APPLE PARROT: A CIRCUIT PARTITIONER

Diogo Fiorentin, Renato Hentschke, Ricardo Reis

Universidade Federal do Rio Grande do Sul – Instituto de Informática
{dfiorent, renato, reis}@inf.ufrgs.br

ABSTRACT

The Apple Parrot is a tool developed to divide big circuits into smaller ones. Placement is a complex problem and as the circuit sizes increase, the partitioning is a necessary step. Through the partitioning process proposed, several circuits are created from one circuit. The circuit is partitioned in four blocks and these blocks are partitioned recursively until all of them have fewer cells than a pre-stipulated limit. The Fiduccia-Matteyses method is taken to decide to what partition each cell will be assigned. The experiment results show that the circuits created have almost the same area and number of cells. However the number of nets linking the partitions is still too high.

1. INTRODUCTION

As the circuits grow in size, the placement problem becomes even more complex. A circuit that has a great number of cells needs a long time to be placed. This problem can be minimized by partitioning great circuits in several smaller ones because the placement time is not a linear function. Several small circuits are placed faster than one great circuit.

The problem consists in creating smaller circuits from one big circuit. Each small circuit has to be independent of the others as much as possible. Thus, it is necessary minimize the cells connection between different circuits.

The partitioner works with circuits described in a logic netlist. These files are in the Spice format. As output, it returns several Spice files. Every new file contains a group of cells from the original circuit. The partitioning process is performed by a hierarchical quadratic method while the partitioning itself is performed Fiduccia-Matteyses algorithm [4] that is a famous and largely used heuristic. There are several previous works that are based on quadratic partitioning of the circuit, like [6] [3] [1].

2. THE CELLS DIVISION METHOD

The Apple Parrot partitioner is based on a heuristic quadratic method to separate the cells in blocks. In this process one block derives four new blocks of cells. This process is repeated recursively until all blocks have fewer cells than a pre-stipulated limit.

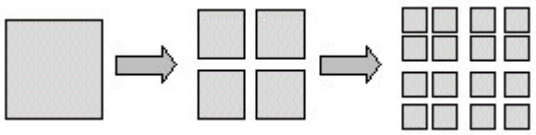

Fig. 1 – Recursive Quadratic Partitioning method

3. THE FIDUCCIA-MATTEYSES METHOD

The method chosen to measure about what partition each cell must stay is the Fiduccia-Matteyses method (FM) described in [4][2]. The objective of FM is to minimize the number of nets that link cells in different partitions. This objective is known in the literature as min-cut [1][5].

The algorithm has some particular novelties that are very interesting to the partitioning of integrated circuits:

- It supports hyper-graphs. In a circuit, the nets are hyper-edges and the cells are the nodes. A net will cut the partitions if there is cells in different partitions.
- There is a balance function that is able to reach area equilibrium for the partitions.
- Fast cuts update. As the FM algorithm deals with hyper-graphs, the nets cutcan be evaluated and updated easily.
- FM is able to perform some iterations that deteriorate the current solution if it can achieve a more optimized solution in the next steps.

Fiduccia-Matteyses is an iterative heuristic. The initial solution is generated randomly. In [2] there are some experiences showing that the random algorithm is a good starting point to FM since the partitions have similar number of cells. The algorithm works with single movements of cells from one partition to another. The net cut is a metric to evaluate if a movement is good or not.

FM uses a balance criterion function to the area equilibrium. If the function were not used probably all cells would be put in just one partition, which is a valid partitioning with a zero cut! The algorithm will not allow movements that violate the balance criterion.

4. CREATING NEW CIRCUITS

This is the last step of the Apple Parrot. Each partition is converted to a Spice netlist file. The main tasks of this step are:

1. Converting nets linking different partitions in input/output pins;
2. Distributing the original circuit's pins among the new circuits;
3. Identifying where every new Spice netlist file must be placed in the full circuit.

Three things must be considered to create i/o pins from nets: the name, the side and the type. The name is the net name. The side is chosen by an arithmetic average function considering the position of all cells connected. The block position is given by the quadratic organization of the partitions. The type is chosen evaluating in what

partition is located the net driver. If it is on the current partition, the pin will be an output pin. Otherwise, the pin will be an input pin.

The position of the block can be identified adding an ID in the file's name. Each new circuit has a code that is generated during the partitioning process. Every time a cell block is divided, a character from '1' to '4' in its last position increases its code. It is showed in the figure 2

5. RESULTS

Six circuits were submitted to the Apple Parrot partitioner. The partitioning was performed in a Pentium III 450 MHz. The results are shown in table 1. SD is the standard deviation.

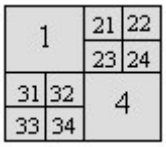

Fig. 2 – Block Position Identification

Tab. 1 – Partitioning results

Circuit	Number of partitions	CPU time (s)	Cut (nets)		Cells/partition		Estimated Area/partition (μm²)	
			Average	SD	Average	SD	Average	SD
Alu4 (5142 cells)	4	367	242.00	20.65	1285.50	2.06	83880.00	64.52
	16	404	260.25	18.21	321.38	2.32	20970.00	59.53
Apex1 (7172 cells)	4	907	380.50	20.16	1793.00	5.20	117997.50	53.56
	16	993	375.00	22.27	448.25	4.29	29499.38	46.50
Apex7 (2225 cells)	4	52	166.50	4.50	556.25	6.42	35943.75	54.47
	16	85	244.38	14.75	139.06	3.36	8985.94	46.98
Misex3 (5477 cells)	4	463	567.50	11.54	1369.25	10.1	40181.25	35.77
	16	521	519.12	29.18	342.31	6.18	10045.31	35.07
C6288_2x2ce (3073 cells)	4	201	266.50	7.76	768.25	5.36	52211.25	118.29
	16	225	122.25	11.69	192.06	2.41	13052.81	95.79
C1355_2x2ce (629 cells)	4	9	69.5	4.15	157.25	6.34	10961.24	50.17
	16	12	31.75	6.07	39.31	2.14	2740.31	85.30

The table above shows that the balance criterion is well succeeded to keep the partitions areas with approximately the same size and same number of cells. This result does not depend on the number of sub-circuits created neither on the circuit. It is good because all circuits created by the partitioning will have similar placement complexity.

However the cut is too high. For the circuit Alu4 about 250 I/O pins must be created in each partition to connect the cells. By the experiments in table 2, decreasing the number of cells in each partition does not decrease the cut of the partitioning.

6. CONCLUSIONS AND FUTURE WORKS

This paper presents Apple Parrot, a partitioning tool to decrease the complexity of the placement problem of large circuits. The experimental results show that the tool is able to partition a design in several sub-circuits with similar area and similar number of cells

The proposed method generates sub-circuits with I/O pins that can be independently placed and routed. After the process, the blocks must be placed to complete the routing. The Apple Parrot suggests one placement derived from the quadratic arrangement of the partitions.

As limitations of our method, we see that the cut is still too high.. As future work, we are planning to verify this by running the full Parrot tool kit that, in addition to the Apple Parrot, it includes a cell placer, a layout generator and router. Also, we observed that we were not able to reduce the cut by decreasing the partition size. That is a huge limitation and it must be solved.

Also, as future work we will experiment other heuristics for bi-partitioning.. Other attempt to reduce the net cut is by the implementation of terminal propagation. This would make cells connected stay near each other in neighbor partitions.

7. REFERENCES

[1] CALDWELL, A.; KAHNG, A.; MARKOV, I. **Can Recursive Bisection Alone Produce Routable Placements?**. In: DESIGN AUTOMATION CONFERENCE, DAC, 37., 2000, Los Angeles.

[2] HENTSCHKE, R. **Algoritmos para o Posicionamento de Células em Circuitos VLSI**. Dissertação de Mestrado. PPGC – UFRGS, 2002.

[3] PARAKN P., BROWN R., SAKALLAH K. **Congestion Driven Quadratic Placement**. In Proceedings of 35th Design Automation Conference IEEE/ACM, 1998.

[4] SAIT S., YOUSSEF H. **VLSI Physical Design Automation – Theory and Practice**. New York: IEEE, 1995.

[5] WANG M., LIM S., CONG J. , SARRAFZADEH M. **Multi-way Partitioning Using Bi-partition Heuristics**. In: ASIA AND SOUTH PACIFIC DESIGN AUTOMATION CONFERENCE, 2000

[6] YILDIZ, M.; MADDEN, P. **Improved Cut Sequences for Partitioning Based Placement**. DESIGN AUTOMATION CONFERENCE, DAC, 38., 2001, Las Vegas.