations problems and has been used for detailed routing of symmetrical FPGAs, whose routing architecture consists of rows and columns of logic cells interconnected by routing channels.**
RAISE (Router using AadaptIve Simulated Evolution) searches not only for a possible solution, but tries to find the one with minimum delay. Excelent routing results have been obtained over a set of several benchmark circuits getting solutions close to the minimum number of tracks.

I. Introduction

In the last years, the use of Field-Programmable Gate Arrays (FPGAs) has been widely accepted as an attractive means of implementing digital circuits. There is a wide range of comercial FPGAs, but one of the most important types is the symmetrical FPGA, which consists of rows and columns of logic blocks with horizontal routing channels between rows and vertical routing channel between columns. This type of FPGAs was first introduced by Xilinx in 1986, but currently it can be found in some of the Altera and Quicklogic families.

Symmetrical FPGAs can reach very high logic capacities; for this reason, a key problem in the design of this kind of FPGAs is the structure of their routing channels. The use of short segments improve chip area (less segment length is wasted using short segments) but to provide long connections, the interconnection of short segments via programmable routing switches is required, reducing speed performance. On the other hand, the uses of long segments wastes chip area but improves speed performance (less segments are required to make long connections passing through only a few switches).

This tradeoff forces the design of complex routing channels, with different lenght segments, which requires sofisticated Computer Aided Design (CAD) Tools.

Five stages are usually involved in mapping a circuit: design entry, logic optimization technology mapping, placement and routing. The last one is made in two step: global routing and detailed routing. This paper presents RAISE, a new detailed router adapted for generic symmetrical FPGAs.

II. RAISE: Router using AdaptIve Simulated Evolution

RAISE is based on SILK [3], a simulated evolution program for channel routing. Before running RAISE, for each point to point net, a set of possible paths is generated (for example, using the technique called Coarse Graph Expansion (CGE) [1] [2]). RAISE takes this set and searches for a path subset that make possible the routing of all the nets, while minimizing the delays.

Theese steps are carried out by RAISE:

1. Initial Routing.
2. Rip-Up and Rerouting.
3. Postoptimization.

A. Initial Routing

The algorithm, of statistical nature, needs a seed to start the iterative process. This seed or solution, does not need to be feasible, that is, it can have conflicts, which have to be solved in the following steps. Our detailed router takes for each point to point net the path with minimum delay. The delay can be calculated with the RC-Tree algorithm of [4].

B. Rip-Up and Rerouting

The rip-up and rerouter solves the conflicts generated in the initial routing. To this purpose, RAISE uses the Simulated Evolution technique.
Basically, a cost is generated, for each point to point net using a special function cost, which accounts for the paths delay and the conflict with other nets; then this cost is scaled in the range $[0.1, 0.9]$; for each point to point net, a random number between 0 and 1 is generated, if this number is less than the scaled cost of the routed path, the path is removed. After end of this process, there will be a set of routed point to point nets and another set of non-routed point to point net. Next, for each multipoint net, in a random order, all the non-routed point to point nets are routed, choosing the path with minimum cost. This process is repeated until a solution with no conflicts is obtained or until a maximun number of iterations is reached.
Using a random number generator to select the non-routed nets, allows the algorithm to exit from local minimums. Note that in the selection process, the nets with a high costs have a high probability of being removed. However randomly removing some good nets also helps to avoid getting stucked at a local minimum.
A key point in such algorithms is the function cost. This function should contain at least a delay and a conflict term. But other terms can be added to improve the convergence:
From the problem definition, we know that point to points nets from the same multipoint net can share segments. To improve chip area, the number of shared segments in a point to point net should be maximized, as this we consume less FPGA routing resources.
Besides, it would be desirable to get out some advantages of each iteration, i.e., if for each iteration we know if the nets are valid or not, we could learn not to do the sames errors we made in previous iterations.
This is included in the following function cost:

$$\begin{aligned} cost &= \alpha \cdot (num\_shared\_wires) \\ &\quad + \beta \cdot (history\_cost) \\ &\quad + \gamma \cdot (\frac{path\_delay}{min\_path\_delay}) \\ &\quad + \delta \cdot (\frac{num\_non\_shared\_wires}{min\_num\_non\_shared\_wires}) \end{aligned}$$

each term is explained as follow:

- *num_shared*: number of multipoint nets shared segments.
- *history_cost*: demand of each segment in previous iterations.
- *path_delay*: self explanatory.
- *min_path_delay*: minimum path delay of the set of possible paths for this point to point net.
- *num_non_shared_wires*: number of non shared segments between this point to point net and the others of the same multipoint net.
- *min_num_non_shared_wires*: minimun number of non shared segments between this point to point net and the others of the same multipoint net.

The history_cost term can be calculated easily if we remember wich segments were shared in previous iterations. In our case, it is calculated as follow:

$$\begin{aligned} HistCost(Wi, K) &= 0.5 \cdot HistCost(W_i, K-1) \\ &\quad + NetsUsingW(W_i, K) \end{aligned}$$

where NetsUsingW is the number of multipoint net that use wire W_i, and K is the number the actual iteration. The minimum number of not shared segments between one point to point net and the others of the same multipoint net, can be calculated from the netlist of the global router supposing each corner of the net can be reached with only 1 segment.
The α, β, γ and δ parameters have to be well tuned to reducte the number of iterations and to get a fast convergence.

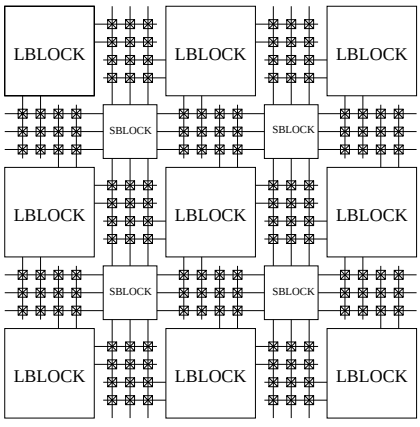

Figure 1: FPGA structure

C. Postoptimization

This phase is reached when a feasible solution has been formed. Then, for each point to point net in a random order, the paths with the least delay from those that do not conflict with present solution, are selected. This phase is repeated until no a change is accepted in an iteration.

Circuits	Channel Density	RAISE	SEGA Area	SEGA Speed	Sega Anneal
9symml	9	9	11	12	11
term1	10	10	11	11	13
C499	10	12	12	15	12
C1355	11	12	12	14	13
vda	14	15	15	15	16

Table 1: Minimum number of tracks per channel required for a successfully routing

W	RAISE			SEGA Area		
	Av. Delay	Max. Delay	Exec. time (s)	Av. Delay	Max. Delay	Exec. time (s)
9	3.599	32.414	159.80	-	-	-
10	3.837	35.316	4.80	-	-	-
11	3.863	37.063	3.07	3.949	32.571	0.53
12	3.895	35.167	2.09	4.350	36.549	0.64
13	4.120	39.930	1.73	4.384	45.569	0.71
14	4.310	40.658	1.86	4.563	45.771	1.03
15	4.349	41.386	1.90	4.363	37.691	1.14

Table 2: Average and maximum delays generated by RAISE and SEGA Area for 9symml and different channel density

III. Results

To test the performances of RAISE, different routing solutions have been obtained with a set of benchmark circuits. The FPGA structure we used can be seen in figure 1, the C blocks have a switch for each segment, i.e. in SEGA terminology, fc=W; the routing structure of an S block is shown in figure 2: all segments excepted the first of each channel (segment 0 in the picture) have a connection pattern like segment 1, then $fs > 3$. For simplicity, the vertical and horizontal routing channels have only one track group with W segments, offset 1, and lenght 3.
As well we set α parameter to 2.0 , β parameter to 0.5, γ to 1.0 and δ to 1.0.

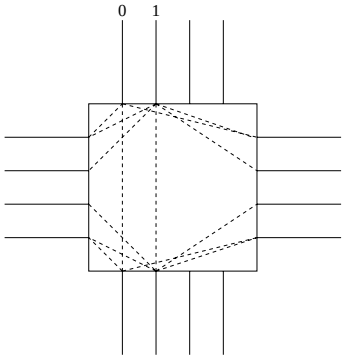

Figure 2: S block routing structure

We can see the results in table 1 for a set of benchmark circuits. For this FPGA architecture, the number of wiresegments in each routing channel, needed to route the circuits is very close to the minimum number told by the global router. Note that RAISE reaches solutions that other routers can't find. In figure 3 we show a RAISE solution for the 9symml circuit with nine track per channel. From table 2, we see the maximum and average path delay for different number of wiresegments per channel, for the 9symml circuit. We can see that RAISE normally obtains better solutions than SEGA Area router and can be used to find solutions in dificult circuits with hugely saturated channels. The price to be paid for this better performances is computetime cost. Like other statistical based optimization programs, RAISE take a time searching for new solutions, as can be seen in the execution time column of table 2.

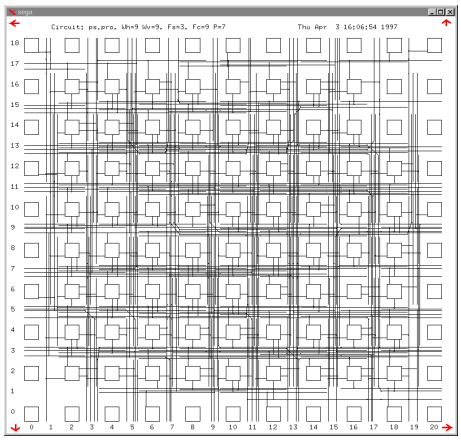

Figure 3: 9symml RAISE routing solution with nine track per channel

IV. Conclusions

This paper has presented RAISE, a simulated evolution router for FPGAs. RAISE uses a statistical technique to explore the solutions space. It has been shown that RAISE normally obtains better solutions than different versions of SEGA. Furthermore it find solutions that other routers can·t find.

V. Acknowledgments

The authors would like to acknowledge financial support by the European Union through the ESPRIT project FIPSOC and by CICYT through the TIC86-0860 project.

References

[1] Stephen Dean Brown, "Routing Algorithms and Architectures for Field-Programmable Gate Arrays",*Thesis, Department of Electrical Engineering*, University of Toronto, Canada. January 1992.

[2] G. Lemieux and S. Brown, "A Detailed Router for Allocating Wire Segments in FPGAs", *ACM Physical Design Workshop*, Lake Arrowhead, California, pp. 215-226. April 1993.

[3] Youn-Long Lin, Yu-Chin Hsu, and Fur-Shing Tsai, "SILK: A Simulated Evolution Router",*IEEE Transactions on Computer-Aided Design*, Vol. 8. NO. 10. October 1989.

[4] M. Khellah, S. Brown, and Z. Vranesic, "Modelling Routing Delays in SRAM-Based FPGAs",*Proc. 1993 CCVLSI*, Banff, Canada, pp. 6B.13-6B.18, Nov.1993.