BAD: Behaviorial Area-Delay Predictor

Kayhan Kucukcakar and
Alice C. Parker

CEng Technical Report 90-31

Electrical Engineering Systems

University of Southern California

Los Angeles, CA. 90089-2562

November 13, 1990

BAD: Behavioral Area-Delay Predictor

Kayhan Küçükçakar and Alice C. Parker
Electrical Engineering – Systems
University of Southern California
Los Angeles, CA 90089-0781

November 13, 1990

This work was supported by the Defense Advanced Research Projects Agency and monitored by the Federal Bureau of Investigation under contract No. JFBI90092. The views and conclusions considered in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of Defense Advanced Research Projects Agency or the U.S. Government.

BAD: Behavioral Area-Delay Predictor

Abstract

This paper presents a comprehensive, integrated area-delay prediction methodology and associated software package which can be used to support behavioral synthesis tools. The prediction model includes functional, register, multiplexing, PLA and wiring area and delay predictions. The predictions are based solely on the behavior of the target design and on its area-performance constraints. BAD can also be used to search the predicted design space for designs which meet the constraints. These methods together with a unified statistical representation of prediction results can be used effectively for a variety of purposes including guiding system-level decisions, guiding a behavioral synthesis planner, quickly searching the design space,guiding the partitioning process and evaluating transformations.

1 Introduction

Designs with submicron feature sizes have wiring delays which are comparable to functional delays. Even though automated synthesis can speed up the design process by orders of magnitude, the growing complexity of designs together with the multitude of tradeoffs possible during synthesis make it impossible to completely search the design space, even with synthesis tools which span behavior to layout. In order to improve the quality of generated designs, the designer has to iterate several times by modifying the functional specification (high-level transformations), design constraints and design decisions. This iterative process can be quite time consuming. Fast prediction tools to estimate the impact of tentative design decisions while taking into account physical design effects can be extremely beneficial in reducing the iteration time.

1.1 Overview of BAD

This paper presents a comprehensive, integrated area-delay prediction methodology which can be used to support behavioral synthesis tools. Each of these methods represents an improvement over previous prediction methods in terms of breadth, performance or accuracy. The prediction model includes functional, register, multiplexing, wiring and PLA area and delay predictions. The predictions are based solely on the behavior of the target design and on its area-performance constraints. These methods together with a unified statistical representation of prediction results can be used effectively for a variety of purposes including guiding system-level decisions, guiding a behavioral synthesis planner, quickly searching the design space and evaluating transformations. BAD can also be used to search the predicted design space for designs which meet the constraints.

The input for BAD, the prediction tool presented here, consists of the proposed value of the clock cycle, a directed acyclic data flow graph, a module library (preferably with more than one module style to implement every operation type) and constraints for total area, initiation interval[1] and total circuit delay[2]. The prediction results are in the form of a set of area-delay pairs for non-pipelined designs and area-delay-initiation interval triples for pipelined designs. Each pair or triple forms a point (predicted design) in the predicted design space. The prediction results are in the form of total area (including functional, register, multiplexer, PLA and wiring area), initiation interval (for pipelined designs) and circuit delay (including register, multiplexer, PLA and wire delay in the clock cycle).

BAD generates predicted designs for all possible (and meaningful) implementations of the design. It simply enumerates configurations of all module sets (each module set contains one module per operation type). For each module set configuration, pipelined and nonpipelined predictions are generated for each possible value of initiation interval and number of stages (in terms of clock cycles). The selection of the best feasible predicted design (satisfying design constraints) from the pool of predicted designs also solves the problem of design style selection and module selection.

[1]initiation interval is the time (or the number of stages) between two successive data inputs to the design
[2]propagation delay from inputs to outputs

BAD is also capable of deleting predicted designs which are inferior or cannot meet specified designer requirements. In this way, predicted designs which would eventually be too large or too slow can be eliminated automatically. The tool handles multi-cycle operations [3] but can also be requested to use only single cycle operations. In the latter case, the module-set enumeration phase automatically screens out selection of library modules which would be too slow for the given clock cycle.

BAD is a fast, practical tool which combines former theoretical and heuristic prediction research done at USC along with new additions, using a unified representation. An entire probability density curve for each design characteristic is predicted from predictions of the lower bound, upper bound, and most likely values of the design characteristic. The distribution is approximated by a normal distribution.

The major assumptions underlying BAD are less restrictive than the previous prediction research and are as follows: All operations of a given type are implemented by a single module type, ALU's are not supported, the resynchronization rate is assumed to be zero, the behavioral description should be free of inner loops (loops with determinate iteration counts should be unrolled), and stochastic delays for operations are not supported.

The overall area-delay predictions are obtained by performing several prediction subtasks (corresponding to actual synthesis tasks) in a specific order. The initial data pool (before performing any predictions) consists of the input data given to BAD (clock cycle, data flow graph, design library, constraints and goals). Each prediction subtask contributes some predicted characteristics of each potential design to the data pool. So, each prediction subtask uses the original input data as well as prediction results existing prior to its execution.

Predictions are performed by following the actual behavioral synthesis guidelines but, without dealing with the actual synthesis details. For example, the core task of pipelined scheduling in behavioral synthesis maps the operations of the data flow graph into time slots and sometimes also onto operators given a data flow graph, a module set, and an initiation interval. At the end of scheduling, operation assignments, the functional resource allocation and the number of stages in the pipeline are easily available. But, on the other hand, the prediction methods only estimates the functional resource allocation and the number of stages in the pipeline without actually dealing with details like the assignment of operations to time slots and/or operators.

In the following sections, the statistical model used to represent prediction results is presented and the prediction methods for operators, registers, multiplexers and wiring are discussed. Then, prediction results will be given with comparisons to synthesized designs. The paper will conclude with the evaluation of the methods and future research.

1.2 Related Research

Although most high-level synthesis programs do not use predictions for the design characteristics yet to be determined, we cite BUD [11], ELF [2] and CHIPPE [1] as exceptions.

[3]This multi-cycle architecture is simple but efficient. It is not, by any means, the most complex form of handling multi-cycle operations.

The prediction results are used to drive the synthesis programs to the desired part of the design space. BUD estimates physical design characteristics like area and wire delays by approximate floorplans. ELF predicts the wiring area from RTL characteristics to make design decisions. CHIPPE predicts the worst case wiring delays after behavioral synthesis (without constructing a floorplan) and uses this delay information in the evaluation mechanism of its iterative synthesis process. Also, the ADAM System [4] uses predictors to guide the design search.

Early work on the problem ([6] and [5]) was on the prediction of lower-bound functional area with varying time characteristics for pipelined and non-pipelined design styles. In this work, only single-cycle operations were considered. The clock cycle was chosen by the prediction method such that all operations could be executed within a clock cycle. Some loose theoretical lower bounds are also given for register and multiplexer area in [6]. Prediction research in [6] and [5] is useful to estimate the resource allocation behavior of Sehwa [14] and MAHA [16], the pipelined and non-pipelined scheduling programs in the ADAM synthesis system.

The existing methods for register [12] and wiring [8] area estimation, although fast enough for human interaction, were inherently too slow to be practical when called hundreds of times by other programs. The run time of the predictions was a pressing factor for our applications. Therefore, some modifications and/or approximations have been used to implement these methods.

2 A Unified Representation of Prediction Results

Using predictions in behavioral synthesis requires special care because of the discrete nature of the problem being solved. In addition, there are many types of prediction tools, each dealing with a different aspect of a design. The fact that it is not always possible to have predictions of same nature complicates the problem even further. Some prediction results are in the form of lower-bound values while others might be expected values or upper-bound values. There is usually more data available than just a single expected value or a bound and it is desirable to make use of as much data as possible.

We use a statistical representation which provides a unified environment for prediction results. The PERT modeling technique [10] has been chosen for the prototype software to approximate predicted design characteristics. In the PERT model every predicted point consists of three values (for each design characteristic) : a lower bound, a most likely value and an upper bound. The average and the variance are calculated from these three values using the PERT technique, and standard statistical methods can then be applied.

Let's assume that area is such a design characteristic. *area.lb*, *area.ml*, *area.ub* are the lower-bound, most likely and upper-bound values of the characteristic, respectively. Based on these three values, *area.avg* and *area.var*, the average value for area and variance of area in the PERT model are estimated (also see Figure 1):

$$area.avg = \frac{area.lb + 4 \times area.ml + area.ub}{6}$$

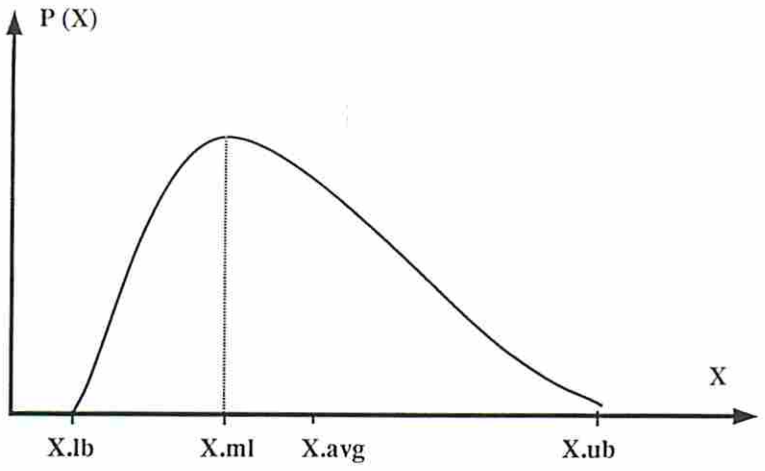

Figure 1: The PERT model used in predictions

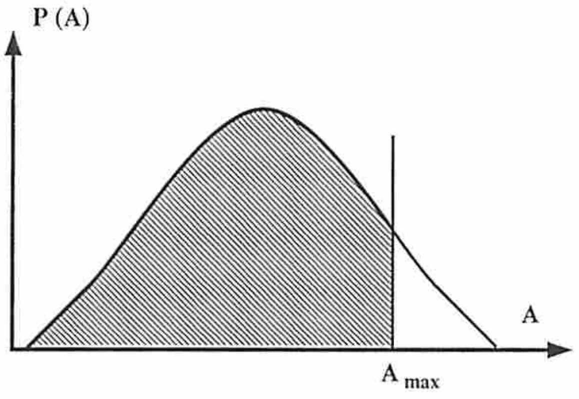

Figure 2: The probability of feasibility as the area under a probability density curve

$$area.var = \frac{(area.ub - area.lb)^2}{36}$$

The following example shows how prediction results are summed to get the total area, A:

$$A.avg = register.area.avg + operator.area.avg + \cdots$$

$$A.var = register.area.var + operator.area.var + \cdots$$

When the total of the prediction results for a particular design characteristic has been computed, infeasible predicted designs are eliminated as follows. The probability distributions for prediction results are approximated by normal distributions. Let A be the total area predicted for the design. Then the probability of satisfying the hard area constraint A_{max} is estimated [17] (see Figure 2):

$$Pr[A \leq A_{max}] = \int_{-\infty}^{\frac{A_{max}-A.avg}{\sqrt{A.var}}} \frac{e^{\frac{-x^2}{2}}}{\sqrt{2\pi}} dx \tag{1}$$

Any predicted design whose probability of satisfying a design constraint is lower than the user-specified minimum probability of feasibility for the same design constraint is eliminated. These user-specified probability figures indicate not only the confidence of the designer in the prediction tools but also the severity of the constraint. If a constraint (e.g the total circuit delay) is not very critical, then it is better to include some of designs which are predicted to violate the given constraint, but are close to the acceptable limits. Another problem is that the error in predictors may not be constant. A predictor which generally gives very accurate results may occasionally show a large deviation from the real results. Therefore, it is important not to eliminate predicted designs which violate some constraints but still stay close to the acceptable limits. The user specified probability figure for each constraint is used as a measure of how strictly that constraint should be enforced. The right hand side of Equation 1 is readily available in a tabular form which makes it suitable for easy, fast calculation [17].

3 The Predictors

A number of separate, distinct predictors have been developed or improved and combined into BAD. Some are based on design theory, some are heuristic, some are empirical, and some are statistical. We briefly summarize the novel aspects of each of these predictors. We begin with operator area prediction, which contributes to total active area, followed by prediction of the number of stages in a design, which, along with the clock cycle, gives total circuit delay. We then briefly describe register and multiplexer area/delay predictions, followed by wiring space/delay predictions.

3.1 Pipelined Operator Estimation

The pipelining theory used by BAD is the same as that used in Sehwa [14]. The resource allocation is highly dependent on the initiation interval set by the schedule. Prediction is performed by varying the initiation interval and calculating the lower bound resource allocation for each initiation interval. The original relation from Jain, et. al. [6] assumes all operations take one clock cycle to execute. The effect of scheduling infeasibilities[4] caused by certain graph topologies is assumed to be reflected in the circuit delay through the pipeline (which was not predicted by Jain), not in the resource allocation.

We have derived a new relation capable of handling multi-cycle operations[5]:

$$o_i = \lceil \frac{n_i^e \times \lceil \frac{d_i}{c} \rceil}{l} \rceil \tag{2}$$

[4]bottlenecks in scheduling

[5]Jain also has a similar model for operator predictions handling multi-cycle operations.

where o_i is the lower bound for the required number of modules of type i, n_i is the number of operations of type i and l is the initiation interval considered. n_i^e is the maximum number of operations of type i which can be executed in parallel. This number is the same as the total number of operations of type i for non-conditional graphs and it is less than or equal to n_i for graphs having conditional blocks [13]. d_i is the delay of module type i and c is the clock cycle. In our pipelining model, it is always possible to achieve the lower bound resource allocation at the expense of longer pipelines; therefore, the lower bound, the most likely and the upper bound values for o_i are the same.

The whole clock cycle at this point is determined by datapath operation delays.

3.2 Non-pipelined Operator Estimation

Non-pipelined operator estimation is performed by varying the number of stages and calculating the lower-bound resource allocation for each case. The original estimation for the number of operators in [5] is for single-cycle operations. The new relation handling multi-cycle operations is

$$o_i = \lceil \frac{n_i^e \times \lceil \frac{d_i}{c} \rceil}{N} \rceil \tag{3}$$

where N is the number of stages.

In a non-pipelined design, it is not always possible to achieve lower-bound resource allocation for any given number of stages, N. The module types which are highly utilized are more likely to be bottlenecks in the schedule. A heuristic approach is used for such cases: When the utilization for a module type i is predicted to be above 80%, the upper bound for o_i is increased by one module, while the most likely value is the lower-bound value.

Here, the whole clock cycle is considered to be composed of datapath operation delays.

3.3 Estimation of the number of stages

As stated earlier, the operator area predictions are performed by varying the initiation interval for pipelined designs and the number of stages for non-pipelined designs. In order to predict the entire area-delay tradeoff curve it is necessary to know the realistic range for the number of stages and the circuit delay through the pipeline as well as the initiation interval. The methods presented in [6] and [5] did not include these predictions.

3.3.1 Non-pipelined Designs

The number of stages for a non-pipelined design is determined by the critical path length of the data flow graph, delays of individual operators and the total amount of processing required.

$$\lceil \frac{C}{c} \rceil \leq N \leq \sum_i n_i \times \lceil \frac{d_i}{c} \rceil$$

where C is the length of the longest path through the data flow graph for a given module library. The upper bound corresponds to the most serial implementation and is a quite loose bound. In the following, a more rational upper-bound estimate is used.

$$N \leq \max(\lceil \sum_i n_i \times \frac{d_i}{c} \rceil, \max_i(n_i \times \lceil \frac{d_i}{c} \rceil) \times 1.25) \tag{4}$$

The second term in Equation 4 models the effect of the operation type with the highest probability of causing scheduling infeasibilities. The number of stages ranges between these lower and upper bounds, which are used to predict the area-delay characteristics of non-pipelined designs.

3.3.2 Pipelined Designs

The initiation interval in our pipelining model is not affected by scheduling infeasibilities. Therefore, the realistic initiation interval range is bounded as follows:

$$1 \leq l \leq l_{max}$$

where the maximum initiation interval considered, l_{max} is given:

$$l_{max} = \max_i(n_i \times \lceil \frac{d_i}{c} \rceil)$$

Designs with initiation interval higher than l_{max} will unnecessarily keep resources idle and are more likely to be inferior. Therefore, no such designs will be considered in the predictions.

The pipeline length is harder to predict because the effect of scheduling infeasibilities has to be taken into account. Currently, the pipeline length is predicted by a simple piecewise linear approximation of empirical observations as described in Figure 3.

According to this approximation model, the expected value of N can be less than $l+1$ or greater than $N.ub$. In such cases, a heuristic is used to make certain adjustments to ensure that results are consistent (e.g. the expected number of stages for pipelined designs should be greater than the initiation interval).

3.4 Register Estimation

The estimation of register area is mainly based on Kurdahi's proposed technique, which have been improved and implemented by Mlinar [12]. In his approach, the maximum cut (maximum number of registers) of the data flow graph is found by using a min-flow max-cut algorithm [9]. This gives the width of the graph. Then, this width is used to estimate the number of registers for each given design point. Although the min-flow algorithm is quite fast for human interaction, it had to be approximated with a faster heuristic because of the performance requirement on BAD.

```
Procedure Estimate Pipeline Length
if all $o_i$ for interval $l$ is same as $o_i$ for $l-1$ then
    Use the estimated number of stages for $l-1$
else
    $N.lb = \lceil \frac{C}{c} \rceil$
    if $l < \frac{l_{max}}{2}$ then
        $N.ub = \sum_i n_i \times \lceil \frac{d_i}{c} \rceil$
    else
        $N.ub = \sum_i n_i \times \frac{d_i}{c}$

    $N.ml = \lceil N.lb + \frac{N.ub - N.lb}{l_{max} - 1} \times (l-1) \rceil$
End.
```

Figure 3: The heuristic for pipeline length estimation

3.4.1 Notation

Some notation concerning graph characteristics which will be used in the remaining sections of the paper is listed below:

ibw_i	The total input bitwidth for type i
obw_i	The total output bitwidth for type i
$icnt_i$	The number of inputs for type i
$icnt_i$	The number of outputs for type i
I	The total number of input bits
I_c	The number of input bits which are constant or don't need storage
O	The total number of output bits
F	The total output bits of operations which only depend on the primary inputs
A	The average bitwidth of operations in the data flow graph
Y	The total output bits of all operations
X	The total input bits of all operations
R	The estimated number of bits of registers
M	The estimated number of bits of 2-to-1 multiplexers
T	The maximum number of stages considered for the design style
N_c	The number of operations on the critical path
$\lceil \frac{N}{l} \rceil$	The number of data sets simultaneously executed in the pipeline

3.4.2 Determination of graph width

R_{max}, the estimated width of the graph (in terms of bits) is estimated by characterizing the shape of the graph based on the global, static characteristics of the graph, without performing extensive graph processing. Figure 4 shows three graph shapes considered in

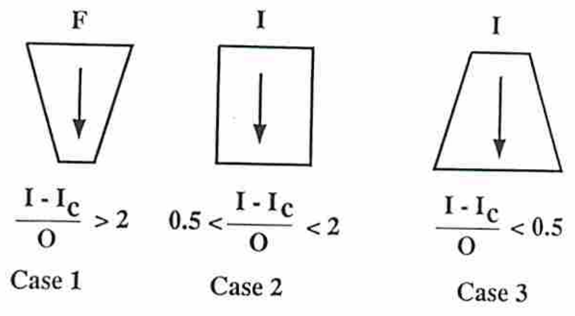

Figure 4: Possible graph shapes considered

the estimation of graph width. R_{max} is a function of the number of inputs to the graph and the average value fanout in the data flow graph, and is estimated by the algorithm shown in Figure 5.

```
Procedure Estimate Graph Width
if $(I - I_c) > 2 \times O$ then                    {The graph narrows at the output}
    $R_{max} = F \times \frac{X}{Y + F}$
else if $O > 2 \times (I - I_c)$ then               {The graph widens at the output}
        $R_{max} = \max(O, (I - I_c) \times \frac{X}{Y + I - I_c})$
    else
        $R_{max} = (I - I_c) \times \frac{X}{Y + (I - I_c)}$
End.
```

Figure 5: Graph width estimation algorithm

3.4.3 Register Area Estimation

After R_{max} is estimated, the heuristic shown in Figure 6 is used to estimate the number of registers. The heuristic is taken from [12] with some modifications. The register area is estimated for each pair of initiation interval and the number of time steps supplied from earlier prediction methods. The estimated register area is mainly a function of estimated graph width, the total bitwidth of graph outputs, the bitwidth of graph inputs, the total bitwidth of constant graph inputs, and functional parallelism.

The register estimate for the non-pipelined case is based on a linear interpolation of register count between the width of the graph (upper bound estimate) and the lower bound. The pipelined register estimate is based on the fact that several instances of the data flow graph will be executed simultaneously requiring storage proportional to the parallelism and furthermore, the primary inputs also need to be stored.

```
Procedure Register Estimation
if $N > l$ then                                              {Pipelined}
    $R.lb = max(I - I_c, O) + \lceil \frac{N}{l} - 1 \rceil \times R_{max}$
    $R.ub = max(I, R_{max}) \times \lceil \frac{N}{l} \rceil$
    if $R.ub > 2 \times R.lb$ then
        $R.ml = \frac{R.ub}{2}$
    else
        $R.ml = R.lb$
else                                                         {Non-pipelined}
    $R.ub = max(O, R_{max})$
    $R.lb = R_{max}$
    if $N = 1$ then
        $R.ml = O$                                           {Only outputs are stored}
    else
        if $N > T$ then
            $N = T$                                          {Saturate $N$}
        $R.ml = \lceil \frac{N-1}{T-1} \rceil \times \max(R_{max} - O, 0) + 1.5 \times O$ {Linear interpolation}
End.
```

Figure 6: Register estimation algorithm

3.4.4 Register Delays

The delay introduced into the clock cycle due to register propagation delay, rd, is constant because of the simple clocking scheme used.

3.5 Interconnect Estimation

Interconnect estimation is based on use of multiplexers only since it is quite hard to characterize and estimate global bus structures. Multiplexer estimates are themselves highly heuristic and the algorithm is shown in Figure 7. When functional operators and registers are shared, there generally exists some multiplexing logic to route the incoming data to the modules. The main idea behind multiplexer estimation is to characterize the multiplexing requirements of each predicted design. This is done by estimating the total number of operator inputs for each module type and the total number of sources incoming to these operator inputs so that the number of multiplexers to route the data from these sources to operator inputs can be estimated. The multiplexer delays introduced to the clock cycle are estimated by using the size of estimated multiplexer trees in front of operators (considering operator chaining) and registers.

The basic assumption behind multiplexer estimation is that each operation type will only be implemented by a single operator type. This assumption allows us to estimate the multiplexing requirement for each operator type based on the resource sharing of the operator type (see calculation of *muxcnt*1 in Figure 7). A second method is also used to estimate operator multiplexing in order to ensure the robustness of the estimates. The second method (see calculation of *muxcnt*2 in Figure 7) is based on the estimation of the number of sources to be routed to the operator inputs. The multiplexer requirements of each operator type (including registers) are summed to get the total interconnect requirements.

```
Procedure Multiplexer Estimation
$muxcnt1 = \sum_i ibw_i \times (n_i - o_i.ml)$                  {Operator multiplexing}
$muxcnt2 = I + R.ml + \sum_i o_i.ml \times (obw_i - ibw_i)$     {Operator multiplexing}
$muxcnt = \frac{muxcnt1 + muxcnt2}{2}$                          {Average of two methods}
$op\_mux = \max_i(n_i - o_i.ml)$                                {The largest operator multiplexer tree}
if $N > l$ then                                                 {Pipelined}
    $x = Y + I$
    $muxcnt = muxcnt + \max(x - R.ml, 0)$                       {Register multiplexing}
    $muxcnt = muxcnt \times 1.40$
else                                                            {Non-pipelined}
    $x = \max(Y, \frac{R.ml}{2} + I + \sum_i o_i.ml \times obw_i)$
    $muxcnt = muxcnt + \max(x - R.ml, 0)$                       {Register multiplexing}
    $muxcnt = muxcnt \times 1.20$
$reg\_mux = \frac{\max(x - R.ml, 0)}{A}$                        {Register Mux Complexity}
$muxdelay = (\lceil \log_2 reg\_mux \rceil + \lceil \frac{N_c}{N} \rceil \times \lceil \log_2 op\_mux \rceil) \times$ 2-to-1 mux delay
if $l = 1$ then
    $muxcnt, muxdelay, op\_mux, reg\_mux = 0$                   {No multiplexing}
$M.lb = 0$                                                      {Empirical Observation}
$M.ml = \frac{muxcnt}{2} \times$ 2-to-1 mux area                {Empirical Observation}

$M.ub = \frac{muxcnt}{1.5} \times$ 2-to-1 mux area              {Empirical Observation}
$md.lb = 0$
$md.ml = \frac{1}{2} \times muxdelay$                           {Multiplexing delay}
$md.ub = muxdelay$
End.
```

Figure 7: Multiplexer estimation algorithm

3.6 Wiring Estimation

3.6.1 Area Estimation

Consideration of only RTL characteristics of a design is not generally enough to make major design decisions. The prediction of layout characteristics is also quite important. In our approach, it is assumed that a standard cell placement and routing package is used although another wiring area estimator can be integrated for whichever design method is being used (provided that there exist some prediction techniques for that method)[6]. PLEST [8] is a program which predicts the wiring area of a design if a standard cell placement and routing approach is used. It has been shown experimentally that the wiring area estimates of PLEST are within 10% of the actual chip areas.

The input to PLEST is the total active area of the design and the number of 2-point nets along with the technology parameters. The estimated active area of the design, *AA* is defined as a summation of estimated operator, register and multiplexer areas.

$$AA = R.ml + M.ml + \sum_i o_i.ml$$

The number of 2-point nets is estimated as

$$nets = (\sum_i o_i.ml \times ibw_i) + R.ml + 2 \times M.ml + O$$

PLEST could have been called repeatedly to estimate the wiring area. But, the run time for PLEST (in the order of tens of seconds on a Sun-3 workstation) was too long to meet our requirements. Therefore, interpolation of previous PLEST results is used. By varying the active area and the number of 2-point nets we ran PLEST several times and we got a two dimensional array of PLEST results ranging from the smallest chip size to the largest size possible and from very few nets to a large number of nets. This is required only once for each technology. A binary search algorithm together with a two-dimensional interpolation method is used to approximate the wiring area, *W.ml*. *W.lb* is estimated as $0.9 \times W.ml$ and *W.ub* is estimated as $1.1 \times W.ml$ as the PLEST results are known to be within 10% of the actual figures. The wiring area approximation, *W.ml* is usually within 1 to 3% of PLEST results. The accuracy could be further improved by increasing the number of sample points at the expense of longer run times. The simple two dimensional interpolation algorithm used in the wiring area estimations is shown in Figure 8.

3.6.2 Delay Estimation

The estimation of wire delays is done in two steps. The first step is to estimate the number of wires in series (due to operation chaining) in a stage. The second step is to estimate the delay for each such wires using an RC model. The RC model taken from [3] gives the expected delay for a given wire. The inputs to this model are the wire length, the

[6] Another technique would be the wiring space estimation methods for the gate array design style proposed by Sastry [18].

```
**Procedure** Estimate Plest($AA$,$nets$)
{aa[K] and net[L] are the arrays of active area and number of 2-point net
figures supplied to PLEST, and r[K,L] is wiring area predictions from PLEST}

Find the smallest row number $p$ such that AA > $aa[p]$ by using binary search
Find the smallest column number $q$ such that nets > $net[q]$ by using binary search

$P = r[p,q] + \frac{AA - aa[p]}{aa[p+1] - aa[p]} \times (r[p+1,q] - r[p,q])$

$Q = r[p,q+1] + \frac{AA - aa[p]}{aa[p+1] - aa[p]} \times (r[p+1,q+1] - r[p,q+1])$

$W.ml = P + \frac{nets - net[q]}{net[q+1] - net[q]} \times (Q - P)$

$W.lb = 0.9 \times W.ml$
$W.ub = 1.1 \times W.ml$
**End.**
```

Figure 8: The interpolation algorithm used to estimate PLEST results

average fanout and the technology parameters. Suppose this model is represented by a function d(length,fanout) for a fixed technology. Then, the minimum delay introduced to the clock cycle due to wire delays, $wd.lb$ happens when the longest path in any stage has the minimum number of wires in series (an operator, a register and no multiplexing) and the wire delays should be the minimum possible, which is calculated as the delay of a zero length wire with no fanout.

$$wd.lb = d(0, 1.0) \times 2$$

The most likely wire delay, $wd.ml$ is the wire delay for an average length wire, avg_wl (estimated from PLEST parameters) with average fanout, $avgf$. There is also some multiplexing assumed. $\frac{N_c}{N}$ is the average operator chaining in any stage and the depth of multiplexers is calculated by assuming that 2-to-1 multiplexers will be used to build the multiplexer trees, resulting in logarithmic growth in multiplexer tree depths.

$$avgf = \frac{(\sum_i o_i.ml \times ibw_i) + R.ml + 2 \times M.ml + O}{(\sum_i o_i.ml \times obw_i) + R.ml + M.ml + I}$$

$$wd.ml = d(avg\_wl, avgf) \times (\lceil \log_2 reg\_mux \rceil + \frac{N_c}{N} \times \lceil \log_2 op\_mux \rceil)$$

where op_mux and reg_mux are the average operator and register sharing, respectively.

Module Name	Type	Bit Width	Area mil^2	Delay ns
add1	Addition	16	4200	34
add2		16	2880	53
add3		16	1200	151
mul1	Multiplication	16	49000	375
mul2		16	9800	2950
mul3		16	7100	7370
register	Register	1	31	5
mux	2:1 Multiplexer	1	18	4

Table 1: Library used in the experiments

The maximum wire delay is calculated for very long wires assuming maximum multiplexing on operators and registers. The length of the longest wire, max_wl is roughly estimated as max($chip.length, chip.width$)[7].

$$wd.ub = d(max\_wl, avgf) \times (1 + \lceil \log_2 reg\_mux \rceil + \lceil \frac{N_c}{N} \rceil \times (1 + \lceil \log_2 op\_mux \rceil))$$

4 Experimental Results

To illustrate how well the prediction methods work, the AR lattice filter from [4] will be used. The synthesized designs were produced by the ADAM synthesis tools [4]. Non-pipelined schedules were created by MAHA [16]. Pipelined schedules were created by using Sehwa's urgency scheduling algorithm which is an internal routine of Sehwa [14]. It would be misleading to accept these results as the best results from Sehwa, which does a much broader design space exploration. RTL designs are generated by MABAL [7]. MABAL was used to generate a single design (not the best possible design) for each schedule. MABAL was asked to generate designs with minimum operator allocation for the schedule, the minimum register allocation and no buses. Since it was quite difficult to layout hundreds of designs to obtain actual wiring areas, MABAL results were fed to PLEST and PLEST results were used to approximate the wiring area of synthesized designs. *Therefore, total areas of synthesized designs as well as predicted designs have estimated wiring areas from PLEST* [8]. The design library in Table 1, which is for 3 micron technology, was used for the experimentation. The clock cycle constraint given to BAD considering only operator delays, was 3000 ns.

[7]a good prediction of maximum fanout is not available at the time. We believe that predicting the wiring delay upper bound using the maximum wire length and the maximum possible fanout would result in a very loose upper bound which is of little use.

[8]However, a paper submitted to Design Automation Conference [15] describes layout experiments using the AR filter, which show that the cost-performance tradeoff curve is preserved with custom layouts.

The first results are given in Figures 9 through 12. The predicted and synthesized designs are pipelined with single cycle operations and the module set $(mul2, add3)$ is used so that no module characteristics could dominate the overall design characteristics. Figure 9 shows the combined operator, register, multiplexer and wiring areas of predicted (dotted line) and synthesized (points) designs for varying initiation intervals. Figure 10 shows the break-down of area characteristics of predicted and synthesized designs shown in Figure 9. The top curve is for the full model[9] (operator + register + multiplexer + wiring). The second curve from the top is for the RTL characteristics (operator + register + multiplexing). The third curve includes operator and register characteristics and the lowest curve is for operator characteristics only. It can be easily seen that not only are the overall characteristics of predicted and synthesized designs very close but also the individual characteristics (operator, register, multiplexer, wiring) are predicted closely.

Figure 11 shows the predicted (dotted line) and the actual (points) number of 2-point nets. The predicted number of 2-point nets is important because the wiring area (especially in the standard cell approach) is highly dependent on the number of 2-point nets. The predicted number of 2-point nets is generally within 10% of actual numbers.

The estimated (lines) and actual (points) number of stages are shown in Figure 12. The solid line is the expected number of stages and the dotted lines show the 3-sigma range for the estimated number of stages. Probability theory tells that more than 99% of actual numbers will be within the 3-sigma range of the solid line if distribution is normal.

The register, multiplexing,pla and wire delays (probable overheads for the main clock cycle) predicted by BAD as well as the estimated delays of synthesized designs (using module set mul2, add3) were within 3% of the clock cycle (3000 ns) so their impact was minimal. But, when the predictions are performed with the same module set, a faster clock (300 ns) and estimates using multi-cycle operations, these delays increase to 30% of the clock cycle. When a technology with a smaller feature size is to be used, then they might easily become comparable to operator delays. Therefore, these delays could be crucial depending on the specific application and the technology.

The results for non-pipelined designs are shown in Figures 13 and 14. Explanations of Figures 10 and 11 are also applicable to these figures. It is important to note that the clock cycle times generated from MAHA are not precisely the same as the clock cycle time given to the predictor due to the fact that MAHA uses operator chaining and the stage time can not be controlled from outside. But, clock cycle times of the synthesized designs shown in Figures 13 are within 10% of the clock cycle time given to BAD. Therefore, the results are still comparable. The newer generation of non-pipelined scheduling tools should do better than MAHA for resource allocation (at least for cases in which scheduling infeasibilities play an important role), so the predictions results will probably be even closer to the synthesized designs for such tools.

Finally, to illustrate another feature of BAD, it was used to search the design space for designs which met constraints. Figures 15 and 16 show total area for all designs considered, pipelined and non-pipelined respectively. For these design points, the same clock cycle was used but, this time multi-cycle operations and all possible module sets were allowed.

[9]PLA characteristics are not considered in the results given.

In order to select the feasible designs, the following constraints were given to BAD and the non-pipelined designs selected by BAD are shown by a $*$ in Figure 16. There were no pipelined design points which could satisfy these constraints. Chip Area = 90,000 mil^2, Performance = 27,000 ns, Circuit delay = 36,000 ns, Prob. of feasibility for Area = 1.0, Prob. of feasibility for Performance = 1.0, Prob. of feasibility for Circuit Delay $\geq$ 0.6.

The results from BAD include not only the overall characteristics (e.g area, throughput and total circuit delay) of designs satisfying design constraints, but also guidance of how to reach those predicted design points. An example would be: use the module set (mul3p, add3q), pipelined design style, initiation interval of n clock cycles, x adders, y multipliers, select a schedule which would result in approximately z registers, and so on.

5 Conclusion and Future Work

In this paper, we have presented a comprehensive set of accurate prediction methods to be used in high-level and system-level synthesis. The RTL characteristics of designs can be predicted accurately by the methods presented here. Validation of wiring space estimation is planned via construction of layouts. There is a related effort to predict characteristics of PLA based controllers [12]. The feasibility of using this approach for BAD is still under consideration[10]. The presence of conditional branches and pipelining are among the complicated issues. The controller may dominate the global design characteristics for highly pipelined implementations of designs with conditionals, and for implementations with multi-cycle operation architecture using fast clocks compared to module delays. For the non-pipelined AR Lattice filter designs in Figure 13, PLA based controller area (actual and estimated) for all design points were in the order of less than 5% of the total area figures.

It can be seen from Figures 12, 11 and 14 that better estimates for the number of stages for pipelined designs and the number of 2-point nets are needed. The estimates for the number of stages in pipelined designs do not currently involve predictions of the scheduling infeasibilities which are needed for better estimations.

This prediction tool and/or the methods presented can be effectively used in a number of design tasks including but not limited to design space exploration prior to high-level synthesis, evaluation of behavioral transformations and guidance for system-level decisions. BAD is now being used in a behavioral partitioning package which uses prediction methods to guide its search. There is a strong need for both accurate and fast predictions during the synthesis process. It is important to be able to quickly predict the final result of any design decision and many of these design decisions must be tried in order to reach globally better designs. Therefore, the speed of the predictions become as important as the accuracy. The run-time for BAD averages about 0.5 msec of CPU time per predicted design on a Sun Sparc 4/460 (180msec for 346 predicted designs), satisfying our run-time requirements for the predictions.

[10]Currently, only a subset of PLA prediction techniques from [12] are used in BAD.

References

[1] F. D. Brewer and D. D. Gajski. A System for Constraint Driven Behavioral Synthesis. *IEEE Transactions on Computer-Aided Design*, CAD-9(7), July 1990.

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

[3] Pravil Gupta. PLA and Wire Delay Analysis. Internal Report, in progress.

[4] R. Jain, K. Kucukcakar, M. J. Mlinar, and A. C. Parker. Experience with the ADAM Synthesis System. In *Proceedings 26th Design Automation Conference*, ACM/IEEE, June 1989.

[5] R. Jain, M. J. Mlinar, and A. C. Parker. Area-Time Model for Synthesis of Non-Pipelined Designs. In *Proceedings International Conference on Computer Aided Design*, ACM/IEEE, November 1988.

[6] R. Jain, A. C. Parker, and N. Park. Predicting Area-Time Tradeoffs for Pipelined Designs. In *Proceedings 24th Design Automation Conference*, ACM/IEEE, June 1987.

[7] K. Kucukcakar and A. C. Parker. Data Path Tradeoffs using MABAL. In *Proceedings 27th Design Automation Conference*, ACM/IEEE, June 1990.

[8] F. J. Kurdahi and A. C. Parker. PLEST: A Program for Area Estimation of VLSI Integrated Circuits. In *Proceedings 23rd Design Automation Conference*, ACM/IEEE, June 1986.

[9] Fadi Joseph Kurdahi. *Area Estimation of VLSI Circuits.* PhD thesis, Department of Electrical Engineering, University of Southern California, 1987.

[10] Richard I. Levin and Charles A. Kirkpatrick. *Planning and Control with PERT/CPM.* McGraw Hill, 1966.

[11] M. C. McFarland. Using Bottom-Up Design Techniques in the Synthesis of Digital Hardware from Abstract Behavioral Descriptions. In *Proceedings 23rd Design Automation Conference*, ACM/IEEE, June 1986.

[12] Mitch J. Mlinar. *System Level Tradeoffs in VLSI Design.* PhD thesis, Department of Electrical Engineering, University of Southern California, 1990. In progress.

[13] N. Park. *Synthesis of High Speed Digital Systems.* PhD thesis, University of Southern California, August 1985.

[14] N. Park and A. C. Parker. Sehwa: A Software Package for Synthesis of Pipelines from Behavioral Specifications. *IEEE Transactions on Computer Aided Design*, 7(3), March 1988.

[15] A. C. Parker, P. Gupta, and A. Hussain. The effects of Physical Design Characteristics on the Area-Performance Tradeoff Curve. In *Proceedings 28th Design Automation Conference*, ACM/IEEE, June 1991. Submitted.

[16] 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.

[17] Enders A. Robinson. *Probability Theory and Applications.* International Human Resources Development Corporation, 1985.

[18] S. Sastry and A. C. Parker. Stochastic Models for Wireability Analysis of Gate Arrays. *IEEE Transactions on Computer-Aided Design*, CAD-5(1), January 1986.

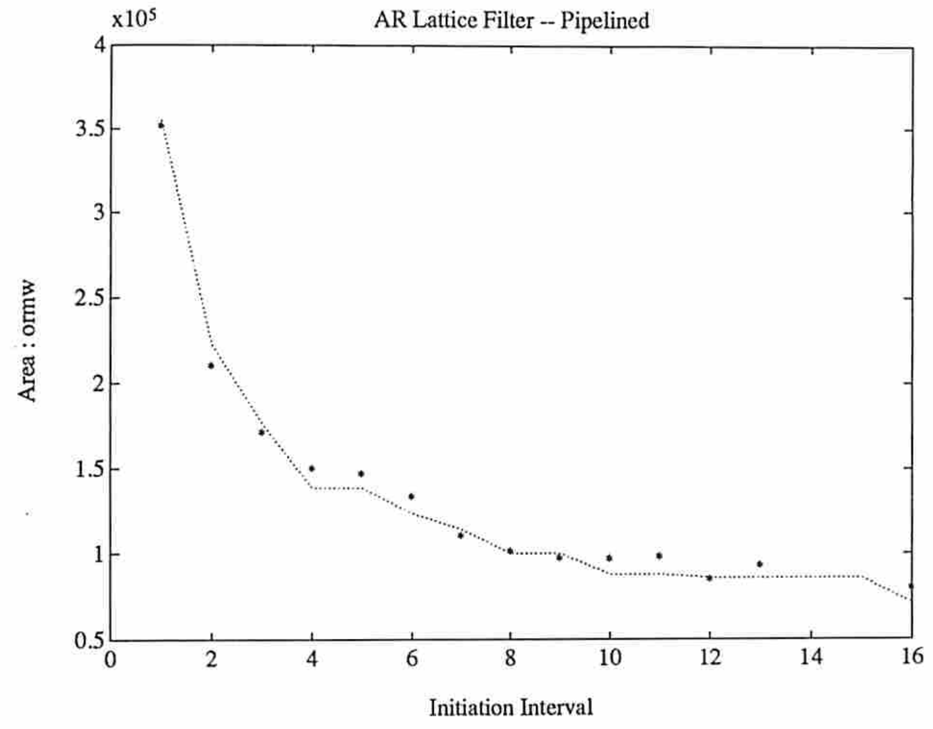

Figure 9: Area - Delay characteristics (using library mul2, add3)

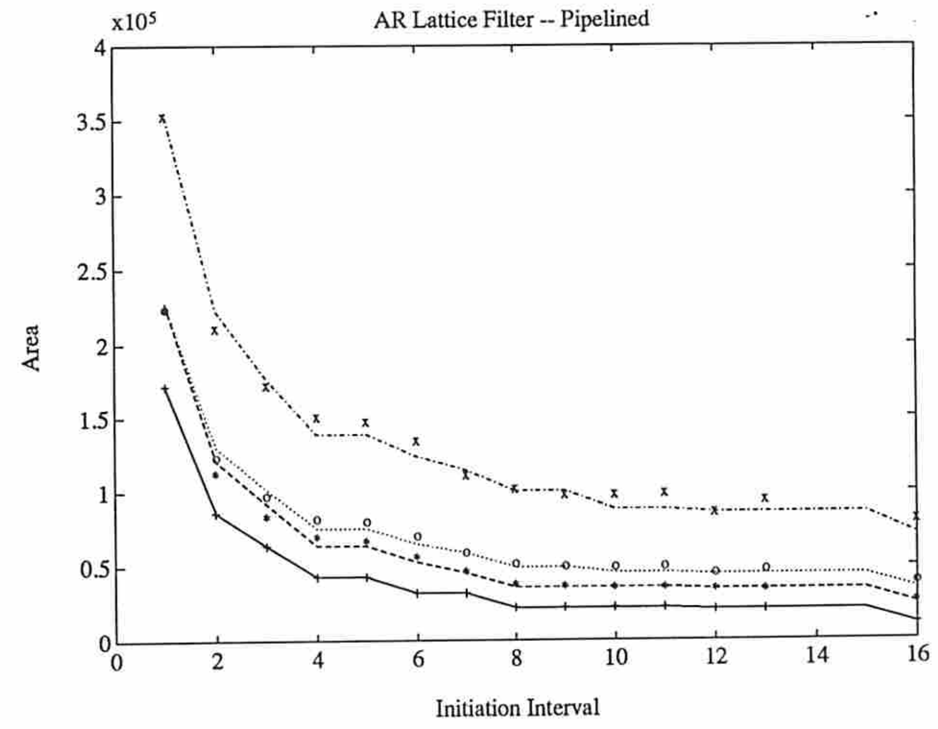

Figure 10: Break-down for Area - Delay characteristics using library mul2, add3)

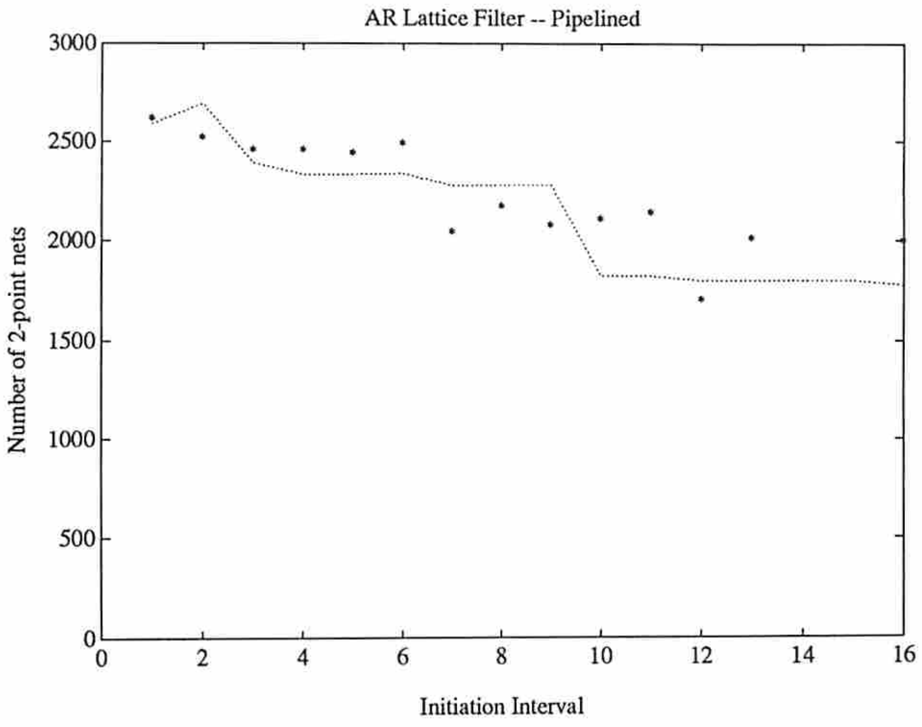

Figure 11: The variation of the number of nets (using library mul2, add3)

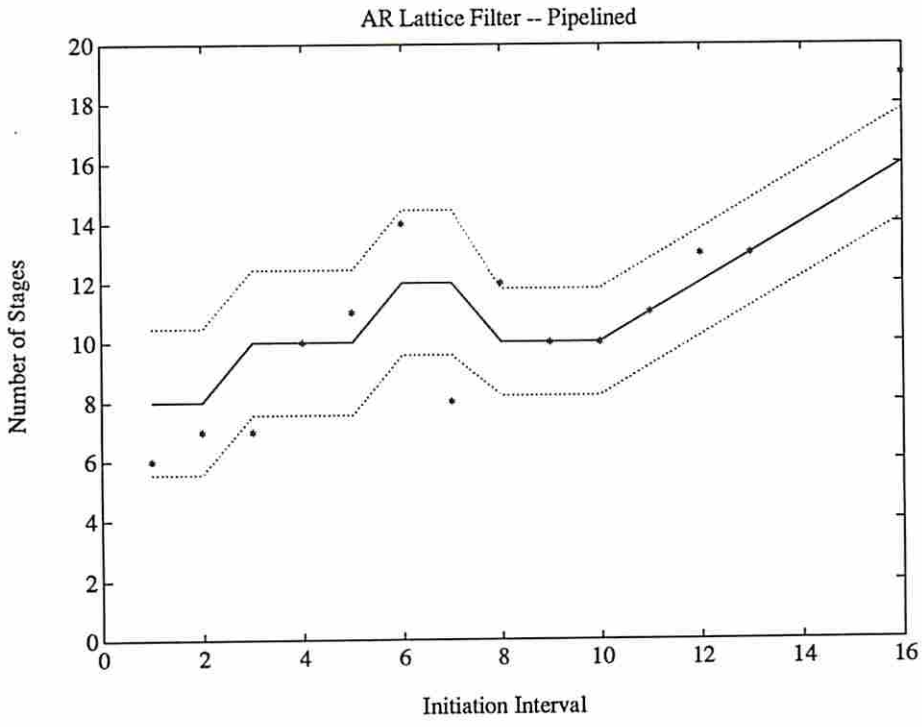

Figure 12: The variation of the number of stages (using library mul2, add3)

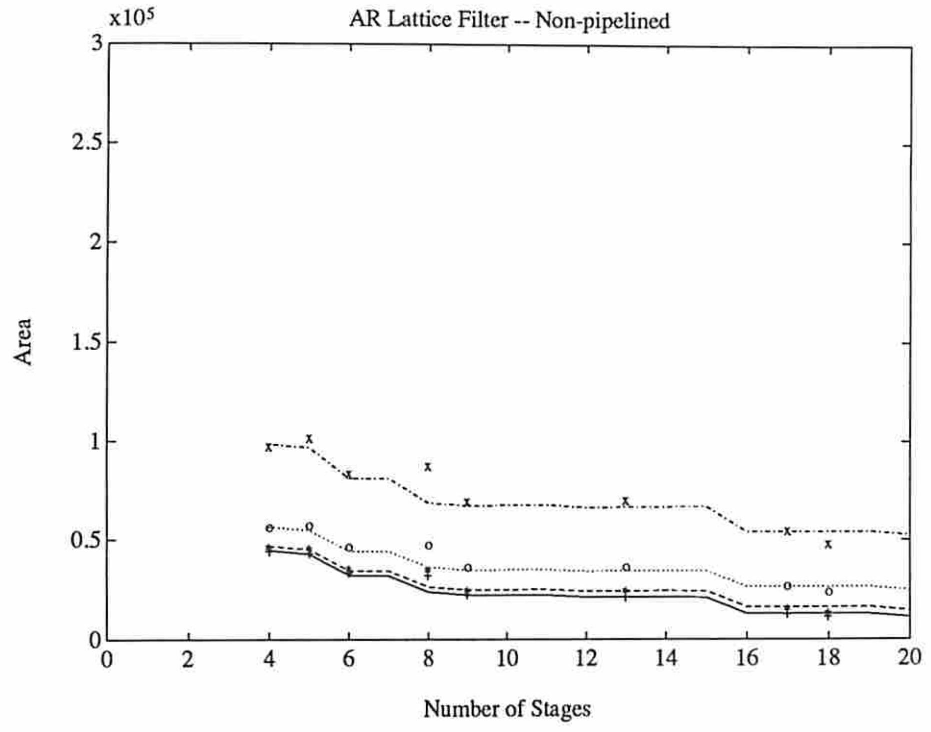

Figure 13: Break-down for Area - Delay characteristics using library mul2, add3)

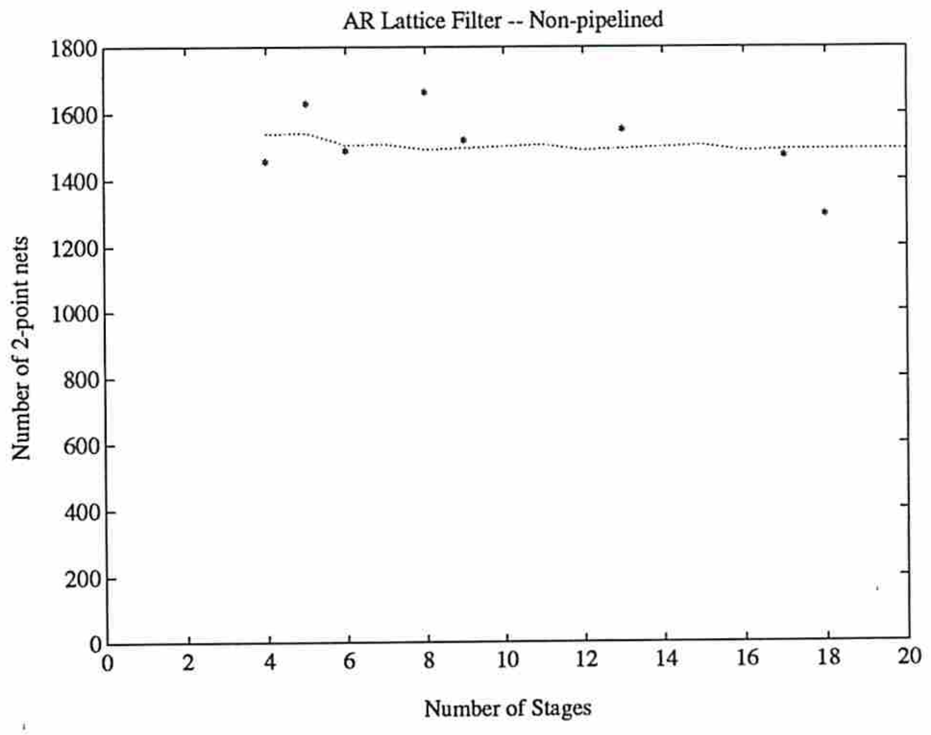

Figure 14: The variation of the number of nets (using library mul2, add3)

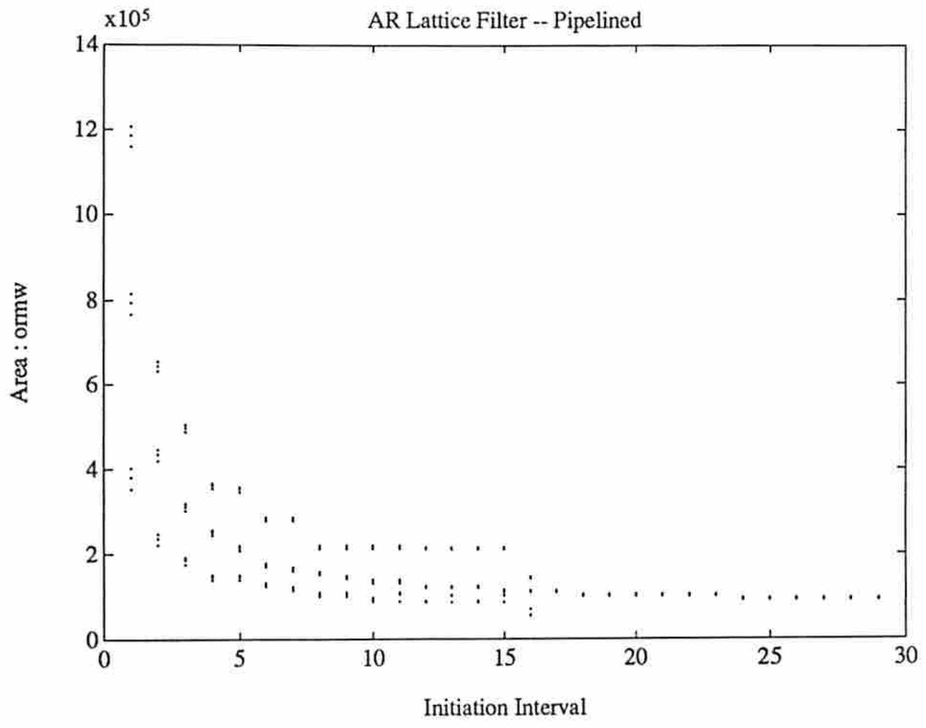

Figure 15: Predicted design points for all possible module sets

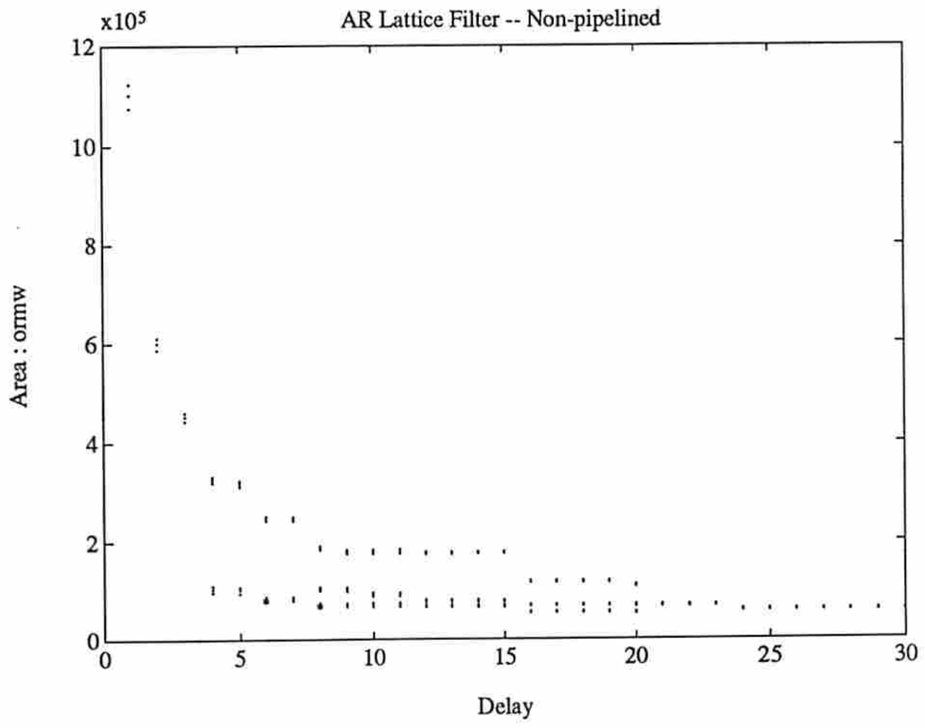

Figure 16: Predicted design points for all possible module sets