Floorplanning with Datapath Optimization

A. SAFIR, B. HAROUN and K. THULASIRAMAN
Department of Electrical & Computer Engineering
Concordia University
1455 de Maisonneuve Blvd. W. Montreal, Quebec, H3G 1M8, Canada
Tel.: (514) 848-3072 haroun@vlsi.concordia.ca

Abstract

This paper presents a floorplanner for datapath with the capability of re-allocating data storage for minimizing the interconnect area and critical path delay without altering the number of Functional units and the schedule. The tool has combined two novel approaches: 1 - A placement & Routing model to handle different architectural topologies (mux. and/ or bus based) suitable for FPGA's. 2 - An efficient formulation for the binding of register/Interconnect & combined Floorplanning. The complexity of the architectural and floorplanning model, and of the cost function, have led us to the use of a stochastic optimization process. The running time of the whole process indicates the viability of the method. We show through various examples how the floorplanner improves the area and critical path delay of the datapath compared to a plain floorplanner. The improvement is about 20% for the critical path delay when this objective is a stringent constraint.

Key words: floorplanning & layout synthesis, FPGA's, Architectural Synthesis, Architectural trade-offs.

1 Introduction

1.1 Motivations

When designing high performance datapaths one cannot ignore the physical aspect of its implementation such as wire loading. For example in the binding process, it has been shown that the simple measure of the number of mux-input or buses is rather weak since the structure and the length of the interconnections is not considered in clock cycle and area determination. Previous research [1][2][3][4][6][7], have recognized the importance of performing floorplanning with different data-path synthesis tasks. What was not addressed as part of the synthesis cost was accurate routing that considers bit-slice implementations [5].

For implementation technologies like gate arrays (GA) and FPGA's, the wiring delay ratio to the combinational logic block (CLB) delay is increasing with the shrinking size of the gates in sub-micron technologies and because of the loading of configuration switches on routing interconnection (for FPGAs). Moreover, routing difficulties can decrease the CLB utilization emphasizing the effect of wiring on both FPGA size selection and performance. Therefore synthesizing datapaths that have a compact floorplan and regular interconnection is highly desired for FPGA's. Hence, by combining the effects of floorplanning and global routing, in datapath synthesis, to account for the wiring delay on the clock cycle time computation and area while determining the structure of a synthesized datapath, is more essential for an a GA or an FPGA implementation. This approach has not been widely adopted though, and this is due to the inherent complexity of the architectural synthesis problem combined with the physical implementation constraints.

1.2 Overview of Floorplanning in Datapath Synthesis.

One can say that the integration of low level information in architectural synthesis design tasks has been roughly made in the literature in three ways. The easiest and oldest method was to limit the problem to a rigid floorplan as in silicon compilers. The second way is to make an estimation of what should be the floorplan or to make a partial or approximate floorplan like in [1][3][4]. These methods can be classified as constructive and analytical methods [4]. The third way, which is the complete floorplanning in parallel with high level design tasks, is generally highly time consuming, especially in the case of constructive floorplanning methods. In this third category, [2] uses an iterative improvement for reducing only wiring delay and allowing an increase in operators, while [7] was using Simulated Annealing, but with the limitation of the slicing tree representation and with the restricted architectural model of a multiplexer based architectures. Very recently,[13] is using a ILP formulation for the binding problem with some limited interactions with floorplanning. Yet, in both [7] and [13], the critical path is not identified and they do not accounting for wiring delay.

Still within the third category, we are not aware in the literature of a complete and independent floorplanning tool that has the capability of redesigning the datapath. But what exists is floorplanning done sequentially after the synthesized datapath. In such a case, a number of iteration is necessary before getting satisfactory design. Moreover, within an iteration it is difficult to quickly guide the design towards a satisfactory outcome.

1.3 Contributions of the paper

Therefore, in this paper, we address the problem of incorporating the register & interconnect binding and the structure of the modules as the bit-sliceability, shape and routing while performing floorplanning and global routing.

Hence, we present a novel floorplanning & routing tool that can have the following properties: a) Satisfy constraints imposed on routing resources such as those found in FPGAs. b) Concurrently perform register and interconnect binding while floorplanning and routing the datapath.

c) Satisfy user set constraints such as: prefixed placement, bit-slice directions and non-rectangular module shapes.

Moreover, the novel floorplanning model used allows accurate and fast computation of total area, actual routing length and network delays. This tool accepts as an input a pre-scheduled and functional unit allocated data flow graph (CDFG). The output is a datapath with all register and interconnect binding performed as well as its global routing and floorplan. The datapath and its floorplan are optimized according to the cost tradeoffs in area and clock cycle.

In section 2, an overview of the model used for data storage in the datapath as well as the Floorplanning model. The datapath redesign/optimization is in Section 3 where the storage binding is explained together with the moves for performing floor planning and binding. Results are shown in section 4 and conclusions are drawn in section 5.

2 Problem Representation and Definitions

2.1 The Storage & Interconnect Binding Model

For our architectural model, the data transferred between two operations can be of either two types, Type-I or Type-II, in Figure 1. For type-I, the data is transferred from the FUx output to the master register R_n on clock cycle *i*, then transferred from the master register R_n to the slave register S_m on clock cycle *k*. For type-II, a bus transfer is added where the data is transferred at clock cycle *s* from master register R_n to Bus_q and then from Bus_q to master register R_p. Note that R_n accesses Bus_q through a tri-state and Bus_q accesses R_p through a mux input. Hence, to reduce bus loading the number of registers (and by implication the number of FU reading or writing in these registers) have to be minimized. The architectural model is discussed in

We assume that the values of clock cycles *i, j, k, s, t*, and FUx's, FU's for each edge in the CDFG has been *assigned*[14] and are an input to our tool.

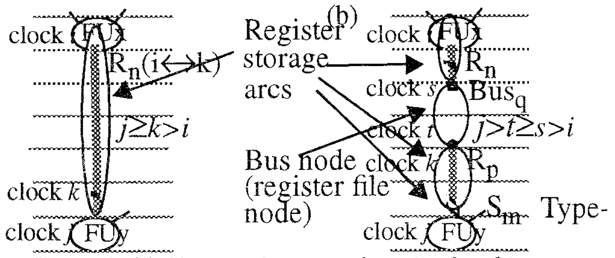


Figure 1: (a) *Shows a data transfer using local interconnections through individual master register Rn* (b) *Shows a data transfer using a pipelined bus. Rn and Rp do not have to be different, but have to have at least a life time of one cycle.*

2.2 The Floorplanning Model:- Modules, holders and tracks.

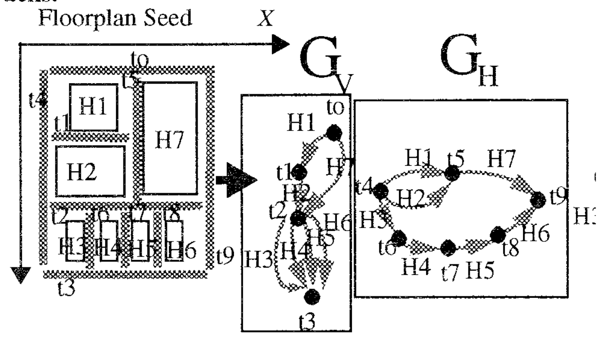


Figure 2: *The floorplan seed shown is one example of a possible arrangement for an arbitrary size of holders (Hi's). That seed can be totaly described by the two polar graphs G_V and G_H . For example, the position of the t5 track is set by the maximum x dimension size of both H1 and H2 and by propagation, 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.*

Our floorplan model, summarized in Figure 2 and detailed in [12] has additional advantages over the plain slicing tree model conventionally used in synthesis:

- The capability of "wheel" or "spiral" representation.
- 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. Neighborhood relations are very easy to identify in this model which helps in reducing the search space.
- Particularly adequate but not specifically 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 routing limitations in FPGAs.
- The seed structure depends on the implementation technology and the direction of bit-slice allowed.

3 Register& Floorplanning Optimization

First of all, recall that all the operation and the bus bindings have been already performed. Referring to figure 1, the only assignments left are for the *storage arcs* to registers and *bus arcs* to a register file (RF) locations. Note that *these arcs* are defined by their source and destination nodes which can be either a FU or a bus.

Bus arc assignment to RF locations is straight forward by using known register minimization (circular-arc coloring) algorithms. *Storage arc* assignment to registers requires considering a number of issues; a) number of registers used, b) local interconnections used to transfer data to register modules from FU modules, c) local connections used to transfer data from registers to FU modules, b) number of multiplexers used in register and FU modules, d) number of troostite drivers used in register modules to drive busses, e) bus and interconnection loading due to fanin, fanout and network length, f) effect of loading on clock cycle. Hence, this step of storage assignment with floor planning is crucial in obtaining an architecture optimal from both performance and area cost aspects. To perform this optimization we have to address the computational costs in implementing a SA search. In the ensuing discussion, we highlight the important features of this formulation.

<u>Cost function to be minimized</u>: It is a weighted sum of: # mux-input + S wire lengths+ # tristate buffers+ # registers + Total Area + Clock cycle. Since the wiring delay is computed individually for each connection and each module knows its sources and destinations, this has made possible the computation of fanin and fanouts for all modules and hence, the critical path (CP) delay.

An initial binding and floorplan is made using a heuristic assignment. Further re-binding of transfers to registers as well as floorplanning and routing re-assignment is performed using a simulated annealing search. When register binding is performed it may result in an introduction of a new register or the annihilation of an existing one. It also implies a new interconnection generated and/or deleted from the floorplan. Also multiplexers can be generated or anhilated.Indeed, it is the floorplan that can actually determine the true area cost of this re-assignment, and the cost function for assessing its impact on performance.

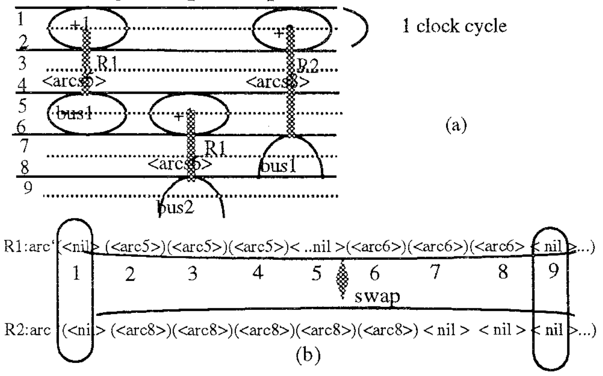

1 clock cycle
1
2
3
4
5
6
7
8
9
+1
+2
R1
R2
<arc5>
<arc6>
bus1
+3
R1
<arc8>
bus1
bus2
(a)
R1:arc
(<nil>(<arc5>)(<arc5>)(<arc5>)<..nil >(<arc6>)(<arc6>)(<arc6>)(<nil>...)
1 2 3 4 5 6 7 8 9
swap
R2:arc
(<nil>(<arc8>)(<arc8>)(<arc8>)(<arc8>)(<arc8>)< nil > < nil >(<nil>...)
(b)

Figure 3: *R1 and R2 store the ordered list of objects <arcs.i> representing the link between the source-destination nodes of the graph in (a). In this way, the move "swap assignment" of registers can be applied on 2 lists of successive data arcs objects. In this Figure, in one move, "<arc5>" and "<arc6>" can be assigned to register 2 and the "<arc8>" can be assign to register 1. The alignment of two "nil" s (encapsulated) in the list is mechanism with which the swap range for the move is determined.*

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. Each new binding for the datapath or a new floorplan is generated using a move in the design space. Therefore, to "walk" from state to state, 7 types of moves are defined; 4 for the assignment process in isolation and 3 for the actual floorplanning. These are:

M1: The swapping assignment move: this is explained in Figure 3.The conditions for a swap assignment moves are:

- A successive sequence of identical "object" <arcsi> cannot be broken.
- The order of any "object" <arcsi> when swapping from one list ":arcs" in register Ri to the list ":arcs" in register Rj must be conserved.

M2: Merging or Splitting of "brothers" arcs. Two arcs are "brothers" if they have the same source node. The Brothers arcs can be assigned to the same register without conflicting.

M3: A random number of arcs within the same register can be assigned to different available registers. For each arc, the registers are sorted in order to get the first available register.

M4: Two variable inputs are interchanged if the operation is commutative.

M5: Swap two modules in a vertical or horizontal direction within a place-holder

M6: Assign a module or the whole content of a holder to a new neighbor holder

M7: Swap two modules from different neighbor place-holders.

M1-M4 are binding moves and M5-M7 are floorplanning moves. This set of moves are selected and applied in two cases. The first case we call binding and floor planning in isolation, while the second case we call combined floorplanning and binding. These cases are:-

I- Only the set of binding moves are applied - no clock cycle or area cost is computed only architecture component count contributes to the cost function. These moves are applied with a SA schedule until the cost function is minimum. This is followed by a sequence of Floorplanning moves where the clock cycle and the area are minimized but no binding moves are used.

II- Both binding and floor planning moves are intermixed and applied in SA schedule. The cost function in this case includes all components.

4 Results

4.1 The 5th order wave digital filter (EWF)

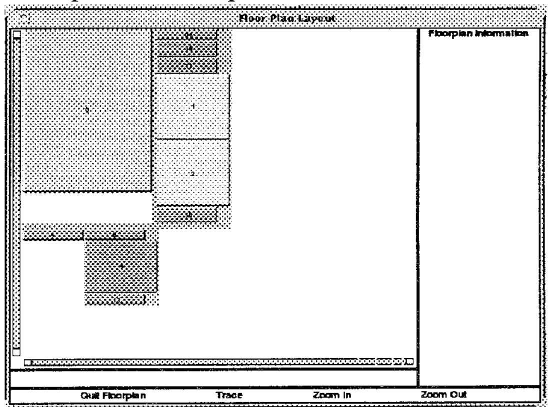
We used a module library specification and technology values to obtain estimates of architecture clock cycles. The relevant data is: Adder 22ns and multiplier stage delay 20 nsec. Read or Write access time of the RF is 7nsec. Set-up time for latches 2nsec. Bus delay is 2 nsec/fanin, 1 nsec/fanout and a mux delay of 2 ns. The following results are obtained from the scheduled data flow graph of the 5th order wave digital filter with 2 adders, one pipelined multiplier and 17 clock cycles.

In the tables below :

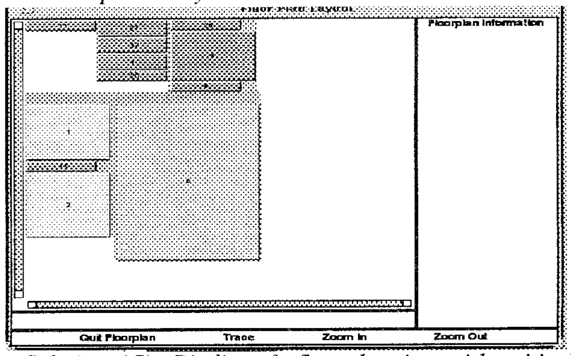
cases A : floorplanning after binding with min. register.
cases B: floorplanning after binding with min. #muxes
cases C: floorplanning & binding in parallel.

Priority	DELAY			AREA		
	A	B	C	A*	B*	C*
#register	7	11	8	7	11	7
#muxes	22	16	25	22	16	20
Clock	58.82	52.44	47.3	83.7	66.7	77.45
Area	40.58	55.7	35.69	32.05	33.29	30.57
#iter	20881	39725	60032	70306	37964	55163

Example for the floorplans of the cases A & C



Solution (A): the Datapath with minimum register. Note that area in this example has a low priority relative to critical path delay.



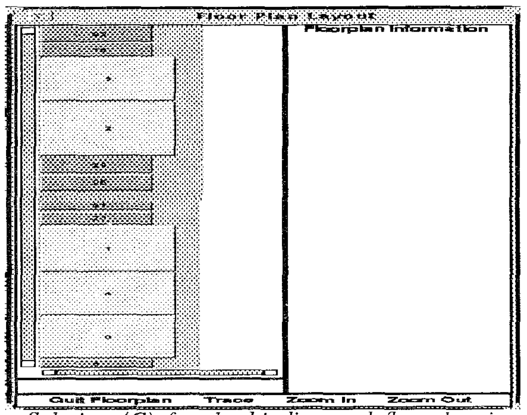
Solution (C): Binding & floorplanning with critical path delay constraint

4.2 Example II

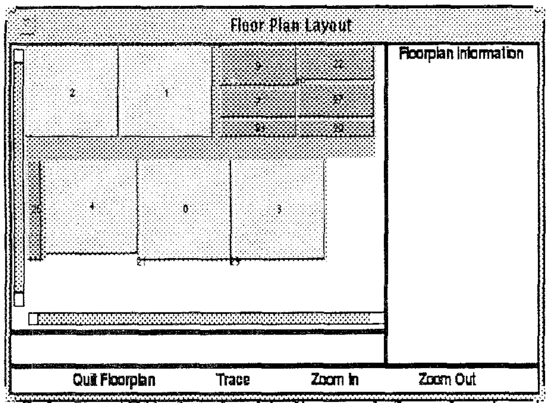
This second example is a wider example in term of F.U's with 5 adders. But the variety of modules is limited to adders and registers.Notice that what is important for performance is to reduce the number of mux-inputs on the critical path and not necessarily the total number of muxes as a whole.

Priority	DELAY			AREA		
	A	B	C	A*	B*	C*
#register	7	10	7	7	10	7
#muxes	26	22	29	26	22	27
Clock	45.9	42.0	38.1	79.5	71.05	60.9
Area	34.62	39.87	24.95	26.92	25.99	23.52
#iter	21487	58882	74476	21839	51348	51142

Below is some pictorial results of the floorplan obtained by the tool in cases C & C*



Solution (C) for the binding and floorplanning in parallel with high critical path delay minimization



Solution (C*) for the binding and floorplanning in parallel with area constraint

The CPU time for the storage binding in isolation without any floorplan considerations is about 10 minutes on a SPARCstation 10 for programs in an object oriented language based on lisp and the routing and placement in isolation is about 10 minutes. The merging of these 2 tasks in parallel leads to a CPU time of 60 minutes as an average of 10 runs.

We can notice a significant improvement of the solution of Floorplanning & Binding in parallel compared to binding followed by floorplanning.

In order to compare the 2 approaches - Binding followed by floorplanning & floorplanning with binding - the examples shown are solutions with a particular stringent constraint: clock cycle time minimization or area minimization. For the binding in isolation we took the two solutions, one with minimum number of registers and one with the minimum number of multiplexor inputs. Then we run the Floorplanning process in isolation with an extreme priority constraint on the area and on the delay.

The solutions obtained from the running in parallel of the binding & floorplanning show that the best solution, is apriori unknown to the user in term of # registers and # of mux-inputs. Indeed, if we apriori minimize the # of registers we maximize the # of mux-inputs and conversely (since these objectives are antagonist) and we have no clues of what could lead to the best solution as we can see in the tables. The best solution with the minimum clock delay requires 8 registers and 25 mux-inputs while the minimum register solution for the binding is 7 registers and 22 mux-inputs, and the minimum Mux-inputs solution is 16 Mux-inputs and 11 registers.

5 Conclusion and future work

We have presented in this paper a method for combined datapath redesign and floorplanning. The redesign is performed by rebinding the interconnections and registers (or register-files) in the datapath. The floorplanning is concurrently performed while taking into account the effects of global routing on clock cycle as well as the datapath and FPGA structural features to implement modules such as the sliceability and their shape as well as fixed placements.

The integration of these complex tasks has been made possible due to two particular aspects of our tool:

(1) The first aspect is the novelty of the floorplanning representation which makes it superior to the slicing tree representation used by others in a number of respects that were discussed.

(2) The second aspect is the stochastic iterative feature of the optimization process. The set of moves and the binding space formulation that we have presented have proved to effectively search the design space and produce good results. The CPU time of the whole process is reasonable.

(3) We have shown that concurrently performing register and interconnect binding with floorplanning and global routing is effective in obtaining high performance architectures as compared with conventional approaches which perform these tasks sequentially.

References

[1] M.C.McFarland, Using Bottom-Up Design Techniques in the Synthesis of Digital Hardware from Abstract Behavioral Descriptions, Proc. 23rd Design Automation Conference, 1986
[2] 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 Very Large Scale Integration 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. of ICCAD-91, pp.34-37.
[6] C. Ewering. Automatic High Level Synthesis of Partitioned Buses. IEEE 1990
[7] A. Safir, B.Zavidovique"Towards a Global Solution to High Level Synthesis" EDAC, Glasgow, Scotland,1990.
[8] T. Ly, J. Mowchenko," Applying Simulated Evolution to HLS", IEEE Tr. CAD, march 1993, pp.389-409.
[9] A. Wu, V. Chaiyakul, D. Gajski, "Layout-Area Models for High Level Synthesis", Proc. of ICCAD-91, pp34-37.
[10] D.F.Wong, C.L. Liu. A New Algorithm for Floorplan Design. 25th DAC 88, pp.306-311.
[11] J.Rose, W. Klebsch. "Temperature Measurement and Equilibrium Dynamics of Simulated Annealing Placements" IEEE trans. on CAD. Vol. 9. No. 3. March 1990.
[12] A. Safir, B. Haroun. "A floorplanner driven by structural and timing constraints" ISCAS-94, London June-94..
[13] M. Ring, A. Mujumbar, R. Jain, R. De Leone. "Optimal and Heuristic Algorithms for Solving the Binding Problem." IEEE Trans. VLSI Systems, Vol.2, No.2, June 1994
[14] A. Safir, B. Haroun, K. Thulasiraman. "A Floorplanner for Datapath Optimization". Submitted to IEEE Trans. on VLSI Systems. October 1994.