3D Scheduling: High-Level Synthesis with Floorplanning

BY

Jen-Pin Weng and Alice C. Parker

Technical Report CEng 90-27

Electrical Engineering Systems

University of Southern California

Los Angeles, CA. 90089-0781

3D Scheduling: High-Level Synthesis with Floorplanning[0]

Jen-Pin Weng and Alice C. Parker
Department of Electrical Engineering-Systems
University of Southern California
University Park
Los Angeles CA 90089-0781
Nov. 5, 1990

Category: High-Level Synthesis – 4.3

Contact person
Jen-Pin Weng
SAL 329
Department of EE-Systems
Univ. of Southern California
Los Angeles CA 90089-0781
E-mail : weng@eve.usc.edu
Phone : (213)-740-4474
Fax: (213) 740-4449

[0]This work was supported by the Semiconductor Research Corporation under contract 89-DJ-075. All appropriate organizational approvals for the publication of this paper have been obtained. If accepted, the author(s) will prepare the final manuscript in time for inclusion in the conference proceedings and will present the paper at the conference.

3D Scheduling: High-Level Synthesis with Floorplanning[0]

Abstract

Submicron feature sizes result in designs in which wiring delay is comparable to functional delay. This paper presents a new approach to the problem of scheduling while simultaneously considering floorplanning. Operators are assigned (and placed) as close as possible to their predecessors in order to minimize the interconnection cost. We also propose an algorithm to reduce interconnection cost by introducing redundant operators. This procedure produces a quite satisfactory result for a practical size example, especially on critical-path dominated cases.

Keywords: Scheduling, Floorplanning, High-Level Synthesis, Wiring Delay

Category: 4.3

[0]This work was supported by the Semiconductor Research Corporation under contract 89-DJ-075. All appropriate organizational approvals for the publication of this paper have been obtained. If accepted, the author(s) will prepare the final manuscript in time for inclusion in the conference proceedings and will present the paper at the conference.

1 Introduction

Submicron integrated circuits have extremely small, fast gates. The gap between functional unit delays and interconnection wiring delays is narrowing, partly due to smaller feature sizes, and partly due to larger designs being integrated onto a chip, requiring proportionally longer wires. Module assignment can affect the performance of the subsequent physical implementation greatly due to long interconnections between operators. "Optimal" register-transfer level schedules can actually be quite suboptimal when wiring delays dominate processing delays. However, interconnection delay can only be determined accurately after floorplanning is completed. Obviously, high-level synthesis tools using submicron technology will not be able to make intelligent scheduling decisions without considering interconnection delay.

This paper describes a new method for scheduling, allocation and module assignment called 3D scheduling[1] which incorporates interconnection delays during the scheduling process. This method has been prototyped and experiments performed. We believe our work is the only program besides BUD [8] which uses floorplanning to take into account wiring delay during high-level synthesis.

The core of data path synthesis [1] is divided into 3 subtasks: scheduling, module allocation and module assignment (binding). Scheduling assigns operations to appropriate time steps. Figure 1 shows an example of a dataflow graph used as input for high-level synthesis. Every node in the graph represents an operation. For example, *add1* represents an addition, and *mul1* means a multiplication. The scheduling of this dataflow graph into 2 time steps is shown with the cross-hatched line in the figure. We must allocate a sufficiently large number of modules (or operators) for each time step, and modules may be shared among

[1] 3D scheduling refers to the problem of simultaneously scheduling time and the X-Y plane.

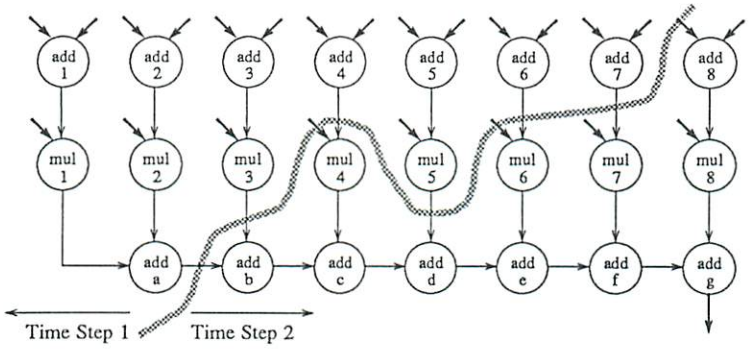

Figure 1: A 2-time-step Non-pipelined FIR Filter Design

operations in different time steps. This task is called module allocation. The phase called module assignment binds the operations to specific modules (or operators). Most current data path synthesis programs [2] [3] [4] only deal with scheduling and module allocation as a first step, but some also perform module assignment concurrently [8].

In the prototype, we simplified a number of attributes of a high-level synthesis system in order to demonstrate the concept. We do not consider the cost and delay of registers, multiplexers, wiring space or control overhead at this time. And, we assume the design is non-pipelined. Such limitations can be relaxed later in a practical implementation.

The basic idea behind this work is the following: as operations are scheduled and functional modules are allocated, we decide their shape and position on the floorplan concurrently. The approach taken is to schedule the operations along the critical path first and assign (and place, if the assigned operator is newly allocated) operators as close as possible to their predecessor(s) according to their data dependencies. Then, the off-critical path operations are scheduled and assigned to operators (and placed, if a new operator is allocated) according to their data relationship. In our example, operations such as: *add1*, *add2* , *mul1*

.... *addg* on the critical path will be scheduled, assigned and placed before the others.

The technique at all times tries to minimize the interconnection length along the critical path(s). Once all the operations on the critical path have been scheduled, the routine checks the time constraints and tries pairwise interchanges to reduce interconnection delays. If it can't achieve the time constraints then it tries to introduce *redundant operators*[2] to alleviate the interconnection delay. For example, the floorplan of the two-time-step design of Figure 1 is shown in Figure 2. In this example, the feasible minimum-allocation design contains only 8 adders and 4 multipliers. However, the software reduced the interconnection delay along the critical path by introducing one more adder, which is shown shaded in Figure 3.

Since we consider the floorplan during scheduling, we can estimate the delay time of designs more accurately than traditional approaches, which may accept a design during high-level synthesis and then find that constraints are not satisfied at a later implementation stage. Our experiments show interconnection delay contributes an additional 20% of functional unit delay to overall delay with a 1.6 micron fabrication process[3]. The overall delay time could be reduced to 5% above functional delay by introducing redundant operators. One of our examples, shown in Figure 4, indicates a design using sequential module assignment followed by a quadratic-based floorplanning technique. This approach results in increased delay time of 10% as compared to our design shown in Figure 3. For finer feature sizes, interconnection cost/delay and redundant operators play more important roles, due to more complex designs fitting on a single chip and lower area overhead when introducing redundant

[2]Redundant operators are operators not required for the minimum feasible design.

[3]The same overhead was found in actual layouts of the AR filter described in a companion paper also submitted to DAC.

operators. Our predictions show the delay time will be increased dramatically by the wiring delays in critical-path dominated systems when submicron fabrication processes are used.

The proposed scheduling technique, module assignment technique and wiring delay model are described in Section 3. The detailed algorithm is given in Section 4. Section 5 gives the experimental results of the FIR filter with 2 time steps. The conclusions are given in the last section.

2 Related Research

Very little synthesis research has taken into account physical design effects. BUD is a unique program which floorplans prior to synthesis [8]. Fasolt [7] floorplans and analyzes area impact during high-level design. ELF is an early system which estimates interconnection effects during synthesis [10]. Chippe is a constraint driven expert system which allows users to specify area, time and power constraints for CMOS gate array design [11]. Chippe predicts wire delay from the structural RT-level design.

Most current approaches consider the scheduling and floorplanning problems separately, which optimizes designs locally. Furthermore, many current floorplan packages [19] [20] [18] minimize the total interconnection cost of a net list as their objective. They pay no special attention to reducing the interconnection length between the operators along the critical path.

Timing Driven Layout Design incorporates timing information to influence the placement and wiring processes [13] [14] [15] [16]. This approach uses the path analysis data produced by a static timing analysis program to generate weights for critical nets of clocks and data paths. These weights are then used to bias automatic placement (and/or routing)

in the layout program. However, this step is currently performed after scheduling. Iteration must be performed to take advantage of timing-driven layout [17].

3 Delay Estimation

Wiring delay can be estimated by the following first order relationship [5]:

$$t_{0.9} = 1.02RC + 2.21(C_tR_t + C_tR + CR_t) \tag{1}$$

where:

R = Total resistance of the wiring.

C = Total capacitance of the wiring.

C_t = Load capacitance.

R_t = Equivalent resistance of the driving transistor.

We calculate R, C, R_t and C_t, as follows:

$$R = r.l \tag{2}$$

$$C = c.l \tag{3}$$

$$R_t = r_t.\frac{L_d}{W_d} \tag{4}$$

$$C_t = \sum_{i=1}^{k} c_{t_i}.W_i.L_i, \qquad k = fanout \tag{5}$$

where:

r = Resistance of wiring per unit length.

c = Capacitance of wiring per unit length.

l = Length of wiring.

r_t = Sheet resistance of driving transistor when it is on.

W_d =Width of driving transistor.

L_d = Length of driving transistor.

c_{t_i} = Gate capacitance per unit area of load transistor i.

W_i =Width of load transistor i.

L_i =Length of load transistor i.

These parameters depend strongly on the design and fabrication process. In the case of metal wiring the wiring resistance, R, is very low as compared to R_t and can be neglected for our purpose. Therefore, Equation (1) reduces to:

$$t_{0.9} = 2.21 R_t (C_t + C) \tag{6}$$

Using Equation(1) (or Equation(6)), we can calculate the worst-case wiring delay. The wire length is measured by the diagonal rectilinear distance of the bounding box containing the two connected operators on the floorplan, and the process parameters are used to estimate interconnection delay. Although this is only a first-order equation, it does produce a very good approximation of wiring delay as verified by SPICE simulation [5].

4 Algorithm Description

Before describing the algorithms used for floorplanning, scheduling and allocation, we give the problem definition. The inputs to the 3D scheduling task are a dataflow graph, library module set and cost (or speed) constraints. The outputs are operation-to-timestep assignments, operation-to-module assignments and a floorplan. The goal is to maximize the performance (or to minimize the cost) while satisfying the cost (or speed) constraints.

The proposed scheduling approach will be described in the following two subsections: (1) scheduling and module assignment and (2) the wiring delay improvement procedure. The first part of the algorithm is a constructive procedure; the second part is an iterative procedure.

4.1 Scheduling and Module Assignment

Our approach to synthesis is to develop the schedule and resource requirements simultaneously. We start from a null module set and add module units only when operations cannot share existing ones. The ordering in the scheduling algorithm is to schedule operations on the critical path first. For the operations along the critical path, we schedule the operations whose predecessors have been scheduled, according to their data dependency along critical path. The scheduling approach is similar to that used by Nagle[12] and used in MAHA[2]. The rest of the unscheduled operations are scheduled by their *freedom* in ascending order. *Freedom* is defined as the difference between the latest possible scheduled time step($ALAP$) and the earliest possible scheduled time step($ASAP$) for a given clock cycle. The operation with larger area will be scheduled earlier, when two or more operations have the same *free-*

dom. For example, we approximate the clock cycle as a multiplier delay plus 5 times the adder delay in the 2-time-step design. The *freedom* of operation *adda* in Figure 1 is 1, since it could be scheduled no later than time step 2, and no earlier than time step 1. However, the *freedom* of operation *addd* is 0, because it could be assigned to time step 2 only.

Once each operation has been scheduled, we need to assign it to a specific module unit. During module assignment, we use the best-first approach. We initially assign the operation to the module available at that time step which performs the same function with minimum wiring cost. These assignments will be iterated later by the improvement procedure introduced in the next subsection. The scheduling algorithm is outlined as follows:

1. **comment:** *Process the operations along the critical path first*
2. **for** all operations along the critical path whose predecessors have been processed **do**
3. **begin**
4. **comment:** *Search all possible time slots on allocated module list*
5. **for** all allowable time steps **do**
6. **begin**
7. *use the best-first approach to find an available module which can perform the operation with minimum wiring cost*
8. **if** found **then continue** to next operation
9. **comment:** *Otherwise, a new module needs to be allocated*
10. **else** *allocate a new module of this operator type*
11. *free all the schedulings of this operator type and reschedule them*
12. **end**
13. **end**
14. **comment:** *Now, assign the positions for new allocated modules*
15. *generate floorplan of allocated modules using the constructive method*

16. **comment:** *Process the rest of unscheduled operations*

17. *repeat the procedure from step 4 to step 11 for the off-critical path unscheduled nodes, except use freedom as the priority function; floorplanning unplaced modules as they are allocated*

4.2 Improvement Procedure

The improvement procedure contains two phases, operation rebinding and redundant operator allocation. The purpose of both phases is to minimize the total wiring delay. Operation rebinding sequentially examines the potential module exchange with respect to reduced total wiring cost. If total wiring length is reduced by this exchange, then we switch the bindings of these two operators. Otherwise, the bindings are retained unchanged.

After searching all the possible reassignments, we try to allocate redundant operators to reduce the total wiring delay under the *generosity* [6] constraint. The *generosity* indicates the maximum tolerance of introducing redundant operators. For example, 10% of *generosity* means the area of redundant operators is constrained to no more than 10% of the total area of the minimum feasible allocation design. Not all redundant operators cause extra cost. The redundant allocation of our example, which is shown by a shaded adder cell on Figure 3, doesn't increase the boundaries of the floorplan.

The outline of the improvement procedure is as follows:

1. **comment:** *Process the operations along the critical path first*
2. **for** all operations along the critical path whose predecessors have been processed **do**
3. **begin**
4. **comment:** *Search all possible exchanges*
5. **for** all modules with the same operator type **do**
6. **begin**

7. if this module is assigned to an operation on critical path
then *skip it* and continue

8. if the total wiring cost is reduced after exchange
then switch the module assignments of these two operations

9. end

10. comment: *Now, allocate a redundant operator to improve the wiring cost*

11. if allocated redundant operators *greater* than generosity allowed
then goto step 1

12. *allocate a redundant operator and place it as close as possible to its predecessor to reduce the total wiring delays*

13. if the total wiring cost is reduced
then put the allocated redundant operator into allocated module list
else free the allocated redundant operator

14. end

15. comment: *Processing the non-critical path operations*

16. for all operations ordered by freedom do

17. begin

18. *repeat the procedure from step 4 to 13, except skip the possible exchanges with operations along critical path*

19. end

The above procedure iteratively improves the wiring cost under the *generosity* constraint. From our experience, one iteration usually produces a satisfactory result.

5 Experimental Results

Since there has been no reported work on scheduling with floorplanning in a program, we can't provide any comparative data on the performance of our software. However, we applied our program to an FIR filter, as shown in Figure 1.

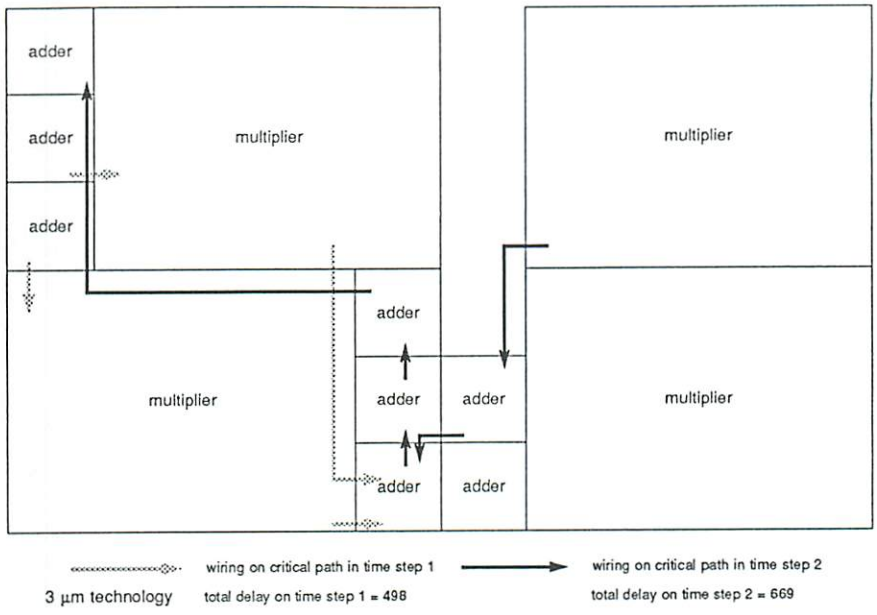

Figure 2: A 2-time-step Non-pipelined FIR Filter Floorplan with Minimum Operators

Technology	16 bit adder		16 bit multiplier	
	Delay (ns)	Area (mil^2)	Delay (ns)	Area (mil^2)
3 μm	34	4200	375	49000
2 μm	22	1867	250	21778
1.6 μm	18	1195	200	13938
1.2 μm	13	672	150	7840

Table 1: First Library Set Used for FIR Filter

The 3D-scheduler generated a 2-time-step design, minimizing the number of operators as shown in Figure 2. Another design with a redundant adder was created and is shown in Figure 3. Those two designs take 0.2 second of CPU time on a SUN 4/460.

Table 1 lists the first library set we used for this example. For comparison purposes, we first linearly scaled down each functional unit area and delay from 3-micron process parameters. We used different device parameters for different fabrication processes to calculate wiring delay in this program. We later show an actual scaled library used to test our concepts. Table 2 lists the predicted delay time for both designs for the linearly scaled

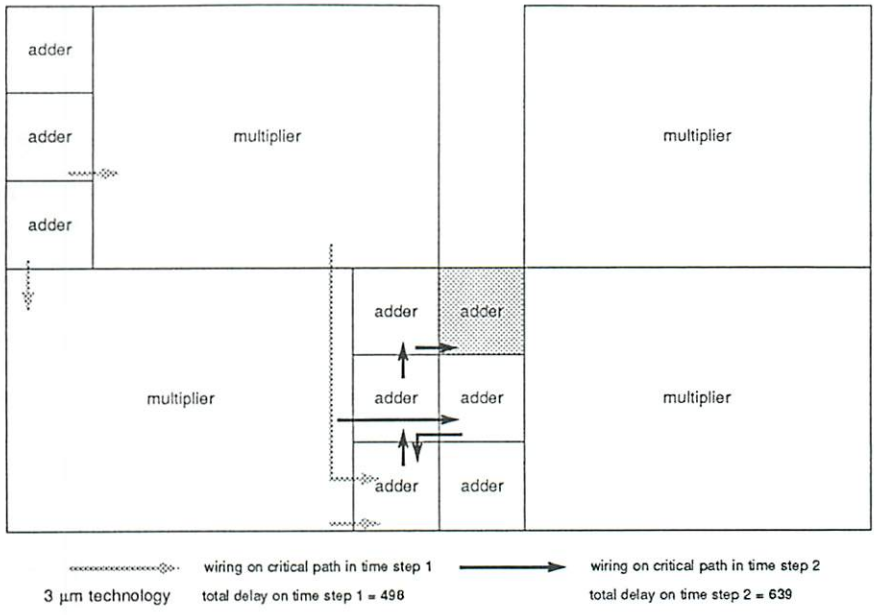

Figure 3: A 2-time-step Non-pipelined FIR Filter Floorplan with a Redundant Adder

Technology	Functional Delay	with Wiring	with Redundant Op.
3 μm	1090 ns	1338 ns	1278 ns (*4.5% less*)
2 μm	720 ns	946 ns	890 ns (*5.9% less*)
1.6 μm	580 ns	760 ns	716 ns (*5.8% less*)
1.2 μm	430 ns	570 ns	536 ns (*6.0% less*)

Table 2: Minimum Operator Design versus Redundant Operator Design

fabrication technology. The errors caused by considering functional delay only are shown clearly in this table. The effect of introducing redundant operators is also obvious, especially with submicron processes, since the area overhead becomes less and less significant. The 1.2 μm design only takes about 16% of the area of the 3 μm design.

To compare the differences between current floorplan packages and our program, we used a quadratic-based floorplan program [9] to generate a 2-time-step FIR filter floorplan using sequential ordering for module assignment. The design is shown in Figure 4. Total wiring cost was optimized during floorplanning. However, since floorplanning was performed

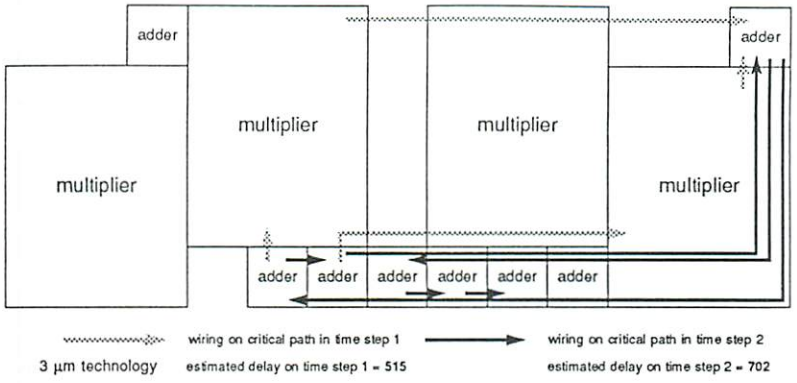

Figure 4: Floorplan Created by a Quadrisection-based Floorplan Program

Technology	Functional Delay		Wiring Delay[5]		Total Delay[6]
	Step 1	Step 2	Step 1	Step 2	
3 μm	443 ns	545 ns	84 ns	184 ns	1458 ns
2 μm	294 ns	360 ns	64 ns	139 ns	998 ns
1.6 μm	236 ns	290 ns	52 ns	114 ns	808 ns
1.2 μm	176 ns	215 ns	41 ns	88 ns	606 ns

Table 3: Design Generated from Quadrisection-Based Floorplan Program

without taking into account the wiring along the critical path, the critical path wiring runs back and forth in this case[4]. The quadrisection results for different fabrication technologies are shown in Table 3. The total delay increases more than 10% over the 3D technique. The data on different fabrication processes has been redrawn in Figure 5 for comparison.

By inspecting Figure 5, we found the differences between functional delay and overall system delay are quite linear. However, this is not true of real process parameters. We believe this phenomenon is due to linearly scaling the area and delay of functional units. For this reason, we used the Seattle Silicon Compiler to create an 8-bit adder and an 8-bit

[4]In all fairness, pairwise interchange might improve the performance of this design. However, we left the floorplan "as is" to illustrate the impact of totally ignoring timing.

[5]Estimated wiring delay on critical path only.

[6]Total delay equals longest step delay (step 2) multiplied by total steps (2).

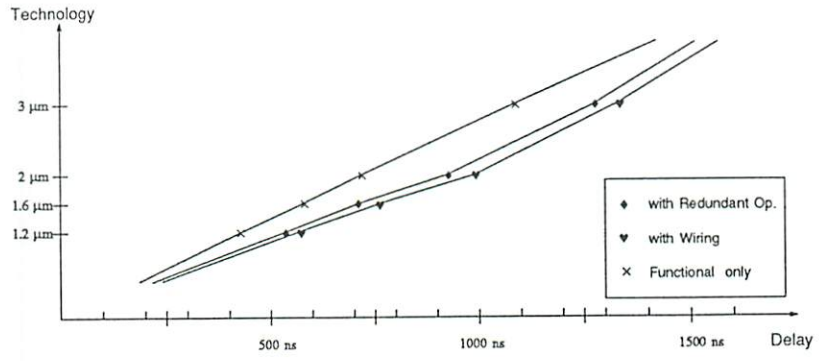

Figure 5: Total Delay versus Fabrication Process

Technology	8 bit adder		8 bit multiplier	
	Delay (ns)	Area (mil^2)	Delay (ns)	Area (mil^2)
1.6 μm	23	112	58	1386
1.2 μm	13	75	32	903
1.0 μm	12	34	30	419

Table 4: Library Set Created by Seattle Silicon Compiler

multiplier for 1, 1.2 and 1.6 micron fabrication technologies. We also resynthesized the data of the FIR 2-time-step example. The library data and resynthesized results are shown in Tables 4 and 5 and in Figure 6. It is obvious the functional area and delay do not scale down linearly. However, the errors caused by considering only functional delay still remain the same (about 20% to 30%). The results of the improvement procedure are not as dramatic as previously. We believe the change of area ratio between the adder and multiplier leads to this result. It should be noted that even though wiring delays decreased for the same design, as future size decrease, smaller feature sizes allow large designs to fit into a single chip. For these large designs, wires will be longer and wiring delays will have a significant effect on performance.

Technology	Functional Delay	with Wiring	with Redundant Op.
1.6 μm	346 ns	414 ns	404 ns
1.2 μm	194 ns	252 ns	242 ns
1.0 μm	180 ns	222 ns	214 ns

Table 5: FIR filter 2-time-step design using Seattle Silicon Compiler Library Set

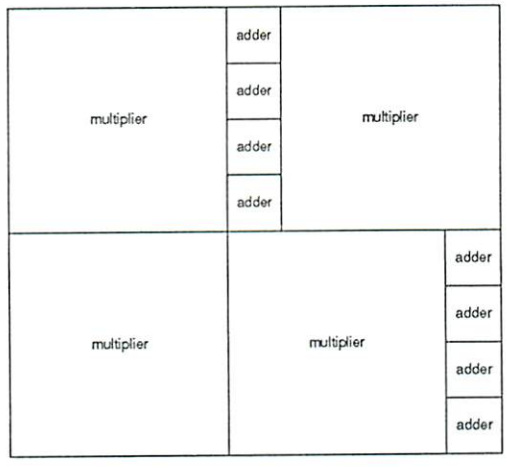

Figure 6: Floorplan Created by using Seattle Silicon Compiler Library Set

6 Conclusion

In this paper, we have presented a new approach which considers scheduling and floorplanning simultaneously during data path synthesis. The main objective of our approach is to minimize the interconnection area and maximize the sharing among allocated operators. A more precise estimate of the delay time of a design can be achieved during the scheduling process using our approach. We have demonstrated the concept of the interconnection delay for the critical path operations during the scheduling process, and showed that the delay may be significantly reduced by introducing redundant operators.

References

[1] M. C. McFarland, A. C. Parker and R. Camposano, "Tutorial on High-Level Synthesis", *Proc. of the 25th Design Automation Conference*, pp. 330-336, Jul. 1988.

[2] A. C. Parker, J. Pizarro and M. J. Mlinar, "MAHA: A Program for Datapath Synthesis", *Proc. of the 23th Design Automation Conference*, pp. 461-466, Jul. 1986.

[3] P. G. Paulin and J. P. Knight, "Algorithms for High-Level Synthesis", *IEEE Design & Test of Computers*, Dec. 1989.

[4] N. Park and A. C. Parker, "Sehwa: A Software Package for Synthesis of Pipelines from Behavioral Specifications", *IEEE Trans. on Computer-Aided Design*. Vol. 7, No. 3, Mar. 1988.

[5] T. Sakurai, "Approximation of Wiring Delay in MOSFET LSI", *IEEE Journal of Solid-State Circuits*, Vol. SC-18, No. 4, pp. 418-426, Aug. 1983.

[6] A. C. Parker, et al. "Experience with the ADAM Synthesis System", *Proc. of the 26th Design Automation Conference*, Jul. 1989.

[7] David W. Knapp, "Feedback-Driven Datapath Optimization in Fasolt", *Proc. of the 27th Design Automation Conference*, Jul. 1990.

[8] Michael C. McFarland, "Using Bottom-Up Design Techniques in the Synthesis of Digital Hardware from Abstract Behavioral Descriptions", *Proc. of the 23th Design Automation Conference*, Jul. 1986.

[9] P. R. Suaris and G. Kedem, "A Quadrisection-Based Combined Place and Route Scheme for Standard Cells", *IEEE Trans. on Computer-Aided Design*, Vol. 8, No. 3, Mar. 1989.

[10] E. Girczyc, "Automatic Generation of Microseqenced Data Paths to Realize ADA Circuit Descriptions", *PhD thesis, Carleton University*, Jul. 1984.

[11] F. Brewer and D. Gajski, "Chippe: A System for Constraint Driven Behavioral Synthesis", *IEEE Trans. on Computer-Aided Design*, vol. 9, no. 7, pp. 681-695, Jul. 1990.

[12] A. Nagle, R. Cloutier and A. Parker, "Synthesis of Hardware for the Control of Digital Systems", *IEEE Trans. on Computer-Aided Design*, Vol. CAD-1, No. 4, pp. 201-212, 1982.

[13] W. Donath et. al, "Timing Driven Placement Using Complete Path Delays", *Proc. of 27th Design Automation Conference*, pp. 84-89, Jun. 1990.

[14] A. Dunlop et. al, "Chip Layout Optimization Using Critical Path Weighting", *Proc. of 21th Design Automation Conference*, pp. 133-136, Jun. 1984.

[15] M. Burstein and M. Youssef, "Timing Influenced Layout Design", *Proc. of 22th Design Automation Conference*, pp. 124-130, Jun. 1985.

[16] R. Nair et. al, "Generation of Performance Constraints for Layout", *IEEE Trans. on Computer-Aided Design*, vol. 8, no. 8, pp. 860-874, Aug. 1989.

[17] F. Kurdahi and S. Sastry. An Optimal Algorithm for Floorplan Area Optimization. *Proc. of 27th Design Automation Conference*, Jun. 1990.

[18] J. Bhasker and S. Sahni. Characterization of Wire Length Distributions in Standard Cell Layouts. *Proc. of The European Design Automation Conference*, Feb. 1990.

[19] T. Wang and D. Wong. An Optimal Algorithm for Floorplan Area Optimization. *Proc. of 27th Design Automation Conference*, Jun. 1990.

[20] Y. Lai and S. Leinwand. Algorithms for Floorplan Design via Rectangluar Dualization. *IEEE Trans. on Computer-Aided Design*, Dec. 1988.