International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 2 Issue 5, May - 2013

Place And Route Algorithm Analysis For Field Programmable Gate Array (FPGA) Using KL- Algorithm

Vaishali Udar
Department of Electronics and telecom. Engg
Priyadarshini college of Engineering
Nagpur, India

Sanjeev Sharma
Department of Electronics and telecom. Engg
Priyadarshini college of Engineering
Nagpur, India

Abstract*:* 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. This paper presents the KL- Algorithm along with the reduction in the circuit as well as the implementation of the algorithm using multiplier which further reduces the cost and power and increases the performances. In the fast growing communication field, requirements of minimization are increasing to reduce the cost and timing of the integrated circuit. The KL partitioning algorithm has been implemented and the result has been observed on the processor based design.

Keywords: KL algorithm, Place and Route, FPGA

I INTRODUCTION

The process of placing and routing for an FPGA is generally not performed by a person, but uses a tool provided by the FPGA Vendor or another software manufacturer. The need for software tools is because of the complexity of the circuitry within the FPGA, and the function the designer wishes to perform. 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. The cost functions include maximum total wire length, wire routability, congestions, and performance and I/O pads locations.

II 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.

III PARTITIONING ALGORITHM (K-L 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

A *Partitioning Algorithms:*

- Iterative partitioning algorithms
- Spectral based partitioning algorithms
- Net partitioning vs. module partitioning
- Multi-way partitioning
- Multi-level partitioning

B *Iterative Partitioning Algorithms*:-

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

IV KERNIGHAN-LIN (KL) ALGORITHM

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.

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.

A. *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 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

$$T_{old} - T_{new} = D_a + D_b - 2C_{a,b}$$

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)

B. KL Algorithm Implementation

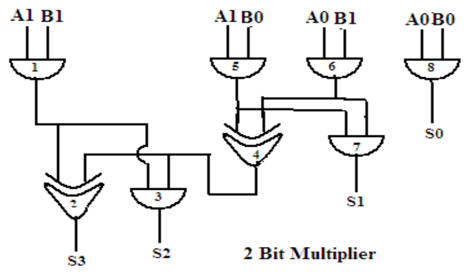

Figure 1 Example 2-Bit Multiplier

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

Node 1 And gate having input A_1B_1
Node 2 Xor gate having S_3 as output
Node 3 And gate having S_2 output.
Node 4 Xor gate
Node 5 And gate having input A_1B_0
Node 6 And gate having input A_0B_1
Node 7 And gate
Node 8 And gate having input A_0B_0

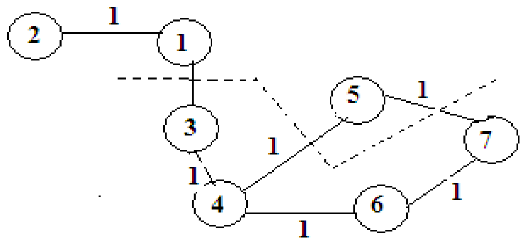

Figure 2 Cut size = 3,

C. Algorithm Steps

Step I: Initialization
Let the initial partition be a random division of vertices into the partition A= {1, 2, 3, 4} and B= {5, 6, 7, 8}.Here let A^l=A= {1, 2,5,8} and B^l = {3,4,6,7}

Step 2: Compute D - values.
$D_1 = E_1-I_1 = 1-1 = 0$
$D_2 = E_2-I_2 = 0-1 = -1$
$D_3 = E_3-I_3 =1-0 = 1$
$D_4 = E_4-I_4 = 1-2 = -1$
$D_5 = E_5-I_5 = 2-0 = 2$
$D_6 = E_6-I_6 = 0-2 = -2$
$D_7 = E_7-I_7 = 1-1 = 0$

Step 3: compute gains
$G_{23} = D_2+D_3-2C_{23} = -1+1-2(0) =0$
$G_{24} = D_2+D_4-2C_{24} =-1-1-2(0) =-2$
$G_{26} = D_2+D_6-2C_{26} = -1-2-2(0) = -3$
$G_{27} = D_2+D_7-2C_{27} =-1+0-2(0) = -1$
$G_{13} = D_{1+}D_3-2C_{13} = 0+1-2(1) = -1$
$G_{14} = D_1+D_4-2C_{14} = 0-1-2(0) = -1$
$G_{16} = D_1+D_6-2C_{16} = 0-2-2(0) = -2$
$G_{17} = D_1+D_7-2C_{17} = 0+0-2(0) = 0$
$\mathbf{G_{53} = D_5+D_3-2C_{53} = 2+1-2(0) = 3}$
$G_{54} = D_5+D4-2C_{54} = 2-1-2(1) = -1$
$G_{56} = D_5+D_6-2C_{56}= 2-2-2(0) = 0$
$G_{57} = D_5+D_7-2C_{57}= 2+0-2(1) = 0$

Largest value of G is G_{53}=3 here we consider (a_1 b_1) = (5, 3) and A^l = A^l - 5= (1, 2) and B^l = B^l - 3 = (4, 6, 7).New A^l = (1,2,8) and B^l = (4,6,7). Both A^l and B^l are not empty, and then we update D values in next step and repeat the procedure from step 3.

Step 4: Update D -values of nodes connected to (5, 3).
The vertices connected to (5, 3) are vertex (1) in set A^l and vertices (4, 7) in set B^l. The new D-values for vertices of A^l and B^l are given by:
$D_1^l= D_1+2(C_{13})-2(C_{15}) =2$
$D_4^l= D_4+2(C_{43})-2(C_{45}) = -1$
$D_7^l= D_7+2(C_{75})-2(C_{73}) =2$
$D_2^l= D_2+2(C_{52})-2(C_{23}) =-1$
$D_6^l= D_6+2(C_{63})-2(C_{65}) =-2$
Repeat step 3
$G_{24} = D_2^l + D_4^l -2C_{24} = -2$
$G_{26} = D_2^l + D_6^l -2C_{26} = -2$
$G_{27} = D_7^l + D_2^l -2C_{27} =1$
$G_{14}= D_1^l + D_4^l -2C_{14}= 1$
$G_{16}= D_1^l + D_6^l -2C_{16}= 0$
$\mathbf{G_{17}= D_1^l + D_7^l -2C_{17}= 4}$
Here the G value for G_{17} is large. Hence pair (a_2, b_2) is (1, 7).
$A^l = A^l$-1 = (2, 8) and $B^l = B^l$ – 7 = (4, 6).
The new D values are
$D_2^{ll}= D_2^l +2(C_{21})-2(C_{27}) = 1$
$D_4^{ll}= D_4^l +2(C_{47})-2(C_{41}) = -1$
$D_6^{ll}= D_6^l +2(C_{67})-2(C_{61}) = 0$
$G_{24}= D_2^{ll} + D_4^{ll} -2C_{24}= 0$
$G_{26}= D_2^{ll} + D_6^{ll} -2C_{26}= 1$
The last pair (a_3, b_3) is (1, 7) and the corresponding gain is G_{17}.

Step 5: determine the values of X and y
X = a_1 = 1 and Y = b_1=7
The new partition that will obtained from moving X to B and Y to A is A={1,2,5,8} and B={3,4,6,7}.
The entire procedure is repeated again with this new partition as the initial partition. Verify that the second iteration of the algorithm is also the last, and that the best solution obtained is A={1,2,5,8} and B={3,4,6,7}.
The overall procedure is repeated with gain having maximum value was taken and then the cut size was calculated. There after the second pass was implemented, here we had locked the nodes with the maximum gain. This process is repeated for all the

passes and combinations. At the end of the above process we get the minimum cut size which in turns reduces the wire delay and increases the performance.
We observe from figure 1 that the cut size initially is 3 and thereafter finding the highest gain and swapping the nodes we had got the final partition to be as required.

V CONCLUSIONS AND FUTURE WORK

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.

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.
[7] Patterson and Hennessy, Computer Organization and Design, 3rd Edition, Morgan Kaufman, 2005.
[8]John F. Wakerly “Digital Design Principles and Practices”.
[9]S. Brown, “FPGA Architecture Research: A Survey,” IEEE Design and Test of Computers, pp. 9-15, Nov./Dec. 1996.
[10] J. Rose, A. El Gamal, and A. Sangiovanni-Vincetelli, “Architecture of Field-Programmable Gate Arrays,” Proc. IEEE, vol. 81, no. 7, pp. 1013-1029, July 1993.
[11] 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.
[12] Xilinx, “Achieving Higher System Performance with the Virtex-5 Family of FPGAs”, White Paper, 2006. http://www.xilinx.com.
[13] Altera, “Improving FPGA Performance and Area Using an Adaptive Logic Module”, White Paper, 2004. http://www.altera.com.