# A Floorplanner driven by Structural and Timing Constraints.

Abdelhakim SAFIR, Baher HAROUN and K.THULASIRAMAN
Department of Electrical Engineering
Concordia University
1455 de Maisonneuve Blvd. W.
Montreal, Quebec, H3G 1M8, Canada

## ABSTRACT

This paper presents a novel layout model and floorplanning tool particularly suitable for taking into account user defined layout constraints on specific sets of modules and specific locations. The user defined layout constraints can be the setting of any common topological property associated with a group of specific modules such as the neighboring property for example. Or the use of any topological regularities in a design such as regular bus structure or the use of the structural property such as the bit-sliceable or non bit-sliceable feature of a module set, or their similar shape. The exploitation of these structural information helps in producing more compact layout especially for datapath oriented architectures. Moreover, in addition to the area and total wiring length, the critical path delay is systematically minimized through a global cost function. The potential candidates for the critical path computation can be specifically defined by the user. The core of the optimization process is based on Simulated Annealing (S.A.).

## INTRODUCTION

The wiring and connectivity in a VLSI circuit is becoming more preponderant since the size of the transistors in the silicon are shrinking more significantly than the wiring size emphasizing the effect of wiring on both area and delay. As a consequence, placement of architecture components has a significant effect on the over all architecture performance. This has lead to include methods of predicting delays by performing approximate placement during architectural design. Approximate placement was also incorporated in architectural synthesis [1]-[5] to control wire delay and satisfy the timing requirements.

Conventional timing-driven placement can be classified, mainly, into the following categories:

- Path delay control [6]where every path from inputs to outputs of a chip is checked to fulfil the timing requirement.

- Net delay control[7] where either higher weights are assigned to the nets in the critical path or bounds [8] on nets lengths are set.

These approaches assume that no specific higher level details are available for the components of the netlist. Higher level information on the structure of the modules can be very useful in placement and for accurate estimation of routing area and hence network delays. This is especially true in high throughtput datapath oriented designs such as the ones used in digital signal processing applications. In datapath oriented designs a large number of the modules that consume a major part of the floorplan are high-bit width modules. Moreover, certain design libraries have mega-modules composed of a number of hardwired modules that are already characterized for performance. Such mega-modules (some times also called cores) do not have essentially rectangular boarders and allow limited on top-of (through) the module routing. Other structural features for datapath modules also can affect the method with which routing is done either around or through the modules as well as possibility of module abuttment.

For example, if a module is bit slice-able it can abut with other bit-sliceable modules of the same bit width. Also, routing can sometimes be done through the bit-slices of a module. Such structural information of modules bit-sliceability, shape or any physical or functional aspect could be exploited to have a more compact layout.

On the other hand, routing can also be affected by the architectural style or by specific requirement of a technology. We also address such restrictions that can be inflicted from either the architectural or the technological aspects. For example, fixed layout architectural cores (mega-modules) usually exist in VLSI design libraries. In such cases, some of these cores can be used in larger designs. The possibility of using a fixed placement of these cores, as well as using available the routing resources within (on top) of these mega-modules is specifically appealing to accomodate in a floorplanner.

Therefore, in this paper, we address the problem of incorporating the structural and timing information of modules and interconnects in floorplanning. We specifically address bit slice-able modules that can abut with other bit slice-able modules in a vertical or horizontal direction in layout while allowing for over (or through) the bitslice routing. We also support Mega-module cores composed of non slicable modules, bitsliceable modules or both with shapes not essentially rectangular.

The plan of the paper is as follows. The floorplan and routing model is exposed in section 2. The optimization process is explained in section 3. Results on a typical datapath example are shown in section 4 and conclusions are drawn in section 5.

## The FLOORPLANNING MODEL

*Modules, holders and tracks*

Our main concern in providing for the floorplanning model is to allow for accurate area and routing, hence delay, modeling. Slicing tree representation for floorplanning have been used previously for datapath oriented designs(e.g. see [2], also [9] for a slicing tree combined with a simulated annealing search). But slicing tree representation has limitations: a- neighboring nodes are not explicitly represented, this property is important if one wants to fix the placement of a subset of modules that represent a (hardwired) mega-module, b- exchanges between *some* neighbor modules cannot be done in one step, this is very important for abutment style placement, c- moves of clusters of nodes (hierarchical moves) are not supported, d) routing is not accurately done (manhattan distance is usually used), hence, accurate delay due to routing distance is not possible. e) Slicing tree is not suitable for handling bitsliceable floorplans, especially if restrictions on feedthrough routing is required. Moreover, the relative geometric aspects of the modules used in the floorplan are not exploited.

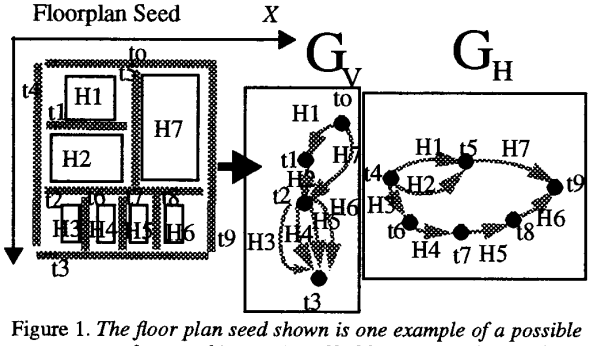


Figure 1. *The floor plan seed shown is one example of a possible arrangement for an arbitrary size of holders (Hi's). That seed can be totaly described by the two graphs $G_V$ and $G_H$. For example, the position of the t5 track is set by the maximum x dimension size of both H1 and H2 and by propagation, the position of the destination track t9 is set by both the position of t5 and t8 and the x dimension of h7 and h6. The track t0 (t4) is the source of $G_V$ ($G_H$) and track t3 (t9) is the termination of $G_V$ ($G_H$). t0-3 are vertical tracks and t4-9 are horizontal tracks.*

Because our first concern is the wiring delay and area estimation, we consequently provide a novel conception of floorplanning by using the concept of "*holders*" and "tracks". A holder is a location which accepts specific modules *with similar shape* and geometric properties. The modules can be abutted in a given direction (vertical or horizontal) if abutment is allowed by that *specific holder.* Actually the user can define any characteristic of a holder like the direction of stretching, the kind of modules it can get, the user can even assign to a specific holder a module which is fixed to this holder. On the other hand, a "track" accepts only wires. Tracks connect (glue) holders together. A floor plan is composed of an arrangement of holders *and* tracks where some may (or may not) have been assigned respectively modules *and* wires.

*The Floorplan Seed*

In order to use this floorplan model, *a floor plan seed has to be defined.* The definition of the floorplan seed is done by the user by specifying two constraint di-graphs $G_V$ *and* $G_H$ for vertical and horizontal directions respectively as shown in Figure 1. There are no constraints on the specification of these graphs as long as they do not have directed cycles. An example of floor plan seeds that are used to produce results in this paper is shown in Figure 2.

A constraint graph G is formed out of nodes representing routing tracks and edges representing holders. A node of the horizontal graph (resp. vertical graph) is only able to have its vertical (resp. horizontal) position changed. A node position can change because of the dimension of the incident edges. The dimension of an incident edge is set by the vertical (or horizontal) dimension of the corresponding holder(see Fig. 1). A vertical track $t_v$ has a vertical width

$W_{tv}$ = [max( $\Sigma$ width of incident edges to $t_v$, $\Sigma$ width of emerging edges from $t_v$)]

The width of a horizontal track is defined similarly. Note that a holder can have dimensions of zero if no module is assigned to it. Yet, the holder does not vanish from the graph, just from the floorplan (we usually indicate its existence in our diagrams by a line).

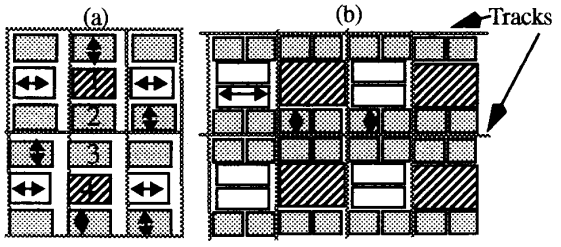


Figure 2. *Two instances of the adaptive floorplan model. In (a) the modules the non-slice-able and slice-able have roughly the same shape. In (b), the biggest non slice-able module is about twice higher and larger than biggest slice-able module. Each rectangle is called a holder and a holder can be only stretched in the horizontal or vertical direction in order to receive new modules. Note that only the crossed hatched holder receive a unique module.*

A holder "i" is said to be neighbor of the holder "j" iff they share the same track. *Any* two modules assigned to neighboring place holders can be exchanged in *a single* move in the stochastic search (small move). By allowing more than one module in a place holder, and allowing a move in the stochastic search of all the contents of a holder to another (or exchanging them), we have managed to reduce the number of moves in the search. Our approach is equivalent to a hierarchical search of floorplan alternatives.

Figure 2 shows one such layout model that we used for our results. We specify two types of holders, the first accepts functional units that are not bit sliceable like multipliers or large RAMs. This type of holder accepts only one such module. The reason for one module per holder is because no routing is allowed through (top of) the non slicable modules, and we assume that their interconnection is through the surrounding channels. Another type of holder, accepts the bit-slicable modules where more than one module can abut in either vertical or horizontal direction as indicated by the model. Since for the slice-able modules routing can be done through the cells at the bit level, the number of

interconnections/buses within a slice-able holder increases the height of the holder in the direction of the bit slicing. The size of the slice-able holder stretches with the assigned module(s) and the interconnections within. If no module is assigned to any holder it assumes a null size (but does not disappear). Notice that null holders may be used in the next iteration of the stochastic search and can be assigned a module.

The optimization of the area floorplan is done by minimizing the critical path in the horizontal and vertical direction. This floorplan model is represented by 2 graphs the horizontal and vertical one. It also allows incrementally computing the new area. This is done by propagating its new dimension through the (horizontal and vertical) graphs from the changed node which got the new module assignment down to a descendant node. If the descendant node is the final destination node than the area has been changed, otherwise, it means that the floorplan move did not affect the total area. This incremental capability is very important in stochastic search as it reduces the "cost" calculation time .

One more important feature of our floorplan model is that it takes into consideration the bit slice-ability of modules and its direction. It also can take into account limitations on routing through slice-able modules as well as the direction of the slicing.

A connection between two holders (i.e., between 2 modules from different holders) is assigned to a specific tracks set and this assignment is already memorized in a matrix where for any couple of holders, a list of tracks is associated.

The advantages of our floorplan representation over the plain slicing tree representation are:

- The capability of "wheel" or "spiral" representation.

- The easy and accurate representation of the wiring paths which are not simply "virtual" Manhattan distances over the total area as used in slicing tree but are actually assigned to specific wiring tracks. This allows to take into account the size of these tracks in the total area computation. Moreover, this would allow an accurate delay computation for buses.

- Easy possibility of fixing specific modules on a specific location. This is useful not only to fix large module like multipliers or RAMs but also to fix positions of I/O ports.

- The search space is judiciously reduced by repeatedly gathering modules with "similar shape" in one holder and possibly moving all of them together to another holder in the floorplan to reduce routing length or area.

- Particularly adequate but not limited to implementation in FPGA's where interconnection is restricted. One can easily put restrictions on the number of networks in specific tracks to account for some of the limitations in FPGAs.

## OPTIMIZATION Using SIMULATED ANNEALING

The basic optimization algorithm used in the system is Simulated Annealing. SA requires formulating the following four points for an optimization problem

1- An initial solution or starting point.

2- A cost function to minimize

3- Some transformations of the current solution or "moves" to scan the solution space. The moves result in a change in the status of the solution. A new status is achieved if the move is accepted.

4- A schedule of a control parameter called temperature in order to gradually converge towards a good solution.

Initial floorplan solution.

The initial floorplan solution is simply build by randomly assigning only one module to a holder.

Generating solutions

One of the most important condition for efficiently generating a good solution with S.A. lies in the ability to quickly "walk" from any given configuration to the optimum. To do so, it is necessary to limit the number of transitions or moves to reach this optimum but, simultaneously the tuning of the solution space must be fine enough to avoid missing the optimum. Therefore, to "walk" from state to state, 3 types of moves are defined So a new solution is generated with the following moves:

M1: Swap two modules in a vertical or horizontal direction within a place-holder

M2: Assign a module or the whole content of a holder to a new neighbor holder

M3: Swap two modules from different neighbor place-holders.

The cost function is a weighted sum of:

$F = \Sigma$ wire lengths+ Area + Clock cycle critical path delay.

Since by default the wiring delay is computed individually for each connection and each module knows their sources and destinations, this has made possible the computation of the critical path delay which is the largest delay of the local connections.

The delay for a bus (network) can be estimated through a simple RC delay model.

delay = $(\Sigma i\ Rout(comp_i))\ (\Sigma i\ Cout(comp_i)+Cw+\Sigma j\ Cload\ (comp_j))$.

"i" are the number of sources, "j" are the number of destination. Cw are the capacitance the wire. Of course, Cw depend on the wire length. Cload ($comp_j$) are the capacitance of the destination j.

## RESULTS

The floorplanning tool was used to place and rout a number of different signal processing architectures with strict performance requirements. We used a module library specification and technology values to obtain estimates of architecture clock cycles. Adders, registers, multiplexers, tristate buffers and register files are bit sliceable, multipliers are not bit sliceable. The relevant data for delay is: Adder 20ns and multiplier stage delay 20 nsec. Read or Write access time of the RF is 6 nsec. Set-up time for latches 2nsec. Bus delay is 1nsec/fanin, 0.5 nsec/fanout and a mux

delay of 1 ns. The first example is that of an elliptic filter .

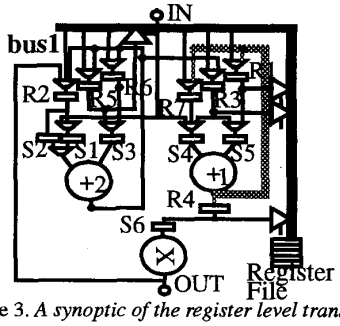


Figure 3. *A synoptic of the register level transfer implementation of the 5th order wave digital filter S# are slave registers, R# are master registers. 7 master registers, 6 slave registers, 24 mux. inputs, one pipelined bus with 4 inputs and 6 outputs connected to a five location register file are used. The critical path is set by the pipelined bus then immediatly followed by the net in shadow.*

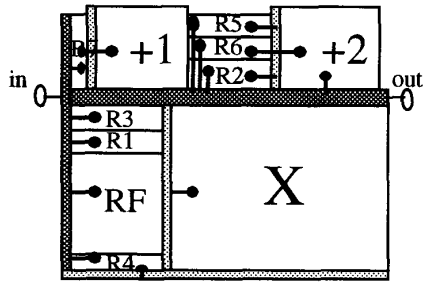


Figure 4. *The Associated Floorplan Multiplexers are merged with their output module in this floorplanning module. For example, note that the R2 module is slightly wider since it includes an extra mux input than R6.A total of 11 nets are used in this layout.The critical path delay is 102ns (which is set by the pipelined bus represented by the dark shadow). For the sake of clarity, we did not draw the empty holders.*

The CPU time for the routing and placement is about 2 minutes on a SPARCstation 10 for programs in an object oriented language based on lisp. To demonstrate the use of the floorplanner, in case of a restricted placement, we show figure 5. In that figure, a restriction was done on the placement of the two multipliers to be aligned and on top of each other, which resulted into a significantly different placement.

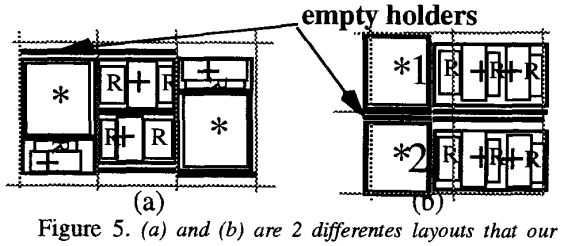


Figure 5. *(a) and (b) are 2 differentes layouts that our floorplan model (cf fig. 5)can handle. One can see in (b) the empty holders that have not been used. In (b) we have used the model of fig.5 (a) and for the purpose of illustration, we wanted the multiplier 1 directely above the multiplier2, therefore, we have forbidden any module binding in holders 2 and 3 and we have preassigned a fixed placement, the multiplier1(resp.2) to holder1(resp.2)*

## CONCLUSION

The novelty of the floorplanning representation results from the idea of representing the floorplanning problem as a binding of components to physical locations, each of them, called holder may have any kind of restrictions.We have shown through examples, how general and flexible enough is the floorplanning model. Moreover, this generality is not at the detriment of the specificity of particular design style like bus style architecture using full custom or FPGA's technologies. Indeed, we do not make rough approximation (like Steiner tree) of the netlist of connected modules to compute a bus length but we actually make (predefined) track assignments to buses and local connection. We also have shown in the FIR example that one can assign any desired location to a specific set of modules. And this capability is necessary when implementing for example components of the circuit with different size and technologies like standard cells and gate-arrays. Our floorplanning approach used in the presented tool produces *fast but accurate* estimates of area and network delays. Because of this, it has been possible to incorporate this floorplanning approach with architectural candidate evaluation in architectural design automation [10], where floorplanning was performed in conjunction with other architectural synthesis tasks. This demonstrates also the viability of our floorplanning approach in high level synthesis.

## References

[1] M.C. McFarland. "A fast Floor Planning Algorithm for Architectural Evaluation" 1989 (ICC(A)D)pp 96-99

[2] J. Weng, A. C. Parker, "3D Scheduling: High Level Synthesis with Floorplanning", DAC-91, pp.668-673.

[3] F.J. Kurdahi, C. Ramachandran. "Evaluating Layout Area Tradeoffs for High Level Applications" IEEE Trans. on VLSI Systems, Vol. 1. No. 1, March 1993.

[4] M. Nourani, C. Papachristou. "A Layout Estimation Algorithm for RTL Datapaths". 30th DAC 1993.

[5] A. Wu, V. Chaiyakul, D. Gajski, "Layout Area Models for High Level Syntehsis", Proc. ICCAD-91, pp.34-37.

[6] W.E. Donalth, R.J. Norman, et.al., Timing driven placement using complete path delays,DAC90, pp. 84-89, .

[7] M. Marek-Sadowska and S.P. Lin, Timing driven placement, Proc. ICCAD 89, pp. 94-97, 1989.

[8] H. Youssef,"Timing Issues in Cell Based VLSI Design", Ph.D. Thesis, CS. Dept., University Minnesota, Jan90.

[9] D.F.Wong, C.L. Liu. A New Algorithm for Floorplan Design. 25th DAC 88, pp.306-311.

[10] A. Safir, B. Haroun, K Thulasiraman. "Full Placement and Routing for Fast Architectural Synthesis" to be Submitted toICCD