Voltage Scheduling Under Unpredictabilities: A Risk Management Paradigm

AZADEH DAVOODI and ANKUR SRIVASTAVA
University of Maryland

This article addresses the problem of voltage scheduling in unpredictable situations. The voltage scheduling problem assigns voltages to operations such that the power is minimized under a clock delay constraint. In the presence of unpredictabilities, meeting the clock latency constraint cannot be guaranteed. This article proposes a novel risk management based technique to solve this problem. Here, the risk management paradigm assigns a quantified value to the amount of risk the designer is willing to take on the clock cycle constraint. The algorithm then assigns voltages in order to meet the expected value of clock cycle constraint while keeping the maximum delay within the specified "risk" and minimizing the power. The proposed algorithm is based on dynamic programming and is optimal for trees. Experimental results show that the traditional voltage scheduling approach is incapable of handling unpredictabilities. Our approach is capable of generating an effective tradeoff between power and "risk": the more the risk, the less the power. The results show that a small increase in design risk positively affects the power dissipation.

Categories and Subject Descriptors: B.5.2 [**Register-Transfer-Level Implementation**]: Design Aids—*Automatic synthesis, optimization*

General Terms: Algorithm, Reliability, Design

Additional Key Words and Phrases: Predictability, voltage scheduling, low power, design closure

1. INTRODUCTION

Estimation in design automation is marred by inaccuracies. Important design objectives like power are extremely hard to predict, especially at high levels of design flow. On the other hand, optimization of design objectives at system the level has a tremendous impact on design quality. Critical optimizations need to be performed in unpredictable scenarios.

Risk management tries to control the maximum amount of error/ unpredictability associated with any estimation. Under this paradigm, the user specifies a risk which signifies the amount of inaccuracy the designer can "risk." This risk is the "violation likelihood" of the constraint for potential gains in design quality.

Authors' addresses: Electrical and Computer Engineering Department, University of Maryland, College Park, MD 20742; email: {azade,ankurs}@eng.umd.edu.

This article demonstrates this paradigm through the voltage scheduling problem, which assigns voltages to individual operations in a data flow graph (DFG) for power minimization. Assigning multiple voltages to operations is a very strong technique for power optimization. This comes from the quadratic dependence of voltage on power. This methodology was primary investigated by [Raje and Sarrafzadeh 1995] and generalized by [Chang and Pedram 1995]. Both these approaches assume accurate information about power and delay are available through estimation engines. Since the complete implementation information is not known, this assumption is far from valid. In this work, we extend voltage scheduling in a risk management paradigm.

Instead of characterizing the delay and power at each voltage by exact values, we represent them by probability distributions. This is more realistic since the estimation is inaccurate. The designer specifies a clock constraint C and a risk factor R which represents the maximum number of clock cycles the designer is willing to "risk." The algorithm then assigns voltages such that both the expected value of clock delay is $\leq C$ and the maximum latency is $\leq R$ while minimizing the power.

The advantage of a risk management paradigm is that it gives control of the unpredictabilities to the designer. Depending on the amount of "risk" the designer is willing to take, the design quality changes. If the acceptable "risk" is high, the algorithm will be relaxed and will generate a lower power solution. Hence we can expect a "risk" versus power tradeoff. The designer should then be able to pick the appropriate point on this tradeoff curve which indicates a balance between design risk and design quality.

In this endeavor, the expected value of the clock cycle constraint is never relaxed; hence this approach is different from slack-based techniques. We also demonstrate that small increase in "risk" can significantly affect power. The designers can experiment with this new parameter to generate solutions that meet their power constraints with a reasonable degree of predictability. The main contribution of this article includes proposing the risk management paradigm and demonstrating its effectiveness through the voltage scheduling problem.

The rest of this paper is organized as follows. Section 2 contains a brief discussion on the issue of unpredictabilities. Section 3 reinvestigates the low-power voltage scheduling problem. Section 3.2 presents our risk management algorithm and Section 4 contains the experimental results.

2. UNPREDICTABILITIES: AN INTEGRAL ASPECT OF DESIGN AUTOMATION

Design automation is a step by step procedure mainly comprised of optimization algorithms driven by estimation engines. Estimation of the design quality especially at a system level of design flow is an extremely complicated task to perform. Accuracy of estimation is severely limited at the system level since various design parameters are not known. On the other hand, design decisions taken at the high level greatly impact the design quality and the time to market. Hence critical design decisions need to be made in the presence of high degrees of estimation inaccuracies.

There has been a tremendous amount of effort aimed at improving the prediction accuracy, especially with respect to power [Pedram 1996], in the past. Typically, most approaches try to increase the amount of implementation information (like wire-delay, leakage power) at the earlier stages in the design flow, hoping to increase the accuracy of prediction. Estimation tools try to predict the course of future/low-level optimizations in order to increase the implementation details. Unfortunately, even state-of-the-art estimation techniques cannot guarantee a reasonable degree of accuracy. Unpredictabilities creep into any such strategy. This article addresses the issue of unpredictability by proposing a strategy which does an accuracy/risk versus design quality tradeoff, hence managing the unpredictability and design quality.

2.1 Unpredictability: Sources and Impact

Unpredictability is defined as the quantified form of accuracy [Srivastava and Sarrafzadeh 2002].

There are many sources of unpredictabilities for various cost functions such as power. At higher stages of design flow, the unawareness of the exact logic structure of the functional module makes exact estimation of power, area, delay, etc., impossible. Another important source of unpredictability in power estimation is unawareness of exact switching activity. Furthermore, exact values of low-level details like wire delay or wire capacitance cannot be determined either, forcing estimation to have inaccuracies. Hence inaccuracies/unpredictabilities creep in due to an unawareness of exact implementation details.

From a practical point of view, it is not possible to estimate the exact implementation details. Even if we have excellent estimation engines which capture each and every parameter responsible for a particular design objective, there is no way to predict the exact value of those parameters. One strong reason for this fact is the interdependence between various cost functions. At later stages of design flow, aggressive optimization of one design objective usually drastically affects other design objectives, hence invalidating any system level decisions/optimization based on early estimation.

2.2 Risk Management: Tradeoff Between Design Quality and Predictability

In any optimization problem, the design constraints must be satisfied to generate a valid/feasible design. More specifically in this work, let us suppose the constraint C is the clock cycle constraint of a DFG and O is the power optimization objective obtained through voltage scheduling. Even if the generated solution satisfies C, there is no guarantee that it will be satisfied after the low levels of design optimization (logic synthesis, physical design). This might result in forcing several iterations.

Hence, if area/noise is heavily optimized at the placement stage, the clock cycle constraint may get violated. Now let us assume that we know the kind of unpredictabilities associated with each solution of the optimization problem. *This means for each solution to the voltage scheduling problem, we know not only the expected value of the clock cycles C but also its range*. Let us also suppose the designer specifies a risk factor he/she is willing to take on C. This specifies the maximum violation of C the designer is willing to tolerate. For example, C might be 10 clocks but the designer is willing to tolerate C up to 12. In the presence of such a scenario, an *unpredictability/risk management* paradigm would approach the problem in a following style:

(1) The expected value of the latency should always be less than C (previously C itself was the constraint).
(2) The associated unpredictability is such that it is always less than the designer-specified risk R.
(3) The objective value O is the best possible among this range of accepted (valid) solutions.

Hence the unpredictability management paradigm manages the associated error to keep it within acceptable bounds. Note that the expected value of the constraint must still be less than C. It's just that the worst case likelihood of the constraint can be tuned.

It should be noted that the work done in modeling and optimization under uncertain/unpredictable cost functions is completely in contrast to our approach. As an example Jyu and Malik [1993, 1994], Tomiyama et al. [1998] and Tomiyama and Yasuura [1998] addressed the issue of predictability in their own individual ways. Tomiyama et. al. [1998] and Tomiyama and Yasuura [1998] addressed the unpredictability in delay estimation due to manufacturing defects. They presented techniques for resource binding and module selection such that the likelihood of failure is minimized. Jyu et. al. [1993, 1994] again captured the manufacturing defects and tried to maximize the probability of meeting the performance constraints in the presence of manufacturing variations. They proposed optimization methodologies and delay models to this effect.

In contrast, the approach here is to make the designs more predictable in the presence of low-level optimization uncertainties such that the high-level decisions can be taken with greater confidence. Here, we expect a tradeoff between risk and design quality. If the designer's specified risk is low then the design quality (power in the voltage scheduling problem) will be worse. If the risk is infinity then the problem will be the same as the traditional approach.

3. LOW-POWER VOLTAGE SCHEDULING: TRADITIONAL VERSUS RISK MANAGEMENT APPROACHES

In this section we will initially overview the multiple supply voltage scheduling problem. Power is quadratically dependent on the supply voltage, as illustrated below [Chandrakasan et al. 1992]:

$$Power = K.V^2.\beta, \tag{1}$$

where K = proportionality constant, V = supply voltage, and β = switched capacitance factor

Numerous techniques have been proposed to optimize the various components of this expression. Chen et al. [2001] described a strategy that reduces the power dissipation through voltage scaling but without any performance penalty. A substantial amount of power is dissipated in clock trees. This is because its switching activity is the highest. Many articles have addressed the issue of clock power. Tellez et al. [1995] presented one of the approaches.

At the behavioral level, a wide class of transformations can be done in order to achieve lower switched capacitance. These include low-power binding and resource allocation [Chang and Pedram 1995; Kruse et al. 2001]. Dynamic power management is a system-level power management technique for controlling the power dissipation by shutting down idle components [Chung et al. 1999].

Power has a nonlinear relationship with the voltage. Reduction of voltage reduces the power dissipation but also increases the gate delay. Hence there is a power performance tradeoff. Voltage scaling which reduces the supply voltage quadratically affects the power but also increases the delay, which can be presented as follows:

$$Delay = K.V/(V - V_t)^{\alpha}, \tag{2}$$

where K = proportionality constant, V = Supply voltage, and V_t = Threshhold voltage.

3.1 Traditional Approach

System-level voltage scheduling techniques take a DFG as input and assign voltages to each operation such that the sum of the overall operation power is minimized and the given clock constraint is satisfied. This problem was tackled in Raje and Sarrafzadeh [1995] and solved optimally for general directed acyclic graphs (DAGs) assuming the same voltage-delay/power curves for all nodes. Chang and Pedram [1997] relaxed this assumption and solved the problem optimally for tree-like DFGs.

The voltage scheduling problem has the following inputs and outputs:

(1) *Input*: DFG, voltage/delay, voltage/power curves for all operations. This can include availability of different architectures for each operation type, which means each operation can have multiple voltage/delay, voltage/power curves. A delay constraint in clock cycle C is also known.
(2) *Output*: Voltage (and architecture) assignments to each operation such that power dissipation is minimized while the clock constraint is satisfied.

Figure 1 illustrates a typical variation of voltage/delay and voltage/power. The formulation assumes a set of predecided voltages (V1, V2, and V3 in this case) which will be available on the chip. The algorithm for tree-like DFGs has two iterations: the forward pass and the backward pass. In the forward pass, the DFG is traversed topologically from primary inputs (PIs) to primary outputs (POs). By the end of the forward pass, the cooptimal solutions that meet the clock delay constraint are determined. The optimal solution that results in minimum power is then chosen. This is followed by a backward pass where the voltage assignment corresponding to the optimal solution is determined. Next, more details on these two passes are provided.

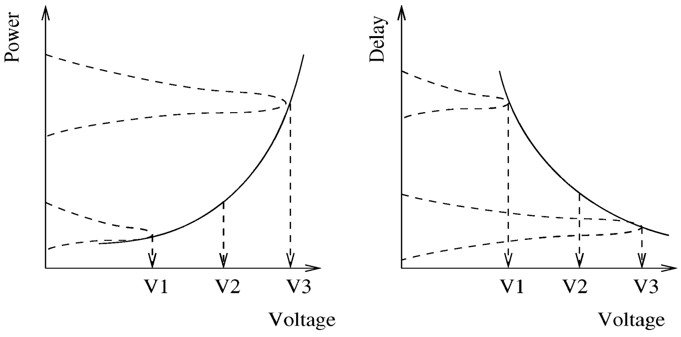

Fig. 1. Power/voltage, delay/voltage variation.

3.1.1 *The Forward Pass.* Initially we introduce the notation used to describe the delay/power curves in Figure 1. The solution at a node i, referred to as $nodeInf_i$, includes these curves. $NodeInf_i$ contains information for each predecided voltage and its associated delay and power values. In the forward pass, initially the DFG is traversed from PI to PO in topological order. At each node, the minimum power dissipation of the subgraph rooted at that node is stored as a delay curve corresponding to that node. This is represented as a set of ordered tuples (t, p) that contains a solution with the arrival delay of t clock steps and minimum power of p in the subgraph rooted at that node. Here we will be referring to this ordered set as *arrivalInf* for each node.

$ArrivalInf_n$ contains information about the times the signal from node n becomes available. Given the *arrivalInf* of the fanins for any node, a function *max* is defined which considers the arrival times of all the fanins of the node simultaneously. The output of *max* is another delay curve referred to as *arrivalMax* for each node. Since the output of different fanins becomes available at different times, the signal coming from all the fanins is valid at the maximum of all these arrival times. This is done by merging the *arrivalInf* of each fanin. Considering any node i, *max* function looks at the *arrivalInf* of all the fanins of i and creates one delay curve (*arrivalMax*) representing the availability of the signal from the fanins. Note since different delay combinations might have the same maximum, among all combinations with the same maximum, the one with minimum power will be stored. This is briefly represented in equation below:

$$arrivalMax_i = max(arrivalInf_j \quad \forall j \in fanin\ i). \tag{3}$$

The function *max* symbolically finds the maximum of the *arrivalInf* curves for the fanins of each node. For each delay combination, power is simply the addition of the powers of each fanin. The power of fanin will be specified based on the assigned delay of the fanin. This is represented in equation below:

$$p_i = \sum_{\forall j \in fanin_i} p_j. \tag{4}$$

When computing $arrivalMax_i$ for each node i, the delay values greater than constraint C will be discarded. Hence the size of this function is always bounded. The *arrivalMax* obtained after *max* on all the fanins is then subjected to a *sum* function. The input to *sum* is *arrivalMax* and *nodeInf* of the operation. Recall that each point on *nodeInf* reflects a voltage assignment that results in a known delay and its power. The *sum* function adds these curves to generate *arrivalInf* of the node. *ArrivalInf* signifies the arrival delay at the output of the corresponding node/operation and the total power dissipation of the subgraph rooted at that node. Each combination of *arrivalMax* and *nodeInf* will be added to generate a point on the *arrivalInf* of the node. Therefore *arrivalInf* is in fact the convolution of the two curves. This is computed for all nodes in forward topological order. By the end, by looking at the solutions stored at the primary output(s), the one with minimum power will be selected as the best solution. Note that the best solution is in a compact form reflecting the overall power of all nodes. A reverse topological traversal is necessary to specify the delay associated with each node.

3.1.2 *The Backward Pass.* In this stage, the DFG is traversed from PO to PI in topological order. At each step, the exact voltage and architecture decisions of the operations are made. At each node, the indices of *nodeInf* specify the solution for that node. The indices from *arrivalMax* specify the referring indices for the fanins of the node. Therefore the reverse topological traversal ensures that in the end at the PI the assignment for all operations has been done and the solution is complete. This approach will be optimal if the DFG is a tree. If the DFG is not a tree, in the backward pass, more than one choice exists for nodes with more than one fanout. Heuristics can then be used to make a good choice. Chang and Pedram [1997] discussed the use of level converters to maintain the integrity of the signal.

Now that the traditional approach has been explained, in the next section the risk management alternative will be presented.

3.2 Risk Management for Voltage Scheduling

Referring back to Section 2, we propose a new voltage scheduling methodology to address unpredictabilities. The previous algorithm assumed accurate information about delay/voltage and power/voltage variations. This is definitely not a realistic assumption. Figure 1 illustrates that, instead of having a fixed delay and power value for each voltage, we might have a distribution (the dotted distribution for each voltage). In this case, the traditional assumption of having accurate estimates made by the optimization algorithm is completely wrong. Let us suppose that somehow we know the distribution (range of values with associated probabilities) for both delay and power at each voltage. Let us assume that each distribution has a range of interest. The likelihood of the value falling outside this range is very low (typically within a distance of $3 \times$ variance from the expected value). With this information, we would like to redefine our objective as follows:

(1) Assume the delay/voltage and power/voltage curves for all architectures for each operation are provided such that for each voltage choice we also have the distribution of delay and power. A delay constraint of C clocks and a risk factor R ($R \geq C$) are also provided.

(2) The objective is to minimize the expected value of power such that the expected value of delay $\leq C$ clocks and the worst case delay of the associated delay distribution is always less than the risk R.

Since we add the power dissipation of all nodes in the final solution, replacing the power estimate for each operation by its expected value should be enough. This is because the expected value of a sum of n variables is the sum of the expected values. Handling delay constraint and delay risk requires a sophisticated algorithm which will be described later.

3.2.1 *Estimating Operation Unpredictabilities.* An important question is: "How do we know the distribution in delay and power for each voltage?" This is a very tricky problem. The basic premise of this work is that estimating distributions is easier than estimating exact values. Distributions will strongly depend on the kind of optimizations the subdesign will be subjected to in future, which in turn will depend on how critical a certain objective function becomes in that subdesign. It also depends on the *sensitivity* between different cost functions. A complex estimation engine that has models for tools (like gate sizing, buffer insertion) and not just libraries, along with consideration of *sensitivity*, will be needed. Such an estimation engine will take a subdesign and, depending on the criticality of different constraints, generate a range of values for each design objective. Of course, to the best of the authors' knowledge, *no such estimation system exists*, although development of such a system is underway. Any further discussion of unpredictability estimation is beyond the scope of this work. The rest of this article assumes that these ranges are available as input.

3.2.2 *Algorithms for Risk Management.* Once again, we are given delay distributions for all voltages. Let us suppose that the delay distributions are provided in terms of clock steps (the general delay distribution can be easily transformed to generate this if the clock frequency is given).

Hence we know, for each operation and each modular architecture for that operation, the expected value, maximum value, and distribution for delay in clocks for each prespecified voltage. The problem is to assign voltages and architectures to operations such that the new objective enumerated above can be satisfied. Once again the algorithm contains two passes over the DFG. The forward pass traverses DFG in forward topological order and the reverse pass in reverse order.

In regard to *the forward pass*, before the forward pass starts, some preprocessing is performed. For each node, we first generate a double dimensional array which stores relevant information about that node and is referred to as $nodeInf_n$ for a node n. $NodeInf_n$ is essentially some way of representing the power and delay distributions for each voltage in a compact form. This is done through indexing the array in terms of the expected delay and the taken risk. In this array, the rows correspond to the expected delay, ranging from 1 to C. The columns signify the taken risk, ranging from 1 to R. The values stored at the (i, j) cell signify the minimum (expected) power solution with expected delay of i clocks and max delay (or risk) of j clocks. This is stored in the power field of each cell ($nodeInf_n[i][j].expPow$). Note that since, in the final objective, the expected value of power is minimized, we are storing the expected value of each power distribution in the power field.

The associated probability distribution of delay (which has an expected value of i clocks) is also stored in $nodeInf_n[i][j].prob$. We also remember the voltage and architecture that result in this solution. Note that this data is computed independently without any consideration in the inputs of these operations.

Now we proceed with the forward topological traversal of the DFG. At each node, we compute another double dimensional data called $arrivalInf_n$ with the same rows and columns as $nodeInf$. $arrivalInf_n$ essentially represents the times (resulting from different combinations of delays of previous nodes) that the signal from node n becomes valid and available. Therefore from the definition it can be concluded that $arrivalInf_n$ should include all different delay combinations of the subgraph rooted at n to be able to reflect the arrival time of the signal for n. It should be clear that if node n is a primary input then $nodeInf_n$ and $arrivalInf_n$ are the same. The (i, j) location of $arrivalInf_n$ contains the solution with minimum (expected) power dissipation of the subgraph rooted at node n and expected arrival delay of exactly i with max delay (risk) of exactly j clocks.

The term arrival delay signifies the number of clock cycles it would take for the data of n to become available. Conceptually it is similar to arrival time in gate-level circuits. $arrivalInf_n[i][j].expPow$ stores the power value and $arrivalInf_n[i][j].prob$ stores the distribution in arrival delay. If a node n is not a primary input then the computation of $arrivalInf_n$ is more involved.

Let us point out the following: the arrival time for a node is defined as

$$\begin{aligned} arrivalInf_n[i][j].prob \;=\; & max(arrivalInf_k[i'][j'].prob|k \\ & \in Fanin(n)) + nodeInf_n[i''][j''].prob \end{aligned} \tag{5}$$

Here $arrivalInf_n[i][j].prob$ corresponds to one of the delay distributions stored at $arrivalInf_n$. Similarly $nodeInf_n[i''][j'']$ corresponds to a possible delay distribution for node n stored at $nodeInf_n$. Here all fanins of node n should be considered.

Equation (5) states that, given a combination of arrival delay distributions for fanins of a node n and a delay distribution for the node itself, the arrival distribution for the node will be the maximum of the arrival distributions of its fanins summed with the delay of the node. Note that Equation (5) should be considered for all valid combinations of fanins and node delay distributions (all valid i, j, i', j', i'', j''). Recall the definitions of the rows and columns of the arrays. A row indexed by i indicates an expected arrival time of i clock steps. A column indexed by j indicates an exact risk of j units for the corresponding stored solutions. Based on these definitions, the following conditions should hold for any valid combination:

—$i' \leq i$ and $i'' \leq i$ to ensure that the expected arrival of solution stored at row i is by clock step i;

—$j' + j'' = j$ to ensure all solutions stored at column j have a risk equal to j.

Algorithm 1. MAX-PROB(P1,P2)

```
INPUTS: P1,P2: Input delay distributions
OUTPUT: P : Distribution which is Max(P1,P2)
comment: P1 and P2 are bounded by R, the maximum risk in delay
for (i =1; i ≤ R; i++)
   P[i] = 0;
   for (j = 1; j ≤ i; j++)
          P[i] + = P1[i] * P2[j]
   for (j = 1; j ≤ i; j++)
          P[i] + = P2[i] * P1[j]
return P
end
```

Algorithm 2. ARRIVAL-MAX(n1,n2)

```
INPUTS: Operations n1,n2
OUTPUT: arrivalMax(Max(n1,n2))
comment: arrivalInf(n1)and arrivalInf(n2) has been computed
Allocate Memory for Temp
for (i = 1; i ≤ C; i++)
        for (j = 1 j ≤ R; j++)
          d for (k = 1; k ≤ C; k++)
              for (l = 1; l ≤ R; l++)
Temp[i,j,k,l].prob =
MAX-PROB(arrivalInf[n1][i][j].prob,arrivalInf[n2][k][l].prob)
Temp[i,j,k,l].expPow =
arrivalInf[n1][i][j].expPow + arrivalInf[n2][k][l].expPow
for (i = 1; i ≤ C; i++)
        for (j = 1; j ≤ R; j++)
              Find k,l,m,n such that
              EXPECT(Temp[k,l,m,n].prob) = i and
              MAX(Temp[k,l,m,n].prob) = j, and with minimum power.
              arrival-max[i][j].prob = Temp[k,l,m,n].prob
              arrival-max[i][j].expPow = Temp[k,l,m,n].expPow
return arrival-max
end
```

We need to first compute the *max* of all fanins probabilistically and add it with the delay of the node. Since we are traversing the DFG topologically, the $arrivalInf_n$ for all fanins is known. The computation of *max* is illustrated in Algorithms 1. Algorithm 1 describes standard *max* between two probability distributions. Algorithm 2 merges the *arrivalInf* of two fanins using the *max* function. Note that *arrivalInf* of each fanin contains distributions reflecting different scenarios where the signal becomes available from the fanin output. Therefore, Algorithm 2 uses Algorithm 1 to combine two sets of distributions corresponding to two fanins to generate a new set of distributions reflecting the arrival time of the two nodes. For all possible combinations of arrival delay distributions at the outputs of fanins, the algorithm calls MAX-PROB, which computes the max.

Out of all these combinations, the ones with minimum (expected) power are chosen. In other words, if two resulting distributions have the same expected delay and risk factor indices, the one that has the minimum expected power will be stored. This data is stored in a temporary *arrivalMax* array.

Algorithm 3. SUM-PROB(P1,P2)

```
INPUTS: P1,P2: Input distributions
OUTPUT: P : Distribution which is SUM(P1,P2)
comment: P1 and P2 are bounded by R, the maximum risk
comments: P can be more than R
for (i = 1; i ≤ R; i++)
      for (j = 1; j ≤ R; j++)
            P[i + j] + = P1[i] * P2[j]
return P
end
```

Algorithm 4. arrivalInf(n)

```
INPUTS: n
OUTPUT: arrivalInf for n
compute arrival-max for n using Algorithm-2
for (i = 1; i ≤ C; i++)
    for (j = 1 j ≤ R; j++)
       for (k = 1; k ≤ C; k++)
           for (l = 1; l ≤ R; l++)

Temp[i,j,k,l].prob =
SUM-PROB(arrival-max[n][i][j].prob,nodeInf[n][k][l].prob)
Temp[i,j,k,l].expPow =
arrival-max[n][i][j].expPow + nodeInf[n][k][l].expPow
for (i = 1; i ≤ C; i++)
    for (j = 1; j ≤ R; j++)
          Find k,l,m,n such that
          EXPECT(Temp[k,l,m,n].prob) = i and
          MAX(Temp[k,l,m,n].prob) = j, and with minimum power.
          arrivalInf[i][j].prob = Temp[k,l,m,n].prob
          arrivalInf[i][j].expPow = Temp[k,l,m,n].expPow
return arrivalInf
end
```

Finally, we have one *arrivalMax* data structure which contains the first term of Equation (5). This needs to be added to the delay of the node n in order to compute *arrivalInf*$_n$. This again needs to be done probabilistically. This procedure is illustrated in Algorithm 4, which basically describes the convolution of two distributions which corresponds to their additions. All possible delay solutions of an operation are stored in $nodeInf$. This is merged with the computed *arrivalMax* using a probabilistic addition function (Algorithm 3). Finally, for each expected delay i and risk j, the solution with minimum power is stored.

After forward traversal, the *arrivalInf* for all the primary outputs is investigated to select the solution with minimum power.

In regard to *the backward pass*, after reaching the PO of the DFG, we choose the solution with the minimum power dissipation. This corresponds to a certain expected clock value $\leq C$ and a certain risk $\leq R$. Taking this solution, we traverse the DFG in topological order from PO to PI. At each step the fanout of the pertinent node forces a certain expected delay and risk factor ((i, j) location in *arrivalInf*$_n$). This corresponds to a certain architecture and voltage for the node n and certain expected and max delay values for the fanins. In this fashion, the voltage and architecture for all the operations can be determined.

If the DFG is not a "tree," then there will be nodes with more than one fanout. Therefore in the backward traversal the solution of such node can be potentially decided by any of its fanouts, and this might result in a suboptimal solution. This corresponds to several choices of (i, j) locations in $arrivalInf_n$ imposed by different fanouts. We solve this deadlock by accepting the solutions in the following priority order E, W. The first priority E selects the selects the solution with minimum row index i. This corresponds to the solution with the minimum expected value of arrival time at n. If E is the same for two solutions, we choose the option that has the smallest worst-case arrival delay W (or column index j). In this way, we try to ensure that the expected value constraint of the delay can be satisfied with a higher priority than the user-specified risk constraint.

The algorithm for voltage scheduling with risk management generates an optimal solution for tree-like DFGs. In the forward pass, definitely the minimum expected power can be verified due to the dynamic programming approach. If the DFG is tree-like in the backward pass, this minimum solution can be exactly verified by verifying the solution of each node from its *one* fanout.

3.2.3 *Postscheduling Resource Binding.* We believe that resource binding can be conducted in a similar fashion as discussed in Chang and Pedram [1997]. Another aspect ignored in the described algorithm is the use of level converters to enable communication between different voltage levels. This algorithm can be trivially extended to consider level converters. This can be incorporated into Algorithm 4. When adding the power of two subsolutions, an extra term for level converter power can be added.

4. RESULTS

The primary objective of this article is to propose the idea of risk management. The basic philosophy is to have a user-controlled parameter, which we call *risk*, which controls the amount of possible risk in meeting the constraints at the penalty of design quality.

There are a few things that we want to demonstrate through our experiments. First, in the presence of unpredictabilities, traditional design methodology (voltage scaling in this case) results in invalid solutions. Second, we want to demonstrate how the penalty in design quality (power in this case) varies as the risk factor is varied.

In our experiment to compute the power and delay distributions, we assumed the existence of four distinct voltages (5, 3.3, 2.4, 1.5). The estimated power and delay at these voltages were then calculated using Equations (1) and (2). Now each delay/power estimated should be assigned an associated unpredictability/distribution. We assume that, at each specific voltage, the distribution of delay and power follows a Gaussian distribution with the value predicted by Equations (1) and (2) as the mean. The variance was set as 20% of the mean. In reality, these values should be generated by an unpredictability estimation engine. Since no such system exists, we had to make this assumption. Now for large data sets, the pertinent statistics are usually Gaussian. Hence the Gaussian assumption should be accurate. But this

Table I. Comparison of Traditional Versus Risk Management Approaches

Bench	T-Clk(ns)	Traditional			Risk Management		
		Const.	Risk	Power	Expect.const.	Risk	Power
dct	8	25	50	212.63	25	25	301.35
ecbenc-4	8	25	50	128.65	25	25	225.92
ellipt	12	40	80	356.29	40	40	523.50
fft2	8	15	30	319.11	15	15	543.00
fir	8	35	70	157.93	35	35	197.00
jdmer-4	8	20	40	253.26	20	20	429.00
jdmer-3	8	8	20	399.86	8	8	472.78
jdmer-1	8	8	20	381.40	8	8	441.50
motion-2	8	14	30	404.00	14	14	655.38
motion-3	8	14	30	404.00	14	14	655.00
noise-2	8	8	20	773.21	8	8	837.44

needs to be experimentally validated. Proceeding further, we calculated the expected value for each delay/power and gave that as the estimate for the traditional algorithm proposed by Chang and Pedram [1997]. Then we used our proposed algorithm to generate a solution in the risk management paradigm.

Table I illustrates the comparison between the traditional and risk-driven approaches. The benchmarks are a mix of MediaBench [Lee et al. 1997] and traditional high-level synthesis bench-set. For the traditional approach, the provided delay constraint is reported in column 3. The power of the solution generated by the traditional approach is reported in column 5. Once the traditional solution is generated, the risk associated with it is evaluated, and is reported in column 4. As can be seen, this risk is very high compared to the input delay constraint. For the risk management approach, columns 6 and 7 report the expected value of delay constraint and maximum allowed risk provided as input. The expected value of delay constraint is the same as the delay constraint in the traditional solution. However, the maximum allowed risk is given as an input equal to the delay constraint. This means the designer is not willing to take any risk on timing constraint. It can be seen that the risk-driven approach always results in a valid solution, for which the power is reported in column 8.

If the designer is not willing to risk (as in the risk management case), the result of the traditional approach should be considered as invalid. This illustrates the superiority of our approach over the traditional one.

The next set of experiments illustrate the variation of design quality (power) as the risk is changed. Hence, without changing the expected value of timing, we illustrate the variation of power as the risk changes. Figure 2 reports this data for fft2 (the rest are omitted due to space considerations) for various expected delay constraints. It can be seen that as the risk factor is increased (which means that the designer can tolerate more risk on the expected delay constraint), the power dissipation reduces. Hence there is clearly a risk/design quality tradeoff. It can also be seen that a small increase in the risk factor tremendously affects the design quality, especially in Figure 2.

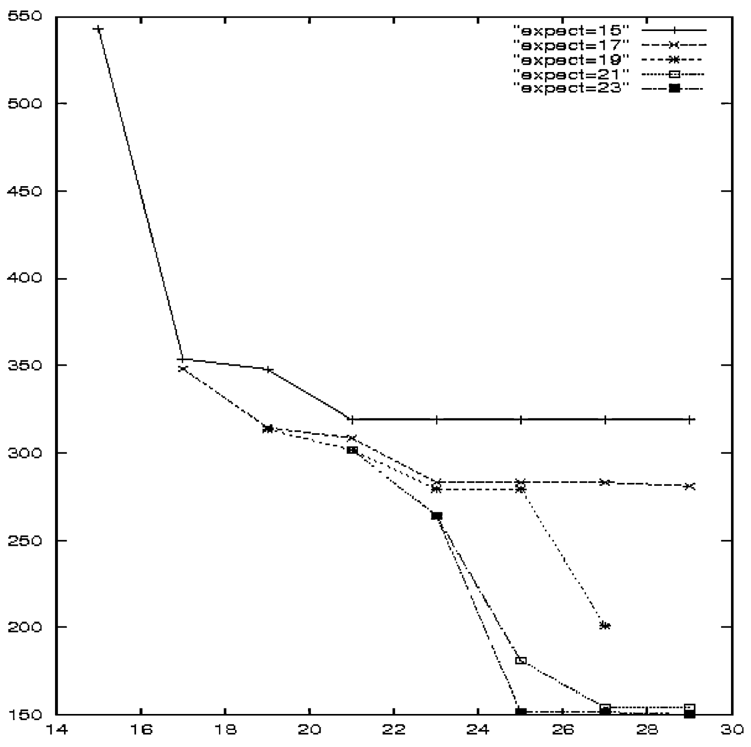

Fig. 2. FFT2: X-axis risk, Y-axis power.

5. CONCLUSION AND FUTURE WORK

This research proposes a formal methodology of addressing unpredictabilities at the system level. The idea is to associate a risk factor with each constraint, and by controlling the risk factor an appropriate solution that is guaranteed to be within the prescribed limits can be generated. This was demonstrated using the voltage scheduling problem. The delay constraint was assigned a risk factor and an optimal algorithm was proposed which generates the minimum power dissipating solution within the clock and risk constraints. This illustrates a formal way of handling unpredictabilities. Our future work will include the development of an unpredictability estimation engine. Development of a formal synthesis system that addresses unpredictabilities is also underway.

REFERENCES

CHANDRAKASAN, A. P., SHENG, S., AND BRODERSEN, R. W. 1992. Low power CMOS digital design. *IEEE J. Solid State Circ. 27*, 4 (April), 472–484.

CHANG, J. M. AND PEDRAM, M. 1995. Low power register allocation and binding. In *Proceedings of the Design Automation Conference*. 29–35.

CHANG. J. M. AND PEDRAM, M. 1997. Energy minimization using multiple supply voltages. *IEEE Trans. VLSI Syst. 5*, 2.

CHEN, C., SRIVASTAVA, A., AND SARRAFZADEH, M. 2001. On gate level power optimization using dual suppply voltages. *IEEE Trans. VLSI Syst. 9*, 5 (Oct.), 616–629.

CHUNG, E. Y., BENINI, L., AND DEMICHELI, G. 1999. Dynamic power management using adpative learning tree. In *Proceedings of the International Conference on Computer Aided Design*.

JYU, H.-F. AND MALIK, S. 1993. Statistical timing optimization of combinational logic circuits. In *Proceedings of the International Conference on Computer Design*. 77–80.

JYU, H.-F. AND MALIK, S. 1994. Statistical delay modeling in logic design and synthesis. In *Proceedings of the Design Automation Conference*. 126–130.

KRUSE, L., SCHMIDT, E., JOCHENS, G., STAMMERMANN, A., SCHULZ, A., MACII, E., AND NEBEL, W. 2001. Estimation of lower and upper bounds on the power consumption from scheduled data flow graphs. *IEEE Trans. VLSI Syst. 9*, 1 (Feb.), 3–14.

LEE, C., POTKONJAK, M., AND MANGIONE-SMITH, W. H. 1997. MediaBench: A tool for evaluating and synthesizing multimedia and communications systems. In *Proceedings of the International Symposium on Microarchitecture*.

PEDRAM, M. 1996. Power minimization in IC design: Principles and applications. *ACM Trans. Des. Automat. Electron. Syst. 1*, 1 (Jan.), 3–56.

RAJE, S. AND SARRAFZADEH, M. 1995. Variable voltage scheduling. In *Proceedings of the International Workshop on Low Power Design*.

SRIVASTAVA, A. AND SARRAFZADEH, M. 2002. Predictability: Definition, analysis and optimization. In *Proceedings of the International Conference on Computer Aided Design*.

TELLEZ, G. E., FARRAHI, A., AND SARRAFZADEH, M. 1995. Activity-driven clock design for low power circuits. In *Proceedings of the IEEE International Conference on Computer Aided Design*. 62–65.

TOMIYAMA, H., INOUE, A., AND YASUURA, H. 1998. Statistical performance driven module binding in high level synthesis. In *Proceedings of the International Symposium on System Synthesis*. 66–71.

TOMIYAMA, H. AND YASUURA, H. 1998. Module selection using manufacturing information. In *IEICE Trans. Fund. E81-A*, 12 (Dec.), 2576–2584.

Received July 2003; revised March 2004; accepted January 2005