Simultaneous Dynamic Voltage Scaling of Processors and Communication Links in Real-time Distributed Embedded Systems

Jiong Luo, Li-Shiuan Peh and Niraj Jha
Dept. of Electrical Engineering
Princeton Univ., Princeton, NJ 08544

Abstract— **Dynamic voltage scaling has been widely acknowledged as a powerful technique for trading off power consumption and delay for processors. Recently, variable-frequency (and variable-voltage) parallel and serial links have also been proposed, which can save link power consumption by exploiting variations in bandwidth requirement. In this paper, we address joint dynamic voltage scaling for variable-voltage processors and communication links in such systems. We propose a scheduling algorithm for real-time applications, with both data flow and control flow information captured. It performs efficient routing of communication events through multi-hops, as well as efficient slack allocation among heterogeneous processors and communication links to maximize energy savings, while meeting all real-time constraints.**

I. Introduction

As the demand for system bandwidth and correspondingly power dissipation is increasing, both chip-to-chip and on-chip interconnection networks are becoming power/energy limited. One powerful technique for reducing power consumption is dynamic voltage scaling, which refers to dynamic adjustment of the supply voltage to the minimum level required for a processor to work at a desired clock frequency. Besides commercially available voltage-scalable processors, variable-frequency links have been proposed for both parallel links [1] and serial links [2]. They can adaptively regulate the supply voltage to a desired link frequency, in order to exploit variations in the bandwidth requirement to reduce the link power consumption. Variable-frequency links, which make dynamic voltage scaling on links feasible, provide a new dimension for system power optimization. The work in [3] optimizes processor and link power jointly. However, it does not explore the usage of variable-voltage links. In this paper, for the first time we tackle the problem of simultaneous dynamic voltage scaling of processors and communication links for real-time distributed systems with chip-to-chip interconnection fabrics (our techniques are applicable to on-chip interconnection fabrics as well).

II. Network switching techniques

There are a number of switching techniques for interconnection networks, such as circuit switching, wormhole flow control, statically-scheduled switching, virtual channel flow control, as well as their combinations. In circuit switching, a route is established for a message. Multiple flits of the message can be pipelined through this path until the path is released. Previous works have used circuit switching to handle real-time traffic. In statically-scheduled switching, such as in the MIT RAW machine, a compiler schedules routes and buffers in advance of program execution. We use statically-scheduled circuit switching for real-time traffic. The routes for all real-time traffic are statically determined and pre-reserved.

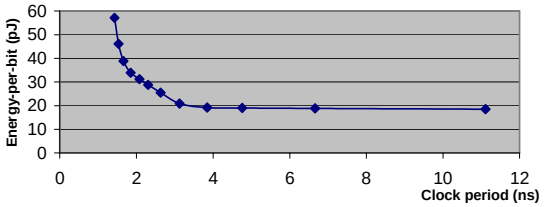

Fig. 1. Energy vs. delay curve for communication links (adapted from [2])

III. Input specification

We assume that allocation of PEs has been done, the network topology determined, as well as the mapping of tasks to different PEs performed. The real-time workload is represented as a set of directed acyclic graphs (DAGs), where nodes represent tasks and edges represent communications. The parent node of an edge is the sender of a communication and the child node the receiver. In each DAG, we also allow the existence of conditional edges to represent control flow (in terms of conditional branches). Each edge can be either a simple edge, representing only data dependency, or a conditional edge, which is annotated with a conditional variable, representing both data and control dependencies. A task with output conditional edges is a branch fork node. Alternative paths meet at a branch join node. The DAG is called a conditional task graph, which is associated with a period, indicating the time interval at which it is invoked again. Different graphs may have different periods. The least common multiple of all the periods is called the hyperperiod. Hard deadlines exist for every sink node as well as some intermediate nodes.

IV. Energy vs. delay modeling for dynamic voltage scaling

The dynamic power consumption, p, can be represented as a function of the supply voltage, V_{dd}, clock frequency, f, switching activity, N, and capacitance, C, as:

$$p = 1/2 f N C V_{dd}^2 \quad (1)$$

For today's deep sub-micron CMOS technology, the processor clock frequency, f, can be represented in terms of the supply voltage, V_{dd}, and threshold voltage, V_t, as:

$$f = k(V_{dd} - V_t)^{\alpha}/V_{dd} \quad (2)$$

where k is a constant, and $1 < \alpha \leq 2$.

The energy vs. delay curve for variable-frequency links is shown in Fig. 1, based on data adapted from experiments presented for a variable-frequency serial link [2], which has a supply voltage range of $0.9V$ to $2.5V$ (correspondingly, bit-rate range of $450Mb/s$ to $3.7Gb/s$ and clock frequency range of $90MHz$ to $740MHz$). We can see that the energy vs. link delay function is convex.

When the clock period of a processor or link is extended by time dt, the supply voltage can be scaled down correspondingly, and the energy consumption is reduced as well. We define an energy reduction rate, G, as the negative of

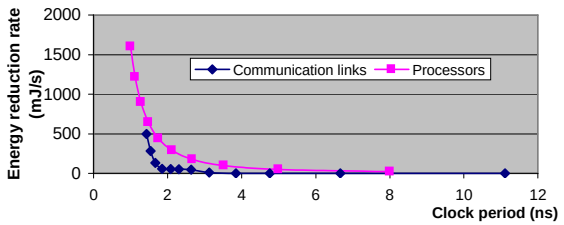

Fig. 2. Energy reduction rate curves for communication links and processors

the derivative of energy consumption with respect to the task or communication event delay t under supply voltage V:

$$G(V) = -\frac{\partial E(V)}{\partial t} \tag{3}$$

The energy reduction rate curves for communication links and processors are shown in Fig. 2. The curve for processors is based on the model in Equations (1) and (2), assuming $\alpha = 2$, $V_t = 0.6V$, the voltage range to be 1.6V to 0.85V, and the power consumption at the maximum clock frequency ($1GHz$) to be $1.6W$. For both processors and communication links, the energy reduction rate is a monotonically decreasing function with respect to the clock period.

Dynamic voltage scaling saves energy by allocating slack to tasks or communication events, so that the clock frequency for executing or transmitting them can be decreased. Suppose the overall slack in the system is ζ. If we represent the overall slack as an interval $[0, \zeta]$, then the energy reduction resulting from voltage scaling after the execution and communication times of all the tasks and communication events are extended by a total of ζ can be expressed as:

$$\delta E = \int_0^{\zeta} G(t)dt \tag{4}$$

where $G(t)dt$ refers to the energy reduction achieved by allocating dt unit of slack $[t, t + dt]$ from overall slack to execution or communication of some event. As the voltage and frequency scale down, the energy reduction rate drops. An optimal energy reduction is achieved by maximizing the integral of energy reduction rates over the overall slack. An effective slack allocation can be achieved by allocating slack to the execution or communication cycles of the set of extensible events with the highest energy reduction rate. An event is defined as non-extensible if extending its execution or communication time leads to violations of its own deadline or that of other events. The allocation of slack to a set of events at the highest energy reduction rate should be done in a balanced way such that their energy reduction rates can be kept at the same level.

V. Variable-voltage Scheduling algorithm

A static schedule is generated for all the instances of tasks and communication edges in the hyperperiod. Each PE or router stores a local schedule table for events assigned to that PE or router, which imposes an execution ordering of events. To support control flow, a schedule table entry can be annotated with conditionals. A conditional variable, when it is ready, is used to activate or de-activate the corresponding branches in the local schedule table of a PE or router. The static schedule can be regenerated for a new hyperperiod when the real-time workload is updated in the system.

The initial schedule is generated using list scheduling under the maximum clock frequency. We adopt the method from [4] to identify mutually exclusive events. When a communication event is ready to be scheduled, we need to first find a routing path for it. The routing path, depending on the network topology, can be multi-hop, consisting of a set of physical links. We utilize an earliest-finish-time driven algorithm based on the Bellman-Ford algorithm for this purpose. After routing is finished, we use a macro-event to represent the communication event, which may occupy several physical links based on the routing path. The macro-event has a delay equal to the communication latency, and an energy consumption equal to the summation of energy over all the hops.

After a valid schedule is generated through list scheduling and the order of scheduled events is determined, a new graph $G(V, E)$ can be created based on the conditional task graphs as well as the constraints imposed by resource sharing and link contention after scheduling. In $G(V, E)$, V is the set of vertices, containing all the scheduled events in the initial schedule, and E is the set of directed edges between vertices. An edge is inserted from one event to another if one is a direct predecessor of another in the task graphs, or if one is scheduled just ahead of another on the same PE or physical link. For two mutually exclusive events, no edge needs to be added to impose any execution ordering. To account for the transition time overhead for supply voltage and clock frequencies, we also insert a dummy event between two events, wherever a possible voltage and frequency transition might occur. Then a slack allocation scheme similar to the one presented in [5] is employed based on $G(V, E)$. The supply voltages and clock frequencies for executing tasks and transmitting communication events can be determined based on the slack allocated to them.

VI. Experimental Results

We tested our algorithm on five embedded applications $TG1$-$TG5$, which are from sonar data processing, digital camera, networking, smart camera, and video service applications, respectively. In Table I, we compare three different schemes: (1) no dynamic voltage scaling; (2) dynamic voltage scaling on PEs alone; (3) dynamic voltage scaling on both PEs and links. A channel refers to two pair-wise communication links. The link characteristics are obtained from [2] and [6]. On average, Scheme 3 consumes 52% (27%) less power than Scheme 1 (2). The CPU time for scheduling is less than .05s on a Pentium-III $933MHz$ machine with $256MB$ memory.

TABLE I

Comparison of different schemes

Test	#Tasks/ #Edges	#PEs/ #Channels	Power(W)		
			(1)	(2)	(3)
TG1	120/40	3/3	0.74	0.51	0.44
TG2	17/14	3/6	1.37	1.11	0.71
TG3	26/10	4/5	2.40	1.81	1.47
TG4	5/4	5/7	0.11	0.037	0.025
TG5	52/26	7/14	0.44	0.31	0.21

VII. Conclusions

In this paper, we presented an efficient simultaneous dynamic voltage scaling algorithm for voltage-scalable processors and communication links in an embedded system.

References

[1] G. Wei, J. Kim, D. Liu, S. Sidiropoulos, and M. Horowitz, "A variable-frequency parallel I/O interface with adaptive power-supply regulation," *J. Solid-State Circuits*, vol. 35, no. 11, pp. 1600-1610, Nov. 2000.

[2] J. Kim and M. Horowitz "Adaptive supply serial links with sub-1V operation and per-pin clock recovery," in *Proc. Int. Solid-State Circuits Conf.*, pp. 216-217, Feb. 2002.

[3] J. Liu, P. H. Chou, and N. Bagherzadeh, "Communication speed selection for embedded systems with networked voltage-scalable processors," in *Proc. Hardware/Software Co-design Wkshp.*, pp. 169-174, May 2002.

[4] Y. Xie and W. Wolf, "Allocation and scheduling of conditional task graph in co-synthesis," in *Proc. Design Automation & Test in Europe Conf.*, pp. 620-625, Mar. 2001.

[5] J. Luo and N. K. Jha, "Power-profile driven variable voltage scaling for heterogeneous distributed real-time embedded systems," in *Proc. Int. Conf. VLSI Design*, Jan. 2003.

[6] http://www.ti.com/sc/docs/news/2000/00107b.htm.