RTL Synthesis with Physical and Controller Information

Min Xu[†]
Fadi J. Kurdahi [‡]
[†] Department of Information and Computer Science
[‡]Department of Electrical & Computer Engineering
University of California, Irvine, CA 92697

Abstract

The current technology advances towards deep submicron have made it indispensable to consider layout and controller effects during all phases of chip synthesis. This paper proposes a paradigm for incorporating such information when synthesizing an RTL design from a scheduled behavioral specification. Experimental results corroborate the fact that layout and controller effects on chip area and performance are significant and can not be ignored.

1 Introduction

High-Level Synthesis (HLS) typically uses generic, abstract models of hardware during the tasks of scheduling, allocation and binding. The use of these models simplifies HLS algorithms and standardizes the output of HLS to a generic format so that it can then be implemented in a particular technology through register-transfer-level (RTL) synthesis (e.g., logic synthesis, technology mapping and physical design).

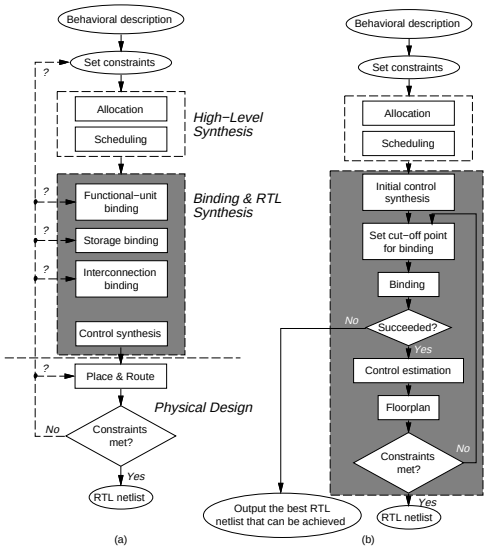

Figure 1: (a) A typical design methodology of an automatic behavioral synthesis system (b) A new approach which incorporate physical information into task of binding

However, experimental evidence indicates that there is tremendous variation in hardware attributes based not only on the target technology chosen, but also on the physical design of each implementation. BUD [1], Chippe [2] and Fasolt [3] clearly indicated the significance of interconnect and other layout effects –traditionally considered as second order in HLS– on the overall implementation area and delay. For HLS algorithms (e.g., scheduling, allocation and binding) to make effective decisions that eventually result in high-quality layouts, we need to incorporate physical design information during HLS. We must account for not only place and route effects, but also global considerations such as RT wiring, component styles, aspect ratio, floorplanning, and the combination of "all of the above". Without such information, the RTL designs may produce unpredictable results when implemented on silicon.

Another important factor that has been typically ignored during HLS is the controller. A complete model of the controller from state diagram necessitates a modeling of the logic as well as the physical design phases. Modeling the impact of logic synthesis is an extremely complex task which has received very little attention in the past.

Figure 1(a) shows the flow of scheduling, allocation, binding and physical design in a typical design methodology of an automatic behavioral synthesis system. This traditional flow treats synthesis (both behavioral and logic) and physical design independently and suffers from four major drawbacks: (1) it is not known whether the design will meet the constraints or not until the end of the time-consuming phase of place & route; (2) when the constraints are not met, it is difficult to identify where the problem comes from and at which level the design should be modified, and there is no way to identify the constraint which leads to a situation when no feasible solution can be found; (3) the three subtasks (FU, storage, and interconnection binding) are tightly related to each other, and the deadlock situation between them is still an open problem in HLS. (4) the control synthesis is time consuming and if run repeatedly, would significantly increase the runtime of the overall HLS procedures.

In contrast with previous approaches, we incorporate physical and controller information into the task of binding as shown in Figure 1(b). The main features of this work are the following: (1) the use of control estimation makes it possible to incorporate the impact of the controller into HLS procedures and hence make the HLS's final decision more

ED&TC '97 on CD-ROM

realistic and predictable; (2) the final result is evaluated without actually going through the time consuming phase of place & route; (3) when time constraints are met, the algorithm will output not only a structural RTL netlist, but also its corresponding physical topology which can be carried through silicon implementation in a predictable manner; (4) whenever time constraints are not met, our binding techniques provide a means of exploring the design space in a realistic and efficient way; with this exploration, our binding techniques will provide feedback to the previous tasks if the constraints can not result in any feasible solution, and will output the best implementation that can be achieved; (5) we break the deadlock situation among FU, storage, and interconnection binding by performing these three subtasks simultaneously, and take into account physical and controller information as well.

While our proposed approach is valid for any technology, we benchmark the results with respect to the Field-Programmable Gate Array (FPGA) design style because of its popularity. Specifically, the Xilinx XC4000 series is assumed to be the layout design style for the remainder of this paper.

2 Previous Work

3-D [4] presented an approach to the problem of binding while simultaneously considering floorplanning. However, this approach didn't consider the cost and delay of registers, multiplexors, and wiring space overhead.

GBA [5] and BITNET [6] also considered binding with physical information. However, GBA applies only to one dimension (linear) bit-slice design, and BITNET does not consider interconnection delay.

Ewering [7] and ApplaUSE [8] addressed the binding with physical information problem by moving placement earlier before bus and register assignment, but no physical information is taken into account when FU binding is performed.

SMB [9] presented an integrated approach for minimizing critical path delay by simultaneously performing FU binding and floorplanning. But their approach has to start with a fixed floorplan and does not account for the shape and delay of multiplexors which affect the delay of the critical path. Furthermore, it is not clear whether SMB can handle multi-cycled FUs or not, and no controller information is taken into consideration.

In [13], abstracted layout area and timing models for HLS were presented. These models were experimentally shown to accurately and efficiently reflect the effects of the data path design tradeoff on the final layout. However, these models concentrated on modeling the datapath and the controller separately and did not consider the impact of floorplanning and logic optimization which could generally be a significant factor in area and delay. In [14], a model for estimating the controller *complexity* was proposed. But the model does not incorporate delay estimates and furthermore, does not account for wiring effects since no netlist is produced.

On the other hand, our approach does not rely on any given floorplan, and we take the shape and delay of multiplexors and the controller into consideration. Furthermore, during datapath binding, we consider clock period (register-to-register delay) in the datapath as our main process object rather than FU-to-register or register-to-FU delay as the main concern as in SMB.

3 Architectural Model and Problem Definition

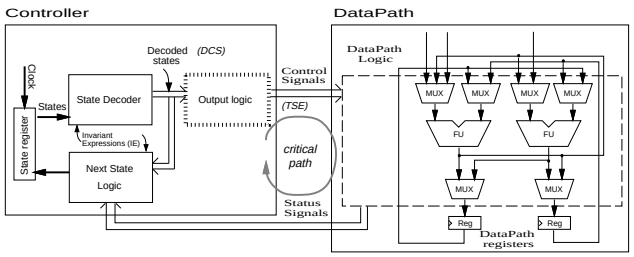

Figure 2: Architectural model

Typically HLS systems use a Finite State Machine with a Datapath (FSMD) [13] design model, which consists of two important components: (a) a controller which can be represented as a finite state machine and synthesized into a state register and combinational logic; (b) a datapath which contains the functional units and storage units that perform the required computations. Although our approach can handle both multiplexor-based and bus-based architectures, we confine our scope in this paper to the multiplexor-based architecture model (Figure 2). Operation chaining is supported in this model by allowing connections from the output ports of some FUs directly to the input ports of other FUs. Moreover, operations can execute over several clock cycles: multi-cycled operations are possible.

The control unit consists of the state register, the decoder, the control logic to drive the control lines for the datapath components, and the next-state logic to compute the next state to be stored in the state register. The control unit implements a state machine that sequences through a series of states.

Our problem can be defined as follows:

> *Given (1) a scheduled data flow graph (SDFG), (2) the number of FUs, registers, input and output multiplexors and (3) maximum clock period, which is usually part of the system specification, identify whether there is a feasible RTL solution or not. If there is, after performing binding, generate the RTL netlist and its corresponding floorplan; Otherwise, report it to previous tasks in HLS and output the best solution that could be achieved.*

The example in Figure 3 illustrates the problem. Given are a scheduled data flow graph which consists of two control steps, and an allocation resource which includes 2 adders and 4 registers. The shape functions of the components and their corresponding delay information can be

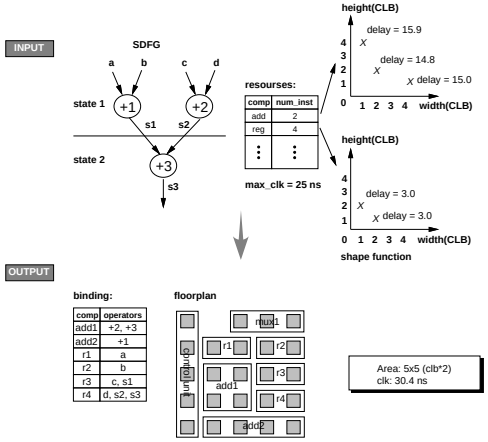

Figure 3: An example illustrating the inputs and outputs of the problem

obtained from the component library. The shape function and delay of the controller can be obtained by invoking our Estimation tools. Our algorithm will output a RTL datapath netlist with all the binding information. Meanwhile, a corresponding floorplan and the clock period (register-to-register delay) which includes wire delay will also be generated.

4 Our Approach

Given a scheduled data flow graph, first, we construct a fully connected netlist in which each FU is connected to every register and each register is connected to every FU. Then, control initial synthesis is performed and a shape function for each datapath component is obtained from the library. After that, ChipEst-FPGA [10] is used to generate an approximate topology of the overall design so that the distance metrics can be used to calculate register-to-register path delays.

The backbone of our datapath layout driven binding approach is a branch and bound algorithm. We introduce a *cut-off point for binding*. We define the number which decides whether the path will be used for binding or not as the *cut-off point for binding*. Only paths with delays smaller than the cut-off point can be used in binding. Furthermore, when the cut-off point is smaller than a certain number, there may be no sufficient paths for binding. We define this limit point as the *cut-off-point threshold*.

After binding, control unit is estimated. Then ChipEst-FPGA can be re-invoked on the whole chip. Details of each of the steps will be discussed in the following sections.

4.1 Datapath Binding

First, register-to-register path delays are calculated based on component delay and wire delay. The component delay can be obtained from the library. From the distance metrics, we can calculate the wire length and use our estimation tool to determine the wire delay [11].

4.1.1 Setting Cut-Off Point for Binding

Knowing all the path delays, we can set the cut-off point such that paths having delays greater than the cut-off point are discarded from the candidate for binding. Let's denote the initial cut-off point for binding as CT_{init} and the cut-off point for the current iteration as CT_{curr}. Let cr_delay_{prev} be the critical datapath delay of the previous datapath synthesis solution and α be the factor for choosing the current cut-off point. The user can decide whether α should be equal to 10, 100, 1000... so that the tradeoff between the time spent on exploration and the number of solutions explored can be made.

The initial cut-off point and the current cut-off point can be obtained by the following equations:

$$CT_{init} = \lceil MAX(Delay_{i,j,k} | i,k = 0,1,...r; j = 0,1,...f) \rceil \quad (1)$$

$$CT_{curr} = \lfloor cr\_delay_{prev} * \alpha \rfloor / \alpha \quad (2)$$

where $Delay_{i,j,k}$ is the delay from the ith register through the jth FU to the kth register in the datapath, r is the number of registers, and f is the number of FUs.

Once the cut-off point is given, we try to find a solution that meets the constraint. During binding, a feasibility check is needed to determine if there are enough paths with delay less than the cut-off point to perform binding. The feasibility check is carried out every time a new object (FU, Ovar, or Ivar) has been virtually bound. This speeds up our search algorithm and will stop the algorithm whenever the cut-off point hits the cut-off-point threshold.

4.1.2 Binding

As we mentioned in Section 4, we use a branch and bound algorithm to search for different binding possibilities one control step at a time sequentially. Within each control step, the virtual binding is carried out in the order of FU, Ovar, Ivar and multiplexors. The FU, Ovar, and Ivar binding call the same recursive binding procedure (which is outlined in [15]) to generate all the different possible solutions while the feasibility check in every step prunes the infeasible solutions as early as possible.

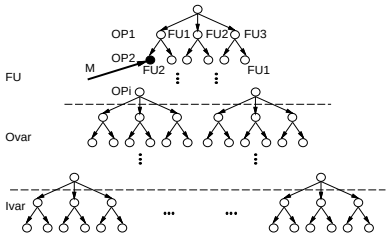

Figure 4: A search space tree

The search space can be shown with a tree having three levels of hierarchy as shown in Figure 4. The first level is for FU, the second level is for Ovar and the third level is for Ivar. At FU level, the depth of the tree is equal to the number of operators in the control step, and each path is a virtual binding for all the previous objects (an object can be FU, Ivar, or Ovar). After finishing FU binding, the

binding procedure proceeds to the Ovar and Ivar level. A backtracking mechanism enables the algorithm to backtrack up to the higher level of the virtual binding solution when the current virtual binding fails and to resume the binding process. The inputs to this algorithm consist of a source object to be bound, a set of target object candidates, and an allocation resources. The algorithm either generates an actual binding solution if one exists under the given cut-off point, or reports that no feasible solution is available, together with the best result that can be achieved.

4.2 Control Unit Estimation

In FPGA, control synthesis mainly consists of two phases: (1) technology independent optimization, and (2) technology mapping. The following two sub-sections will explain how we predict the optimization and use CompEst-FPGA to mimic the technology mapping and place & route.

4.2.1 Predicting Technology Independent Optimization

During binding and RTL synthesis iteration, the design is not being re-scheduled and the number of states remains a constant and so do the transitions between the states. In the Moore machine controller model, the state encoding depends on the state transitions and hence would be invariant during binding and RTL synthesis [2]; On the other hand, the controller output is only dependent on the current state of the design. In other words, the boolean value on the datapath control lines is a function of the Decoder Current State of the datapath (DCS).

Figure 2 shows the Partitioned Control Model(PCM) that we propose for the purposes of use with the binding and RTL synthesis phase of HLS. The partitioned control model consists of two sets of expressions: the Invariant Expressions(IE) consisting of the next state logic and the state decoder equations, and the Transformation Sensitive Expressions (TSE) consisting of the output logic equations.

State encoding and initial logic synthesis are performed only once in the initial control synthesis step by running IE.

To determine a relation between TSE and DCS (decoded current state), TSE_i of a datapath control line can be expressed as

$$TSE_i = \sum_{j=1}^{N} (V_{ij} * DCS_j) \qquad (3)$$

where, V_{ij} is a boolean variable and N is the number of decoded states. This relationship can also be expressed in the form of a bipartite graph as shown in Figure 5(a). The left nodes of the graph are the DCS_j's and the right nodes are TSE_i's, The nets of the graph are the V_{ij}'s and a net exists if the value of the corresponding V_{ij} is 1.

[2]This is a basic assumption in our system. However, we note that datapath changes could sometimes affect the status lines, and re-encoding of the states may result in changing the next state logic structure.

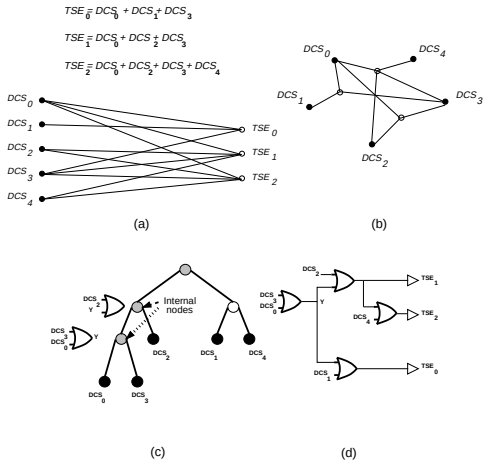

Figure 5: (a) Bipartite graph model of TSE (b) hypergraph of bipartite graph (c) hierarchical cluster tree of bipartite graph (d) gate level netlist

We can now cast the logic optimization problem as the bipartite graph clustering problem. Let C_j be a boolean variable such that it is equal to 1 when all the DCS_j nodes in the cluster K_l are connected to a node TSE_i:

$$C_j = \prod (V_{ij}) \quad \forall DCS_j \in K_l \qquad (4)$$

Hence, the clustering problem is that of determining K_l such that C_j is maximized.

We have transformed the bipartite graph to a hypergraph as shown in Figure 5(b). The nodes of the new graph are the left nodes of the bipartite graph or the DCS_j. The nets of the graph are TSE_i. TSE_i is connected to a DCS_j such that V_{ij} is 1. The hypergraph can now be partitioned such that the cost function C_j shown in expression 4 is maximized. The internal nodes of the tree can now be directly mapped to the OR gates in the technology library. This way, we can derive a logic netlist of the output logic as shown in Figure 5(d). For details, please refer to [12].

4.2.2 Predicting Technology Mapping

Once we obtain the gate level netlist for output logic from control estimation, the next state logic and the encoding logic, we use our physical level estimation tools CompEst-FPGA [11] to obtain an approximate topology of the layout for output logic. CompEst-FPGA is a component level estimation tool which predicts the area and delay of a given RTL component netlist. Benchmarking has shown that CompEst-FPGA can estimate area with about 2.5% accuracy and static delay with about 2-13% accuracy. For details, please refer to [11].

4.3 Layout & Clock Period Adjustment

Once we get the shape function for the output logic, layout adjustment will be performed by running the RTL netlist comprised of the datapath and the new controller through ChipEst-FPGA [10]. The clock period for the whole chip is determined by the longest *register-to-register* delay and is estimated by ChipEst-FPGA. Typically, the

path through the control logic, as shown in Figure 2 has the largest delay. Therefore, in our estimation, the clock period *clk*, is approximated using the following equations:

$$clk = t_{dp} + t_{ctr} + w_{cd} \quad (5)$$

$$t_{ctr} = t_{dec} + t_{output} + t_{ns} + w_{cu} \quad (6)$$

where t_{dec}, t_{output}, t_{ns} are delay of decoder, output logic and next state logic, respectively. w_{cd}, w_{dp}, and w_{cu} are wiring delay between controller and datapath, datapath and control unit, respectively.

After layout adjustment, if the cycle time still can not satisfy the maximum clock period constraint, we need to decrease the cut-off point and redo the binding. This iteration will continue until the cut-off point hits the cut-off-point threshold.

5 Experimental Results

We have implemented our layout-driven RTL binding techniques for HLS in C on the Sun SPARC workstation. The designs used to test our binding techniques are from some well-known HLS benchmarks: (1)*the 2nd order differential equation solver*; (2)*the 5th order elliptic wave filter (EWF)*; (3)*Discrete Cosine Transformer (DCT)*. The bit-width of all the examples is 4.

Examples	Resources	RTL constraints (FU, reg only) (ns)	Cut−off point for binding (ns)	datapath_delay w/o adjustment (ns)	datapath_delay (w adjustment) (ns)	Clock period w controller (ns)	Chip Size (CLBxCLB)
DCT	8A,4M,15R	44.5	168.0	167.1	74.6	105.5	26x26
		44.5	167.0	166.5	71.1	103.7	26x26
		44.5	166.0	163.0	70.8	101.8	26x26
		44.5	162.0	161.7	72.6	110.4	26x26
		44.5	161.0	160.8	73.2	110.3	24x24
		44.5	160.0	159.9	72.3	109.1	24x24
		44.5	159.0	158.8	72.0	109.2	24x24
EWF	3A,2M,12R	44.5	120.0	119.1	71.2	109.1	16x16
		44.5	119.0	118.9	71.6	102.6	16x16
		44.5	118.6	116.6	67.5	105.7	16x16
HAL	2A,2M,8R	44.5	92.0	91.8	62.5	93.8	12x12
		44.5	91.0	90.3	62.7	94.1	12x12
		44.5	90.0	90.0	62.3	85.3	10x10
		44.5	89.9	89.9	65.1	84.7	12x12
		44.5	89.5	89.5	65.4	82.1	12x12

Figure 6: Experimental Result

Figure 6 shows the results. The first column is RTL constraints which include only FU and register delays since they have no interconnection and layout information. The second column is the cut-off point for binding obtained by using the formulas 1 and 2, or set by the designers. The third column is the cycle time before adjustment. If the binding succeeds, our estimation tools can be used to estimate the controller shape function and construct the actual RTL netlist and to find its actual cycle time. In Figure 6, datapath_delay is the worst case *register-to-register* delay in the datapath after adjustment, while the clock period with controller is the chip clock period including both wiring delay as well as controller delay. These results clearly indicate that: (1) the controller has a large impact (up to 30%) on performance when controller effects are considered; (2) the datapath with the smallest *register-to-register* delay may not necessarily result in highest performance; (3) as the RT-Level delay constraints decrease, the candidate datapaths will be confined into a smaller group, so that datapath sharing will increase. This may result in more complex output logic and hence may decrease the performance; (4) layout and interconnection delays are significant since they may contribute up to 50% of the overall delay; and (5) by varying the cut-off point, we can explore a set of alternative binding solutions with varying clock cycle time.

6 Conclusion

We presented a RTL synthesis approach which simultaneously binds FUs, registers and interconnections in the datapath synthesis and consider both physical and controller impact on overall design quality and also uses an accurate layout estimator to simultaneously produce an RTL solution and a corresponding floorplan.

7 References

[1] M. C. McFarland, "Using Bottom-Up Design Techniques in the Synthesis of Digital Hardware from Abstract Behavioral Descriptions," *Proc. 23rd DAC*, pp. 474-480, July 1986.

[2] F.Brewer and D.Gajski, "Chippe: A System for Constraint Driven Behavioral Synthesis," *IEEE Trans. on CAD*, Vol. 9, No. 7, pp. 681-695, 1990.

[3] David W. Knapp, "Fasolt: A Program for Feedback Driven Data-Path Optimization," *IEEE Trans. on CAD*, Vol. 11, No. 6, pp. 677-695, 1990.

[4] P. Gupta and A. C. Parker, "SMASH: A Program for Scheduling Memory-Intensive Application-Specific Hardware," *Proc. 7th International Workshop on HLS*, pp. 54-59, May 1994.

[5] Hyuk-Jae Jang and Barry M. Pangrle, "A Grid-Based Approach for Connectivity Binding with Geometric Costs," *Proc. ICCAD*, pp. 94-99, 1993.

[6] Ashutosh Mujumdar, Minjoong Rim, Rajiv Jain, and Renato De Leone, "BITNET: An Algorithm for Solving The Binding Problem," *7th International Conference on VLSI Design*, pp. 163-168, 1994.

[7] C.Ewering, "Automatic High-Level Synthesis of Partitioned Busses," *Proc. EURODAC*, pp. 304-307, 1990.

[8] Elof Frank, Thomas Lengauer "APPlaUSE: Area and Performance Optimization in a Unified Placement and Synthesis Environment," *Proc. ICCAD*, pp. 662-667, 1995.

[9] Y.M. Fang and D.F. Wong, "Simultaneous Functional-Unit Binding and Floorplanning," *Proc. ICCAD*, pp. 317-312, 1994.

[10] M. Xu and F.J. Kurdahi "ChipEst-FPGA: A Tool for Chip Level Area and Timing Estimation of Lookup Table Based FPGAs for High Level Applications," *Proc. ASP-DAC*, Jan. 1997.

[11] M. Xu and F.J. Kurdahi "Area and Timing Estimation for Lookup Table Based FPGAs," *Proc. of ED&TC*, March 1996.

[12] C. Ramachandran, F. J. Kurdahi, "Incorporating the Controller effects during Register-Transfer Level Synthesis," *Proc. EDAC* 1994

[13] D. Gajski, N. Dutt, A. Wu, and S. Lin, "High-Level Synthesis: Introduction to Chip and System Design," *Kluwer Academic Publishers* 1992

[14] B. Mitra et. al, "Estimating the complexity of synthesized designs from FSM specifications," *IEEE Design and Test*,vol. 10, pp. 36-42, Mar. 1993

[15] M. Xu and F.J. Kurdahi "Layout-driven RTL Binding Techniques for High-Level Synthesis," *Proc. 9th ISSS* 1996