Predictability: Definition, Analysis and Optimization

Ankur Srivastava
Department of Electrical and Computer Engg.
University of Maryland, College Park
ankurs@glue.umd.edu

Majid Sarrafzadeh
Computer Science Department
University of California at Los Angeles
majid@cs.ucla.edu

Abstract

Predictability is the quantified form of accuracy. We propose a predictability driven design methodology. The novelty lies in defining and using the idea of predictability. In order to illustrate the basic concepts we focus on the low power binding problem. The binding problem for low power was solved in [3], [5], but in the presence of in-accuracies, their claims of optimality are imprecise. Our experiments show that these inaccuracies could be as high as 33%. Our methodology could improve this unpredictability to as low as 11% with minimal power penalty (7% on average).

1 Introduction

The design flow for VLSI systems is a systematic and step by step process. In each step a cost function that is assumed to capture the design objective is optimized. The estimation of this cost function can have many inaccuracies which can lead to unpredictabilities in final solution. The objective of this paper is to present a methodology that synthesizes designs that are more robust to inaccuracies and hence are more predictable w.r.t. power. Essentially, predictability is defined as a quantified measure of accuracy of an estimate. The gain in predictability would directly affect the overall synthesis time. Moreover improved predictability enables efficient design space exploration and better design management. We target the low power resource binding problem to demonstrate the idea of predictability driven design flow.

Binding is the process of ordering the operations on available resources such that the computation could be done successfully. [3], [5] try to minimize the switched capacitance of a binding solution and solve the problem optimally. We show that claims of optimality are not practical in the presence of inaccuracies. Our experiments with Media-bench benchmark suite showed that the unpredictability associated with power driven binding methodology was as high as 33%. Our predictability driven binding methodology which integrates SUIF, commercial tools like VSS, Synopsys D.C. and our own high level optimization engines could reduce this unpredictability in power to as low as 11% with very little penalty in power dissipation (less than 7%). Due to page limitations, some details have been omitted. A detailed description of our methodologies can be found in [2], [1].

The rest of the paper is organized as follows. Section 2 describes the concept of (un)predictability, it's sources and tradeoffs. Section 3 describes the low power binding formulations and Section 4 discusses the various objective functions and algorithms that capture predictability. Section 5 describes the experimental results followed by conclusion in section 6.

2 Understanding Unpredictability

The automated design of digital systems optimizes many cost functions that capture different design metrics. Estimation of these cost functions at any step is prone to inaccuracies. We define predictability as the quantified form of this accuracy. It is the range around the estimate in which the cost function can occur. Take for instance the following example

Let us suppose that there is a functional module which has three different (delay,power) values ((10,2), (8,4), (6,6)). These values could be a result of three different optimization strategies in logic synthesis. Let us suppose that the clock frequency is 15. Hence all the implementations meet the clock constraint. Unless we synthesize the design, we cannot estimate the right (delay,power) value for this module. If the estimated value for power is 4 (which is the average), then the estimate can vary by 2 units. There is an inaccuracy of 2 units OR 50%. This quantified value of 50% is defined as the unpredictability associated with the estimation. In this paper we address the predictability issues in binding for low power. Binding for low power has been studied in [5] and [3]. Both of them try to minimize the switched capacitance. In the next couple of subsections we will discuss some of the sources of unpredictability in the estimation of switched capacitance.

2.1 Unpredictability: Various Sources

Unpredictability Due to Logic Synthesis and Other Downstream Optimization: Most resource binding methodologies (low power or not power) depend highly on pre-characterization of functional modules. In a low power binding scenario, the modules are characterized for switched capacitance [5]. Estimation is based on a typical logic structure. The exact logic structure that this module will have after logic synthesis is unknown. Hence there is an unpredictability associated with the estimate. The source of this unpredictability is the unawareness of downstream optimizations.

Table 1 shows a typical power variation for two adder architectures. The results were obtained using synopsys design compiler through five different optimization options. The power values are obtained by gate level simulation [6]. The last column signifies the maximum variation in power dissipation from the average value. It is interesting to note that although Ripple Carry architecture dissipates lesser power, it has more variation and hence more unpredictability. If the exact architecture is not known, the estimation becomes even harder. Table 2 shows the variation in power dissipation for different adder architectures.

Unpredictability Due to Switching Activity: Switched Capacitance heavily depends on the switching activity. This switching activity depends on the structure of the circuit and also on the stochastic properties of the inputs. If the estimation engine does not capture these stochastic properties effectively then the estimation will be inaccurate.

Other Sources: Apart for the most prominent sources of unpredictability in power estimation discussed above, there can be other sources also which affect the accuracy of estimate. Sources like glitching power, power in wire segments and buses will also contribute to inaccuracies. In the next couple of subsections we will study the effects of the sources of unpredictability discussed above on different binding solutions.

2.2 Unpredictability and Resource Binding

[5] and [3] try to optimally solve the low power binding problem such that the total switched capacitance is minimized. We studied the variation in unpredictability in various binding solutions. Unpredictability of a binding solution was measured in the following way. Each binding instance was optimized using five different optimization options of design compiler such that the clock frequency constraints are met. The maximum variation from the average of these five power values was designated as the measure of unpredictability. This is a simple yet effective measure. Figure 1 indicates the variation in unpredictability for a sample data flow graph. Table 1 indicated that a ripple carry adder has

0-7803-7607-2/02/$17.00 ©2002 IEEE

Adder Architecture	Option-1	Option-2	Option-3	Option-4	Option-5	Average	Max Variation
Ripple	12.89uw	12.09uw	14.09uw	19.38uw	14.09uw	14.5uw	33.6%
Carry Look-Ahead	15.22uw	13.99uw	17.38uw	17.47uw	17.38	16.3uw	14.06%

Table 1: Variation in Power for Different Logic Level Optimizations

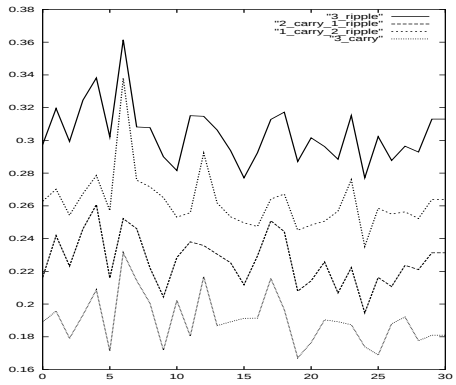

Figure 1: DFG1: X-axis: Different Binding Instances, Y-axis: Unpredictability

	Ripple	CLA	Avg.	Max Var
Option-1	13.233uw	17.85uw	15.54	14.8%
Option-2	12.3287uw	17.74uw	15.06	17.99%
Option-3	14.4uw	14.30uw	14.35	0.3%
Option-4	19.73uw	15.35uw	17.54	10.8%%

Table 2: Power Variation for Different Adder Architectures

more associated unpredictability. Hence we can expect a design with more ripple carry adders to be more unpredictable. In figure 1, it can be seen that as the number of ripple carry adders increase, the unpredictability also increases, for all the bindings. One very direct conclusion that one can draw from these experiments is that unpredictability varies massively between different binding solutions.

2.3 Why is Unpredictability Undesirable?

As mentioned already, predictability is the measure of level of accuracy or confidence that can be associated with a prediction. A more predictable estimate means that after all the optimization steps of the design flow have been executed the difference between the actual value of the cost function and the one predicted at high level is lesser compared to a less predictable estimate. This means better interaction between high level and low level tools and lesser chances of re-iteration. Predictability is key in a power management (design management) perspective. More predictability in estimation will enable early design planning and better early on design space exploration. This would finally enable the synthesis of better designs.

3 The Low Power Binding Problem

In order to illustrate our unpredictability driven design flow, we need to describe the low power binding problem. The low power binding problem strives to minimize the switched capacitance of the functional resources. It has been studied in detail in [5], [3]. Figure 2 illustrates the formulation of the binding problem.

Given a scheduled graph as shown in figure 2(A), a compatibility graph is built out of it (figure 2(B)). Two operations are compatible if they can be computed on the same resource. The node list of the compatibility graph is augmented with a source and sink and directed edges are added from all the nodes to the sink node and from the source to all the nodes (figure 2(C)). The low power binding problem can have two variations.

1. The architecture of the functional module is known

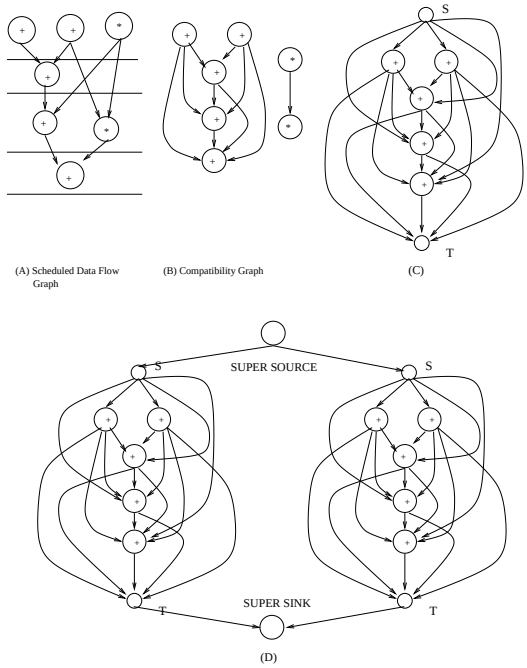

Figure 2: The Low Power Binding Problem

2. The architecture of the functional module is not known and we need to pick the right architecture from a library of architectures

In the first version, the edges of the compatibility graph are assigned a cost where this cost indicates the estimated switched capacitance on the given module if the source and destination of the edge form a predecessor and successor pair of operations on that resource. The objective is to send **R** units of flow from source to destination with minimum cost such that each node is visited exactly once. The objective function represents the overall switched capacitance of the binding solution. More formally the objective and constraints can be represented as

$$Minimize \sum_{ij \in E} X_{ij} C_{ij} \quad (1)$$

$$\sum_{\forall i \, such that \, si \in E} X_{si} = R \quad (2)$$

$$\sum_{\forall i \, such that \, it \in E} X_{it} = R \quad (3)$$

$$\sum_{\forall j \, such that \, ij \in E,} X_{ij} = 1 \; \forall i, i \neq s, t \quad (4)$$

$$\sum_{\forall i \, such that \, ij \in E,} X_{ij} = 1 \; \forall j, j \neq s, t \quad (5)$$

$$X_{ij} should \; be \; positive \; integer \quad (6)$$

Where
C_{ij}: Switching Cost of an Edge ij, Cost=0 if either i $\in$ s,t or j = $\in$ s,t
R: Number of Resources

[5] shows that this problem can be optimally solved using min cost flow problem. An extension of this problem tries to find the best architecture from a library of architectures such that the overall switched

Bench	$MinSwitchedCap$			$Min\delta_{avg}$			$Min\delta_{max}$		
	Power	δ_{bind}	Max Power	Power	δ_{bind}	Max Power	Power	δ_{bind}	Max Power
fft-2	76.57	0.27	97.24	98.31	0.18	116.01	89.76	0.18	105.92
jctrans-1	159.99	0.33	212.79	173.12	0.11	192.16	173.32	0.12	194.12
jctrans-2	42.10	0.29	54.31	48.96	0.11	54.35	51.14	0.12	57.28
$jdmerge-4$	16.03	0.16	18.59	18.16	0.14	20.70	18.45	0.15	21.22
motion-2	95.92	0.27	121.82	121.95	0.16	141.47	120.96	0.16	140.31
motion-3	92.60	0.27	117.60	112.03	0.16	129.95	117.98	0.17	138.04
noiseest-2	66.98	0.26	84.39	75.45	0.16	87.52	75.45	0.16	87.52
ecb-enc-4	105	0.28	134.40	122.1	0.13	137.7	127.79	0.14	145.68
jdmerge-1	89.08	0.28	114.02	107.24	0.13	121.18	110.42	0.13	124.77
jdmerge-3	121.52	0.29	156.76	143.25	0.13	161.87	146.27	0.13	165.29

Table 3: Switched Capacitance and Unpredictability Values for Various Optimization Objectives

capacitance is minimized. In order to solve this problem, the formulation shown in figure 2(C) needs to be extended to consider multiple architectures. Let us suppose there are **M** different architectures and as before we also have a resource constraint of **R**. The problem is binding of operations together on **R** resources and deciding the architecture for each resource such that the overall switched capacitance is minimized. The graph shown in figure 2(C) is replicated **M** times (figure 2(D)). Each s node in the graph is connected to a node called Super-source and each t node in connected to a Super-sink. We need to send **R** units of flow from the Super-source to Super-sink with minimum cost. The constraints force us to use an operation in exactly one resource. Each of the replicated sub-graph represents a particular architecture. The cost values of an edge in that sub-graph represent the estimated switched capacitance on that particular architecture. We believe that this problem is NP-Complete because of it's closeness to the multicommodity flow problem. It could be solved optimally using any ILP solver. We have also proposed a heuristic based on LP-relaxation of the problem. Basically the heuristic iteratively finds R paths in the underlying compatibility graph using LP-Relaxation. For brevity we skip the detailed discussion of this heuristic.

4 Binding for Predictability

This section is devoted towards solving the predictability driven binding problem. Formally we can state it as

Given a compatibility graph of operations in a DFG. Given the edge costs C_{ij}. Given the edge unpredictabilities which represent the maximum variation possible from the corresponding cost. Bind the operations into **R** *resources such that the variation of the estimated switching capacitance of the binding solution is minimized.*

As mentioned before, the edges of the compatibility graph have an associated cost which signifies the estimated switching activity. Due to reasons discussed earlier, this estimate is bound to have inaccuracies and hence unpredictability. In the presence of these inaccuracies, we would like to bind the operations such that the inaccuracy in the switching activity of the binding solution is minimized. Let us suppose C^{ij} are the costs of the edges ij in a binding solution. Let δ^{ij} be the unpredictability associated with each of the cost values. The estimated switching capacitance of the binding solution is given by $\sum_{\forall ij \in Binding-solution} C^{ij}$. The maximum variation in the presence of unpredictabilities is given by

$$\sum_{\forall ij \in Binding-solution} C^{ij}\delta^{ij} \tag{7}$$

Hence the associated unpredictability in the binding solution is

$$\delta_{bind} = \sum_{\forall ij \in Binding-solution} C^{ij}\delta^{ij} / \sum C^{ij} \tag{8}$$

A smaller value of δ_{bind} would mean lesser variation and hence more predictability in the binding solution. The idea is to solve the binding problem such that δ_{bind} is minimized.

This is a highly non-linear objective function and makes the design of optimal polynomial time algorithms or even Integer Linear Programming formulations very hard. Hence we propose two alternative objective functions and study their closeness to the objective function of equation 8.

4.1 Minimizing Maximum Unpredictability

Let us make the following observations.

$$if \delta_{max} \geq \delta^{ij} \; \forall \; edges \; ij \; \in \; Binding - Solution \tag{9}$$

$$\delta_{max} \geq (\sum_{\forall ij \in Binding-solution} C^{ij}\delta^{ij}) / \sum C^{ij} \tag{10}$$

If δ_{max} is the largest of all the δ^{ij} in the *binding solution*, then δ_{max} is also the upper bound δ_{bind} (directly from equations 9 and 10). Hence minimizing δ_{max} of binding solution could be a good objective function to maximize the predictability. The problem could formally be stated as follows

Given a compatibility graph with edge costs and associated unpredictabilities (δ^{ij}), find a binding solution such that δ_{max} is minimized.

The objective function could be formally stated as

$$Minimize \; z \tag{11}$$

$$z \geq X_{ij}\delta^{ij} \; \forall ij \in E \tag{12}$$

This objective could be augmented with constraints of the low power binding problem to formulate the complete optimization problem. Algorithmically, there are two distinct formulations that need to be solved: Single Architecture and Multiple architecture. The single architecture problem can be solved as follows.

Sort the edges in decreasing order of δ^{ij}. Remove the edge with largest δ^{ij} and solve the Mincost Flow formulation (with power as objective [5]) to find a feasible flow. Repeat the above steps till no feasible flow exists. The smallest δ^{ij} for which a solution can be found is the minimum δ_{max}.

Note that for single architecture instance this formulation can be solved optimally. In Multi-architecture case, the same methodology could be used. Instead of solving Mincost flow optimally in each iteration optimally, we can use some heuristic or ILP formulation. Also note that each iteration essentially generates a tradeoff point between power and predictability. Hence this algorithm could also be used to make an effective tradeoff.

4.2 Minimizing Average Unpredictability

Minimizing δ_{max} may not be a very good idea since we cannot say anything about the tightness of upper bound. An average based cost function could follow the function in equation 8 (δ_{bind}) more accurately.

$$\delta_{avg} = \sum_{\forall \; edges \; ij \in Binding-Solution} \delta^{ij}/n \tag{13}$$

Here
n: Number of edges of the compatibility graph in the Binding Solution.

As shown in equation 13, δ_{avg} is the average of all δ^{ij} in the binding solution. Another point to note is that in any binding solution, the number of edges will remain the same. This is because each operation has to appear in a resource exactly once. Hence minimizing δ_{avg} is same is minimizing the sum of all δ_{ij}.

In order to minimize δ_{avg} we can replace the edge costs by edge unpredictabilities in the algorithmic formulations discussed in section 3 and solve for minimizing the sum of the δ^{ij} in a binding solution. Since the number of edges in a solution remain the same, minimizing the sum corresponds to minimizing the average. When both predictability and cost are of concern, then we can replace the cost of an edge by a linear combination of edge cost and edge unpredictability and solve the problem using the same formulations as discussed in section 3.

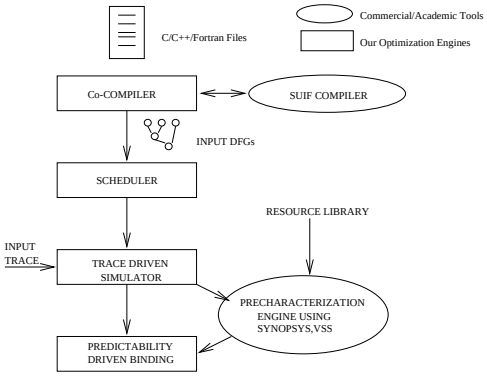

Figure 3: Experimental Flow

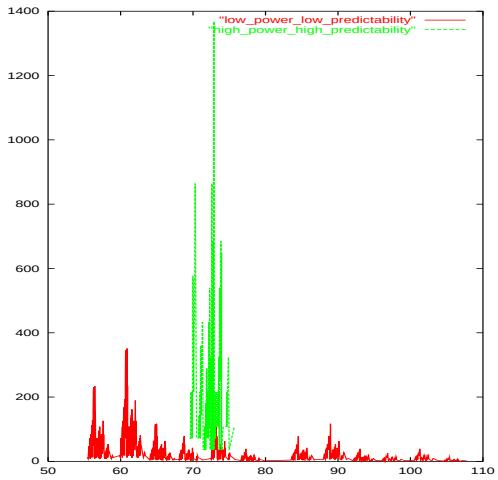

Figure 4: Power Prob Distribution: Whole RTL Design : JDMERGE4, X-Axis Power, Y: Axis Freq. of Occurance

5 Experimental Results

We built an experimental flow shown in figure 3 which enabled us to experiment with some of the concepts. We take functions written in C and extract data flow graphs from them using SUIF/Machine-SUIF. These DFGs are then scheduled using a path based scheduler [7]. This scheduled data flow graph is then simulated with an input trace to evaluate the values of internal variables. All this information is used by the pre-characterization engine (Synopsys d.c. and VSS-Simulator) to generate the edge unpredictabilities and costs. A number of different synopsys optimization scripts are used to compute the edge cost (average of all optimization scripts) and unpredictabilities (max variation). The binding engine then generates the solution depending on the cost function.

Table 3 reports the results obtained on Mediabench benchmark suite [4]. We binded the benchmarks using three separate cost functions: Minimum Switched Capacitance (cost), Minimum δ_{max} and Minimum δ_{avg}. Note that for δ_{max}, δ_{sum} based cost function, the optimization algorithm did not control the increase in switched capacitance. For each optimization objective, we report the value of δ_{bind} and switched capacitance and the maximum power dissipation in presence on unpredictabilities. The reported results are only for the addition operations in the DFG. We had two adder architectures in the module library: Ripple Carry and Carry Look-ahead.

We can observe that although there is a large gain in predictability, in many cases the increase in power dissipation offsets for this gain. Table 3 illustrates that the maximum power dissipation in the presence of unpredictability in power driven binding solution is still lesser that the maximum power in cases where predictability is optimized. Hence generating a power predictability tradeoff is imperative. If the tradeoff point is picked wisely, the maximum power in the presence of unpredictabilities could become much lesser than the case when only switched capacitance or only unpredictability is maximized. Table 4 reports the results when power and predictability are considered together, for all the benchmarks. These results show that there is a massive increase in predictability without much increase in switched capacitance (power) (7% average). Moreover in most of the cases the maximum power was lesser than the case when DFGs were binded for minimum switched capacitance. We also synthesized the whole RTL design for the JDMERGE4 benchmark with 3 Adders, 3 Multipliers and 1 Subtractor using the different binding formulations discussed above. Figure 4 indicates the probability distribution of power for the case when binding is done for low power vs predictability. The variation in power occured due to different gate level implementations for the same RTL design. It can be seen that our techniques were very effective in increasing predictability.

6 Conclusion and Future Work

This paper provides an initial impetus towards a predictability driven design flow. We illustrated the idea through the low power binding problem. We found that after effective tradeoff between power and predictability, designs with better power and accuracy could be generated. In this work, we primarily focus on data flow graphs. We believe this is the first step in predictable power estimation/optimization. Extension of this work to control data flow graphs with uncertainties due to branches is under consideration. Interesting course of future work would include development of statistical cost functions and their optimization, capturing other sources of unpredictability and their quantification. Extension of predictability to other levels of design and other cost functions is another interesting future direction.

Bench	Power	Power Penalty	δ_{bind}	Max Power
fft-2	84.9	10.87%	0.18	100.38
jctrans-1	173.12	8.20%	0.11	192.16
jctrans-2	44.54	8.92%	0.12	49.88
jdmerge-4	16.3	1.68%	0.16	18.83
motion-2	102.98	7.36%	0.19	122.56
motion-3	101.6	9.72%	0.17	118.90
noiseest-2	70.02	4.54%	0.18	82.44
ecbenc-4	111.39	6.1%	0.14	126.66
jdmerge-1	95.24	6.9%	0.13	108.07
jdmerge-3	129.38	6.46%	0.14	146.87

Table 4: Power/Predictability Tradeoff

References

[1] A. Srivastava and M. Sarrafzadeh . "Unpredictability in Binding: Concepts, Cost Functions and Algorthms". In *Technical Report No. 020006, UCLA*, Feb 2002.

[2] A. Srivastava, E. Kursun and M. Sarrafzadeh . " Predictability in RT-Level Designs ". In *Journal of Systems, Circuits and Computers (To Appear)*, 2002.

[3] A. Raghunathan and N. Jha. "An ILP Formulation for Low Power Based on Minimizing Switched Capacitance During Datapath Allocation". In *Procs of IEEE Symposium on Circuits and Systems*, 1995.

[4] C. Lee, M. Potkonjak and W.H. Mangione-Smith. "MediaBench: A Tool for Evaluating and Synthesizing Multimedia and Communications Systems". In *International Symposium on Microarchitecture*, 1997.

[5] J.M. Chang and M. Pedram. "Low Power Register Allocation and Binding". In *Proc. Design Automation Conference*, pages 29–35, June 1995.

[6] R. Birch, F. Najm, P. Yang and T. Trick. "McPOWER: A Monte Carlo approach to power estimation". In *IEEE/ACM International Conference on Computer-Aided Design*, pages 90–97, Nov 1992.

[7] S. Ogrenci-Memik, E. Bozorgzadeh, R. Kastner and M. Sarrafzadeh. "A Super Scheduler for Embedded Reconfigurable Systems". In *Procs of International Conference on Computer Aided Design*, pages 391–394, Nov 2001.