A LOW POWER SCHEDULING SCHEME WITH RESOURCES OPERATING AT MULTIPLE VOLTAGES

Ali Manzak and Chaitali Chakrabarti

Department of Electrical Engineering
Telecommunications Research Center
Arizona State University
Tempe, AZ 85287-5706
istanbul@asu.edu, chaitali@asu.edu

ABSTRACT

This paper presents a new resource constrained *and* latency constrained scheduling scheme that minimizes power consumption for the case when the resources operate at multiple voltages. The scheduling scheme is list based. In each control cycle, the ready nodes are assigned depending on the number of available resources and the difference between the actual number of cycles left and an estimate of the number of cycles required to schedule the remaining nodes. The switching activity of the nodes as well as the priority (which is related to the capacitance) of the nodes is taken into account. We assume that the resources can be operated at 5V, 3.3V, 2.4V and 1.5V. Moreover, we assume that the resources (multipliers and adders) have different voltage-delay curves. Experiments with some HLS benchmark examples show that the proposed scheme achieves significant power reduction. For instance, when the latency constraint is 1.5 times the tight latency constraint, the average reduction is 59.1%.

1. INTRODUCTION

Power considerations have become an increasingly dominant factor in the design of both portable and desk-top systems. An effective way to reduce power consumption is to lower the supply voltage level for a circuit. Reducing the supply voltage, however, increases the circuit delay and reduces the throughput. To maintain the throughput, parallelism and/or pipelining has to be incorporated [1]. The resulting circuit consumes lower average power while meeting the global throughput constraint at the cost of increased circuit area. Another way of maintaining the throughput is to use resources operating at multiple voltages [2], [3], [4], [6], [8], [7]. This has the advantage of allowing modules on the critical paths to be assigned to the highest voltage levels (thus meeting the required timing constraints) while allowing modules on non-critical paths to be assigned to the lower voltages (thus reducing the energy consumption). This scheme tends to result in smaller area overhead compared to parallel architectures.

In this paper we address the problem of scheduling a dataflow graph under resource *and* latency constraint for the case when the resources operate at multiple voltages. We assume that the resources respect different voltage-delay curves. The scheduling algorithm minimizes power consumption for the case when the resources operate at multiple voltages (5V, 3.3V, 2.4V and 1.5V). It is worth mentioning that the operating voltages do not have to be restricted to 5V, 3.3V, 2.4V and 1.5V. In fact, the resources can operate on any voltage for which the corresponding delay is known. The proposed algorithm is list based, where in each control cycle, the node assignment is a function of the number of available resources and the difference between the actual time available to finish the computation and an estimate of the time required to schedule the remaining nodes. Simplistically, if this difference is large, the nodes are assigned to low voltage resources while if the difference is small, the nodes are assigned to high voltage resources. The switching activity of nodes is taken into account. If two nodes have the same freedom, then the node with the lower switching activity is assigned to the high voltage resource. Finally the nodes are prioritized based on the capacitance of the resource it gets assigned to. For instance, multipliers have higher priority compared to adders. Experiments with some HLS benchmarks (DIFFEQ, AR lattice and EW filter) show the effectiveness of these approaches in reducing power. When the latency constraint is tight [1], the average reduction is 32.1% while when the latency constraint is 2 times the tight latency constraint, the average reduction is 66.8%. The computational complexity of our algorithm is $O(r^2L^2)$, where r is the number of resources and L is the latency.

There are several scheduling algorithms for multiple voltage resources in the literature today [2]-[8]. These algorithms can be classified into (i) *only* latency-constrained (i.e., latency is a hard constraint and resources are minimized) [3], [4], (ii) *only* resource-constrained (i.e., resource is a hard constraint) [5], and (iii) latency *and* resource constrained (i.e., both latency and resource are hard constraints) [6],[7]. While the (only) latency-constrained and (only) resource-constrained algorithms have polynomial or pseudo-polynomial time complexity, the latency *and* resource-constrained algorithms are based on integer linear programming and have (worst case) exponential time complexity. In this paper, we propose a heuristic algorithm for latency *and* resource-constrained scheduling that produces

[1] Tight latency constraint corresponds to the minimum time required to schedule all the nodes of the DFG.

comparable results with only polynomial time complexity.

The rest of the paper is organized as follows. Section 2 deals with the preliminaries. Section 3 presents the algorithm. Section 4 includes the results on some HLS benchmark examples. Section 5 concludes the paper.

2. PRELIMINARIES

The input to our algorithm is a data flow graph, a timing constraint, a resource constraint, where the resources operate at multiple voltages, and the switching activity of the nodes. The timing constraint (also referred to as the latency constraint $= L$) is the maximum time to finish the computation. The switching activity is the probability that a power-consuming transition will occur during a single data period. The resource constraint is specified as follows.

1. The number of computational units for each type is given. Each computational unit can only be operated at a specific voltage. In other words, the same computational unit cannot be operated at different voltages.
2. The voltage delay curve of each type of computational unit is given.
3. The number of level shifters is not user defined. (The proposed algorithm tries to reduce the number of level shifters.)

3. RESOURCE AND LATENCY CONSTRAINED SCHEDULING

The proposed resource and latency constrained algorithm is list based. In each control cycle, a parameter called the latency freedom, $k(i)$, is computed for each type of resource. This parameter is used to decide whether the resource will be assigned to an available low voltage resource or a high voltage resource.

A summary of the main algorithm is given below.

For control cycles 1 through L=latency constraint, do

Step 1: Make/Update the ready set table.

Step 2: Calculate $k_j(i)$ for each type of node.

Step 3: Use $k_j(i)$ to schedule the functional units according to the *latency freedom* scheduling algorithm described in Section 3.2.

3.1. Calculation of $k_j(i)$

We next describe the steps to calculate $k_j(i)$. This involves calculating the following parameters.

- $C_j(i)$: the minimum number of control cycles that are required to schedule all the nodes of type j in cycle i.
- $C_{next}(i)$: an estimate of the number of control cycles that are required to schedule all the nodes in the ready sets of cycles $i+1$ through L.
- $k_j(i)$: the mobility or the freedom of a node of type j in ready set i.

Step A. Calculation of $C_j(i)$:

Let $R_{j,X}$ be the number of resources of type j and $D_{j,X}$ be the delay of such a resource operating at X volts. Then $N_j(i)$ (the number of nodes of type j in cycle i) and $C_j(i)$ are linked as follows.

$$N_j(i) = \lfloor \frac{C_j(i)}{D_{j,1.5}} \rfloor * R_{j,1.5} + \lfloor \frac{C_j(i)}{D_{j,2.4}} \rfloor * R_{j,2.4} + \lfloor \frac{C_j(i)}{D_{j,3.3}} \rfloor * R_{j,3.3} + \lfloor \frac{C_j(i)}{D_{j,5}} \rfloor * R_{j,5}$$

So given $N_j(i)$, $C_j(i)$ can be easily calculated.

Step B. Calculation of $C_{next}(i)$:

To compute $C_{next}(i)$, we first calculate for each of the remaining cycles ($i+1$ through L), an upper bound, $max_j\{C_j(k)\}$ on the number of cycles that would be required to schedule all the nodes in that cycle. Next, we sum up these upper bounds over cycles $i+1$ through L to get $C_{next}(i)$.

$$C_{next}(i) = \sum_{k=i+1}^{L} (max_j\{C_j(k)\})$$

Calculation of $k_j(i)$: The latency freedom value is calculated differently for different type of resources (in this case, multipliers and adders). This is done so that multiplication nodes have higher $k(i)$ values than addition nodes and thus can be assigned to lower voltage resources. The justification for this is that multipliers have larger capacitance compared to adders and this greater power savings can be obtained by assigning more multiplication nodes to low voltage resources.

For multipliers, $k_M(i) = C_R(i)$-$C_{next}(i)$, where $C_R(i) = L - i + 1$. For addition nodes, the procedure is as follows.

if $(\sum_{j=i}^{L}(N_M(j)) > 0)$ and $(N_M(i) = 0)$ and $(k_M(i) > C_A(i))$
 $k_A(i) = C_A(i)$
else
 $k_A(i)$=$C_R(i)$-$C_{next}(i)$

3.2. Latency Freedom Scheduling Algorithm:

In this algorithm, the *latency freedom* parameter $k_j(i)$ is used to determine the way the nodes of type j in the ready set of cycle i are scheduled. The algorithm tries to schedule all the ready nodes $N_j(i)$ in $k_j(i)$ cycles giving first priority to the 1.5V resources, the second priority to the 2.4V resources, the third priority to the 3.3V resources and the fourth priority to the 5V resources. If the freedom of the nodes is the same, then the nodes with the high (low) switching activity are assigned to the low (high) voltage resources.

Algorithm:

$AVAIL_{j,X}(l)$ is the number of available control cycles in cycles i through $i + k_j(i) - 1$, for the l^{th} resources of type j at X volts. $N_{j,X}$ is the number of nodes of type j that can be scheduled to $R_{j,X}$ in cycles i through $i + k_j(i) - 1$. $N_{j,X} = 0$ for all X at initialization.

1. /*calculate the number of nodes that can be assigned to 1.5V resources.*/

for($l = 1; l \leq R_{j,1.5}; l++$)

$N_{j,1.5} = N_{j,1.5} + \lfloor \frac{AVAIL_{j,1.5}(l)}{D_{j,1.5}} \rfloor$

if ($N_j(i) \leq N_{j,1.5}$)

go to step 6

2. /*if the number of nodes is larger than what can be assigned to 1.5V resources, calculate the number of nodes that can be assigned to 2.4V resources.*/
for($l = 1; l \leq R_{j,2.4}; l++$)
$N_{j,2.4} = N_{j,2.4} + \lfloor \frac{AVAIL_{j,2.4}(l)}{D_{j,2.4}} \rfloor$
if ($N_j(i) \leq N_{j,1.5} + N_{j,2.4}$)
$N_{j,2.4} = N_j(i) - N_{j,1.5}$
go to step 6

3. /*if the number of nodes is larger than what can be assigned to 1.5V and 2.4V resources, calculate the number of nodes that can be assigned to 3.3V resources.*/
for($l = 1; l \leq R_{j,3.3}; l++$)
$N_{j,3.3} = N_{j,3.3} + \lfloor \frac{AVAIL_{j,3.3}(l)}{D_{j,3.3}} \rfloor$
if ($N_j(i) \leq N_{j,1.5} + N_{j,2.4} + N_{j,3.3}$)
$N_{j,3.3} = N_j(i) - N_{j,1.5} - N_{j,2.4}$
go to step 6

4. /*if the number of nodes is larger than what can be assigned to 1.5V, 2.4V and 3.3V resources, calculate the number of nodes that can be assigned to 5V resources.*/
for($l = 1; l \leq R_{j,5}; l++$)
$N_{j,5} = N_{j,5} + \lfloor \frac{AVAIL_{j,5}(l)}{D_{j,5}} \rfloor$
if ($N_j(i) \leq N_{j,1.5} + N_{j,2.4} + N_{j,3.3} + N_{j,5}$)
$N_{j,5} = N_j(i) - N_{j,1.5} - N_{j,2.4} - N_{j,3.3}$
go to step 6

5. /* if none of the other conditions satisfy (The possibility of infeasibility)*/
$k_j(i) = C_j(i)$
go to step 1

6. /* Schedule the nodes.*/

   (a) First, order the ready nodes with respect to freedom.

   (b) Then schedule $\min(R_{j,5}(i), N_{j,5})$ nodes with the lowest freedom to the resources operating at 5V.

   (c) Then schedule $\min(R_{j,3.3}(i), N_{j,3.3})$ nodes with the lowest freedom to the resources operating at 3.3V.

   (d) Then schedule $\min(R_{j,2.4}(i), N_{j,2.4})$ nodes with the lowest freedom to the resources operating at 2.4V.

   (e) Then schedule $\min(R_{j,1.5}(i), N_{j,1.5})$ nodes with the lowest freedom to the resource operating at 1.5V.

   (f) If the freedom of the nodes are the same, assign the node with the high switching activity to the low voltage resource and assign the node with the low switching activity to the high voltage resource.

   (g) In case of schedule conflict, assign the children of the nodes that have been scheduled in the previous control cycle to the same voltage resource (This is an attempt to reduce the number of level shifters).

3.3. Example

Figure 1. Data flow graph of the example

The energy and delay values used here have been obtained from [4]. We list these values for some key modules in Table I. For $t_c = 20ns$, $D_{M,5} = 5$,$D_{M,3.3} = 9$, $D_{M,2.4} = 15$, $D_{M,1.5} = 36$, $D_{A,5} = 1$, $D_{A,3.3} = 2$, $D_{A,2.4} = 3$, $D_{A,1.5} = 8$ (see Table I).

The hardware constraint is $R_{M,5} = 1$, $R_{M,3.3} = 1$, $R_{M,2.4} = 1$, $R_{M,1.5} = 1$, $R_{A,5} = 1$, $R_{A,3.3} = 1$, $R_{A,2.4} = 1$ and $R_{A,1.5} = 1$. Switching activities at all nodes were assumed to be 0.5 in figure 1. In other words the probability of a transition on any selected 1 bit signal is 50 % in any one sample interval. $T_{comp} = 500ns$ which implies that $L = \lfloor \frac{500}{20} \rfloor = 25$. The final schedule is as follows.

3.4. When the given latency is less than the tight latency constraint

When the given latency is less than the tight latency constraint, i.e. if $L < \sum_{j=1}^{L} max(C_M(j), C_A(j))$, we can not guarantee a schedule with the late ncy bound. In this case, $k_j(i) = 0$ and algorithm schedule the ready nodes as soon as possible. In the following cycle latency freedom value $k_j(i)$ might be bigger than 0. When this is happens, there is a schedule exist given latency and resource constraint and algorith schedule the nodes considering to decrease power consumption. If $k_j(i)$ is never bigger than 0, DFG cannot be scheduled given latency bound, algorithm gives the minimum latency for this DFG.

3.5. Complexity Analysis

In this section we show that the worst case complexity of this algorithm is $O(r^2L^2)$, where r is the number of resources and L is the latency. The complexity of this algorithm is dominated by Step 6 of the latency freedom scheduling algorithm. In this step (in the worst case),

Table 1. E is energy (in pJ) and D is normalized delay for switching activity of 0.5 as in the modules in [4]

all the ready nodes are sorted and r nodes chosen. Let n_i be the number of nodes that are scheduled in cycle i ($\sum_{i=1}^{L} n_i = n$). In the worst case, in cycle 1, n nodes are ready, in cycle 2, $n - n_1$ nodes are ready and so on. So the total number of operations is

$nr + (n - n_1)r + (n - n_1 - n_2)r + \ldots + (n - n_1 - \ldots - n_{L-1})r$
$= r(\sum_{i=1}^{L} n_i + \sum_{i=2}^{L} n_i + \ldots + \sum_{i=L}^{L} n_i)$
$= r(n_1 + 2n_2 + 3n_3 + \ldots + Ln_L) = r\sum_{i=1}^{L} in_i$

Since $max(n_i) = r$, the above summation is $r^2 L(L+1)/2$. The complexity of this algorithm is thus $O(r^2 L^2)$. Note that, switching activity of the nodes can be calculated based on behavioral simulation results and it is not taken into account for the complexity analysis of the algorithm.

4. RESULTS

In this section we present the results obtained by running our algorithm on some high-level synthesis benchmarks. We present the results when actual energy consumption values in [4] are used. In all these examples, the switching activity of the nodes is assumed to be 0.5.

In Table III, E1 is the total energy consumed in the hardware resources (and not the interconnect) for the case when all the resources operate at 5V and there is no resource constraint. E2 is the energy consumed (including the level shifter power) for the case when the resource constraint is $R_{M,5} = 1$, $R_{M,3.3} = 1$, $R_{M,2.4} = 1$, $R_{M,1.5} = 1$, $R_{A,5} = 1$,

Table 2. Final schedule of the example

$R_{A,3.3} = 1$, $R_{A,2.4} = 1$ and $R_{A,1.5} = 1$. E_{LS} is the energy consumed by the level shifters. The % reduction is $(E1 - E2)/E1$. The results for each benchmark are presented for three timing constraints. The entry with a * corresponds to tight timing constraint (i.e. the minimum time required to schedule all the nodes of the DFG). The second entry corresponds to the case when the timing constraint is 50% larger than the tight timing constraint and the third entry corresponds to the case when the timing constraint is 100% larger than the tight timing constraint. Note that the power consumed by the level shifters is significant. This is the price for giving multiplication nodes a higher scheduling priority. For DFGs where the multiplication and addition nodes alternate, the multiplication nodes are assigned to low voltage resource while the addition nodes are assigned to high voltage resource resulting in level shifters being required between multipliers and adders.

5. CONCLUSION

In this paper, we present a new scheduling scheme under resource and latency constraint that minimizes power consumption for the case when the resources operate at multiple voltages. The worst case running time of our algorithm is $O(r^2 L^2)$, where r is the number of resources, and L is the latency. The proposed scheme considers different voltage-delay curves for the different resources, gives higher prior-

Table 3. Energy consumption of some benchmark examples when actual energy consumption values from [4] are used.

ity to multiplication nodes (compared to addition nodes), considers switching activity of the nodes and takes into account the power consumption of the level shifters. For tight latency constraint, the average reduction obtained by our algorithm was 32.1%. When the latency constraint was 1.5 times the tight latency constraint, the average reduction was 59.1% while, when the latency constraint was 2 times the tight latency constraint, the average reduction was 66.8%. Thus we conclude that significant power savings can be obtained by using resources operating at different voltages.

REFERENCES

[1] A. P. Chandrakasan and R. W. Brodersen, “Minimizing power consumption in digital CMOS circuits,” Proceedings of the IEEE vol83, 1995, pp. 498-523.

[2] K. Usami and M. Horowitz, “Clustered voltage scaling technique for low power,” Proc. of the Int. Symp. on Low Power Design, 1995, pp. 3-8.

[3] S. Raje and M. Sarrafzadeh, “ Scheduling with multiple voltages,” Integration the VLSI Journal, Oct. 1997, pp. 37-60.

[4] J-M. Chang and M. Pedram, “Energy Minimization Using Multiple Supply Voltages” Proc. of the Int. Symp. on Low Power Design, 1996, Monterey CA.

[5] W.-T.Shiue and C. Chakrabarti, “Low Power Scheduling with Resources Operating at Multiple Voltages,” Proc. of the Int. Symp. on Circuits and Systems, May 1998.

[6] M. C. Johnson and K. Roy, “Datapath scheduling with multiple supply voltages and level converters” ACM Transactions on Design Automation of Electronic Systems, July 1997, p227-248.

[7] Y-R. Lin, C-T Hwang and A. C-H. Wu, “Scheduling techniques for variable volta ge low power design” ACM Transactions on Design Automation of Electronic Systems, 1997, pp. 81-94.

[8] M. Igarashi, K. Usami, K. Nogami, “A Low-Power design method using multiple supply voltages” Proc. of the Int. Symp. on Low Power Design, 1997, pp. 36-41.