A Tutorial on FPGA Routing

Daniel Gomez-Prado *Maciej Ciesielski*
dgomezpr@ecs.umass.edu *ciesiel@ecs.umass.edu*

Department of Electrical and Computer Engineering, University of Massachusetts, Amherst, USA

I. Introduction

The entire CAD process that is necessary to implement a circuit in an FPGA (from the RTL description of the design) consists of the following steps:

- **Logic optimization.** Performs two-level or multi-level minimization of the Boolean equations to optimize area, delay, or a combination of both.

- **Technology mapping.** Transforms the Boolean equations into a circuit of FPGA logic blocks. This step also optimizes the total number of logic blocks required (area optimization) or the number of logic blocks in time-critical paths (delay optimization).

- **Placement.** Selects the specific location for each logic block in the FPGA, while trying to minimize the total length of interconnect required.

- **Routing.** Connects the available FPGA's routing resources[1] with the logic blocks distributed inside the FPGA by the placement tool, carrying signals from where they are generated to where they are used.

Routing is an important step of the process as most of the FPGA's area is devoted to the interconnect [21], and the interconnection delays are greater than the logic delays of the designed circuit. Therefore an efficient routing algorithm tries to reduce the total wiring area and the lengths of critical-path nets to improve the performance of the circuit; and for this, the router needs the interconnect information of the target FPGA architecture. This means that the problem of routing is architecture dependent and therefore the number of routers needed to route FGPAs is as varied as FPGA architectures there are in the market.

To understand better this dependency between routing and the target architecture, an overview of one of the most important commercially available FPGAs is shown below:

- **Xilinx**

  The general architecture of Xilinx FPGAs consists of a two-dimensional array of programmable blocks, called Configurable Logic Blocks – CLBs [24], with horizontal and vertical routing channels between CLB's rows and columns. The routing resources available on this architecture are:

[1] The clock net is not considered here as it is usually routed via a dedicated routing network in commercial FPGAs

1) Connection boxes: The C boxes connect the channel wires with the input and output pins of the CLBs. It has two major properties that can affect the routability of a design: its flexibility, *Fc*, which is the number of wires that each logic block pin can connect to; and its topology, which is the pattern of switches[2] that make the connection (especially if *Fc* is low). For example in figure 1, for a C box with Fc = 2, topology 1 can not connect pin A with pin B, meanwhile topology 2 can.

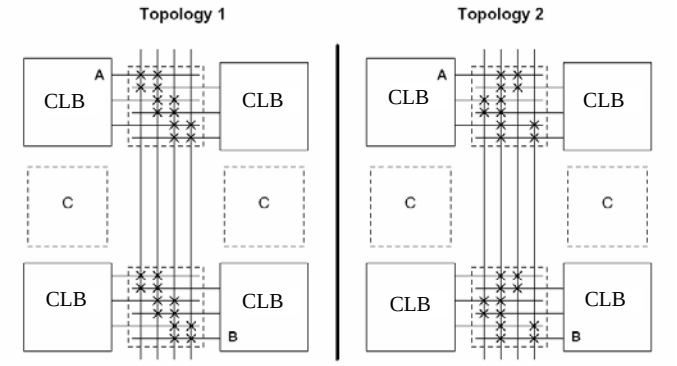

Figure 1. Connection box topology

2) Switch boxes: The S boxes allow wires to switch between vertical and horizontal wires. Its flexibility, Fs, defines for a wiring segment entering the S block the number of other wiring segments it can be connected to. The topology of the S blocks is very important since it is possible to choose two different topologies with the same flexibility *Fs* that result in very different routabilities. For example, figure 2 shows that meanwhile topology 1 can't connect wire A with B, topology 2 can.

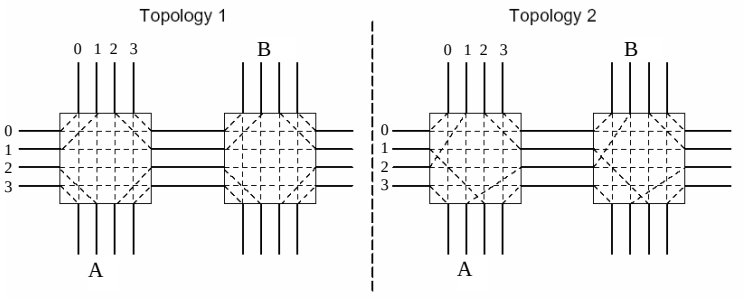

Figure 2. Switch box topology

Switch boxes that connect only tracks in the same domain, i.e. 0-0, 1-1, are called planar or subset switch boxes. Switch boxes that allow connection to any other domains, i.e. 0-3, 1-2, are called Wilton switch boxes, and they are broadly used as they provide greater flexibility on routing.

3) Single-length lines. They are intended for relatively short connections among CLBs and they span through one CLB only. See figure 3.b.

4) Double-length lines. They are similar to the Single-length Lines, except that each one spans two CLBs, offering lower routing delays for moderately long connection.

[2] The switches can be pass transistors or multiplexers

5) Long lines. They are appropriate for connections that require reaching several CLBs with low-skew. See figure 3.c.

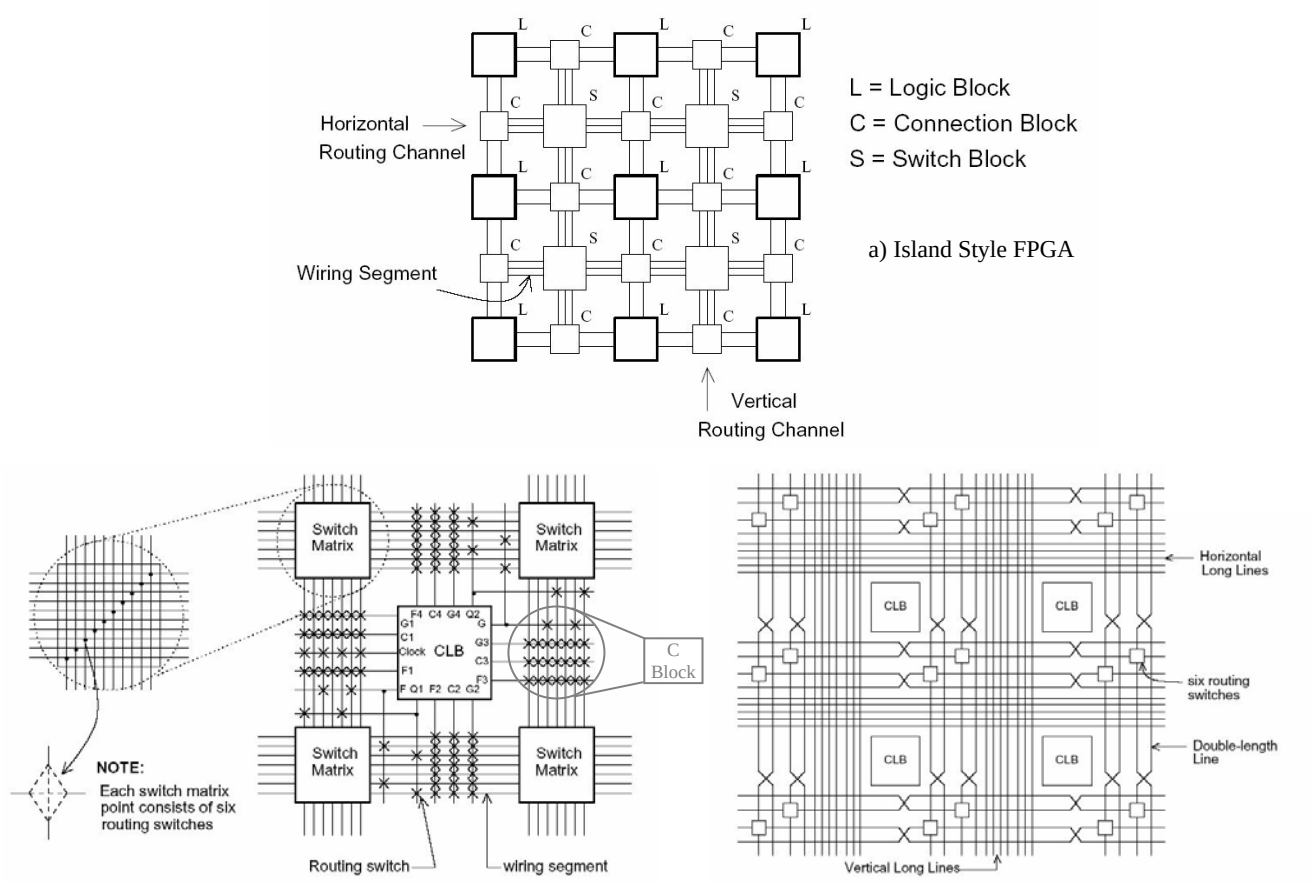

Figure 3. Island Style Architectecture

Increasing the flexibility of the switch box, the connection box and the number of wires per channel makes routing a trivial problem [17] as all possible interconnections are available. But increasing routing resources has the drawback that waste area and transistors in the FPGA, as only a fraction of those resources will be used for a given design, even worse it increases the number of interconnect transistors which are the principal reason of delay on FPGAs.

As FPGAs have prefabricated routing resources, the router must work within the framework of the architecture's resources, deciding exactly which routing resources will be used to carry the signals between logic blocks, and making sure that no more connections are made through a region than there are resources to support them. Thus the router must consider the congestion of signals in a channel, and through multiple iterations rip out and reroute those congested areas and wires. This search of possible connections to route the placed logic blocks is not ensured to be feasible and it is possible that after a given number of iterations, 40 for example, the circuit can't still be routed and the placement has to be redone. Therefore, together with the routing algorithm a routability detection algorithm is clearly desirable to avoid long routing iterations on designs that eventually will be determined to be unroutable.

II. The FPGA Model

Academic research has adopted as FPGA architecture a simplified version of the island style model from Xilinx. The main reason is that FPGA market share is divided in mainly three companies: Xilinx with the highest share has an average presence of roughly half of the total market[3], Altera has roughly one-third, and Actel has one-six of the market. From these three companies Actel and Altera have respectively solved their routing problems by adapting channel-style ASIC routing algorithms [1] or over assigning routing resources [2]; so the active research area left to academia is the island style architecture from Xilinx FPGAs, nevertheless this is an important architecture as it is responsible of half of the entire FPGAs production [3] on the market.

In academia the most common simplifications made to the island style model are:

- Each logic block has 4 inputs pins and 1 output pin, and all logic blocks are alike.

  Commercial FPGAs have logic blocks with different number of inputs, ranging from 3 to 7, and they provide two or more outputs.

- The C box is implemented with pass transistors rather than multiplexers for input connections. This allows two or more tracks to be electrically connected via the input pin by turning on individual switches in the C box. This is called input pin doglegs.

  Commercial FPGAs implement the C box via multiplexers to save area, so only one track may be connected to the input pin and no input pin doglegs are possible. See figure 4.

- The wire segments span only one logic block before terminating. This means that all interconnections have to pass as many C boxes and S boxes as logic blocks there are between the two connecting points.

  Commercial FPGAs have double-length and long wires to speed up this kind of connections, and avoid congesting the C and S boxes.

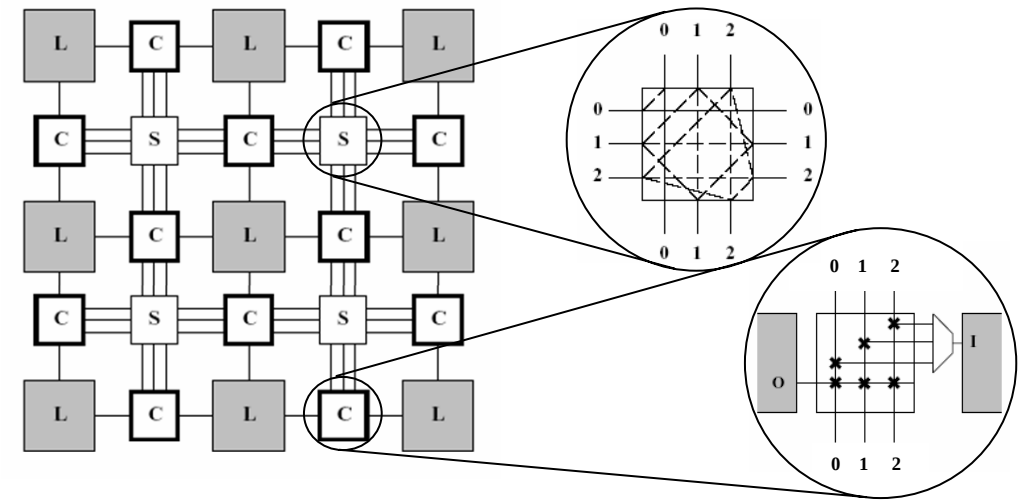

Figure 4. The FPGA Model

[3] FPGA market share research by www.rhk.com/rhk/research and www.icinsight.com

III. General Background for Routing

Routing is an NP complete problem[4] [23] that is generally separated in two phases using the divide and conquer paradigm [8]: a global routing that balances the densities of all routing channels, and a detailed routing that assigns specific wiring segments for each connection [17][18][25]. These two phases avoid congestion and optimize the performance of the circuit, making sure all nets are routed minimizing wirelength and capacitance on the path. By running both algorithms a complete routing solution can be created.

Of course there are a number of routing algorithms that solve the problem using a mixed routing, both global and detailed routing at the same time, based on the idea that a higher integration of the two phases can prevent inaccurate estimation and the routing result will be better. The drawback of this approach is that as circuit size grows this mixed routing becomes more complex and less scalable [13].

3.1 Global routing

The global router performs a coarse route to determine, for each connection, the minimum distance path through routing channels that it has to go through. If the net to be routed has more than two terminals the global router will break the net into a set of two-terminal[5] connections and route each set independently.

The global router considers for each connection multiple ways of routing it and chooses the one that passes through the least congested routing channels. By keeping track of the usage of each routing channel, congestion is avoided; and the principal objective of the global router, balancing the usage of the routing channels, is achieved.

Once all connections have been coarse routed, the solution is optimized by ripping up and re-routing each connection a small number of times. After that, the final solution is passed to the detailed router.

3.2 Detail routing

The detail router determines for each two point connection the specific wiring segments to use in the routing channel assigned by the global router. To do this, detail routing algorithms construct a directed graph from the routing resources to represent the available connection between wires, C blocks, S blocks and logic blocks within the FPGA.

The search performed on this directed graph is usually based on Dijkstra's algorithm to find the shortest path between two nodes. The paths are labeled according to a cost function that takes into account the usage of each wire segment and the distance of the interconnecting points. The distance is estimated by calculating the wire length in the bounding box of the interconnecting points using a Manhattan metric. Most of the routers relax the bounding box constraints and allow searching for possible solutions in the surrounding routing channels of the bounding box. This is done to avoid subsequent iterations of ripping out and re-routing if the solution lies on the near outside of the bounding box.

The most common detail routing algorithms are:

[4] There is no polynomial algorithm that can solve the problem.
[5] By breaking the multiple output net in a set of two-net connections, the global router is (most likely) allowing dogleg pin.

- Maze Router

The Maze routing algorithm is based on a wavefront expansion technique that attempts to find the shortest path between two points while avoiding any used routing resources [4]. This algorithm is an iterative process that rips up and re-routes some of the routes to eliminate congested routing channels.

The principal drawback of the maze routing is that it does the routing without taking into account that the path found can block the routing of the subsequent nets. This means that the performance of the algorithm is net ordering dependent, and different orderings will yield different results. For i.e.: if the order in which the two nets are routed in figure 5 is reversed, a better solution is found.

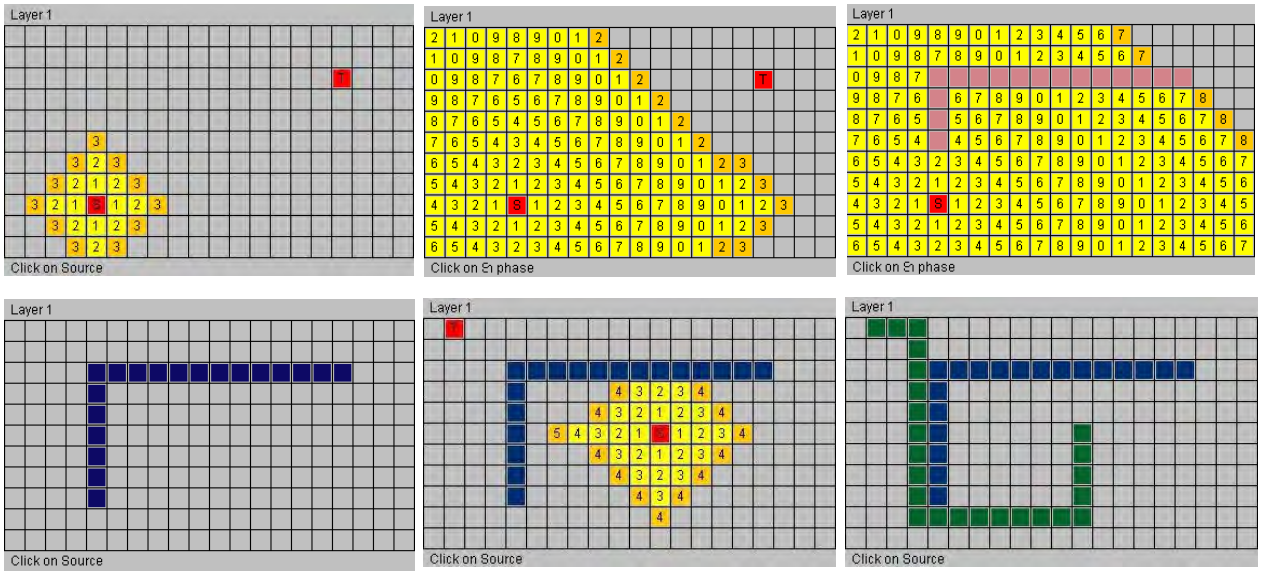

Figure 5. The Maze router wavefront[6]

- A* Search Routing

The maze routing is a special case of the A* routing. The A* routing allows to tune the path search from a breadth-first search algorithm into a shorter depth-first search algorithm. The BFS is an exhaustive search that consider all possible paths and will find the best path if there is any but has the drawback that it can be slow; meanwhile, the DFS may not find the minimum cost path but can be fast. See figure 6.

Weighting a scaling factor α between 0 and 1 the A* routing tunes the search from BFS to DFS. The cost function used to evaluate the directed graph for each node i is [15]:

$$f_i = (1 - \alpha) \times (f_{i-1} + c_i) + \alpha \times d_i$$

Where c_i is the node cost and indicates the current usage of the node and it is used to penalize nodes occupied by previous routes; f_{i-1} is the total cost of the previous path, and d_i is the estimated cost of the path from the node i to the destination.

[6] source: http://foghorn.cadlab.lafayette.edu/cadapplets/MazeRouter.html

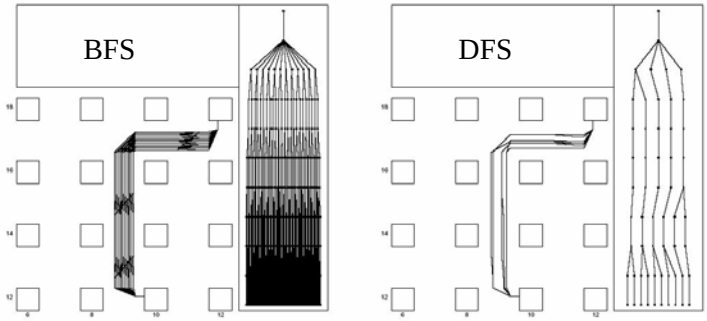

Figure 6. BFS and DFS algorithms

In FPGA architectures with planar or subset switch boxes, wires can only change domain at the output of a logic block or at an input dogleg pin; this means that the route from output to input is confined to the same track domain. Therefore in a DFS search, it is important to attempt routing first in domains that have high probabilities of completion, so that subsequent DFS searches in different domains will not be needed. This is the concept of domain negotiation.

Domain negotiation consists on ranking the domains based on the usage of its wires adjacent to the output pins before routing. Then the cost function is modified by [15]:

$$f_i = (1 - \alpha) \times (f_{i-1} + c_i) + \alpha \times d_i + r_d$$

Domains with lower congestion will have a lower rank, r_d, thus promoting routing in less congested domains first.

- The Pathfinder

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. The cost function implemented by the pathfinder is [10]:

$$f_i = (1 + h_n * h_{fac}) \times (1 + p_n * p_{fac}) + b_{n,n+1}$$

where $b_{n,n+1}$ is the penalty of bending the wire, p_n is the cost of using a specific wire, h_n is the history that keeps track of the usage of the wire during previous iterations; and, h_{fac} and p_{fac} are the respective weighting factors.

Subsequent iterations rip up and re-route all nets, and the process goes on until no overuse of routing resources exist. This process of ripping out and re-routing every net allows the pathfinder algorithm to minimize the net ordering problem of the maze routing.

IV. The State of the Art in Routing

The routers described in this section represent the trend in FPGA routing research. Even thought these are academic tools and they don't actually route any real FPGA, they are important because modifying the used model architecture, the core algorithms implemented on these tools can be effectively use to route commercial FPGAs.

4.1 VPR: Versatile Place and Route

The VPR router is one of the most versatile routers available in academia as it allows describing the targeting architecture. It can be used to route island style FPGAs as well as row-based FPGAs [19]. In this router the type of switch boxes for the FPGA can be chosen to be [20]: planar, Wilton or universal; different length of wires can be defined, input dogleg pins can be allowed or disallowed; and the parameters of the cost function can be modified. The VPR router can perform a global routing or a combined global-detailed routing; being the VPR combined router able to change the current global routing configuration when it can not easily find a detail routing solution [6].

This router is based on a modified version of the Pathfinder algorithm, and it can run in two different flavors to target two different main objectives:

- VPR's routability-driven router

  The primary objective of this algorithm is routing a design successfully with minimum track count. For this, the routability-driven incorporates a modified routing cost model as show below [22]:

  $$cost_n = b_n * h_n * p_n + bend_{n,m}$$

  where b_n is the base cost, usually 1 or 0.95 for most routing resources and 0 for sinks, the latter is to prevent the algorithm to keep searching for possible connections if the sink can already be reached. Note that congestion in the sink is not possible as it will mean that the design requires an input to be driven by two different sources, therefore a base cost of 0 for the sink improves the running time of the algorithm without affecting its quality. $bend_{n,m}$ penalizes bending the wire when routing and it is only taken into account by the global routing. p_n is the present congestion penalty and its value is the difference between the number of nets using a channel and the number of wires that can be placed on that channel. It is call present congestion because its value is updated within an iteration of the algorithm to avoid overusing a channel. Its cost is given by:

  $$p_n = 1 + max(0 , [1 + occupancy_n - capacity_n] * p_{fac})$$

  with p_{fac} equal to 0.5 in the first iteration and to 1.5 or 2 times its previous value in subsequent interations. h_n is the historical congestion penalty and it keeps track of previous cost of the resources, thus avoiding reusing a channel in subsequent iteration. It starts with a value of 1 in the first iteration and then it is:

  $$h^i_n = h^{i-1}_n + max(0 , [1 + occupancy_n - capacity_n] * h_{fac})$$

  where h_{fac} is a constant value between 0.2 and 1. The present congestion and the historical congestion are computed similarly, but one is computed dynamically inside the

iteration of the algorithm, when routing every net; and the other one is computed once at the beginning of each iteration.

- VPR's timing-driven router

  The objective of this algorithm is to increase hardware circuit speed. To do this, it adds an Elmore delay model to the function cost, so the routing gives preference to those solutions with minimum delay. To set an upper bound on delay, this algorithm starts routing the nets with most distant connections first. The imposed ordering on routing produces suboptimal track counts, and faster results.

Another modification common to both approaches is that for multiple output nets the maze wavefront expansion is modified. As mentioned before the global route breaks all *n* terminal net into *n-1* two-terminal nets, and it performs *n-1* iterations of the wavefront expansion to connect the nets. The normal maze router empties the wavefront expansion for each iteration, meanwhile the VPR router does not empty the current wavefront, it adds all the routing resource segments required to connect the reached terminal to the wavefront with a cost of 0, and it continue expanding normally; therefore, the next terminal will be reached much more quickly than if the entire wavefront expansion would have been started from scratch. Figure 7 shows a) a wavefront expansion; a normal maze router when reaches a terminal net empties the wavefront and restart it from the beginning, as shown in b), meanwhile the VPR router adds the last net found to the wavefront with a cost of 0 and continues expanding it, see c).

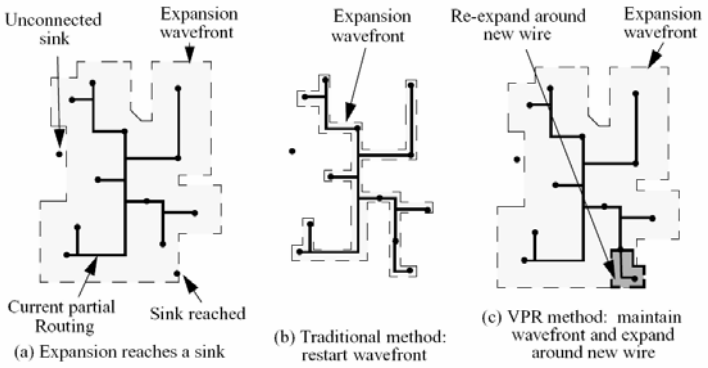

Figure 7. VPR wavefront expansion

Other than the modifications stated above the VPR algorithm behaves as a Pathfinder algorithm [19] routing each net by the shortest path it can find regardless of any overuse of routing resources, and ripping up and re-routing every net in the circuit and recalculating the cost of using a given routing resource.

4.2 ROAD: An Order-Impervious Optimal Detailed Router for FPGAs

The routers described so far have been based on the rip-out and reroute paradigm. The ROAD router is based instead in the bump and refit B&R paradigm. The main idea of this paradigm is to modify the nets already routed when a new conflicting net is found. It starts by routing the nets one by one until a conflict is found, if there are other tracks that can successfully route the conflicting net, the problem is solved and the next net is routed. In the case that all routing resources have been used and no other tracks are available, the router bumps all tracks conflicting with the resource needed, and then all those unrouted net segments are refitted, as at

least one of them wont be able to fit properly in the design, this would cause the unfitted track to be routed through another channel possibly bumping another tracks. In this way, the B&R paradigm makes net congested areas to be depopulated. Therefore in the B&R algorithm when a track is bumped, the bumped track can be propagated producing a path with many bump searches until a vacant resource or a spare routing area is found; and if all possible paths are exhausted and no solution exists or a cycle is detected (a previous bump in the path is revisited) a backtrack to the initial conflicting resource is done and another track is bumped instead.

Even thought this represents no problem for an incremental router, in which some nets have been added to an existing routing in an FPGA, and the goal is to route the new connections without changing the global topology of the existing nets; for a detail router this represents a main problem as many more of the prior routed nets are bumped, thus leading to extensive and time consuming depth first based searches.

To overcome this problem three major enhancements are done to the space search in the B&R algorithm:

- Learning-based search space pruning: This technique records the unsuccessful bumps of a net, and if later on, in another depth first search process we try to bump the same net again, a comparison on the search graphs is done. If both search graphs are found to be isomorphic the bumping of the net is disregarded as it will be unsuccessful, thus saving search time.

- Clique-based search space pruning: This technique dynamically determines the presence of cliques among nets and its size *k* is used to determine the minimum number of different tracks needed to route successfully the nets in the clique. If we attempt to bump a net in the clique while routing a path ϕ, and the maximum tracks per channel available in the FPGA is *t*, then if bumping the net produces $(k+m) > t$ the depth first search is prune as the solution will be unfeasible. The number *m* in the inequality above is the number of unusable tracks for the clique, this is the *m* nets in the clique whose adjacent are ancestor of the path ϕ; and therefore they can't be used to route the actual path as they will cause a cyclic conflict.

- Lookahead transition cost functions: This is a cost function that measures which transition of the net n_i on track T_j to the track T_k , $n_i^{T_j \to T_k}$, is more likely to succed so fewer searches are performed and backtracked. Two principal factors considered on the cost function are the total wirelength of the bumped nets and the flexibility of the bumped nets. This flexibility means that if there is a solution with one net of wirelength 9, and there is another solution produced by 3 smaller nets each of wirelength 3, the set of smaller nets will be more likely to produce a feasible solution as it is easier to move smaller nets than a big one. The cost function will be [7]:

$$TC_1(n_i^{T_j \to T_k}) = \frac{\sum_{n_j \in ad\ j^{T_k}(n_i)} l(n_j)}{\sqrt{\left| ad\ j^{T_k}(n_i) \right|}}$$

Where $ad\ j^{T_k}(n_i)$ are the neighbors of n_i that are on the track T_k , and $l(n_j)$ is the length of n_j in terms of the tracks segment it occupies. The previous function can be

further improved by looking ahead to the next transitions, this is done by calculating the minimum cost of going from T_j to the track T_k and then from T_k to the track T_l , the final cost is given by:

$$TC_2(n_i^{T_k \to T_l}) = \frac{\sum_{n_j \in ad\ j^{T_k}(n_i)} \min_{\forall T_l} TC_1(n_i^{T_j \to T_k})}{\sqrt{\left| ad\ j^{T_k}(n_i) \right|}}$$

The three enhancements do not affect the quality of the routing result as they only prune results that are suboptimal or search spaces that do not yield any result; and with these enhancements the basic B&R algorithm is sped up 604 times, which makes the algorithm fast enough to perform not only incremental but complete detail routing.

The ROAD detail router based on B&R is implemented such as if no solution exits (the circuit is unroutable) one track is added to the channel so the router can find the minimum track solution for the given placement and global assignment. This router is said to be independent of the net order in which it routes because bumping previous routes is equivalent to reversing previous routing decisions, or changing the order in which the nets are routed.

4.3 Routing Approach Via Search-Based Boolean Satisfiability

This approach addresses the routing problem completely different, transforming the complex interactions of the routing constraints as a Boolean function. It has two main virtues: all paths for all nets are considered simultaneously as the routing constraints are a set of equations that need to be satisfied simultaneously; and the Boolean function is satisfiable if and only if the design is routable. The latter means that if there is no satisfying assignment for the Boolean function, it is proven that no routing solution exist for the design, for the given placement and global route assignment.

The Satisfiability-Based Detailed Router (SDR) takes as input the connections assigned for a global router and produces two types of constraints [6]: The connectivity constraints, ensure that a net has a continuous path between the net terminals; these constraints form a list of tracks and C boxes that can possibly form the path in channel segment. And the exclusivity constraints, ensure that different nets are assigned to different tracks in a channel so no overlap occurs. In the intersection of horizontal and vertical channels (S boxes) the constraints of a same net are connected by the logic operation AND.

Once these constraints have been formulated they are transformed and encoded into Boolean equations represented in Conjunctive Normal Form – CNF clauses. The conjunction of all connectivity and exclusivity constraints for all nets form the routing constraint Boolean function, which models the routing problem as a whole. The Boolean SAT solver takes as input the routability function and tries to satisfy the assignments or to prove that the given layout is not satisfiable. If the layout is satisfiable, the solution is an assignment of binary values 1 or 0 to the Boolean variables which encode the track variables. This information is transformed into an assignment of actual routing resources (tracks, C boxes and corresponding S boxes) to nets which forms the actual FPGA routing solution. If it is not satisfiable, then no legal detailed routing solution exists with the given placement and global routing topology.

It is important to note that the equations are written in the CNF form and they are not represented as a BDD[7]. A BDD satisfiability approach explicitly represents all possible satisfying assignments as paths through the BDD directed acyclic graph, and any path found in this graph to 1 is said to satisfy the problem, and if such a path doesn't exist it is said to be unsatisfiable. The problem with BDDs is that without a good variable ordering the BDD graph can become memory-unmanageable in intermediate computations; and finding a good variable ordering is an NP-complete problem.

Instead of BDDs the SAT instances created from the routing constraint Boolean function are solved using GRASP [5][6][16], a generic search algorithm for the satisfiability problem. GRASP is based on search techniques that analyze conflicts on the graph and base on this it can prune large portions of the search space. The analysis yields the causes that produce conflicts, and this information is recorded to recognize similar conflicts on the graph and assignments that are necessary for a solution to be found. This means that GRASP searches to find one satisfying assignment, and must search exhaustively to conclude that no satisfying assignments exist; a trade-off of more search-time for manageable memory sizes.

V. Contrast of the approaches

To thoroughly understand the differences among the routers previously described, it is necessary to compare them based on the ideal objectives of any router:

- **Unroutability detection**

  Only the SAT approach is able to prove that the layout is unroutable for a given placement and global route assignment, though this conclusion can take long, as it has to determine that the SAT problem is unsatisfiable and this means that all possible search combination have to be done. VPR can not determine routability and it stops searching after 30 iterations assuming by then that the circuit is unroutable, during these iterations VPR global-detailed router can modify the global assignment if it simplifies the detail routing operation. This characteristic allows VPR to find routing solutions that the SAT solver determines as unroutable, as the SAT solver relies heavily on the global assignment, and the VPR actually performs modification on the global assignment if needed. The ROAD approach is not really concern with determining routability of the given layout as its ultimate goal is to route the circuit with minimum track count, so if it can't find a solution with a given track count it will add one track to the channel and it will continue routing the design.

  The overall unroutability detection classification of the routers is then SAT, VPR and ROAD.

- **Running time**

  The fastest router is the ROAD algorithm, with the enhancement perform to the basic B&R algorithm ROAD is able to route in average 2 times faster than VPR routability-driven router. To perform this comparison VPR is run as a combined global-detailed router and as global only router, the difference of these values is assume to be the time spent in the detail routing for VPR, and the comparison is against this value.

[7] Binary Decision Diagram is a graph representation for Boolean functions based on the Shannon expansion.

VPR timing-driven router is [20] 2 to 10 times faster than VPR routability-driven router, so in average VPR timing-driven router is still faster than ROAD by 1 to 5 times faster.

To establish the running time of SAT we compare the benchmarks provided on [6] and [7] (see table below), only two circuits can be compared ALU2 and VDA. A straight forward comparison of these circuits is misleading and concludes that VPR is 3 times faster than SAT and for those specific circuits ROAD is 18 times faster than SAT. Such comparison is false as ROAD and VPR benchmark were run on a 550Mhz Pentium III and SAT was run on a 170Mhz Ultra 5 Sparc, so a more fair comparison speeds up SAT results by a factor of 3 yielding that SAT and VPR have approximately the same running time and ROAD is 6 times faster for those specific circuits. The latter comparison only gives us an idea of the running time performance of SAT and more benchmarks needs to be compared before a final conclusion on SAT running time can be made.

Ckt name	VPR	ROAD	SAT
ALU2	8.54	1.41	26.52 (8.84)
VDA	34.13	4.99	98.14 (32.71)

Table 1. Comparison from [6] and [7] [8]

The overall running time classification of the routers is then VPR timing-driven, ROAD, VPR routability-driven and SAT.

- **Minimum Track count**

Both ROAD and VPR routability-driven router achieve the same minimum number of tracks per channel on all benchmarks. VPR timing-driven router requires 5% more tracks than ROAD and SAT router requires about 25% more than ROAD.

These results are heavily correlated with the cost function implemented inside the routers. VPR has two different kind of cost function for each of its routers, being the routability-driven algorithm concerned with achieving minimum track count. ROAD does not really have a cost function oriented toward minimum track count, it is more search-base pruning oriented; but the fact that ROAD looks for the clique interconnectivity and no tracks are added to the channel unless it is mandatory makes this router to always find the minimum track count. SAT doesn't have any cost function implemented in its algorithm and its search is completely "flat", as it only looks for a solution regardless of its optimality.

The overall minimum track count classification of the routers is then VPR routability-driven, ROAD, VPR timing-driven and SAT.

- **Memory utilization**

Even though memory utilization has not been addressed as an objective for any of the routers described, a small comparison can be performed by realizing the correlation between memory and running time of an algorithm. Thus VPR timing-driven algorithm being the fastest has the least memory requirements, ROAD with its search-based pruning and a running time faster than VPR routability-driven has second least memory

[8] The table shows the common benchmarks for the three routers, the values in parenthesis correspond to the equivalent value if the experiment would have been done in a 550Mhz Pentium III.

requirements, VPR routability-driven is the third and the non-directed search of SAT that needs to solve simultaneously all routing constraints has the highest memory requirements.

The overall memory utilization classification of the routers is then VPR timing-driven, ROAD, VPR routability-driven and SAT.

- **Circuit speed**

  The only router concerned with this objective is the VPR timing-driven algorithm, so all the other approaches will show slower performance and higher delays.

It can be seen that different approaches to the same problem inherently targets different main objectives, VPR heuristically search for minimum tracks by minimizing the net ordering problem while its modified version looks for fast running times allowing slightly higher track counts, B&R finds the optimal minimum track by overcoming the net ordering problem, and SAT can formally determine which circuits are unroutable.

VI. Summary

It has been shown that FPGA routing is a complex problem and even thought historically it has been underestimated by VLSI designers, due to its fixed routing resources that should make the routing easier, it has been all the contrary. Fixed routing resources makes routing in FPGA a much harder problem since multiple and all constraints have to be satisfied to successfully route the design.

Most approaches to FPGA routing have been based on the divide and conquer paradigm, in which the routing has been split in two phases, a global route that sparse the track requirement throughout the FPGA and a detail router that does the actual assignment of routing resources. From these two phases the detail router is a much harder problem as it has to consider in deep the architecture of the FPGA. For the detail router, the maze algorithm has been the starting point and different modifications and improvements have been done to the basic algorithm with the A* search and the pathfinder algorithm, and finally this approach has reach is state of the art with the VPR tool.

Of course the maze & rip-out & reroute algorithm used by VPR has not been the only approach to the routing problem, and two other different approaches have been shown, the ROAD router based on bump & refit paradigm, and the SAT router based on the satisfiability constraints of an equivalent Boolean function of the routing problem. The summary performance of these three different approaches can be seen in the next table.

	VPR Timing-driven	VPR routability-driven	ROAD	SAT
Unroutability detection	Heuristic	Heuristic	None	Formal Prove
Running time	Best	Good	Very Good	Bad
Minimum track count	Good	Best	Best	Bad
Memory requirement	Best	Good	Very Good	Bad

Table 2. Router comparison

The above comparison shows that ROAD approach is a nice trade off between the two different flavors of VPR; and that more research has to be done in the SAT approach to make it competitive, maybe adding a specialize metric cost on the search to reduce the number of tracks and speed the running time.

The approaches presented here are not the only ones, and there are many more that can outperform the approaches described in a particular objective. For example Just In Time routing [13] intended to place the routing algorithm in hardware so it can reroute and reprogram another FPGAs achieves outstanding running times and very few memory requirements with the penalty of requiring more tracks; another approach differentiate the so far combined delay minimization and wirelenght optimization [11], by using Steiner graphs to obtain better circuit performance, and some others are capable of detecting routability as early as in the first iteration of the pathfinder router using some heuristic [9] [14].

The different approaches in routing and the different performs obtained do not mean that research on some trends should be pruned as they have not outperform any previous router. All research in the area enlighten the routing problem from a different perspective thus helping to refine or to improve existing algorithms or even to combine some of them.

As architectures evolve and the logic inside each logic block grows, routing resources will be more scarce and routing will be more constraint, therefore FPGA routing will always be an active topic of research throughout the life of FPGA technology.

VII. References

[1] Actel Inc, Axcelerator family FPGA, 2004.

[2] Altera Corporation, Stratix II Device Handbook, Volume 1, Jul 2004.

[3] Electronics Weekly; ABI/INFORM Trade & Industry, Feb 25, 2004, pg. 12

[4] F. Mo, A. Tabbara and R. Brayton, A Force-Directed Maze Router, *Department of EECS, University of California at Berkeley.*

[5] G. Nam, K Sakallah and R. Rutenbar Satisfiability-Based Layout Revisited: Detailed Routing of Complex FPGAs Via Search-Based Boolean SAT, *ACM/SIGDA International Symp on FPGA*, 1999, pp. 167-175.

[6] G. Nam, K. Sakallah and R. Rutenbar, A New FPGA Detailed Routing Approach Via Search-Based Boolean Satisfiability, *IEEE Transactions on computer-aided design of integrated circuits and systems,* vol. 21, no. 6, june 2002.

[7] H. Arslan and S. Dutt, ROAD: An Order-Impervious Complete Detailed Router for FPGAs, *ICCD* 2003 pp.350-356.

[8] H. Arslan and S. Dutt, An Effective Hop-Based Detailed Router for FPGAs for Optimizing Track Usage and Circuit Performance. *GLSVLSI* 2004.

[9] J. Swartz, V. Betz and J. Rose, A fast routability-driven router for FPGAs, *In 6th International Workshop on FPGAs*, 1998, Monterrey, CA.

[10] L. McMurchie and C. Ebeling, PathFinder: A negotiation-Based Performance-Driven Router for FPGAs, *ACM FPGA Symp.* 1997, pp. 111-117.

[11] M. Alexander and G. Robins, New Performance-driven FPGA routing algorithms, *32nd ACM/IEEE Design Automation Conference*, San Francisco, CA, 1995, pp. 562-567.

[12] R. Jayaraman,Physical Design For FPGAs, *ISPD* 2001, Sonoma, CA.

[13] R. Lysecky, F. Vahid and S. Tan, Dynamic FPGA Routing for Just-in-Time FPGA Compilation, *DAC* 2004, San Diego, CA.

[14] R. Tessier, Fast Place and Route Approaches for FPGAs, PhD thesis, *Massachusett Institute of Technology*, 1999.
[15] R.Tessier, Negotiated A* Routing for FPGAs, *in Proceedings of the Fifth Canadian Workshop on Field-Programmable Devices*, 1998.
[16] R. Wood and R. Rutenbar FPGA Routing and Routability EstimationVia Boolean Satisfiability, *ACM* 1997, Monterey CA.
[17] S. Brown, Routing Algorithm and Architectures for Field programmable Gate Arrays, PhD thesis, *Department of Electrical Engineering, University of Toronto*. 1992.
[18] S. Hauck and A. Agarwal, Software Technologies for Reconfigurable Systems, *IEEE Computer*, 1997.
[19] V. Betz and J. Rose, VPR: A New Packing, Placement, and Routing Tool for FPGA Research, in Proceedings, Field-Programmable Logic, *Oxford, U.K.* , 1997.
[20] V. Betz, VPR and T-VPack User's manual – version 4.30, 2000.
[21] V. Betz and J. Rose, FPGA Routing Architecture: Segmentation and Buffering to Optimize Speed and Density.
[22] V. Betz, J. Rose and A. Marquardt, Architecture and CAD for deep-submicron FPGAs, *Kluwer Academic Publishers, ISBN 0-7923-8460-1*, 1999.
[23] V. Gudise and G. Venayagamoorthy, FPGA Placement and Routing Using Particle Swarm Optimization, i*n Proceedings of the IEEE Computer Society Annual Symposium on VLSI Emerging Trends in VLSI Systems Design, ISVLSI* 2004.
[24] Xilinx Inc, Virtex II Data Book, 2004.
[25] Y. Changy, S. Thakury, K. Zhuz, and D. Wong, A New Global Routing Algorithm for FPGAs, *ACM* 1994.