CHAPTER  5

TIME CONVERGENCE IN SWITCHING AND ROUTING, INTEGRATED AUTOMATED ROUTER SYSTEM
5.1 Introduction

Convergence is an important notion for a set of routers that engage in dynamic routing. For a set of routers to have converged, it collects all available topology information from each other via the implemented routing protocol, the information they gathered must not contradict any other router's topology information in the set, and it must reflect the real state of the network. In other words: In a converged network all routers "agree" on what the network topology looks like.

All Interior Gateway Protocols rely on convergence to function properly; it is the normal state of an operational autonomous system. The Exterior Gateway Routing Protocol BGP typically never converges because the Internet is too big for changes to be communicated fast enough.

Due to the low manufacturing cost and time, the FPGA(field-programmable gate array) has became the most popular ASIC (application-specific integrated circuit) for fast them prototyping, the symmetrical array architecture is of the most widely used classes of FPGA architectures [50], the architecture consists of two-dimensional (2D) arrays configurable logic blocks (CLBs), rows, and columns of tiny channels, and programmable switch blocks. Each one of any Boolean functions constrained by the number of routs. Routing channels are predefined wire segments of label length. Switching blocks are programmable to make connections between the wire segments and CLB pins.

Unlike the routing problems of custom layouts, such a standard cells and mask-programmed gate arrays, in which the segments can be drawn almost arbitrarily, routing problem of the symmetrical FPGA[64,65] is very restricted by the limited resources of the (prefabricated) segments and programmable switches. Consequently, general routing approaches for custom layouts may not be propriety for FPGA routing.

Many routing approaches for symmetrical FPGA have a proposed over the past several years. CGE [53] composed each net into a set of two terminal nets and them onto minimum distance paths. The objective is to distribute the wires among the channels such that the maximum channel density is minimized. A Steiner spanning-tree-based routing approach (called a negative reinforcement method) was proposed in [54] to overcome the sequential routing of two-terminal nets. SEGA [52] tried to route critical nets first so that long connections do not suffer long propagation delays through multiple programmable switches. GBP [56] formulated the routing problem
as a two-dimensional interval packing problem. It uses two greedy algorithms iteratively to pack the nets into the tracks. IKMB [62] modeled interconnection resources as a graph and constructed routing trees for the nets by the repeated routing approach for 2D FGPA/FPIC (field programmable interconnect chip) was proposed in [57, 58] is which the main focus is to minimize the channel density to improve the routability of the designs. TRACER-fpga [59] initially routed nets sequentially according to their criticalities and rout abilities. The nets/paths violating the routing resources are then resolved iteratively by employing a simulated evolution-based optimization technique.

In this we present a new performance and rout ability driven algorithm for symmetrical array-based FPGAs on STP called Time convergence in switching & routing we have also combined all the features of STPRSA, UIMR, HFRD, TSR [Ch 2, 3, 4] with symmetrical FPGA’s spanning tree on symmetric and generic routers called Integrated automated router system (IARS). One key feature of our approach is that it overcomes critical limitation of the previous works, namely, the existing routing algorithms use the “channel density” as the measure of the refutability in routing. However, the channel density reflects the degree of net-congestions at the connection blocks of symmetrical FPGA only and does not take into account the net-congestions at the switch blocks properly. Consequently, this leads to routing solutions with a suboptimal use of routing resources in the switch blocks. Clearly, maximally utilizing the routing resources in the switch blocks as well as in the connection blocks is a key to achieve an optimal routing result in terms of net/path delay and routability for this reason, we analyze the interconnection structure of symmetrical FPGA and derive highly reliable that not only take into account net-congestions at the connection blocks but also at the switch blocks. We then design a new routing algorithm to utilize all the features more effectively. Our goal in designing a new routing algorithm to produce more efficient routing solutions in terms of net/path delays and routability while not spending long execution times.

We employ a more predictive way of finding a minimum distance spanning tree by estimating the routing density of the switch resources. We have conducted a series of experiments to demonstrate the strength of the proposed routing algorithm. The results show that our routing solutions are very effective in reducing net/path delays.
The rest of the chapter is as follows, section 5.2 describes the motivation of this work. Section 5.3 presents the architecture and algorithms of TSR. Section 5.4 describes the implementation of TSR. Section 5.5 presents our experimental evaluation. Section 5.6 draws conclusions and discusses some of the remaining issues.

5.2 Motivation

We first argue for the need of an integrated automated router system and then discuss the limitations of existing approaches.

When a routing protocol process is enabled, a router will attempt to exchange information about the topology of the network. The extent of this information exchange, the way it is sent and received, and the type of information required vary widely depending on the routing protocol in use.

A state of convergence is achieved once all routing protocol-specific information has been distributed to all routers participating in the routing protocol process. Any change in the network that affects routing tables will break the convergence temporarily until this change has been successfully communicated to all other routers.

5.2.1 Why time convergence?

Convergence time is a measure of how fast a group of routers reach the state of convergence. It is one of the main design goals and an important performance indicator for routing protocols to implement a mechanism that allows all routers running this protocol to quickly and reliably converge. Of course, the size of the network also plays an important role, a larger network will converge slower than a small one.

RIP is a routing protocol that converges so slowly that even a network of a few routers can take a couple of minutes to converge. In case of a new route being advertised, triggered updates can speed up RIP's convergence but to flush a route that previously existed takes longer due to the hold down timers in use. OSPF is an example of a fast-converging routing protocol. A network of a few routers can converge in a matter of seconds.

Certain configuration and hardware conditions will prevent a network from ever converging. For instance, a "flapping" interface (an interface that frequently changes its state between "up" and "down") might cause conflicting information to...
propagate the network so that routers never agree on its current state. Under certain circumstances it might even be desired to withhold routing information from parts of the network, thereby enforcing an unconverted network.

5.2.2 Limitations of Existing Approaches

Recent FPGA routing results have suggested that a separate global and detailed routing strategy is inferior to a combined routing process [1, 10, 19, 21]. Similarly, the practice of dividing multipoint nets into multiple two-point nets for routing was thought to negatively impact rout ability. In facts, recently published results have shown that combined routers have used significantly fewer routing tracks than the best-known two-step routers, CGE [4] and SEGA [11]. However, results obtained with a new global router, VPR; show that distinct global and detailed routing, combined with multipoint net division, can be competitive with the latest published FPGA routing tools. This is encouraging because separate global and detailed routing of two-point nets may have other practical benefits such as reduced memory use or compute time.

There is an additional concern that separate global and detailed routing may suffer from what [20] calls a mapping Anomaly. This is a condition where the global route forms such a constraint that the channel density greatly under-specifies the minimum number of routing tracks required. After making the architectural assumptions suggested by [20], we sometimes detect the presence of a Mapping Anomaly. Our experimental results indicate that this anomaly is of critical concern if multipoint nets are constrained to a single track domain. However, the anomaly was not found to be present when nets were allowed to be split onto multiple track domains at input and output pins.

Finally, a new lower bound for evaluating the performance of any detailed router is presented. Although the new bound is not completely tight compared to tracer, the SEGA detailed router typically routes bench-marls within two tracks & four circuits of the bound.

Segment Allocator (SEGA) [61], used a different cost function structure to make use of long wire segments. SEGA also made the assumption that a net could be fully expanded into all possible paths along the global route. Consequently, SEGA does not re-expand a net when its path is exhausted. Instead, the cost function
increases a net’s priority as its choices diminish. This approach yielded good results, so CGE-style rip-up was deemed unnecessary to the algorithm.

Since SEGA’S original publication date, a number of different cost functions have been explored to investigate rout ability and speed-performance [5]. The cost function used to produce the results for this paper, called Area, has been the most successful so far in using the fewest wring tracks. The Area cost causes SEGA to first identify the nets which have the fewest number of remaining paths. Among these nets, the path with the lowest Demand cost (akin to CGE’s cost) is chosen.

5.3 The TSR Architecture

This section deals the architecture of time convergence in switching & routing (TSR) using FPGA. First we outline its design rationale & overall operation, next we discuss SEGA & TRACER metrics and then details TSR techniques. Finally we analyze the complexity of TSR.

5.3.1 Overview of TSR

Figure 5.1 outlines the TSR CAD flow. The inputs to VPR consist of a technology mapped net list and a text file describing the FPGA architecture. VPR can place the circuit, or a pre-existing placement can be read in. VPR can then perform either a global route or a combined global/detailed route of the placement. VPR’s output consists of the placement and routing, as well as statistics useful in assessing the utility of FPGA architecture, such as routed wire length, track count, and maximum net length. Some of the architectural parameters that can be specified in the architecture description file are:

- the number of logic block inputs and outputs,
- the side(s) of the logic block from which each input and output is accessible,
- the logical equivalence between various input and output pins (e.g. all LUT inputs are functionally equivalent),
- the number of I/O pads that fit into one row or one column of the FPGA, and
- the dimensions of the logic block array (e.g. 23 x 30 logic blocks).

In addition, if global routing is to be performed, one can also specify:
the relative widths of horizontal and vertical channels, and
the relative widths of the channels in different regions of the FPGA.
Finally, if combined global and detailed routing is to be performed, one also specifies:

- the switch block [1] architecture (i.e. how the routing tracks are interconnected),
- the number of tracks to which each logic block input pin connects (Fc [1]),
- the Fc value for logic block outputs, and
- the Fc value for I/O pads.

The current architecture description format does not allow segments that span more than one logic block to be included in the routing architecture, but we are presently adding this feature. Adding new routing architecture features to VPR is relatively easy, since VPR uses the architecture description to create a routing resource graph. Every routing track and every pin in the architecture becomes a node in this graph, and the graph edges represent the allowable connections. The router, graphics visualization and statistics computation routines all work only with this routing resource graph, so adding new routing architecture features only involves changing the subroutines that build this graph.

Although TSR also be used with row-based FPGAs. VPR is not currently capable of targeting hierarchical FPGAs, although adding an appropriate placement cost function and the required routing resource graph building routines would allow it to target them.

Finally, TSR’s built-in graphics allow interactive visualization of the placement, the routing, the available routing resources and the possible ways of interconnecting the routing resources.
### 5.3.2 SEGA & TRACER

The results of TSR are compared with SEGA & Tracer. So, we have given here the small introduction of these:

SEGment Allocator (SEGA) [51], used a different cost function structure to make use of long wire segments. SEGA also made the assumption that a net could be fully expanded into all possible paths along the global route. Consequently, SEGA does not re-expand a net when its paths are exhausted. Instead, the cost function increases a net's priority as its choices diminish. This approach yielded good results, so CGE-style rip-up was deemed unnecessary to the algorithm. Since SEGA's original publication date, a number of different cost functions have been explored to investigate routability and speed-performance [52]. The cost function used to produce the results for this paper, called Area, has been the most successful so far in using the fewest wiring tracks. The Area cost causes SEGA to first identify the nets which have the fewest number of remaining paths. Among these nets, the path with the lowest Demand cost (akin to CGE's cost) is chosen.
TRACER-fpga was able to achieve very dense routing for two suites of benchmark circuits using a reasonable amount of CPU time. The long wiring delay makes TRACER-fpga unsuitable for high performance design. However, the track efficiency makes it suitable for slow-speed applications such as hardware emulation where the clock rate ranges from 1 to 10 MHz.

The largest circuit in our benchmark suites has 586 CLB’s. As the technology advances, the number of CLB’s in a circuit will increase.

TRACER-fpga will slow down as the routing graph grows. It is important to improve the programming and data structuring efficiency so that TRACER-fpga can keep up with the technology progress. Furthermore, to improve the wiring delay of TRACER-fpga, we have to integrate the delay model into the expansion router.

That is, instead of searching for the shortest connection, it should look for the fastest connection.

5.3.3 Layered Approach

Using the above optimization techniques in software & hardware, we have built in the previous work. Now, we have implemented TSR(time convergence in switching & routing)in spanning tree with FPGA’s, TSR consists of combined features of global & detailed routers. They are

- Routing architecture
- Detailed Routing

Routing architecture consist of 3 types of components considered of logic blocks, configurable IO blocks interconnection resource. It consists of wiring segments switch points that are organized into horizontal routing channels. A wire segment is connected between CLB pins and switch points in the connection switch blocks accordingly it’s shown emphatically that flexibility of routing resources is achieved when connection blocks are fully flexible.

Detailed Routing

In this phase the problem is to assign the nets in a connection or a switch block ti the routing resources in a way that minimizes the number of routing resources used
the resource assignment is usually modeled as a graph in which nodes represent nets and an edge between 2 nodes if the corresponding nets cannot share a track. Finding the min number for a graph will determine the min number of tracks used by the nets.

A number of efficient techniques including the implicit enumeration technique can be used to solve the problem; we may use any of the conventional detailed routing algorithms for field programmable gate arrays. However from the analysis of the route types at a switch point we can utilize tracks more efficiently if we exploit the several possible ways of implementing nets.

5.4 System Implementation

The TSR algorithm combines both global and detailed routing into one step. An FPGA is routed by treating every track domain as a bin and greedily filling that bin with nets until no more will fit. It then proceeds to the next track domain and repeats the process. In this way, TSR is similar to the Best Fit Decreasing bin-packing heuristic. Observations that it did not densely pack the last few track domains led to the Orthogonal Greedy Coupling (OGC) algorithm [51]. By switching from one greedy algorithm to another (which has a different optimization goal) after some track domains have been packed, the last few track domains were more densely packed and fewer routing tracks were used.

A series of one-step routing algorithms was presented above. In these algorithms, multipoint nets are routed one at a time. If a net fails to route, it is moved to the front of the net order and routing is restarted. The FPGA routing resources are represented by a graph which shrinks as nets are routed. The algorithm differs by the way they route a multipoint net through the remaining graph. Five different core metrics were presented, three of which were further enhanced using iteration. Of these eight, four minimized wire length by solving the Network Steiner Tree Problem and four minimized source to sink distance using shortest-paths algorithms. In this, we compare our results to those produced by IKMB of the bench circuits.

The time convergence in switching & routing uses the same net routing strategy described above. It also uses the IKMB algorithm to perform detailed routing. However, before each recursive partitioning step, TSR greedily selects a partial global route for each net based on rectilinear Steiner arborescence and assigns nets to specific S blocks. This allows FPR to balance congestion across each cut and fix the signal entrance or points on each side before cutting each sub partition.

Department of Computer Science, S.K University, Anantapur.
The algorithm also performs combined routing of multipoint nets. It uses a maze router seeded from the source and all sinks to route each net. Initially, all nets are routed by allowing them to share wires. Next, a simulated evolution technique (similar to simulated annealing) chooses nets for rip-up and rerouting; nets sharing resources are more likely to be ripped up. During rerouting, a high cost is used to discourage future sharing. When no more sharing occurs, a solution has been found. The TRACER-fpga algorithm [60] is similar, except that it avoids sharing during initial net routing. Also, it uses slacks to order nets during initial routing and for selection of nets during rip-up. By using slacks, it gives long nets priority for direct connections and allows short nets to route around congestion.

Our proposed detail routing algorithm will maximally increase flexibility of implementing the nets and is composed of 3 steps.

**Step 1:** Decomposition Step – this step is essentially a variant of reliably net decomposition based routing technique. It decomposes every K terminal net into a set of 2 terminal nets. This allows the net to be routed if a net is timing critical, the net be decomposed with an emphasis on the interconnection delay rather than the routability.

**Step 2** – Resource assignment step. We assign the subnets obtained in the decomposition step to routing resource. The objective is to minimize the number of tracks used at each of the connection or switch blocks.

**Algorithm: An algorithm for net decomposition, resource assignment & merging**

- Sort the nets according to the timing criticality; /* The timing criticality is Estimated using the measure in Eq.9 with a1 = 0.*/
  
  For each net ni in the list

  - Sort the edges in the net according to edge-costg(.);
  - if (net ni is timing critical)
    - Perform breadth first search from signal source;
    - Select the k-1 shortest two-terminal nets whose route Match with the signal flow of ni;
  - else
    - Assign edge-costd(.) to every pair of terminals;
    - Construct a minimum cost spanning tree;
    - Generate two-terminal net for each edge of the spanning tree;
  - endif;
- endfor;

// An algorithm for resource assignment (step d-2):

Department of Computer Science, S.K University, Anantapur.
• Construct a coloring graph $G$ for the routes of subnets obtained from net-decomposition step;
• Determine the color of each node of $G$ by applying the implicit enumeration technique;
• Report the chromatic number of $G$;

// An algorithm for route merging (step d-3):
• Sort the nets according to the timing criticality;
repeat
   for each net $n_i$ in the list
      o Construct a resource sharing graph $G$ for $n_i$;
   repeat
   • From the source terminal node in $G$ toward the sink nodes,
   Find pair of nodes (with the longest path first) that can share a track;
   if (there is a pair)
      o Remove the switch between the pair
      o Update $G$ by merging the two nodes of the pair into one;
   endif;
   until (no pair of nodes);
endfor;
until (no reduction in the # of switches);

Step 3 – Merging step. This step tries to merge the connection in different connection or switch block to maximize the resource sharing. The sharing of tracks will lead to a reduction in number of switches used which would then directly contribute to the reduction in the routing delay of the net. A merging process in which the most timing critical net processed first and the least timing critical net last is performed repeatedly for the nets.

Figure 5.2: merging of 2 subnets
The possible merging of subnets is then performed from the subnet containing the source terminal towards the subnet containing the sink terminal. For each net we model the merging problem using a graph called resource sharing graph in which a node represents a subnet and an edge exists between two nodes if the corresponding subnets have been connected. We can now simplify the graph by merging the nodes in a way that maximizes the resource sharing. This in turn leads to a reduction of the routing delay without any degradation in the routability. The figure below shows the resource sharing graphs for the subnets in which the resources to be merged are specified on the edges of the graph.

![Figure 5.3: Integrated automated router system](image)

The flow below indicates as follows: In order to make the router algorithm still more efficient as it is specified, we have developed a router system called Integrated automated router system, which is a combination of channeling, zonal system, merging, efficient hardware forwarding router domain, efficient creation and...
retrieval storage in a router, automatically optimizes router table. Intact optimizes the net, path delays, CPU time. The comparison of results are shown in section 5.5.2.

It also sequentially maze-routes each multipoint net by searching out the next closest sink. If a path for a net cannot be found, it is moved to the front of the net order and routing is restarted. To reduce the maze-routing search space, it initially follows paths which advance toward the closest sink.

5.5 Performance Evaluations

We have implemented the proposed routing algorithm in C & C++ Programming language on a sun spark station on a Linux OS, we have tested our algorithm on set of marks reported in SEGA and TRACER. We explore the tradeoffs between the performance and routing density results.

5.5.1 Experimental results & analysis

The below table shows the result of longest net and path delays produced by SEGA, TRACER and TSR (Time convergence in switching & routing).

<table>
<thead>
<tr>
<th>Circuits</th>
<th>SEGA</th>
<th>TRACER</th>
<th>Ours(TSR)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 bit alu</td>
<td>110</td>
<td>354</td>
<td>78</td>
</tr>
<tr>
<td>4 bit alu</td>
<td>163</td>
<td>458</td>
<td>126</td>
</tr>
<tr>
<td>apex 7</td>
<td>78</td>
<td>147</td>
<td>52</td>
</tr>
<tr>
<td>term 1</td>
<td>70</td>
<td>113</td>
<td>39</td>
</tr>
</tbody>
</table>

TABLE & Graph 5.1: Comparison Of Longest net delay
Finally, the results in table & graph 5.1 show how much the longest net delay can be reduced at the expense of routing density. Overall, our algorithm reduces the net delay by up to 53.8 percent on the average with less number of tracks, which implies that the results produced by our algorithm more closely to an optimal solution in terms of both rout ability and speed performance of the implementation circuits.

**TABLE 5.2: Comparison of Path delay**

<table>
<thead>
<tr>
<th>Circuits</th>
<th>SEGA</th>
<th>Tracer</th>
<th>Ours (TSR)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 bit alu</td>
<td>947</td>
<td>714</td>
<td>618</td>
</tr>
<tr>
<td>4 bit alu</td>
<td>1392</td>
<td>1337</td>
<td>1322</td>
</tr>
<tr>
<td>apex 7</td>
<td>339</td>
<td>299</td>
<td>320</td>
</tr>
<tr>
<td>term 1</td>
<td>129</td>
<td>136</td>
<td>131</td>
</tr>
</tbody>
</table>

**TABLE 5.3: Comparison of CPU time**

<table>
<thead>
<tr>
<th>Circuits</th>
<th>Tracer</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 bit alu</td>
<td>66</td>
<td>70</td>
</tr>
<tr>
<td>4 bit alu</td>
<td>207</td>
<td>200</td>
</tr>
<tr>
<td>apex 7</td>
<td>30</td>
<td>18</td>
</tr>
<tr>
<td>term 1</td>
<td>16</td>
<td>12</td>
</tr>
</tbody>
</table>
Table 5.2 depicts the explanation results for benchmark circuits. This mainly comes from the capacity of the global router to form diverse routing trees under different values. The tables 5.2 & 5.3 show the results produced by Tracer & SEGA. The comparisons show that most of the longest path delay by 46.8% & CPU time produced by our router are shorter than those indications, an effective exploration of solution space.

**TABLE 5.4: Tradeoffs between Routability and Performance**

<table>
<thead>
<tr>
<th>Circuits</th>
<th>Routability</th>
<th>Performance</th>
<th>Routability</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Types</td>
<td>Tracer[12]</td>
<td>Delay</td>
<td>SEGA</td>
<td>Delay</td>
</tr>
<tr>
<td>alu2</td>
<td>9</td>
<td>714</td>
<td>11</td>
<td>707</td>
</tr>
<tr>
<td>alu4</td>
<td>11</td>
<td>1037</td>
<td>15</td>
<td>1096</td>
</tr>
<tr>
<td>apex7</td>
<td>8</td>
<td>299</td>
<td>13</td>
<td>258</td>
</tr>
<tr>
<td>term1</td>
<td>7</td>
<td>136</td>
<td>10</td>
<td>103</td>
</tr>
</tbody>
</table>

---

Department of Computer Science, S.K University, Anantapur.
Table 5.4 shows a summary of empirical results on how the tradeoffs between the routing density & speed performance are explored by using the control parameters. The routability and time is measured by taking minimum number of 2 tracks. The above data corresponds to result of bench mark examples with first columns of the table. Shape of the curves clearly reveals empirical evidence and suggests that routing solution satisfy the performance requirement within a few iterations of router while minimizing router density. The qualitative and quantitative metrics is shown in appendix C.

5.6 Conclusion

We first make concluding remarks and then discuss some of the remaining issues associated with new parallel router algorithm.

5.6.1 Concluding remarks

We have presented a new performance and routability driven routing algorithm for STP. The key contribution of our work is the formulation of much more reliable and accurate metrics which is derived from the careful analysis of the interconnection of the switch block. This then led us to develop an efficient routing technique called Integrated automated router system, which is well suited to improve the metrics very effectively. Extensive experimental data showed that the proposed routing algorithm is very effective in improving the overall performance of the design as well as the routability. In summary, compared to the results produced by
SEGA, TRACER for the benchmark circuits, our algorithm reduced the longest net by 53.8 percent (on average) and the delay of the longest path by 46.8 percent even with about 1.1625 times less execution time.

5.6.2 Remaining issues

Expansion router: Although the current IARS techniques have basic updations, an expansion router can have more sophisticated connectivity such as ripup & rerouter in STP, depending on target environments, and IARS can be served as a general framework to support them as well.

Use of multiple techniques: IARS nodes can cooperate obviously to reduce the avoidance & making path & net delays efficient, but the algorithm should have unique approach allowing it to be used over a wide range of different FPGA routing architecture.