Multivoltage Scheduling with Voltage-Partitioned Variable Storage

Amitabh Menon
Texas Instruments India

S.K.Nandy
Indian Institute of Science

Mahesh Mehendale
Texas Instruments India

Categories and Subject Descriptors

B.6.3 [**Optimization**]:

General Terms

Design

Keywords

High Level Synthesis, Multivoltage, Datapath Synthesis

ABSTRACT

Multivoltage scheduling (MVS) for datapaths offers the energy savings of voltage scaling on a per-operation basis with a voltage aware operator scheduling. This work investigates the effect of using multivoltage register file storage in operator graph scheduling for multivoltage datapaths, thus partitioning the variable storage space on the basis of operating voltages. Scheduling operations on multivoltage functional units lowers the energy-delay product by exploiting slack in the schedule. We find that using multivoltage partitioned register files do the same by exploiting the variable lifetimes in the schedule. In a resource constrained context, longer variable lifetimes provide the opportunity to store such variables in low voltage register files. A ZOLP formulation for the resource constrained MVS problem is presented along with energy-delay savings obtained from applying this formulation to image processing transform graphs.

1. INTRODUCTION

The ever increasing scale integration in contemporary SoC devices and constraints of battery operation has made controlling power consumption a major concern for system designers. Considerable work has gone into minimizing power dissipation at the device, circuit and architecture level [1][3]. Lowering supply voltage for CMOS devices is now generally accepted as the most effective way to reduce energy dissipation. Techniques to do this dynamically over time [2][10] and statically using multiple voltages [4][7] have been studied. Dynamic voltage scaling incurs the cost of a latency, typically in the microsecond range, for switching the output of a programmable power supply. This range of delay suits allocation of different voltages to coarse grained computational entities like tasks scheduled by an operating system. Fine grained voltage control at the level of an operation is possible if functional units performing the operations can be run at different voltages. The operation schedule can then be directed towards achieving a lower energy dissipation by scheduling more operations onto those units running at a lower voltage if constraints on latency or resources allow it.

A solution for datapath scheduling with voltage assignment to operators was first proposed in[5]. In [7] a dynamic programming scheduler, and in [8] an ILP formulation for MVS were presented. The work in [9] presented an algorithm to partition operators in terms of voltages and search for a solution with minimum voltage conversions under latency constraints. In [6] two new techniques for MVS under resource constraints were presented and [11] exploited differences in functional unit delays in a multivoltage datapath by clock period scheduling. All the above concentrate on scheduling operations onto functional units with the inherent assumption that variable storage is realized with independent registers with associated level converters [8][9]. In general, the use of register files is more desirable as the storage required can be reduced through reuse as per the variable lifetimes in the scheduled graph.

In this work, we have used register files partitioned on the basis of operating voltage for multivoltage datapaths. This allows the variables in the operator graph to take on a multivoltage nature. Variables with longer lifetimes can be stored in the low voltage partitions so accesses to them consume less energy. We present an ILP formulation for the resource constrained MVS problem that includes the access times for register files and energy dissipation in the storage and datapath. We present achieved energy savings by using this formulation for scheduling a set of image processing graphs.

2. DATAPATH AND MVS GRAPH MODEL

The MVS problem involves simultaneously assigning a supply voltage to every operation in a graph while scheduling under latency or resource constraints and minimizing power/energy dissipation. The presence of multivoltage register files for storage of the graph variables complicates the scheduling problem due to non-uniform register access times. The access time for a variable stored in a register file becomes dependent on the register file supply voltage. If the schedule is done without considering the access times of the register files, this access time variation can render the schedule invalid.

The size of the register file affects the access delay. Ideally

ISLPED'03, August 25–27, 2003, Seoul, Korea.

this size would be exactly the minimal number of registers needed after the register assignment is optimized on the basis of variable lifetimes obtained from the schedule. However for our case, this delay is needed for operator scheduling. We made a simplifying assumption of fixed size register files, so that the supply voltage alone would decide the delay. This implies that in the worst case, all registers in a register file may not be used, but the schedule comprehends the supply voltage related access delay variations and remains valid. By doing a register assignment optimization on this schedule, one could determine the sizes of the register files needed and remove the size and power penalty of the fixed size register files. We modeled the fixed size (32 registers,10 read ports and 5 write ports for us) as a resource constraint while scheduling. Also, while the technique presented here is valid for multiple voltages, we limited the experiments to two voltages as the supply voltages were low to begin with on the technology used.

As an architectural model we limited the work to the context of data-dominated behavioral descriptions the form of acyclic data flow graphs. We also assume a component binding that assures a one-one correspondence between types of operations in the graph and the types of functional units available. The functional unit library consists of identical blocks implementing arithmetic operators at both voltages. The register file is partitioned into two fixed size register files run at one of the two voltages. Interfaces between register file ports and functional units with different supply voltages were equipped with level converters. The level converter circuit used in [8] was used after sizing appropriately for the technology used. With this datapath structure, an operation in the graph being scheduled could have the read of it's operands, the execution of the operation and write-back of the results independently at any of the two voltages.

We modified the graph representation of a behavioral description to model the register file accesses explicitly. Specifically, we added extra nodes before and after every arithmetic/logical operation to model the read and write accesses associated with the functional unit operation. For scheduling, this treats the read and write accesses like any other operation and assigns voltages to these nodes to represent the supply voltage of the register files accessed. This necessitates extra constraints in the scheduling formulation as shown later.

3. CHARACTERIZING ENERGY-DELAY

For our experiments we used a contemporary CMOS technology (TI 0.18μ power optimized library) to implement the functional units and register files as synthesized but tiled and regular structures. The supply voltages in such a process technology are much lower than those in previous studies and the difference in V_{dd} and V_T is small enough to cause a large sensitivity to voltage scaling for both delay and energy consumption. Consequently the voltage differential required to bring about energy-delay savings is small and the delay and energy cost of using level converters is diminished.

We chose supply voltages of 1.25V and 1.95V for our experiments, V_T at this technology node being 0.48V. The netlists, with interconnect delays annotated, were analyzed at both voltages using a transistor level static timing analysis tool to get critical path delays. We took a pessimistic estimate of the register file delays in that we calculated the read access and write access latencies by normalizing against the delay of the most critical path among all ports of the same type. The figure 3 shows the effect of the voltage on delay and energy of the library components. Note that the delays in the read and write accesses paths are comparable to the functional unit delays so cannot be ignored in the schedule.Finally, we lumped the delay for level conversion with the read and write access times as per the voltages in use and discretized the delays to latencies by normalizing against the smallest of the delays which was for the write access path (1.8 ns at 1.95V). Latencies of the operations and register file accesses at the two voltages are shown in table 1.

Figure 1: Library delays and energy vs operating voltage

Operation	# Steps at 1.25V	# Steps at 1.95V
read	2	1
write	2	1
add	2	1
subtract	2	1
multiply	5	3

Table 1: Control steps for computation and register access

For energy estimates of the functional units and register files, we relied on simulations. Simulation vector generation for the functional units was done using a GA based max-power vector generator in the $Powermill^{TM}$ tool. The generated vector pairs, which create near maximum average power dissipation, were simulated on the transistor netlist. We multiplied the power readings from the simulation with the cycle time to get the energy dissipated per cycle at each voltage. Note that the energy reduction in functional units and the register accesses due to lowering the operating voltage is relatively larger than the delay increase due to the same. While each register file access consumes far lesser energy than a single functional unit operation, each operation scheduled is accompanied by two reads and a write, so the cumulative energy consumption in the register file accesses becomes significant.

4. RCS FORMULATION

4.1 Notation used

- G(V,E) denotes the graph to be scheduled. Vertices $v \in V$ represent arithmetic/logical operations in the algorithm, or variable read/write operation. Edges represent data dependencies between these vertices.
- The vertex set V is composed of mutually disjoint subsets V_r, V_o, V_w, where V_o is the set of arithmetic/logical operators in the DFG, V_r is the set of reads in the DFG and V_w is the set of writes in the DFG. n denotes the number of vertices, i.e. $n = |V|$.

- s denotes the total number of scheduling steps. In our model, we add a dummy node in the graph that has a dependency on all the output variables, so that the schedule step of this node is equal to s+1.
- FU_k denote functional units of type k. As read and write accesses to register files are modeled as operators, notional functional units of types read and write are available. $v_i \in FU_k$ if v_i can be executed on functional unit FU_k i.e. the operation v_i is compatible with the type k.
- M_k denotes the maximum available number of functional units of type k. For read and write operations, these correspond to the number of read and write ports in the register file respectively.
- $D(i,g)$ denotes the delay of the vertex v_i scheduled to voltage g. It corresponds to the delay of functional unit FU_k for operating at voltage g, where $v_i \in FU_k$.
- $W(i,g)$ denotes the energy consumed by the operation v_i when scheduled onto a functional unit operating at voltage g. This corresponds to the energy consumed by the functional unit FU_k where $v_i \in FU_k$.
- S_i denotes the starting step of v_i. The starting step of an operation is the control step in which it is scheduled onto a functional unit.
- E_i denotes the ending step of v_i. $E_i = S_i + D(i,g) - 1$ when v_i is scheduled to operate at voltage g. The ending step of an operation is the last control step in which it occupies a functional unit.
- x_{ijg} are {0,1} integer variables associated with v_i such that $x_{ijg} = 1$ if v_i is scheduled into control step j, to operate at g volts. G is the number of voltages used in the schedule.
- asap(i), alap(i) denote the earliest and latest control steps respectively that v_i can be scheduled into. These values are arrived at by doing an ASAP and ALAP schedule of the graph. Since the delay of an operation would depend on it's assigned voltage, ASAP scheduling is done using the highest available voltage for all nodes (and hence the lowest delays) and the ALAP scheduling is done with the lowest available voltage for all nodes (and hence the largest delays). In a schedule using that range of voltages, these asap and alap values would represent the full possible range of start values for the nodes.

4.2 ZOLP formulation for RCS

For weights α and β :

$$\textbf{min}(\alpha * s + \beta * \Sigma_i \Sigma_j \Sigma_g x_{ijg} * W(i,g))$$

such that :

$$\Sigma_j \Sigma_g x_{ijg} = 1 \quad \forall i \in \{1..n\},$$
$$0 \le g < G, \ \ asap(i) \le j \le alap(i) \tag{1}$$

$$E_i < S_j \quad \forall (v_i, v_j) \in E \tag{2}$$

$$\Sigma_{v \in FU_k} \Sigma_g \Sigma_{E_i \ge j} x_{ijg} \le M_k, \ \ \forall FU_k,$$
$$\forall asap(i) \le j \le alap(i) \ , \ \ 0 \le g < G \tag{3}$$

$$S_j = E_i + 1 \quad \forall v_i \in V_r, \ v_j \in V_o, \ (v_i, v_j) \in E \tag{4}$$

$$S_j = E_i + 1 \quad \forall v_i \in V_o, \ v_j \in V_w, \ (v_i, v_j) \in E \tag{5}$$

$$\Sigma_l x_{ilg} = \Sigma_k x_{jkg} \quad \forall v_i \in V_w, \ v_j \in V_r, \ \forall g \tag{6}$$

- The objective function of this formulation is a weighted sum of the number of control steps used and the energy dissipated by the scheduled graph. The weights α and β provide a way for the user to control the emphasis of the optimization on the latency of the schedule or the energy dissipation while exploring the design space.
- Equation 1 constrains the schedule for any operation to be unique, i.e. any node in the DFG can be scheduled to one control step and one supply voltage only.
- Equation 2 is the sequence constraint. It specifies that the schedule respects precedence relationships implied in the DFG.
- Equation 3 is the resource constraint that dictates for any resource type, the maximum number of resources used in a control step cannot surpass the maximum number of available resources of that type.
- Equations 4 and 5 are relative constraints specific to our graph model. They constrain the schedule to have read and write access vertices scheduled exactly one cycle away from the associated arithmetic operation. This ensures that registers are read only when functional units are available and that write ports are available when an arithmetic operation completes. Scheduling without these constraints would present unneccesary false resource constraints.
- The equation 6 is specific to multivoltage register files. It ensures that a value written to a register is read at the same voltage as it was written at. Since the register files in our model are partitioned as per the supply voltage, this implies that a value is read from the same register file it was saved in.

5. EXPERIMENTAL RESULTS

Using the above formulation, we generated schedules for three commonly used image processing transforms - the Haar transform, the Slant transform and a graph for a set of Edge Detection masks. These are medium sized graphs on addition of the read/write nodes. We scheduled these graphs for a single voltage (at 1.95V), with two voltages for the functional units only and with two voltages for both the functional units and the register files. To explore the design space, we used the biasing weights in the cost function. We first scheduled the graphs with no cost function bias (energy and latency get equal emphasis $\alpha = 0.5$, $\beta = 0.5$) and repeated with latency emphasized ($\alpha = 1$, $\beta = 0.3$) to see the effect of tighter latency requirements.

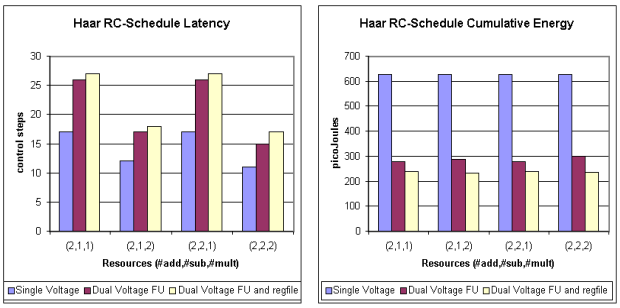

Figure 2: Latency and Energy of Haar transform schedule

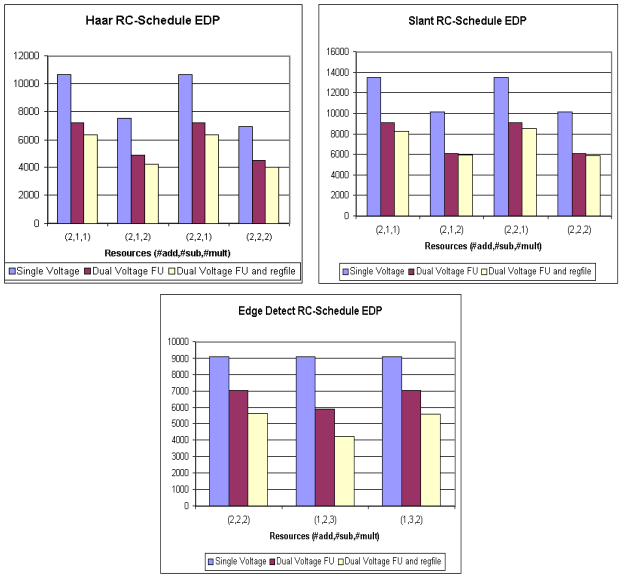

Figure 3: EDP for schedules using $\alpha = 0.5, \beta = 0.5$

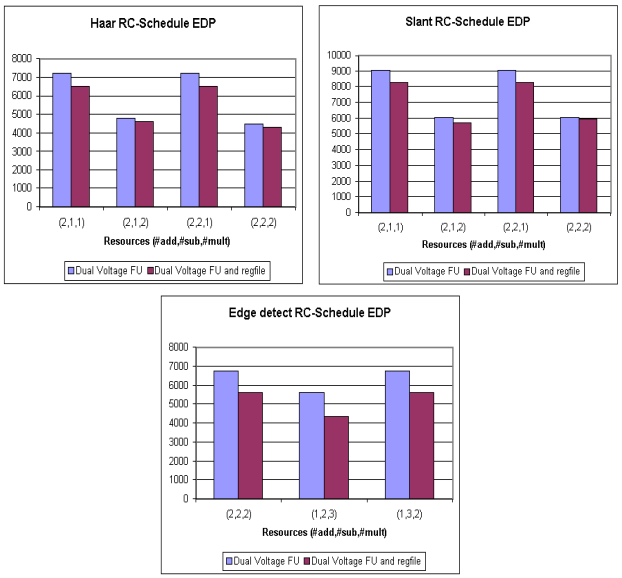

Figure 4: EDP for schedules using $\alpha = 1, \beta = 0.3$

The figures 2 , 3 and 4 show the experiment results. Figures 2 and 3, compare the schedules with various resource constraints, for the baseline of a single higher voltage for all datapath components (case 1), multivoltage functional units only (case 2) and the above formulation where both functional units and register files are multivoltage (case 3). In figure 4 the single voltage case has been omitted.

Figure 2 shows that for equal weightage to latency and energy, the total energy is reduced at the expense of latency. However, figure 3 shows that the Energy-Delay Product reduces overall implying that the latency loss is overcome with the energy gains for both case 2 and case 3 (EDP reduces by 11-28% for case 3 over case 2). Also note in figure 3 that the largest gains in EDP for case 3 over case 2 are when the resource constraints are tightest (single multiplier resource) and for the highly parallel edge-detect graph. With tight resource constraints, the schedule gets more serialized and latency increases for both case 2 and 3, but case 3 shows lower EDP. We examined the schedules produced in case 3 to confirm that the serialization creates larger separations between writes and successive reads, which allows these write and read access nodes to be assigned to the lower voltage register file partition. Figure 4 shows that for the cost function biased towards better latency, the EDP gains are much lower (5-10%) but the latencies achieved are shorter. Even with the bias towards low latency solutions, case 3 shows a marginally reduced EDP.

6. CONCLUSIONS AND FUTURE WORK

In this work we have demonstrated the opportunity of additional reduction of energy-delay by using multiple voltages for register files as well as functional units. We have presented an ZOLP formulation that implements this idea for the RC MVS problem. This approach exploits longer variable lifetimes and hence gives energy savings even when the resource constraints are tight unlike previous work on MVS which depends on the duplication of functional units. The obvious application of this work is in behavioral synthesis of multivoltage hardwired datapaths. We plan to explore the utility of this for VLIW processor datapaths, such that instruction scheduling can be done utilizing multivoltage functional units and register files.

7. REFERENCES

[1] Massoud Pedram "Power Minimization in IC design : Principles and applications", ACM Transactions on DAES, Jan 1996.

[2] A. Chandrakasan, V. Gutknik, T. Xanthopoulos, "Data driven Signal Processing : An approach for Energy Efficient Computing", Proceedings of the ISLPED, 1996

[3] A.Raghunathan and N.K.Jha, "Behavioral Synthesis for Low Power", Proc. of the ICCD,1994

[4] K.Usami and M.Horowitz, "Clustered Voltage Scaling Technique for Low Power design", Proceedings of the ISLPED, 1995

[5] S.Raje and M.Sarrafzadeh, "Variable Voltage Scheduling", Proceedings of the ISLPED, 1995

[6] S.Raje and M.Sarrafzadeh, "Scheduling with Multiple Voltages under Resource Constraints", Proceedings of IEEE conference on Circuits and Systems,1999

[7] J.Chang and M.Pedram,"Energy Minimization using Multiple Supply Voltages", IEEE Transactions on VLSI systems, December 1997

[8] M.Johnson and K.Roy,"Datapath Scheduling with Multiple Supply Voltages and Level Converters", ACM Trans. DAES, July 1997

[9] Mark Johnson, Kaushik Roy "Scheduling and optimal voltage selection for low power multi-voltage DSP datapaths", Proceedings of IEEE International Symposium on Circuits and Systems, 1997

[10] T.Ishihara, H.Yasuura "Voltage scheduling problem for dynamically variable voltage processors", Proceedings of the ISLPED, 1998

[11] S.P.Mohanty, N.Ranganathan, V.Krishna "Datapath Scheduling using Dynamic Frequency Clocking", IEEE Computer Society Annual Symposium on VLSI, 2002

[12] A.Menon, "Multivoltage High Level Synthesis for Low Power Design", M.S. Thesis, Indian Institute of Science, 2002