CRL 88-27

Area-Time Model for Synthesis of Non-Pipelined Designs [1]

Rajiv Jain, Mitchell J. Mlinar and Alice Parker
Department of Electrical Engineering - Systems
University of Southern California
Los Angeles, CA 90089-0781

Abstract

The ability to predict area-time characteristics of designs without actually synthesizing them is vital to producing quality designs in a reasonable time. In this paper, we give a mathematical model for predicting the area-time tradeoff curve for non-pipelined datapaths given a data flow graph and a module set. The model has been validated against designs generated by a program which synthesizes non-pipelined datapaths.

1 Introduction

Many programs which synthesize designs from behavioral descriptions without human intervention have been developed. These programs perform scheduling, resource allocation, module selection, module binding and interconnect allocation to generate an RTL (Register-Transfer Level) design. The designer has to iterate the synthesis process several times and search for a satisfactory design. Comparisons of synthesis results using different module sets, module allocations, and schedules are made to locate the desired design. If a fast prediction tool could estimate the cost-delay parameters of the final design space without actually synthesizing the designs in this space, the number of trial passes and the design time could be significantly reduced. In this paper we present such a tool.

The input to a synthesis system consists of a description of required behavior, library of components, and a set of design goals. An RTL datapath output by such a system consists of modules, and registers interconnected via multiplexers, busses and wiring. A behavioral description can be implemented using a number of candidate RTL designs, some of which are inferior to the others [2]. Major factors which impact the chip area and delay of the RTL datapaths include amount of parallelism, amount of resource sharing, and choice of clock cycle. Collecting the area-delay measures of the non-inferior design points gives the area-delay tradeoff curve for the input description.

The prediction tool we have constructed examines operator cost and delay to predict the *lower bound* non-inferior area-time curve. Other effects such as register, multiplexer, bus and wiring area and delay are omitted at present. A future model will consider all these measures to improve the accuracy.

The input to our prediction tool is a data flow graph (an example is shown in Figure 1) which specifies the required behavior of the hardware to be synthesized, and a set of module styles which will be used to implement the operations. Our estimation tool is able to predict an area-time curve in the design space which forms a lower bound for all possible designs; that is, all the design points lie either on or above the curve; design points which lie on the curve represent optimal designs. The lower bound area-time tradeoff curve has the shape $AT = c \times k$ where k is a constant and c is the clock cycle.

Theoretical results obtained from this estimation tool have been validated by designs generated by MAHA, a non-pipelined synthesis program [5]. MAHA concurrently performs scheduling and module allocation and outputs a scheduled data flow graph, the number of required operators, and the clock cycle. By altering the design constraints MAHA can synthesize a number of designs for the same input description. An example area-time tradeoff curve showing non-inferior design points produced by MAHA for Figure 1 is given in Figure 5.

The predictive model computes the theoretical lower bound on area-time; therefore it provides an independent performance measure for comparing different synthesis systems. In the past, most synthesis systems were compared by example. However, there is no absolute standard and exhaustive search to find optimal designs is impractical. By developing this basic predictive model, synthesis systems can actually quantify the quality of the designs they produce against a known optimum.

The following assumptions have been made in developing the lower-bound model:

- Only functional area of the design is considered. The model does not include register, multiplexer, bus and wiring area/delay.
- An operation must execute within one clock cycle. An operation cannot be scheduled into two or more time steps. Partitioning an operation can be done by *a priori* division into two or more sub-operations. For example, any operation *foo* which can be executed over two clock cycles can be decomposed into *foo1* and *foo2*. The predictive technique described here helps detect situations when such a partitioning is desirable.
- Merging of several different module types into one composite type (for example ALU) is not considered.
- A module set contains exactly one module type which can perform an operation type. Further, the module set is complete in that every operation can be executed by one module type in the module set. For example, consider the design library given in Table 1. A data flow graph with addition and multiplications operations would require a and m type modules in the module set.

Although many synthesis systems exist, none of the published systems uses an area-time estimation model as a front end to prune the inferior design space.

Early research on area-time estimation was published by Davio et. al in 1983 [1]. One major assumption made is that all nodes of the input data flow graph are identical,

[1]This research was supported by the Semiconductor Research Corporation (Contract 86-01075), the Departments of Air Force, Army and Navy (Contract N00039-87-C-0194), and the Department of Advanced Research and Project Agency (Contract N00014-87-K-0861).

[2]*A design implementation is inferior when there exists at least another implementation which performs better in one or more figures of merit, all other figures of merit being equal* [3].

Function	Module Name	Area mil^2	Delay nS
addition	a	4200	340
subtraction	s	4200	340
multiplication	m	49000	375

Table 1: Example Design Library

i.e. they have same delay, and area, and perform the same function. This assumption simplifies the problem as the clock cycle time is readily determined. In this paper, we solve a more general problem where a data flow graph may consist of many operation types.

Kurdahi has derived upper bound register and 2-to-1 multiplexer cost [3]. He proves that the upper bound register count is given by the maximum cut of the data flow graph which can be computed by any *maximal-flow, minimal-cut* algorithm.

2 Area-Time Estimation Model

2.1 Clock Cycle Estimation

Prior to estimating the AT curve itself, a lower bound on the value of clock cycle which is used for the lower-bound estimates must be derived. Since this is a lower bound estimate, latching delays are not included in clock cycle computations. This value is dependent on the module delays (d_i), the delay along the critical path C, and the number of time steps t the data flow graph is partitioned into.

Theorem 1 *The clock cycle which produces the best possible designs from a given data flow graph and a module set can be found.*

$$c = maximum(\frac{C}{t}, maximum(d_i)) \tag{1}$$

where C is the critical path delay, the data flow graph is divided into t time steps and d_i is the delay for each module type i.

Proof: When $t = 1$, then the clock cycle will be the critical path delay, C. If the circuit is partitioned into two time steps, then the best case is when the clock cycle is equal to exactly half the critical delay. Similarly for $t = 3, 4, \cdots$ the best possible value of clock cycle will be $C/3, C/4, \cdots$ respectively. However, as every operation must complete within one clock cycle, the minimum clock cycle is bounded by the delay of the slowest module. This completes the proof. ■

Consider Figure 1 for example. If the delays of the adder and multiplier operators are 250 ns and 500 ns respectively, then the critical path delay $C = 2750$ ns, and $c = maximum(2750/t, maximum(250, 500)) = maximum(2750/t, 500)$. If this data flow graph is partitioned into five or fewer time steps, then $c = 2750/t$. For $t \geq 6$, the clock cycle is equal to $maximum(250, 500) = 500$ ns. The break point for the clock cycle occurs when the number of time steps $t = 2750/500 = 5.5$.

2.2 Lower Bound Area-Time Tradeoff Curve

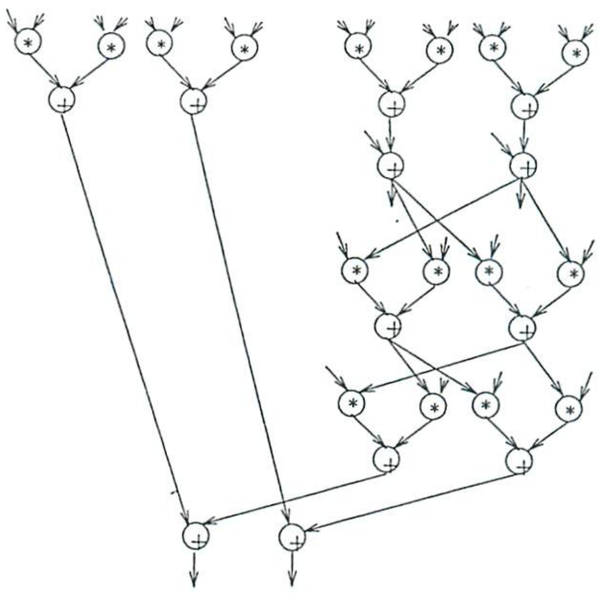

Figure 1: AR-Lattice Filter Data Flow Graph

With a lower bound on the clock cycle time computable given t, the optimal number of modules required for each time step value t is readily determined. These results are combined to form the area-time tradeoff curve for non-pipelined designs. Before explaining how to compute this curve, we provide the following definition. The underlying concept of module-optimality has been taken from [2].

Definition 1 *The utilization of each module type $0 \leq i \leq m-1$ is defined as*

$$u_i = \frac{n_i}{t \times o_i} \tag{2}$$

where n_i is the number of nodes of type i in the data flow graph and o_i is the number of modules of type i used in the design. A utilization of one is module-optimal.

Definition 2 *For a module-optimal design $u_i = 1$ for all module types, $0 \leq i \leq m-1$.*

Of course, in practical designs, it is often not possible to utilize all the modules in every cycle, resulting in suboptimal designs. The following theorem uses the above definition to provide a basis for our estimation technique.

Theorem 2 *Given a data flow graph and a module set, the lower-bound curve of non-pipelined designs is given by*

$$AT = c \times k \tag{3}$$

where total functional area of the design is $A = \sum_{i=0}^{m-1}(a_i \times o_i)$, $T = t \times c$ is the time to execute a set of input data, c is the clock cycle for the design, and k is a constant.

Proof: From Equation 2 module-optimal designs satisfy

$$t = \frac{n_i}{o_i}$$

Multiplying both sides by $a_i o_i$ (a_i is the area for each module of type i) and summing over all m operation types yields

$$t \sum_{i=0}^{m-1} (a_i \times o_i) = \sum_{i=0}^{m-1} (a_i \times n_i)$$

Multiplying both sides by the clock cycle time c and substituting for the definition of area and time results in

$$AT = c \sum_{i=0}^{m-1} (a_i \times n_i)$$

Noting that for a given data flow graph and module set, a_i and n_i are constants, we get Equation 3. ■

Combining Equations 1 and 3, we have

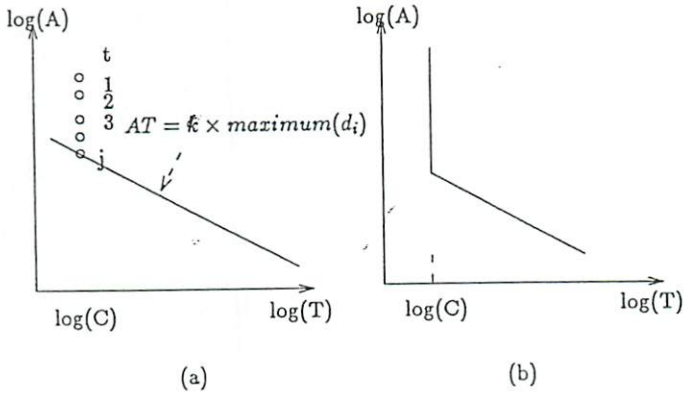

Figure 2: Area-Time Estimation Curve

(i) for $maximum(d_i) < \frac{C}{t}$,

$$AT = \frac{kC}{t} \tag{4}$$

(ii) for $maximum(d_i) \geq \frac{C}{t}$,

$$AT = k \times maximum(d_i) \tag{5}$$

Consider Equation 4 and its graphical representation given by the circles in Figure 2a. (The graph is drawn on a *log - log* scale for better readability.) Each circle corresponds to some value of t. For example, the top-most circle corresponds to $t = 1$, for which $AT = kC$. Equation 5 is shown by the solid sloping line. Joining the circles with the firm line for $T > C$ gives the final lower-bound area-time curve for the data flow graph and is redrawn in Figure 2b. Designs with clock cycle equal to C/t have identical lower-bound delay T and their area A decreases as t increases.

2.3 Non-Optimality of Practical Designs

Practical designs are often non-optimal and these design points have an AT product inferior to the lower-bound design points. There are several reasons for non-optimality, some of which are illustrated as follows. Consider Figure 3a. The delay along the critical path is $C = 3d_{adder}$. If the data flow graph is partitioned into two time steps, then the lower-bound clock cycle is $c = \frac{3}{2}d_{adder}$, which is not possible based on our assumptions. The actual clock cycle time is the ceiling function $\lceil 3/2d_{adder} \rceil = 2d_{adder}$ which degrades the performance from that predicted using the lower bound clock cycle. For any given design, this

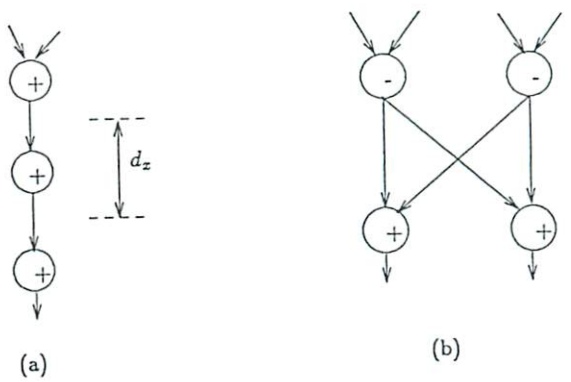

Figure 3: Non-Optimal Design Examples

loss in performance for non-optimal designs is bounded by $\Delta c = \frac{1}{2}maximum(d_i)$.

In addition, a non-optimal design results when all the modules cannot be used every clock cycle. Continuing with Figure 3a, a design with two partitions will require two adders, of which one adder will lie idle for one clock cycle leading to a non-optimal design.

There is a third cause for non-optimality in designs: feasibility of scheduling. An optimal design with two partitions for Figure 3b would require one subtractor and one adder and the clock cycle would be half the delay of critical path. However, to meet the clock cycle requirements, the only possible partition is to put both the subtraction nodes in first partition and both the addition nodes in the second time step. Clearly, this solution is infeasible as the first partition has two subtraction operations to be performed on one subtractor and two addition operations in the second

```
procedure estimate_lower_bound(t)
begin
        $c_1 = maximum(d_i)$;
        $c_2 = \lceil C/p \rceil$;
        $c = maximum(c_1, c_2)$;
        $T = t \times c$;
        $o_i = \lceil n_i/t \rceil,\ 0 \leq i \leq m-1$;
        $A = \sum_{i=0}^{m-1}(a_i \times o_i)$;
        print $A, T$;
end
```

Figure 4: Procedure for Lower-Bound Area-Time Curve

partition. Thus, four time steps will be needed to make the schedule feasible, reducing the performance of the design. This type of non-optimality is not observed in pipelined designs where modules can be kept busy more easily by the execution of different input data sets at the same time.

The actual procedure which generates the lower bound area-time curve given in Figure 4 takes into account the module non-optimality of designs by computing the ceiling function $\lceil n_i/t \rceil$ rather than n_i/t. The procedure is executed for different time steps, $1 \leq t \leq \sum_{i=0}^{m-1} n_i$.

3 Experiments and Results

Three data flow graphs were synthesized using MAHA [5] and the module set given in Table 1. The estimation proce-

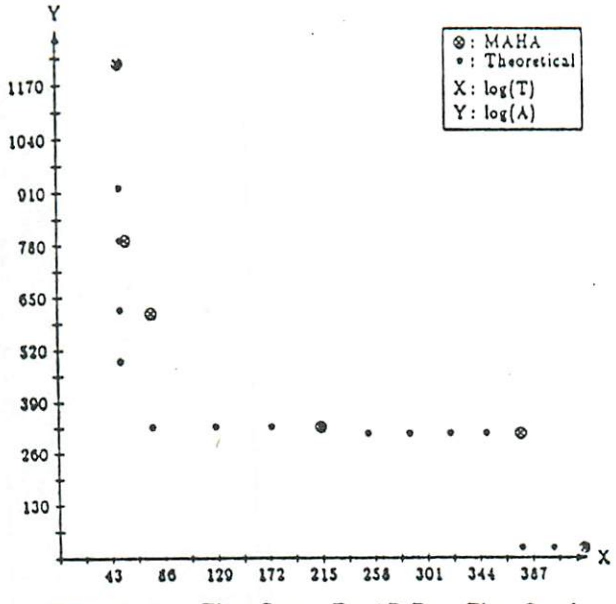

Figure 5: Area-Time Curves For AR Data Flow Graph

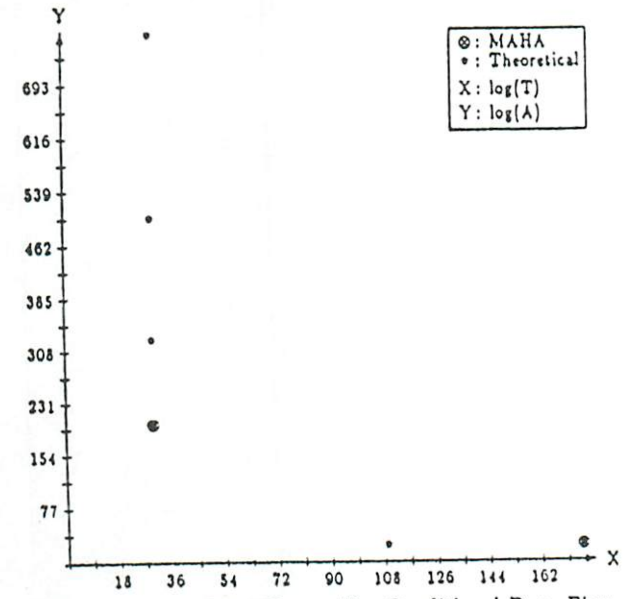

Figure 6: Area-Time Curves For Conditional Data Flow Graph

dure was then executed for the example data flow graphs. The results of MAHA and the estimation procedure are given in Figures 5 through 7.

Some results produced by BUD are given in [4]. The shape of the curve produced by BUD with module cost-delay alone (Figure 3c of [4]) follows the shape of the curves produced by our estimation program. The values of the design points cannot be compared as the input description and the module sets used by BUD were different. BUD also performs certain transformations which our estimation tool does not handle such as module merging and assigning operations over several time steps. Despite these transformations, BUD still produces curves with the same shape as our estimation routine even though the cost-delay values are likely to be different. This strengthens our confidence in the capability of the estimation tool to be used with other similar synthesis packages.

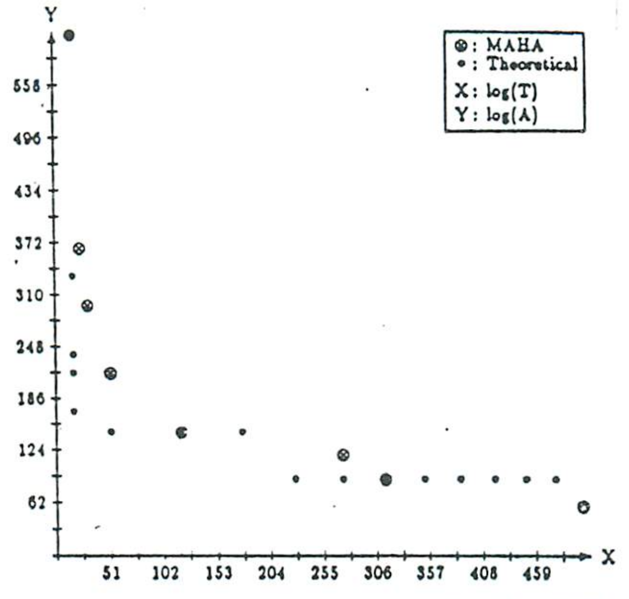

Figure 7: Area-Time Curves For Random Data Flow Graph

4 Conclusion

In this paper, we have given a mathematical model for predicting non-pipelined design points in the design space. This prediction routine has been coded in C and is running on a SUN workstation. It produces results for practical design examples in less than a CPU second.

The current tool has several limitations. It is restricted in that only functional area of the final design point is considered. The cost and delay effects of registers, multiplexers, busses, and wiring are not presently considered, but are under investigation. Furthermore, the final estimation tool should be capable of estimating non-optimal designs while relaxing the assumptions given in this paper. Further research is underway to validate the model against other synthesis packages.

References

[1] M. Davio, J. -P. Deschamps, and A. Thayse. *Digital Systems with Algorithm Implementation.* John Wiley & Sons, New York, 1983.

[2] E. Girczyc. *Automatic Generation of Microsequenced Data Paths to Realize ADA Circuit Descriptions.* PhD thesis, Department of Electronics, Carleton University, July 1984.

[3] F. J. Kurdahi. *Area Estimation of VLSI Circuits.* PhD thesis, Department of Electrical Engineering, University of Southern California, August 1987.

[4] M. C. McFarland. Reevaluating the Design Space for Register-Transfer Hardware Synthesis. In *Proceedings IEEE International Conference on Computer Aided Design*, ACM/IEEE, November 1987.

[5] A. C. Parker, J. Pizarro, and M. J. Mlinar. MAHA: A Program for Datapath Synthesis. In *Proceedings 23rd Design Automation Conference*, ACM/IEEE, June 1986.