AUSTRALIAN JOURNAL OF BASIC AND APPLIED SCIENCES

ISSN:1991-8178 EISSN: 2309-8414
Journal home page: www.ajbasweb.com

Design of FPGA Routing Architecture with Algorithmic Approach

[1]J. Jerline Mano and [2]V. Monisha

[1]*Dr.Sivanthi aditanar college of engineering*
[2]*Dr.Sivanthi aditanar college of engineering*

Address For Correspondence:
J. Jerline Mano, Dr.Sivanthi aditanar college of engineering.
E-mail: jerlinejackson@gmail.com

ARTICLE INFO

Article history:
Received 10 December 2015
Accepted 28 January 2016
Available online 10 February 2016

Keywords:
Computer-aided design (CAD), field-programmable gate array (FPGA) architecture, FPGA routing, satisfiability (SAT).

ABSTRACT

FPGA is grown to be an important technology in VLSI design, Implementation of various complicated reconfigurable Soc and Noc architecture designs are relied on FPGA platform due to its dynamic programmability. Many paper works for performing boosting of FPGA technologies are introduced throughout the literature FPGA path routing is one of the complex and time consuming routing architecture because of the boolean satisfiability criteria and the No of possible cluster connections. A method of Hybrid pathfinder style routing mechanism is proposed to group the wide wires of switch matrix without affecting the Boolean nature of the Logic to reduce the Routing delay and complexity.

INTRODUCTION

As Field –Programmable Gate Arrays (FPGA) becomes larger, the run required to execute the associated computer-aided design (CAD) tools gets worse. It can now take days to compile the largest industrial FPGA designs from hardware description level to bit stream. In the past, worsening CAD tool runtime was mitigated by increases in the uniprocessor clock speed. However, this is no longer the case. The high current density in modern processors has created a “power wall,” which restricts increases in processor clock speeds. The gap between the size of FPGA devices and the ability of CAD tools to handle them is widening with every process generation. In addition to decreasing productivity for hardware engineers, long runtimes are an impediment to the adoption of FPGAs by software developers, who are used to compilation times measured in seconds or minutes, not hours or days. Previous efforts to reduce FPGA CAD runtimes have focused on algorithmic changes (Swartz, J., 1998; Gort, M. and J.H. Anderson, 2012) or parallelization (Gort, M. and J.H. Anderson, 2010; Wang, C.C. and G.G. Lemieux, 2011 2010). In this paper, we reduce router runtime via a combined CAD and architecture approach. Interactions between CAD and architecture are known to affect FPGA speed, area, and power, however, the impact of these interactions on runtime has not been well studied. We propose a new two-stage FPGA routing algorithm that takes advantage of increased flexibility in the FPGA switch block (SB), resulting in reduced router runtime.

At a high level, our routing approach works as follows: first, a PathFinder-style router (McMurchie, L. and C. Ebeling, 1995) assigns signals to small groups of wire segments, called wide wires, rather than to single wire segments. This is made possible by coarsening the routing resource (RR) graph, which allows PathFinder to terminate early. Second, an embedding stage assigns each signal from a wide wire onto one of the wire segments contained within. We express the embedding problem as a Boolean satisfiability (SAT) problem and use a SAT solver (Luu, J., 2009) to solve it. Our approach bears some resemblance to two-stage global-detailed routing,

To Cite This Article: J. Jerline Mano and V. Monisha., Design of FPGA Routing Architecture with Algorithmic Approach. ***Aust. J. Basic & Appl. Sci.,*** *10(1): 358-363, 2016*

Australian Journal of Basic and Applied Sciences, 10(1) January 2016, Pages: 358-363

where global routing assigns sig- nals to entire FPGA channels, and detailed routing assigns signals to wire segments within those channels.

II. Back Ground:

A.FPGA Architecture:

Fig. 1 depicts a basic island-style FPGA. Each LB has a capacity, which represents the number of lookup tables (LUTs) and flip-flops that can be contained therein. LBs connect to wire segments through programmable switches in the connection blocks (CBs). Wire segments connect to other wire segments using programmable switches in the SBs. Connections between LBs are made by turning on the appropriate switches in the SBs and CBs. The degree of connectivity inside SBs and CBs is described using the parameters Fs and Fc , respectively. Fs describes the number of wire segments coming out of an SB than can be reached by each wire segment coming into an SB, on average. In other words, it represents the average fanout of each wire coming into an SB. Fc describes the fraction of wire segments in a channel that can be directly reached by an LB pin through a CB. Fcin refers to the connectivity to LB input pins while Fcout refers to the connectivity to LB output pins.

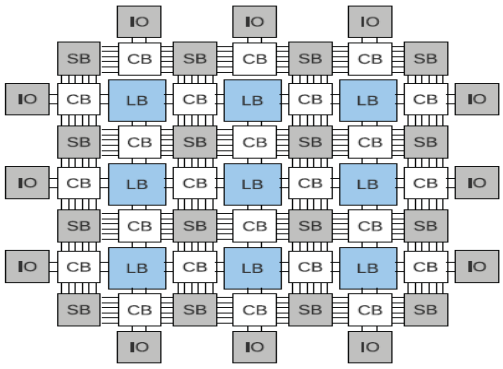


Fig. 1: FPGA Architecture.

B. FPGA Routing:

FPGA routing is one of the most time-consuming stages in the CAD flow. It is responsible for finding routes for connections between LBs, using wire segments and program- mable switches. Naturally, no two signals may use the same wire segment. The two largest FPGA vendors, Xilinx and Altera, use a variant of the PathFinder negotiated congestion routing algorithm in their commercial routers .PathFinder is also used in the publicly available VPR FPGA placement and routing framework, which we modified and used in this paper.First, all signals in a placed design are routed in the best manner possible (e.g., minimum delay), permitting shorts between the signals (two or more signals may use the same wire). Then, the penalties associated with the shorts are increased, and the signals are rerouted, avoiding shorts where possible. The process of increasing the penalties for shorts and rerouting the signals continues iteratively until all shorts are removed and the routing is feasible.VPR PathFinder uses a RR graph to represent the FPGA interconnect. Graph nodes represent wire segments, input pins, or output pins. Programmable switches between wires or pins are represented as edges between the nodes. Each node has a capacity, which indicates the number of signals it can accommodate, and an occupancy, which indicates the number of signals that currently occupy the RR node. Routing continues until the occupancy of each node is not greater that its capacity, at which point the routing is feasible.

III. Design Flow:

A. Embedding Stage:

The output of the coarsened routing stage is not a complete routing solution because signals are not assigned to individual wire segments. A decoarsening or embedding stage must be included in order to generate a legal detailed routing solution from the partial solution, assigning each signal to a single wire segment. The embedding difficulty depends on the connection patterns of the underlying routing architecture (Gort, M. and J.H. Anderson, 2012). If the routing architecture has full connectivity between wires within each wide wire, embedding is trivial, since any assignment of signals to tracks is legal. On the other hand, if only one-to-one connectivity exists, meaning that a wire segment within a wide wire only connects to a single wire within a different wide wire, then it is possible that no legal embedding exists. In such a case, Pathfinder could be rerun on an uncoarsened RR graph. A routing with no legal embedding the routes through an SB of four signals ($n1$, $n2$, $n3$, $n4$). In this fig, solid lines indicate wide wires with $n = 2$. Each is labeled with the signals (up to 2) that use it. The wide wires have been expanded (flattened) so that, each solid line represents one wire segment.

Observe that no assignment of signals to wire segments can avoid shorts. However, by adding some flexibility to the connectivity between the wide wires.

It is possible to arrive at a legal embedding. Speculate that an intermediate amount of connectivity will allow the vast majority of coarse routing solutions to be embedded while avoiding the significant area overhead of having full connectivity between constituent wire segments of connected wide wires.

B. Sat formulation:

Formulate the embedding problem as a Boolean SAT problem in the CNF form. Formulation is inspired by the SAT-based detailed routing approach presented in which, given a global routing, formulates detailed routing as a SAT problem instance. The two types of clauses to represent routing constraints exclusivity constraints and liveness constraints. SAT formulation uses the same types of constraints; though express them differently to best serve embedding problem.

Fig. 2 shows the flow used to evaluate connectivity patterns. Before beginning this flow, we generate coarse routes for each benchmark using VPR routing, for each value of *n* that explore. Regenerating coarse routes saves experimentation time, since the first phase of routing does not have to be performed for every SB connectivity pattern that explore. The first step in the flow, after having generated coarse routes, is to generate a new SB architecture using our Monte Carlo SB generator. A SAT formulation is then generated using the coarse route of each benchmark and the SB architecture (Betz, V., 2008). The SAT formulation is then provided as input which then produces either a legal routing or else an UNSAT result.

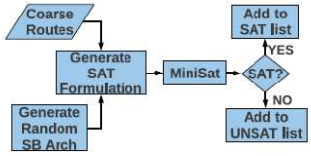


Fig. 2: switch block connection flow.

C. Pathfinder style routing algorithm:

The pathfinder algorithm is based on the maze router, but speeds up the algorithm by routing every connection on a free obstacle environment and allowing routing resources to be overused.

After a single iteration of the algorithm, all nets are routed once as if they were the only connection to be routed; and the cost of using every resource is calculated according to its demand. Subsequent iterations rip up and re-route all nets (McMurchie, L. and C. Ebeling, 1995), and the process goes on until no overuse of routing resources exist. This process of ripping out and re-routing every n*et al*lows the pathfinder algorithm to minimize the net ordering problem of the maze routing. It is responsible for finding routes for connections between LBs, using wire segments and programmable switches. Naturally, no two signals may use the same wire segment. The two largest FPGA vendors, Xilinx and Altera, use a variant of the PathFinder negotiated congestion routing algorithm in their commercial routers.

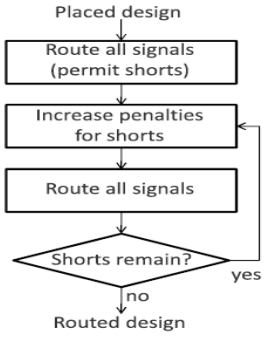


Fig. 3: Negotiated congestion routing flow.

Fig. 3 gives an overview of the negotiated congestion approach used in the VPR PathFinder. First, all signals in a placed design are routed in the best manner possible (e.g., minimum delay), permitting shorts between the signals (two or more signals may use the same wire). Then, the penalties associated with the shorts are increased, and the signals are rerouted, avoiding shorts where possible. The process of increasing the penalties for shorts and rerouting the signals continues iteratively until all shorts are removed and the routing is feasible (Luu, J., 2009).

PathFinder uses an iterative framework to achieve eliminating congestions and minimizing delay of critical paths simultaneously. Unlike the typical two step routing approaches, this algorithm tries both global and

detailed routing at the same time. It starts by constructing a routing graph whose topology mirrors the complete FPGA routing architecture. Thus, paths in this graph correspond to feasible routes in the FPGA. Using this graph, PathFinder uses sophisticated cost functions to search for routing solutions (i.e., paths in the graph) for all the nets in a circuit.

Although routing resources are initially allowed to be shared among nets, these are eventually assigned to most demanding signals in subsequent iterations. The negotiation process through cost functions on congested routing resources determines which net needs the corresponding routing resource most. This iteration is repeated until no shared routing resource exists. A timing analysis is also performed at every iteration to set a higher priority for more timely critical nets. In spite of the fact that all the routes are rerouted in every iteration, the reported results were excellent achieving a higher degree of routability as well as shorter critical paths on commercial circuits.

If a routing solution exists for a given circuit and placement, Pathfinder will eventually converge toward it after a sufficient number of iterations. If the problem is unroutable, however, the algorithm may not converge and execution must be aborted after a suitable time limit. This convergence problem is inherent to all heuristic one-net-at-a time routing algorithms: these algorithms cannot decide the routability of the given circuit placement. Even when only a few nets are not routed in a circuit, these algorithms must resort to time-consuming rip-up-reroute procedures which may or may not lead to a routing solution. The key observation of the hybrid algorithm search-SAT is that by combining the PathFinder algorithm with a Boolean SAT-based routing formulation, the convergence drawback of PathFinder can be corrected and the applicable circuit size is not limited to small-scaled instances.

Routing is an important step in the FPGA tool flow. FPGAs have a finite number of discrete routing resources, and the effectiveness of an FPGA router directly impacts the performance of an application net list on a target device. Path finder is the current, state-of-the-art FPGA routing algorithm. Pathfinder uses an iterative, negotiation-based approach to solve the FPGA routing problem. During the first routing iteration, nets are freely routed without paying attention to resource sharing. Individual nets are routed using a shortest path graph algorithm. At the end of the first iteration, resources are generally congested because multiple nets have shared them. During subsequent iterations, the cost of using a resource is increased based on the number of nets that share the resource, and the history of congestion on that resource.

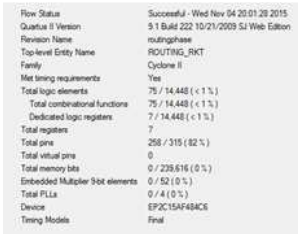
In effect, nets are made to negotiate for routing resources. If a resource is highly congested, nets that can use lower congestion alternatives are forced to do so. On the other hand, if the alternatives are more congested than the resource, then a net may still use that resource. Pathfinder has proved to be one of the most powerful FPGA routing algorithms to date. Pathfinder's negotiation-based framework is a very effective technique for routing nets on FPGAs. More importantly, Pathfinder is a truly architecture adaptive routing algorithm. The algorithm operates on a directed graph abstraction of an FPGA's interconnect structure, and can thus be used to route net lists on any FPGA that can be represented as a directed routing graph. Pathfinder's adaptability is one of the main reasons for its widespread acceptance.

D.Hybrid Path finder style routing algorithm:

Including one small term hybrid path finder style routing algorithm. The difference between path finder and hybrid path finder is providing the source and destination. From that path finder are not giving any source and destination value .But in hybrid path finder style routing already giving the source and destination. In this path finder if any error occur in between the source and destination means its once again repeat the signal back to the sender .Then the sender send the data to another way. But in this hybrid path finder style routing whether any error occur, it will be take another way to reach thedestination there is no need to send the data back to sender.

IV. Simulation Results:

Hybrid pathfinder style routing:



Flow Status	Successful - Wed Nov 04 20:01:28 2015
Quartus II Version	9.1 Build 222 10/21/2009 SJ Web Edition
Revision Name	routingphase
Top-level Entity Name	ROUTING_RKT
Family	Cyclone II
Met timing requirements	Yes
Total logic elements	75 / 14,448 (< 1 %)
Total combinational functions	75 / 14,448 (< 1 %)
Dedicated logic registers	7 / 14,448 (< 1 %)
Total registers	7
Total pins	258 / 315 (82 %)
Total virtual pins	0
Total memory bits	0 / 239,616 (0 %)
Embedded Multiplier 9-bit elements	0 / 52 (0 %)
Total PLLs	0 / 4 (0 %)
Device	EP2C15AF484C6
Timing Models	Final

Fig. 4: Flow Summary report of hybrid pathfinder style routing.

Hybrid pathfinder style algorithm use 75 total logic element the total register is 7.Total pin is 258

Hybrid pathfinder style algorithm setup time is 7.701ns clock to output time 7.369ns hold time is 3.318ns

Hybrid pathfinder style static thermal power dissipation is 47.40mW. Dynamic thermal power dissipation is 0mW. Input output thermal power dissipation is 49.75mW.Total thermal power dissipation is 97.16mW.

Table 1.shows time power and area of hybrid path finder routing style algorithm .Total power dissipation is 97.16,setup time is 7.071ns and total LUT is 75.

Timing Analyzer Summary

	Type	Actual Time
1	Worst-case tsu	7.071 ns
2	Worst-case tco	7.369 ns
3	Worst-case th	3.318 ns
4	Clock Setup: 'clk'	Restricted to 420.17 MHz (period = 2.380 ns)
5	Total number of failed paths	

Fig. 5: Timing summary report of hybrid pathfinder style routing.

PowerPlay Power Analyzer Status	Successful - Wed Nov 04 20:05:47 2015
Quartus II Version	9.1 Build 222 10/21/2009 SJ Web Edition
Revision Name	routingphase
Top-level Entity Name	ROUTING_RKT
Family	Cyclone II
Device	EP2C15AF484C6
Power Models	Final
Total Thermal Power Dissipation	97.16 mW
Core Dynamic Thermal Power Dissipation	0.00 mW
Core Static Thermal Power Dissipation	47.40 mW
I/O Thermal Power Dissipation	49.75 mW
Power Estimation Confidence	Low: user provided insufficient toggle rate data

Fig. 6: Power summary report of hybrid pathfinder style routing.

Table I: Power,Area And Time.

Method	Area (LUT)	Time (ns)	Power (mw)
Hybrid pathfinder style routing	75	7.071	97.16

Above the simulation using benchmark file is Intelligent Storage Elements s298.This benchmark totally have 75 gates. When the clock, enable input is 1 the routing is done parallel. Right hand side it shows the shortest path of our input file. The difference between path finder to hybrid pathfinder is the input, source and destination was given only. The bench mark which has the one big circuit from that the value of clk ,enable, a in, bin, add out which can be given by directly .All the connection is controlled by the switch matrix for purpose using 8*8 matrix for 2d clusterarray .The memory element also storing the value of the result. When the clock, enable input is 1 all the operation done in parallelly how they can reach the destination. After that grouping is completed. Then the result of shortest path is determined. Then the shortest path value in the rage of 1001, 1010, 0110, 1111, 0101,1001,0111,0100

Fig. 7: Hybrid pathfinder style routing output structure.

Conclusion:

A successful demonstration of pathfinder style routing in FPGA architecture is designed in VHDL and simulated using MODELSIM tool. A dedicated routing logic which organize wide wire clustering by maintaining Boolean satisfiability all along. The performance for area, time and power is calculated using Quartus synthesizer tool, from these finally conclude that the adopted algorithm optimistically proven to be best in every aspect with the simplicity and efficiency of Path finder style algorithm. This technique is found to be simpler and highly efficient for Time and Power reduction

Future Work:

Since all the chip aspect like area, power and time are improved; future work can be focused to reduce the routing failure in FPGA nodes by deflecting into alternate path, resulting in the LE size reduction by modifying the Lookup table structure thus reducing repetition in LUT entry and improving the area utilization.

REFERENCES

Swartz, J., V. Betz, and J. Rose, 1998. "A fast routability- driven router forFPGAs," in Proc. ACM/SIGDA FPGA, 140–149.

Sankar, Y. and J. Rose, 1999. "Trading quality for compile time: Ultrafast placement for FPGAs," in Proc. ACM/SIGDA FPGA, 157–166.

Tessier, R. 2002. "Fast placement approaches for FPGAs," ACM Trans. Design Autom. Electron. Syst., 7(2): 284–305.

Maidee, P., C. Ababei, K. Bazargan, 2003. "Fast timing-driven partitioning based placement for island style FPGAs," in Proc. ACM/IEEE Design Autom. Conf., 598–603.

Gort, M. and J.H. Anderson, 2012. "Accelerating FPGA routing through parallelization and engineering enhancements," IEEE Trans. Comput.- Aided Design Integr. Circuits Syst., 31(1): 61–74.

Gort, M. and J.H. Anderson, 2010. "Deterministic multi-core parallel routing for FPGAs," in Proc. IEEE Field-Program. Technol., 78–86.

Betz, V., A. Ludwin, K. Padalia, 2008. "High-quality, determinstic parallel placement for FPGAs on commodity hardware," in Proc. ACM/SIGDA FPGA, 14–23.

Wang, C.C. and G.G. Lemieux, 2011. "Scalable and deterministic timing driven parallel placement for FPGAs," in Proc. ACM/SIGDA FPGA, 153–162.

McMurchie, L. and C. Ebeling, 1995. "PathFinder: A negotiation-based performance-driven router for FPGAs," in Proc. ACM/SIGDA FPGA, 111–117.

Luu, J., I. Kuon, P. Jamieson, T. Campbell, A. Ye, M. Fang, and J. Rose, 2009. "VPR 5.0: FPGA CAD and architecture exploration tools with singledriver routing, heterogeneity and process scaling," in Proc. ACM/SIGDA FPGA, 133–142.