FPGA PLACE AND ROUTE ALGORITHM ANALYSIS USING KL-ALGORITHM

Rachana Borghate[1], Samrudhi Shirgaonkar[2], Apurva Ganar[3]
Lecturer, Electronics Enggineering, RGCER, Nagpur, India[1]
Lecturer, Electronics Engineering, SRMCEW, Nagpur, India[2]
Lecturer, Electronics Enggineering, RGCER, Nagpur, India[3]

Abstract: In the fast growing communication field, requirements of minimization are increasing to reduce the cost and timing of the integrated circuit. Efficient placement and routing algorithms play an important role in FPGA architecture research. Together, the place and route algorithms are responsible for producing a physical implementation of an application circuit on the FPGA hardware. The KL- Algorithm along with the reduction in the circuit as well as the implementation of the algorithm is shown using processor design which further reduces the cost and power increases the performances.

Keywords: KL algorithm, Place and Route, FPGA

1. INTRODUCTION

Improvements in computer performance have traditionally been the result of higher clock speed, pipelining, cache memories, multiple execution units, hardwired control units, and RISC. With the appearance of reconfigurable devices such as field programmable gate arrays (FPGAs), a new means for increasing computer performance has become possible. High-quality partitioning is necessary to achieve acceptable utilization of the reconfigurable resources. The goal of this paper is to look at the partitioning problem for Digital Logic Circuit implemented on FPGA and present algorithm to solve variants of the partitioning problem specific to particular digital logic circuit.

The need for software tools is because of the complexity of the circuitry within the FPGA, and the function the designer wishes to perform. **Place and route** is a stage in the design of printed circuit board, integrated circuits, and field-programmable gate arrays. As implied by the name, it is composed of two steps, placement and routing [1]. The first step, placement, involves deciding where to place all electronic components circuitry, and logic elements in a generally limited amount of space. This is followed by routing, which decides the exact design of all the wires needed to connect the placed components. This step must implement all the desired connections while following the rules and limitations of the manufacturing process.

The FPGA is designed using logic diagrams containing both digital logic and Very High Speed Integrated Circuits Hardware Description Language (VHDL), or Verilog [9]. These will then be put through an automated place-and-route procedure to generate a pin out, which will be used to interface with the parts outside of the FPGA. The final layout of early ICs and PCBs was stored as a tape out of Rubylith on transparent film.

Gradually, electronic design automation automated more and more of the place and route work [10]. At first, it merely sped up the process of making many small edits without spending a lot of time peeling up and sticking down the tape.

Later design rule checking speed up the process of checking for the most common sorts of errors.

Later auto routers speed up the process of routing.

Some people hope that further improvements in auto-placers and auto-routers will eventually

produce good layouts without any human manual intervention [5].

Probably the most successful heuristic for partitioning large graphs is the multilevel graph partitioning approach which was initially introduced to the graph partitioning field in the nineties by Barnard and Simon [18] to speed up spectral partitioning techniques.

After applying an initial partitioning algorithm to the smallest graph, the contraction is undone and, at each level, a local search method is used to improve the partitioning induced by the coarser level [16].

High-quality partitioning is critical in high-level synthesis. To be useful, high level synthesis algorithms should be able to handle very large systems. Typically, designer‟s partition high-level design specifications manually into procedures, each of which is then synthesized individually [19]. Generally the FPGA design-flow map designs onto an SRAM-based FPGA consist of three phases. The first phase uses synthesizer which is used to transform a circuit model coded in a hardware description language into an RTL design. The second phase uses a technology mapper which transforms the RTL design into a gate-level model composed of look-up tables (LUTs) and flip flops

(FFs) and it binds them to the FPGA‟s resources (producing the technology-mapped design). During the third phase, the place and route algorithm use the technology-mapped design to implement on FPGA.

The routing and placing operations may require a long time for execution in case of complex digital systems, because complex operations are required to determine and configure the required logical blocks within the programmable logic device, to interconnect them correctly, and to verify that the performance requirements specified during the design are ensured. The delay introduced by logic block and the delay introduced by interconnection can be analyzed by the use of efficient place and route algorithm.

The placement algorithms use a set of fixed modules and the netlist describing the connections between the various modules as their input. The output of the algorithms is the best possible position for each module based on various cost functions. We can have one or more cost functions depending on designs.

2. PLACING AND ROUTING

These operations are performed when an FPGA device is used for implementation. For designing, Placing is the process of selecting particular modules or logical blocks of the programmable logic device which will be used for implementing the various functions of the digital system. Routing consists in interconnecting these logical blocks using the available routing resources of the device.

3. PARTITIONING

Partitioning is a problem with any design using more than one component. It is a particularly interesting problem in embedded systems because of the heterogeneous hardware/software mixture. Partitioning can be classified on four main characteristics:

- The specifications model supported
- The granularity
- The cost function
- The algorithm

Explored algorithm classes include greedy heuristics, clustering methods, iterative improvement and mathematical programming. Partitioning is done together with scheduling, since the overall goal is to minimize response time. An initial partitioning is obtained by classifying blocks according to whether or not they are synthesizable and whether or not the communication overhead justifies a hardware implementation. The initial partitioning is then improved by Kernighan-Lin iterative swapping process.

Originally, this algorithm was developed for a formulation of the circuit partitioning problem (Kernighan and Lin, 1970). Its aim is to partition a graph into two parts of equal size with a minimal number of cutting edges. It is a so-called iterative improvement algorithm, meaning that it starts from an arbitrary partition, and swaps pairs of nodes in order to improve the cost of the partition The reason for the success of the KL is that it is fast as a greedy algorithm.

3.1 Partitioning Algorithm

Basic purpose of partitioning is to simplify the overall design process. The circuit is decomposed into several sub circuit to make the design process manageable.

Now partitioning can be done at different levels.

1. System Level Partitioning.
2. Board Level Partitioning.
3. Chip Level Partitioning.

At the system level, a system is partition into a set of subsystems such that each subsystem can be designed and fabricated independently on a single PCB. At the board level, the circuit assigned to a PCB is partitioned into sub circuits such that each sub circuit can be fabricated as a VLSI chip. And then, the circuit assigned to a chip is partitioned into smaller sub circuits. Physical partitioning at the chip level is not necessary.

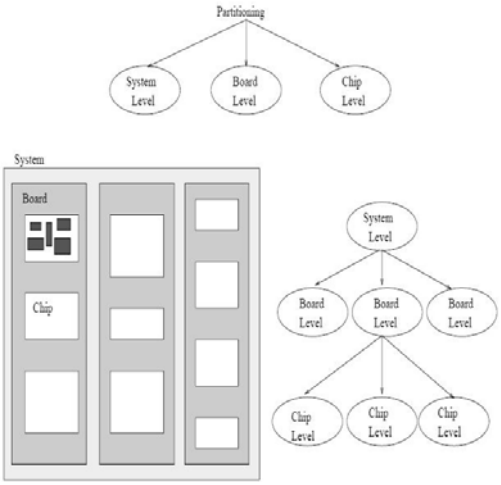

Fig 1 Partitioning done at different levels.

The partitioning problem can be formulated as follows: the objective is to minimize the number of interconnects between partitions and minimize the delay due to partitioning. These objectives need to be achieved while satisfying the following constraints the number of terminals in each subsystem should be less than a max value, there is an upper limit and lower limit on the area of each partition and number of partitions. The most important constraint is that the critical path should not cut partition boundaries.

Decomposition of a complex system into smaller subsystems, each subsystem can be designed independently speeding up the design process. Decomposition scheme has to minimize the interconnections among the subsystems. Decomposition is carried out hierarchically until each subsystem is of manageable size.

Here we have two sample placement results. If you notice the IO pads for the placement on the figure 2.a) are distributed along the periphery, while for the placement on the figure2.b) the IO pads are clustered in one place. As you can see there is no congestion for wire routing so that avoids any hotspots, we are using shorter wires which reduce area, power and delay. It also reduces the number of metal levels. On the other hand, for the bad placement we have congestion, long wire length, more delays, more metal level and higher power dissipation leading to an inefficient design.

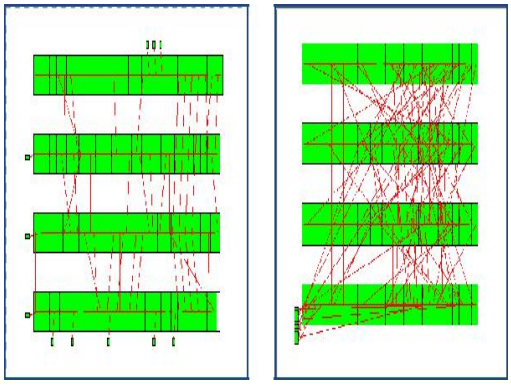

Fig 2.a) Good placement Fig 2.b) Bad placement

Importance of Circuit Partitioning:-

1. Divide-and-conquer methodology - The most effective way to solve problems of high complexity E.g.: min-cut based placement, partitioning-based test generation,

2. System-level partitioning for multi-chip designs inter-chip interconnection delay dominates system performance.

3. Circuit emulation/parallel simulation partition large circuit into multiple FPGAs (e.g. Quickturn), or multiple special-purpose processors (e.g. Zycad).

4. Parallel CAD development Task decomposition and load balancing

Different Partitioning methods: Top-down partitioning

- Iterative improvement
- Spectral based
- Clustering methods
- Network flow based
- Analytical based
- Multi-level

Bottom-up clustering

- Unit delay model
- General delay model
- Sequential circuits with retiming

Partitioning is assigning of logical component to physical packages:

1. Circuit is too large to be placed on a single chip.
2. I/O pins limitation.

Iterative Partitioning Algorithms:-

1. Greedy iterative improvement method
   - [Kernighan-Lin 1970]
   - [Fiduccia-Mattheyses 1982]
   - [krishnamurthy 1984]
2. Simulated Annealing
   - [Kirkpartrick-Gelatt-Vecchi 1983]

4. KERNIGHAN-LIN (KL) ALGORITHM

The Kernighan-Lin algorithm is used to view external against internal cost between nodes of a graph that emerges from the given dataset to be clustered. The figure below, figure 3, shows two partitions; Partition A and Partition B. The nodes of the graph are assigned to each of the partitions. The internal cost is the cost of an edge between two nodes within the same partition. The external cost is the cost of the edge of a node in one partition to a node in the other partition. The Kernighan-Lin algorithm and used in this thesis uses two partitions only. The cost within each partition, the internal cost, is rated lower than the cost between nodes across the partition, the external cost.

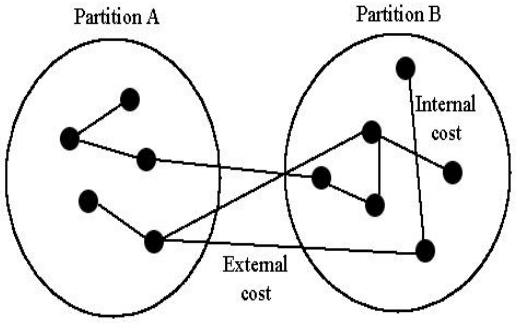

Fig 3 Internal Cost Vs External Cost

The algorithm tries to move each node between partitions so that the graph is maintained but with as low cost, the total sum of internal and external, as possible. Thus it means that the internal cost should be maximized while the external cost is minimized. The Kernighan-Lin heuristic partitioning of interconnected graphs has been used in circuit partitioning for a time. And it has been proven to be very successful. The results are close to the optimum and the time-complexity is almost linear.

The Kernighan-Lin can be easily implemented in hardware or software or a combination of the two, and over time the Kernighan-Lin has proven difficult to outperform. There are faster algorithms, and more accurate but as a combination Kernighan-Lin is one of the best. The results for Kernighan-Lin algorithm are a result of an extension provided by Fidduccia/Mattheyses and their addition made the Kernighan-Lin run in linear-time.

The K-L (Kernighnan-Lin) algorithm was used for bisecting graph in VLSI layout which was first suggested in 1970. The algorithm is an iterative algorithm; which Starts from a load balanced initial bisection, it will first calculate each vertex gain in the reduction of edge-cut that may result if that vertex is moved from one partition of the graph to the other. For every inner iteration it moves the unlocked vertex having the highest gain, from the partition with more vertices to the partition which it requires which has less in number. Then the vertex is locked and the gains are updated.

The procedure is repeated until all of the vertices are locked even if the highest gain may be negative. The last few moves that had negative gains are then undone and the bisection is reverted to the one with the smallest edge-cut so far in this iteration. Here one outer iteration of the K-L algorithm is completed and the iterative procedure is restarted again. If an outer iteration will results in no reduction in the edge cut or load imbalance, then the algorithm is terminated.

Node a	ALU
Node b	Accumulator
Node c	Temporary register
Node d	Register Bank
Node e	Control Unit
Node f	Memory

If an outer iteration gives no reduction in the edge-cut or load imbalance, the algorithm is terminated. The K-L algorithm is a local optimization algorithm, with a capability for getting moves with negative gain.

4.1. How KL Works

Let we have a graph G (V, E), and let V be the set of nodes and the E set of edges.

The algorithm attempts to find a partition of V into two disjoint subsets A and B of equal size, or unequal such that the sum T of the weights of the edges between nodes in A and B is minimized.

Let I_a be the internal cost of a, that is, the sum of the costs of edges between a and other nodes in A, and let $E_a$be be the external cost of a, that is, the sum of the costs of edges between a and nodes in B.

Furthermore, let D_a, $D_a = E_a - I_a$ be the difference between the external and internal costs of a. If a and b are interchanged, then the reduction in cost is

$$T_{old} - T_{new} = Da + Db - 2Ca,b$$

Where $C_{a,b}$ is the cost of the possible edge between a and b.

The algorithm will try attempts to find an optimal series of interchange operations between elements of A and B which maximizes $T_{old} - T_{new}$ and then executes the operations, producing a partition of the graph to A and B[5].

We can try all possible bisections. Choose the best one. If there are 2n vertices, then numbers of possibilities are $(2n)! / 2(n!)^2$ .For 4 vertices (A, B, C, D), possibilities are three:

1. X = (A, B) and Y = (C, D)
2. X = (A, C) and Y = (B, D)
3. X = (A, D) and Y = (B, C)

4.2KL Algorithm Implementation A Processor Design:

A weighted graph G = (V, E) with Vertex set V. (|V| = 2n) Edge Set E. (|E| = e) Cost c(A,B) for each edge {A, B} in E. These are the required inputs and the outputs are 2 partitions X & Y such that total cost of edges "crossing" the partition are minimized. Each partition has n vertices. Try all possible bisections. Choose the best one. If there are 2n vertices, then number of possibilities is $(2n)! / 2(n!)^2$. For 4 vertices (A,B,C,D), possibilities are three:

1. X = {A, B} & Y = {C, D}
2. X = {A, C} & Y = {B, D}
3. X = {A, D} & Y = {B, C}

Start with any initial legal partitions X and Y.

Now the processor design application was converted to six nodes as mentioned above in the designing aspects according to KL algorithm. The nodes are

Table 1 Assigning nodes to the Processor

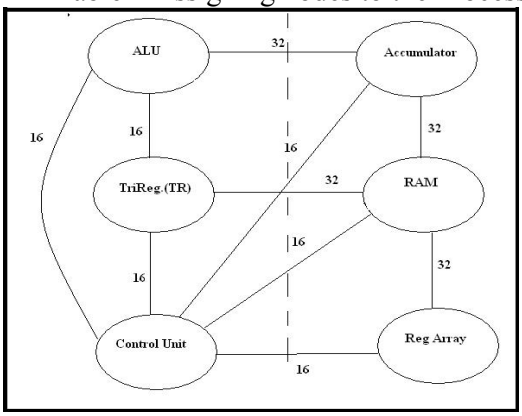

Fig 4 Processor Design before applying algorithm

In the above initial cut size = (32+16+32+16+16) = 112

Initial weighted graph G with V (G) = {a, b, c, d, e, f} Start with any partition of V (G) into A and B, say A= {a, c, e} B = {b, d, f}

Compute the D-values

$D_a = E_a - I_a = 0$ (= 32 – 16 – 16)
$D_c = E_c - I_c = 0$ (= 32 –16 – 16) D_e = $E_e - I_e$ = + 16 (= 48 – 32) $D_b = E_b$

– I_b = + 16 (= 48 –32) $D_d = E_d$
– I_d = – 16 (= 48 – 64) $D_f = E_f$
– I_f = – 16 (= 16– 32) Compute the gains

$g_{ab} = D_a + D_b – 2w_{ab}$ = 0+ 16 – 2□32 = –48 $g_{ad} = D_a + D_d – 2w_{ad}$ = 0 – 16 – 2□0 = –16 $g_{af} = D_a + D_f – 2w_{af}$ = 0 – 16 – 2□0 = – 16 $g_{cb} = D_c + D_b – 2w_{cb}$ = 0 + 16 – 2□0 = +16 $g_{cd} = D_c + D_d – 2w_{cd}$ = 0 – 16 – 2□32 = –80 $g_{cf} = D_c + D_f – 2w_{cf}$ = 0 – 16 – 2□0 = – 16 g_{eb} = $D_e + D_b – 2w_{eb}$ = +16 + 16 – 2□16 =0 $g_{ed} = D_e + D_d – 2w_{ed}$ = +16 – 16 – 2□16 = – 32 $g_{ef} = D_e + D_f – 2w_{ef}$ = +16 – 16 – 2□16 = – 32

We observe from figure 4 that the cutsize initially is 112 and thereafter finding the highest gain and swapping the nodes we had got the cutsize to be 96 as shown in fig.5

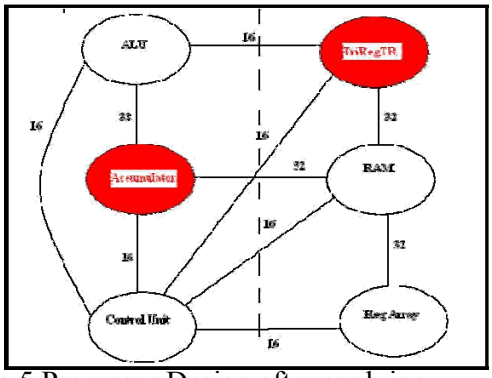

Fig 5 Processor Design after applying algorithm

From the above fig cut size = (16+16+32+16+16) =96

5. RESULTS AND CONCLUSION

```
Initial partition cost = 112
The g matrix is:
   -80   -16   -48
   -32   -32     0
   -16   -16    16

iter = 1, nodes to be exchanged: 6, and 4 Max. g = 16
Updated D values are:
   -32   -16   -16   -16

*** Iteration 2 ***
The g matrix is:
   -48   -48
   -32   -96

iter = 2, nodes to be exchanged: 3, and 2 Max. g = -32
Updated D values are:
    32   -16

*** Iteration 3 ***
The g matrix is:
    16

iter = 3, nodes to be exchanged: 1, and 5 Max. g = 16
Decision ...
Exchange first 1 pairs of nodes
Final partition =
final partition cost = 16
Partition A: 1  3  4
Partition B: 2  5  6
```

Fig 6 Command window history showing Initial cost and final cost Here we have presented the implementation of KL-Algorithm along with the reduction in the circuit as well as the implementation of the algorithm using Processor design which further reduces the cost and power and increases the performances.

Thus it reduces the cost parameters for the digital logic circuit as it is observed by following the results of processor design. It optimized the area and further reduces the routing delay. The KL partitioning algorithm has been implemented and the result has been observed on the processor based design. Thus the algorithm helps to solve placement and routing problems to certain extent for the FPGA.

Also here we have tried o reduce time complexity because of KL- Algorithm initial bisection and cut size. The results show that cut size is reduced after applying the algorithm for the given circuits and the applied custom circuit.

6. CONCLUSION

The quality of the place-and-route algorithms has a direct bearing on the usefulness of the target FPGA architecture. The benefits of including powerful new features on an FPGA might be lost due to the inability of the place-and-route algorithms to fully exploit these features. Thus the advancement of FPGA architectures relies heavily on the development of efficient place-and-route algorithms. KL Algorithms increase the performance by reducing the wire delay. Further work is necessary in the use of *kl*-feasible cuts for the optimization purpose. Analysis of an efficient algorithm for Place and route process would be done, in order to place the components efficiently and create a proper routing path between them on FPGAs. In this paper we have presented a new methodology for Digital circuits here example is considered as multiplier, which in turns reduces the area and increases the performance of the circuit type algorithms for the problem of hardware/software partitioning.

7. REFERENCES

[1] Xilinx Inc., "Spartan-II 2.5 V FPGA Family: Introduction and Ordering Information,"
Xilinx Product Specification Datasheets, 2003

[2] Luca Sterpone, Student Member, IEEE,

and
Massimo Violante, Member, IEEE." A New
Reliability-Oriented Place and Route Algorithm for SRAM-Based FPGAs", IEEE TRANSACTIONS ON COMPUTERS, VOL. 55, NO. 6, JUNE 2006

[3] Chenguang Guo, Yanlong Zhang, Lei Chen, Tao Zhou, Xuewu Li, Min Wang, Zhiping
WenDept. FPGA „A Novel Application of
FPGA-Based PartialDynamic Reconfiguration
System with CBSC" 2012 IEEE

[4] Virtex-II Pro and Virtex-II Pro X Platform FPGAs: Complete DataSheet, Xilinx Corporation, DS083 (v4.7) Nov. 5, 2007.

[5] Osvaldo Martinello Jr, Felipe S. Marques,
Renato P. Ribas, André I. Reis "KL-Cuts:A New Approach for Logic Synthesis Targeting Multiple Output Blocks",777-782

[6] Amr M. Fahim, "Low-Power High-Performance Arithmetic Circuits and
Architectures", IEEE Journal of Solid-State Circuits, Vol. 37, No. 1, pp. 90-94, January 2002