Exploration of Area and Performance Optimized Datapath Design Using Realistic Cost Metrics*

Kyumyung Choi† and Steven P. Levitan
Department of Electrical Engineering
University of Pittsburgh
Pittsburgh, PA 15261
Phone:(412) 624-0558 E-mail: kycst@ee.pitt.edu

ABSTRACT

We present a novel technique for datapath allocation, which incorporates interconnection area and delay estimates based on dynamic floorplanning. In this approach, datapath area is minimized by minimizing the number of wires, routing tracks, and multiplexers, while performance is optimized by minimizing wire length. The simultaneous optimization of these physical cost metrics allows the system to explore realistic design solutions.

I. Introduction

The goal of high-level synthesis is to build an optimum register-transfer level structure from a given behavioral specification of a digital system by exploring chip area and performance tradeoffs. Chip area is based on the size of the control unit and the area of the datapath. Performance is defined as the total execution time of the synthesized design.

The datapath is composed of functional units, memories and wires or buses which interconnect these components. While the lower bound on the functional unit area and the memory area is determined by scheduling, interconnection area can be minimized during datapath allocation. Existing datapath allocation algorithms reduce area by minimizing the number of multiplexers and multiplexer inputs as well as either the number of wires for random topologies, or the number of buses and tri-state buffers for linear topologies. However, these optimizations can only be done effectively if accurate area cost metrics, including routing area, are used during high-level synthesis. These factors are not generally considered during the allocation phases of high level synthesis systems.

Further, the interdependencies among the sub-problems in datapath allocation that affect interconnection area minimization must be considered. These sub-problems are: functional unit binding, memory unit binding and interconnect binding. Existing datapath allocation algorithms do not fully account for these interdependencies.

Performance, or execution time, is determined by both the clock cycle length and the number of control steps. Whereas the number of control steps is fixed in scheduling, the clock cycle length can be minimized in datapath allocation. The clock cycle length consists of the execution delay of datapath units and the data transfer delay. The data transfer delay consists of the loading delays of the data source and the data carrier, as well as the interconnection delay. Execution delay is fixed by the cell characteristics in the cell library. However, the minimization of loading and interconnection delays can be performed during datapath allocation. Few researchers have considered the minimization of these delays during datapath allocation.

In order to solve these problems, more accurate area and performance estimation metrics must be used during high-level synthesis. This is only possible when physical layout effects are considered in high-level synthesis. Some researchers have discussed the effects of incorporating physical layout during high-level synthesis.[1] Others have proposed accurate modeling techniques for layout area and delay estimation in high-level synthesis.[2] Only a few researchers have proposed methods for how cost metrics for physical layout can be incorporated in high-level synthesis efficiently.

3D scheduling[3] considers scheduling, functional unit binding and floorplanning simultaneously, and incorporates interconnection delay during high-level synthesis. However, it considers only functional unit binding and floorplanning, and it does not consider other elements, such as registers, multiplexers and wiring elements which have a large effect not only on datapath area but also on interconnection delay.

GB[4] uses a grid-based connectivity binding approach, and considers the minimization of interconnection lengths with the assumption of a bit-sliced stack architecture. GB calculates arc lengths between functional units during binding in order to minimize the distance between functional units. Then, registers are inserted by several ad hoc methods based on a case by case analysis. However, arc lengths should really be calculated between functional units and registers. Therefore, an excessive number of registers may be allocated, this increases not only actual layout area but also the longest arc length and delays.

In this paper, we propose a novel approach to the problem of datapath allocation which incorporates more accurate estimate of interconnection area and delay. Because our algorithm considers the minimization of the number of routing tracks and the longest wire length along with conventional datapath area metrics, our algorithm can explore diverse area-optimized and performance-optimized datapath structures while maintaining optimal values of conventional datapath area metrics. Further, our algorithm performs allocation and binding sub-problems for all control steps simultaneously, avoiding the sequential nature of other techniques which are sources of sub-optimum solutions.

II. Area and Performance Estimation

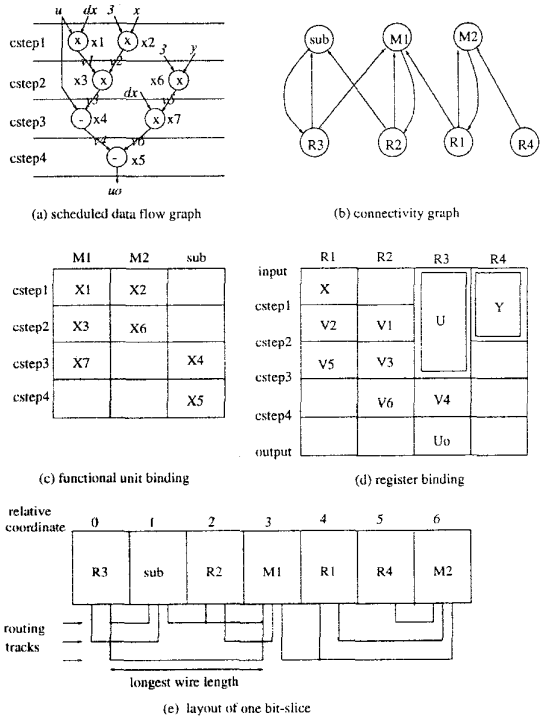
We use a target architecture with a bit-sliced stack and random topology interconnection. In bit-sliced stack architectures, a bit slice is constructed by laying out each bit slice unit using standard cells vertically and connecting different units within the same bit-slice using a vertical routing channel.

To minimize the area of a bit-slice design, we must minimize both the standard cell area and the area of the routing

*This work was supported, in part, by the National Science Foundation under Grant MIP-9102721

†The author was supported, in part, by a Samsung Electronics Fellowship

0-7803-2570-2/95 $4.00 ©1995 IEEE



(a) scheduled data flow graph (b) connectivity graph

(c) functional unit binding (d) register binding

(e) layout of one bit-slice

Figure 1: Binding model

channel. Conventional datapath area metrics, such as numbers of functional units, registers and multiplexers, can be used for minimizing the area of the standard cells. But, in order to minimize routing channel area, the width of the routing channel must also be minimized. This minimization can be achieved by minimizing the number of routing tracks required to implement the nets between the datapath components.

To reduce data transfer delay, both loading delay and interconnection delay can be minimized during datapath allocation. Because the loading delay is fixed by the fanout on each net with a random topology, we can only minimize interconnection delay by minimizing wire lengths during datapath allocation. In bit-sliced stack architectures, the minimization of the longest vertical span in any connection can be used as a cost metric for interconnection delay.

Figure 1 explains how our algorithm handles these area and performance optimization metrics. Figure 1(a) shows a data flow graph after scheduling. Figure 1(c) and (d) show the two-dimensional binding models for functional unit and register allocation.[5] Because we treat *3* and *dx* as constants, we exclude them from entries in the binding model for register allocation.

In our binding model, we include the relative coordinates of each functional unit and register. These values are calculated by the floorplanning phases during datapath allocation. Therefore, gains on the number of routing tracks and the longest wire length as well as the number of wires and multiplexer inputs can be calculated during each move or exchange operation during binding. Figure 1(e) shows the routing tracks and the longest wire length calculation based on this relative coordinate scheme, for the functional unit and register bindings of Figure 1(c) and (d). If cell sizes are available in a library, they can be used to more accurately calculate these relative coordinates and wire lengths.

An important point to note is that when there is a change in the mapping of operations to functional units or the mapping of variables to registers, the change is not always reflected in the numbers of routing tracks, wire lengths, the number of wires and multiplexer inputs. The change of these cost metrics only occurs when there is a change in the interconnection wiring. This interconnection binding is tied to functional unit and register binding by a many-to-one mapping. This fact is more easily understood by introducing the notion of connectivity graph:

$G = \{V, E\}$
V = { functional units and registers }
$E = \{V_i \rightarrow V_j \mid$ *if one or more data transfer exist from* V_i *to* V_j $\}$

Figure 1(b) shows the connectivity graph corresponding to the functional unit and register binding shown in Figures 1(c) and (d) for the the data flow graph of Figure 1(a). Only a change in the edges in this derived connectivity graph would produce a change in the above mentioned cost metrics. This is the reason why we minimize the number of wires in the datapath, even though the number of wires does not directly influence traditional measures of datapath area.

III. Algorithm for Datapath Allocation

The input to our datapath allocator is a description of the scheduled data flow graph. Our scheduler (implemented using FDS algorithm[6]) generates the scheduled data flow graph from a VHDL behavioral description. The output of our datapath allocator is an RTL VHDL structural description and the relative placement of functional units and registers. The algorithm (Figure 2) consists of two phases, initial allocation and allocation improvement.

```
Datapath_Allocation( )
begin
1     S := i := Initial_allocation;
2     FP(best) := Floorplanning;
3     while(stopping criteria are not satisfied) do
4          while(not at equilibrium) do
5               select_improve_mode;
6               j := generate(i);
7               c := cost(j) - cost(i);
8               y := min( 1, exp(-c/T) );
9               r := random(0,1);
10              if( r < y ) then i := j;
11         end
12         FP(i) = Floorplanning;
13         if( FP(i) is better than FP(best) )
                then FP(best) := FP(i);
14         if( cost(i) < cost(S) ) then S := i;
15         update_temperature;
16    end
17    return(S, FP(best));
end
```

Figure 2: Algorithm for datapath allocation

During the initial allocation phase, functional unit binding is performed by a *Clique partitioning algorithm*[7] and register binding is performed by a *Left edge algorithm*[8]. These algorithms each produce individual optimal solutions. Therefore, they are reasonable starting points for the later optimization phases. Next, floorplanning is performed to determine the one dimensional relative coordinates of each functional unit and register. Minimization of the number of maximum cuts is used as an object function for this one dimensional floorplanning. The *mincut algorithm*[9] with terminal propagation is recursively used for this purpose.

During the allocation improvement phase, our system op-

timizes the number of wires and multiplexer inputs, as well as the number of routing tracks and the longest wire length simultaneously. Our system explores various binding alternatives based on a *simulated annealing* scheme[10], and seeks an optimum solution by probabilistically exploring possible datapath structures. The details of how we perform allocation improvement are explained below.

A. Cost function on allocation improvement

As mentioned above, datapath area is minimized by reducing the total area for multiplexers and the routing channel area during allocation improvement. The number of multiplexer inputs is minimized for minimizing the multiplexer area. The number of routing tracks is minimized for minimizing routing channel area. Also, in order to improve performance by minimizing the longest wire length, the longest vertical span for any connection is minimized. Therefore the cost function during allocation improvement is:

$$\text{cost} = P_m \times \#\text{multiplexer inputs} + P_w \times \#\text{wires} + P_{rtrack} \times \#\text{routing tracks} + P_{lmax} \times \text{longest wire length}$$

All the coefficients, $P_m, P_w, P_{rtrack}, P_{lmax}$ are weighting values which indicate the relative importance of each term. Here, we define the number of wires and the number of multiplexer inputs as *primary cost metrics*, and the number of routing tracks and the longest wire length as *secondary cost metrics*. Generally, we set the weights for the primary cost metrics greater than the weights of the secondary cost metrics. The rationale for this decision is twofold. First, the minimization of the number of wires must be weighted more than the minimization of the secondary cost metrics, because the value of the secondary cost metrics will be changed only when the value of the number of wires is changed. Second, the number of multiplexer inputs is a more concrete cost metric than the secondary cost metrics during high-level synthesis, since it is assumed that detailed routing could affect the values of the secondary cost metrics.

B. Improvement scheme

During the allocation improvement phase, a new binding state can be generated by any of following operations:

- move or exchange a functional unit binding
- move or exchange a register binding
- swap the input binding of a functional unit in a symmetric operation

These trials for allocation improvement are generated randomly, and controlled by a *simulated annealing* scheme as shown in Figure 2. The *select_improve_mode* chooses an allocation improvement mode randomly among the operations which are described above. In this way, our system does not sequentially optimize any of the subtasks during datapath allocation but performs the optimizations concurrently.

One feature of our datapath allocation algorithm is that a floorplanning phase is tightly integrated into datapath allocation. Floorplanning is invoked each time after completion of the inner loop of allocation improvement. If a new floorplan produces a better cost for the number of routing tracks or the longest wire length, it replaces the current floorplan. Note that only these two measures are affected by the floorplan. According to our experiments, the use of re-floorplanning produces far better results than keeping the floorplan produced after the initial allocation, with only a small increase in computation time.

The other feature of our datapath allocation algorithm is the efficient modification of the standard simulated annealing algorithm. Line 14 in Figure 2 ensures that the best result so far is stored in S. When the inner loop is completed at a given temperature, that result(i) is compared with the best result so far(S). If the new result is better than the best result, the best result is updated. In either case, the result i is used for the next iteration of the loop. We can always obtain a reasonable result using this modification.

C. Further consideration on the minimization of wire length

So far, we have discussed to reducing the longest wire length to optimize performance. This scheme makes the assumption that the execution delays of functional units are almost even at each control step. Multicycling and chaining schemes used any scheduler could achieve this assumption. In order to remove this assumption, we must emphasize the reduction of the wire length of the critical path rather than the reduction of the longest wire length. This can be done easily by the following technique. First, the critical path on each control step is found before datapath allocation by considering the execution delay of functional units and registers. Then, heavy weights are set for operations on the critical path. These weights are considered for calculating the longest wire length term of our cost function, which will balance the reduction between the longest wire length and the wire length of the critical path.

IV. Experimental Results

We have implemented the system, called *MandM*, and tested it using several benchmark examples. We fixed P_w and P_m at 5 in these experiments for the two reasons. First, because they are the weights for the primary cost metrics, they need to be larger than the weights for the secondary cost metrics. Second, we found that *MandM* produced the best results when we used the same weights for P_w and P_m. In these experiments, we increased the weighting values of the *secondary cost metrics*, P_{rtrack} and P_{lmax}, from 0 to 5. Also we set several values larger than 5 for P_{rtrack} and P_{lmax}.

Table 1 and Figure 3(a) and (b) together show the allocation results of the fifth order elliptical wave filter example[6], with the 19 control steps scheduling result used as the input. The *core cost* is the summation of the primary cost metrics.

We can see the following trends from these tables: The number of routing tracks is reduced greatly as P_{rtrack} is increased from zero, and the longest wire length is reduced when P_{lmax} is increased. The longest wire length is generally reduced when P_{rtrack} is increased, and the number of routing tracks is generally reduced when P_{lmax} is increased, but the amount of this improvement is not large, and the improvement is not uniform. This is because the improvement occurs not due to the weighting value of the respective metric but due to interdependency with the other weighted value. Also, the core cost generally maintains its value as P_{rtrack} and P_{lmax} are increased from 0 to 5. On the other hand, when these values are set larger than 5, the core cost generally increases.

From these examples, we can conclude that if we set the weights for the secondary cost metrics non-zero but smaller than the primary cost metrics, *MandM* produces results which are as good as the best published solutions in terms of conventional datapath metrics, as shown in Table 2. Further, by varying these secondary cost metrics based on user requirements *MandM* can explore diverse area-optimized and performance-optimized datapath structures while maintaining optimal values of conventional datapath allocation metrics.

Table 3 and Figure 3(c) and (d) together show the allocation results of the discrete cosine transform example[5], with the 14 control steps scheduling result used as the input. We can see

P_{rtrack}	Mux	Mux input	Wire	P_{lmax}	Mux	Mux input	Wire
0	6	22	32	0	6	22	32
1	5	18	30	1	7	22	31
2	5	18	30	2	6	19	32
3	6	20	30	3	6	20	33
4	7	21	31	4	6	21	32
5	6	18	30	5	7	24	32

(a) using non-pipelined multiplier

P_{rtrack}	Mux	Mux input	Wire	P_{lmax}	Mux	Mux input	Wire
0	5	18	29	0	5	18	29
1	5	20	30	1	5	18	29
2	5	20	30	2	7	21	29
3	6	19	29	3	5	20	30
4	6	20	29	4	5	22	32
5	5	19	30	5	7	24	32

(b) using pipelined multiplier

Table 1: Elliptical wave filter example

System	FU *	FU +	Reg	Mux	Mux input	Mux 2-1	Wire
HAL	2	2	12	n/a	29	n/a	n/a
EMUCS	2	2	12	n/a	34	n/a	n/a
ELF	2	2	11	10	30	20	n/a
SE	2	2	10	11	31	20	n/a
SALSA	2	2	11	n/a	n/a	16	n/a
STAR	2	2	10	6	27	21	43
MandM	2	2	11	6	22	16	32
HAL	1P	2	12	n/a	26	20	n/a
ELF	1P	2	11	11	30	19	n/a
SE	1P	2	11	9	25	16	n/a
SALSA	1P	2	11	n/a	n/a	16	n/a
MandM	1P	2	11	5	18	13	29

Table 2: Allocation results for elliptical wave filter example

the same improvement trends as in the elliptical wave filter example with even better improvements. We infer that the use of the number of routing tracks and the longest wire length in datapath allocation has an increased impact as the size of synthesized circuit grows.

The run times of these examples ranged from 1-2 minutes for the differential equation example, and about 10 minutes for the elliptical wave filter example, to 15-20 minutes for the discrete cosine transform example, using a Sun Sparc2.

P_{rtrack}	Mux	Mux input	Wire	P_{lmax}	Mux	Mux input	Wire
0	19	50	59	0	19	50	59
1	17	48	59	1	17	47	58
2	18	48	58	2	17	48	59
3	17	48	59	3	17	48	59
4	18	49	59	4	18	48	58
5	17	50	61	5	17	47	58

(a) using non-pipelined multiplier

P_{rtrack}	Mux	Mux input	Wire	P_{lmax}	Mux	Mux input	Wire
0	14	45	57	0	14	45	57
1	15	46	57	1	15	47	58
2	15	44	55	2	15	47	58
3	15	44	55	3	17	47	56
4	14	45	57	4	15	46	57
5	15	45	56	5	15	47	58

(b) using pipelined multiplier

Table 3: Discrete cosine transform example

V. Conclusion

In this paper we presented a novel approach for datapath allocation which incorporates more accurate interconnection area and delay estimates than previous results. To minimize chip area, we considered the number of routing tracks as well as the number of functional units, registers and multiplexers. To optimize execution time, we reduced the longest wire length used in any control step. In order to estimate the number of routing tracks and the longest wire length accurately, floorplanning phases were tightly integrated into the datapath allocation algorithm. Experimental results show our algorithm can explore diverse area-optimized and performance-optimized datapaths, to meet user requirements, while maintaining optimal values of conventional datapath area metrics.

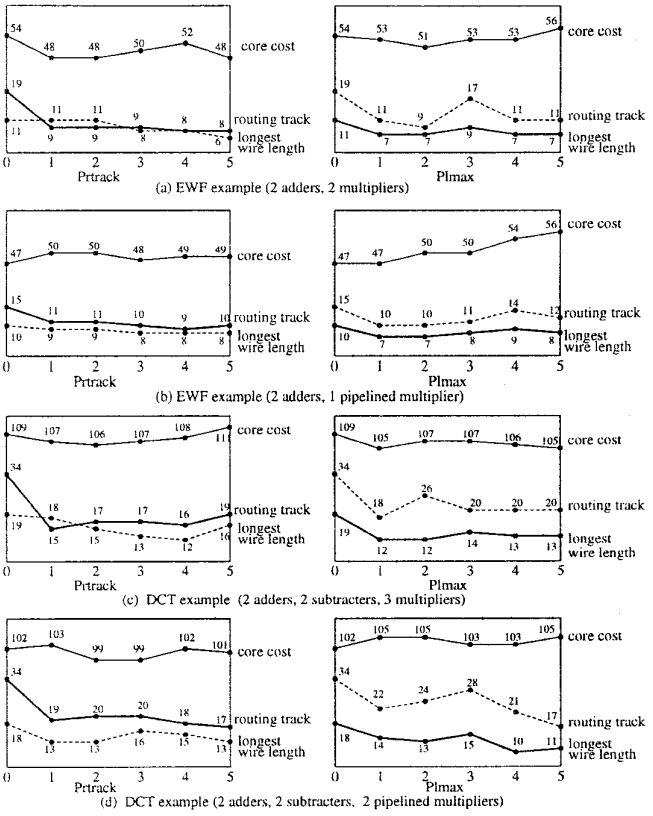


Figure 3: Allocation result

References

[1] M. C. McFarland and T. J. Kowalski, "Incorporating Bottom-Up Design into Hardware Synthesis," *IEEE Trans. on Computer-Aided Design*, vol.9, no.9, pp.938-950, Sep. 1990.

[2] C. Ramachandran, F. J. Kurdahi, D. D. Gajski, A. C. Wu and V. Chaiyakul, "Accurate Layout Area and Delay Modeling for System Level Design," *ICCAD-92*, pp.355-361, 1992.

[3] J. P. Weng and Alice C. Parker, "3D Scheduling: High-Level Synthesis with Floorplanning," *Proc. 28th DAC*, pp.668-673, 1991.

[4] H. Jang and Barry M. Pangrle, "A Grid-Based Approach for Connectivity Binding with Geometric Costs," *ICCAD-93*, pp.94-99, 1993.

[5] G. Krishnamoorthy and J. A. Nestor, "Data Path Allocation using an Extended Binding Model," *Proc. 29th DAC*, pp.279-284, 1992.

[6] P. G. Paulin, J. P. Knight, "Force-Directed Scheduling for the Behavioral Synthesis of ASIC's," *IEEE Trans. on Computer-Aided Design*, vol.8, no.6, pp.661-679, Jun. 1989.

[7] C. Tseng and D. P. Siewiorek, "Automated Synthesis of Data Paths in Digital Systems," *IEEE Trans. on Computer-Aided Design*, vol.5, no.3, pp.379-395, Jul. 1986.

[8] F. J. Kurdahi and A. C. Parker, "REAL: A Program for REgister ALlocation," *Proc. 24th DAC*, pp.210-215, 1987.

[9] B. W. Kernighan and S. Lin, "An Efficient Heuristic Procedure for Partitioning Graphs," *Bell System Tech. Journal*, vol.49, no.2, pp.291-307, Feb. 1970.

[10] S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, vol.220, no.4598, pp.671-680, May 1983.