Routability Prediction for Hierarchical FPGAs

Wei Li
Nortel Technologies,
Ottawa, Ontario
wwli@nortel.ca

D.K. Banerji
Department of Computing & Information Science,
University of Guelph,
Guelph, Ontario
N1G 2W1
dbanerji@snowhite.cis.uoguelph.ca

Abstract

This paper investigates the problem of routability prediction in a FPGA that employs a hierarchical routing architecture. Such a FPGA is called a hierarchical FPGA(HFPGA). A novel model is proposed to analyze various HFPGA configurations. A software tool has been developed to predict the routability of circuits on specific HFPGA architectures. Primary contribution of this work is that routability prediction can be done immediately after the technology-mapping step, rather than after placement. The effect of connection block and switch block flexibility on routability is also studied. The results show that compared to a symmetrical FPGA architecture, we can achieve the same degree of routability on a HFPGA, with much fewer routing switches.

1. Introduction

In recent years, Field-Programmable Gate Arrays(FPGAs) have seen a huge growth in usage because they dramatically reduce design turn-around time and manufacturing costs for prototype circuits and for small volume products. Much research has been done in routing architectures, routability prediction, and routing algorithms. However, most of this work has focused on traditional, symmetrical FPGAs which are represented as an *NxN* array of logic blocks, separated by both horizontal and vertical routing channels(Fig.1).

Section 2 introduces a novel hierarchical architecture for FPGAs as shown in Fig.2. Our experiments show that for HFPGAs, the amount of routing resources required is greatly reduced while maintaining a good routability.

Section 3 describes a statistical model to calculate the routability of an application immediately after technology-mapping, to evaluate the effectiveness of routing architectures without performing placement and routing.

Section 4 presents our experimental procedure and results. These include routing resource consumption of a symmetrical FPGA versus HFPGA, and the effect of connection block and switch block flexibility on routability.

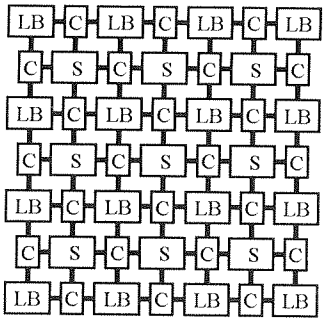

Fig.1. Typical symmetrical FPGA architecture
LB: Logic block, C: Connection block,
S: Switch block

0-7695-0104-4/99 $10.00

2. Hierarchical FPGA architecture

The basic components of the proposed HFPGA architecture are shown in Fig.2. This architecture is influenced by the proposals and results presented in [1], [2], and [4]. The elements of this architecture are similar to those of a symmetrical FPGA, and include logic blocks(LBs) into which logic is mapped, connection blocks, and switch blocks through which routing is performed. The difference is that K=k x k LBs and the corresponding routing blocks, which are called 0th-level elements, are grouped into a supercluster which is called a 1st-level element. Then, K 1st-level elements are grouped into a 2nd-level element. This grouping can be performed recursively until the whole HFPGA is covered by a nth-level element. Such a HFPGA structure is called a K-N HFPGA. While any type of logic block can be used as a 0th-level element, this paper assumes a 4-input lookup table for this purpose.

The connection block and switch block structures are assumed to be the same as those of symmetrical FPGAs described in [1]. However, their flexibility may be different for each level. It is intuitive that lower levels need lower flexibility because they have fewer connections to route.

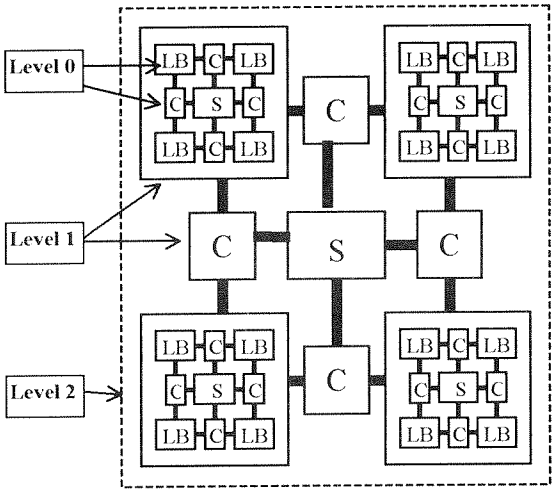

Fig.2. A hierarchical FPGA architecture

At each level, routing is first performed locally. Only if a signal has a connection with other elements, will it be routed out of this element through its connection block. A HFPGA requires less switches to route a net, which in turn reduces the capacitance on the net. Therefore, it can implement much faster logic with much less routing resources, as compared to a standard FPGA.

3. Routability prediction

A stochastic model to predict the routability of symmetrical FPGAs is described in [1]. This model conforms quite well with experimental results. Since the structure of each element in a HFPGA is the same as that of a symmetrical FPGA, the model in [1] can be applied individually to each level if the following parameters are known.

R_i : average connection length at ith level.

C_{T_i} : number of connections at ith level.

However, these parameters remain unknown before the actual partitioning and placing of the circuit. The following describes how to estimate these parameters after technology-mapping, but *before placement*, to enable us to predict the routability as early in the design process as possible.

There is a well-known relationship for partitioning logic into sub-modules, known as Rent's Rule[5].

$$P = \lambda B^r$$

where P is the number of external pins on a module, B is the number of blocks per module, λ is the average number of pins per block, and r is called Rent's exponent which is a measure of interconnection complexity of the logic. P and r are fixed for a given logic net; therefore, λ and B are the adjustable parameters of Rent's rule. The internal details of a logic net are separable from the external environment as long as we provide enough interface pins(in the case of FPGAs, it is connection block flexibility) whose number is determined by Rent's rule. Fortunately, we have found that Rent's rule holds equally well for hierarchical partitioning of logic onto HFPGAs.

Suppose we have a total of B elementary gates, each of which has λ I/O pins, and we partition them onto a K-N HFGPA. Applying Rent's rule, the total number of pins used by ith level and up is $K^{N-i}\lambda K^{ir}$, and the total number of pins used by i+1st level and up is $K^{N-(i+1)}\lambda K^{(i+1)r}$; hence the number of pins used for ith-level connections is $K^{N-i}\lambda K^{ir} - K^{N-(i+1)}\lambda K^{(i+1)r}$. Finally, the number of connections at ith-level is given by:

$$C_{Ti} = \alpha(K^{N-i}\lambda K^{ir} - K^{N-(i+1)}\lambda K^{(i+1)r})$$

where $\alpha = \dfrac{\#connections}{\#pins}$ is assumed to be constant for all levels and can be obtained from the original circuit.

Suppose a logic design is partitioned onto K=k x k elements. If we assume that an element may connect to

all other K-1 peer elements, a good approximation of the average connection length is $\frac{2}{3}\sqrt{K}$. Hence, the average connection length for this design can be calculated as

$$R_i = \frac{\sum_i [C_{T_i} \bullet (avg.\#connections\,at\,level\,i)]}{\sum_i C_{T_i}}$$

$$= \frac{2}{3} \bullet \frac{1-r}{r-\frac{1}{2}} \bullet (B^{r-\frac{1}{2}} - 1)$$

Applying these formulae to the technology-mapped logic is equivalent to placing it on a HFPGA. We then apply the model of [1] to every level to get the routability at that level. The final routability of the whole design on the HFPGA is given by:

$$Routability = \frac{\sum_{i=0}^{N-1} Routability_i}{N}$$

The detailed derivation of these expressions is not provided due to space limitations.

4. Experimental study

The purpose of our experiments is to first compare the routing resource consumption, namely the number of switches used in connection blocks and switch blocks, of a HFPGA and symmetrical FPGA. Next, we want to determine parameter values which result in high routability. Many MCMC benchmarks were used for this study; the results for five big benchmarks are reported in Table 1. The experimental procedure is as follows: (1)logic optimization using SIS[8]; (2)technology mapping using chortle[7]; and (3)routability prediction using our method.

Table 1: MCNC benchmark circuits

Circuit	#LBs	#Pins	#IOs	#Connections
Elliptic	3604	17246	245	12520
Frisc	3556	17098	136	12565
Spla	3690	17452	62	13762
s298	1931	8884	10	6945
Apex2	1878	8567	41	6689

4.1 Switch consumption

This experiment compares switch counts for HFGPA and symmetrical FPGA architectures. The routability of the benchmark circuits is predicted on 16-3 HFPGAs and on a 64x64 symmetrical FPGA. The switch counts are plotted up to routability of 95%. Fig.3. shows the results obtained. From the results, we may see that HFPGAs require fewer switches to implement the circuits than the symmetrical FPGAs, thus leading to less delay and increased operating speed. The switch savings could be up to 65% for this particular experiment. This is because a HFPGA takes advantage of the hierarchy and locality of nets for routing.

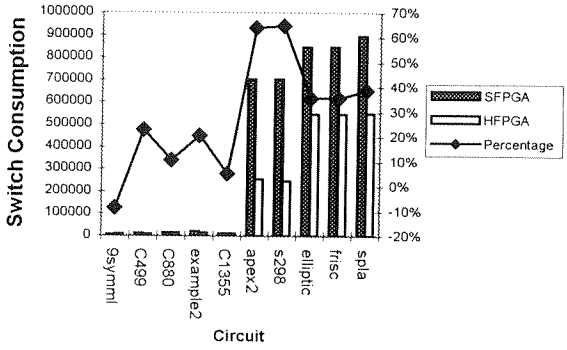

Fig.3. Switch counts for 95% routability

4.2 Optimal architecture

Since switch block flexibility(Fs) follows the same logic as in [1], we are mainly concerned with connection block flexibility(Fc) here.

We first predict the routability of the same circuits on 64-2, 16-3, 4-6 HFPGAs and 64x64 Symmetrical FPGAs. Figure 4 shows the results. It is obvious that HFPGAs require less switches than do symmetrical FPGAs (SFPGAs). The architecture with a small super-cluster requires less switches than an architecture with a large super-cluster. In this particular experiment, the savings over SFPGAs are 40%, 55%, and 75% respectively.

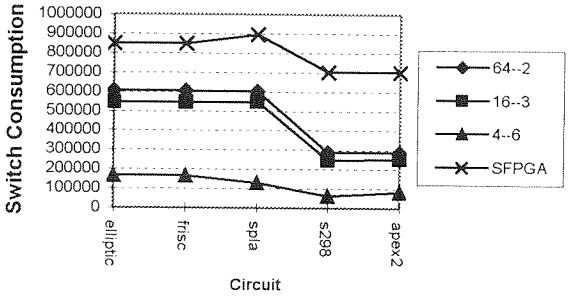

Fig.4. Routability vs. number of levels

Then we vary Fc_i from 1 to W_i(channel width at level i), and fix the parameters at other levels. A typical

result is shown in Fig. 5. If the flexibility at level 1 and level 2 is one, then very small percentage of circuits are successfully routed even when Fc_0 is increased to eight. This is because a low flexibility at any level acts as a bottleneck for the successful routing of a circuit. Only after Fc_1 and Fc_2 are greater than five, the circuit can be deemed routable. The routability reaches the maximum at Fc_0=4; after this point, increasing flexibility at any level does not result in significant increase in routability.

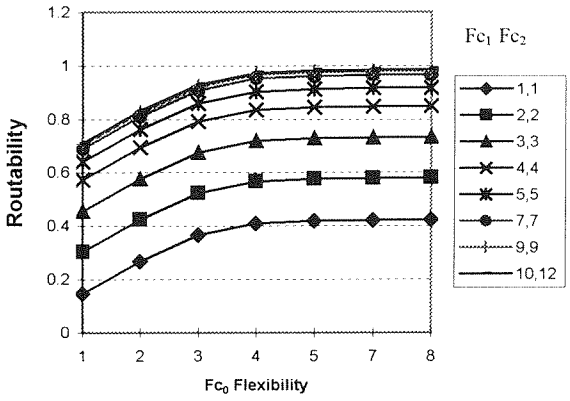

Fig.5. Routability vs. Fc_0

From an analysis of the results, we can conclude that:

a) There is an optimal value for Fc(usually > W/2); larger values do not result in significant routability increase. Higher levels in a HFPGA tend to need higher flexibility(Fc), because they have more connections to route.

b) HFPGAs with smaller clusters require fewer switches to route a design. The 4-6-HFPGA needs far fewer switches than the 16-3-HFPGA(Fig.4). But practical considerations may put constraints on the size of clusters. For example, the logic may not be so fine-grained as to fit into a small cluster.

c) Reducing Fs and Fc at lower levels will result in more switch saving than doing it at higher levels.

d) Greater number of levels in the hierarchy results in lower switch requirements.

5. Conclusions

We have presented a statistical model to predict the routability of a design on HFPGAs, *prior to placement.* Experiments were conducted to investigate the desirable properties of the routing architecture. Results show that HFPGAs require less routing resources to implement a design, thus reducing delay and increasing circuit performance.

References

[1] S. Brown, R. Francis, J. Rose, and Z. Vranesic, Filed-Programmable Gate Arrays, Kluwer Academic Publishers, Boston, MA., 1992.

[2] Yen-Tai Lai, and Ping-Tsung Wang, "Hierarchical Interconnection Structures for Field Programmable Gate Arrays", *IEEE Trans. on VLSI Systems*, Vol. 5, No.2, June 1997, pp.186-196.

[3] K. Chung, S. Singh, J. Rose, and P. Chow, "Using Hierarchical Logic Blocks to Improve the Speed of Filed-Programmable Gate Arrays", *Proceedings of the 1990 Custom Integrated Circuits Conference (CICC-90)*, pp. 31.2.1-31.2.7.

[4] A. Aggarwal and David. M. Lewis, "Routing Architectures for Hierarchical Field Programmable Gate Arrays", *IEEE International Conference on Computer Design: VLSI in Computers & Processors*, 1994, pp. 475-478.

[5] W. E. Donath. "Placement and Average Interconnection Lengths of Computer Logic." *IEEE Transactions on Circuits & Systems, CAS-26:* pp. 272-277, 1979.

[6] Dirk Stoobandt, "Improving Donath's Technique for Estimating the Average Interconnection Length in Computer Logic", *ELIS Technical Report DG 96-01* June 1996.

[7] R. Francis, J. Rose, and Z. Vranesic, "Chortle-crf: Fast Technology Mapping for Lookup Table-based FPGAs," in *Proc. 28th DAC*, 1991, pp. 227-233.

[8] Ellen M. Sentovich, R. Brayton, and A. Sangiovani-Vincentelli, "SIS: A System for Sequential Circuit Synthesis", *Department of Electrical Engineering and Computer Science, University of California, Berkeley*, May 1992.

[9] Xilinx Corp., The Programmable Logic Data Book, 1996.