Voltage Reduction of Application-Specific Heterogeneous Multiprocessor Systems for Power Minimisation

Allan Rae, *Student Member, IEEE* Sri Parameswaran, *Member, IEEE*
Department of Computer Science and Electrical Engineering
University of Queensland,
St. Lucia, Queensland, Australia.
{rae,sridevan}@csee.uq.edu.au

Abstract— **We present a design strategy to reduce power demands in application-specific, heterogeneous multiprocessor systems with interdependent subtasks. This power reduction scheme can be used with a randomised search such as a genetic algorithm where multiple trial solutions are tested. The scheme is applied to each trial solution after allocation and scheduling have been performed. Power savings are achieved by equally expanding each processor's execution time with a corresponding reduction in their respective operating voltage. Lowest cost solutions achieve average reductions of 24% while minimum power solutions average 58%.**

I. Introduction

A. Motivation

As the levels of system integration continue to rise designers of all types of integrated circuits are needing to deal with power consumption. No longer is power consumption the bane only of designers of battery-powered devices. For while low-power consumption can obviously lead to longer battery life there are other compelling reasons to reduce power consumption. These reasons include increased reliability, reduced manufacturing cost (through packaging and heatsink savings) and reduced time to market (less demanding of power supplies and cooling so fewer difficulties in development).

The rise of the system-on-a-chip is making it possible to integrate multiple processor cores and peripheral devices on a single silicon chip. One area of system-on-a-chip design is the field of heterogeneous multiprocessor (HeMP) synthesis where multiple processors of differing functionality are selected to work together to achieve higher performance than a homogeneous multiprocessor might.

Application-specific multiprocessor systems range from high-performance signal processing systems and robot arm controllers to video-cameras and automobile engine and transmission management systems. Any embedded system or subsystem will have performance, cost and power considerations. In this paper we investigate a strategy for reducing the power requirements of application-specific HeMP systems. Our strategy can be applied to multiple processor core system-on-a-chip systems or even to multiprocessor systems using off-the-shelf components.

TABLE I
CPU Voltage and Power

CPU Type	V_t	V_{min}	V_0	P_0(mW)	Cost
a	0.6	1.5	5.0	180	2
b	0.7	1.3	5.0	240	4
c	0.55	1.4	5.0	200	5

The power consumption of a CMOS processor core can be reduced considerably by lowering the operating voltage. Since power is proportional to the voltage squared even small reductions in voltage can provide large power savings. The downside of voltage reduction is increased circuit delay and a corresponding reduction in clock speed. Therefore voltage reduction techniques can only be applied where a schedule has room to expand before it meets a deadline.

Our goal for the research presented in this paper is to assign a single optimal voltage to each processor present in an application-specific HeMP system (where an application is made up of interdependent subtasks) such that interprocessor communication dependencies are satisfied without disrupting a previously determined processor allocation and task schedule at the nominal operating voltage.

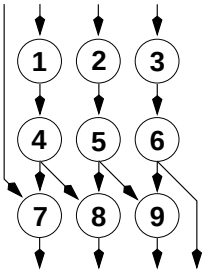
B. Motivational Example

To demonstrate the effectiveness of the voltage reduction technique we present a simple worked example of the power minimisation strategy. This example is an extension of the nine-subtask problem presented by Prakash and Parker [1]. The extra data consist of nominal voltage (V_0), threshold voltage (V_t) and nominal power dissipation (P_0) and these are shown in Table I along with P&P's original processor costs. The voltage and power data are estimated from embedded processor or microcontroller data sheets. The subtask execution times are shown in Table II while the task-data flow-graph is shown in Fig. 1(a).

Given a target time of $t_M = 12$, the search for the lowest cost, lowest power solution will involve testing a number of possible solutions. The allocation and scheduling of one possible

TABLE II
PP2 SUBTASK TIMES

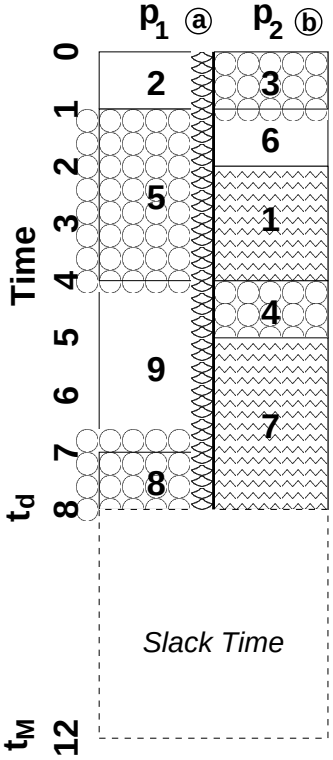
CPU Type	1	2	3	4	5	6	7	8	9
a	1	1	2	-	3	1	4	1	3
b	2	2	1	1	1	1	3	-	1
c	3	1	1	3	1	2	1	2	1



(a) TDFG

t_d	8
t_M	12
Cost	7
P_0(mW)	420
P_t(mW)	124.4

(b) Results



(c) Schedule

Fig. 1. pp2 Task Data Flow Graph and Schedule

solution for the task is shown in Fig. 1(c) which also shows on the time line the amount of slack time ($D = t_M - t_d$). The power minimisation process simply aims to stretch the execution time of a task by reducing the processor operating voltages and clocks such that the slack time is soaked up but also ensures that synchronisation/communication points remain respected. The allocation and relative scheduling of the subtasks therefore remain the same while the execution of the task now covers the interval {0,12} instead of the period {0,8}.

The two processors have different threshold voltages, V_t, so will also have different reduced operating voltages. The voltage required to stretch the execution time of the subtasks on p_1 is 3.533Volts which corresponds to a power level of 53.33mWatts. Similarly processor p_2 will be operating at 3.566Volts and dissipating 71.11mWatts. Thus both processors are operating above their minimum reliable operating level. Thus the total reduced operating power, P_t, is 124.44mWatts which is a 70% reduction. A summary of the time, cost and power consumption levels before and after power minimisation are shown in Fig. 1(b).

What if we were to shut down the processors during the slack time instead of reducing their operating voltages? Let us assume that during shutdown no power is dissipated. Then for this example we would be operating with a duty cycle of 66.7%. So the average power consumption would be $0.667P_0$ or 280mWatts. This results in over twice the power dissipation of operating at reduced voltage levels with 100% duty cycle. So for power reduction it is better to reduce operating voltages than to shutdown provided there is slack time available for expansion of the execution time.

C. Definitions

To make the rest of the paper a little easier to understand some definitions are provided below:

- V_0 the nominal operating voltage of the device
- V_t the threshold voltage of the transistors making up the device
- V_{min} the minimum usable operating voltage of the device. This voltage is dependent on the noise margins required for reliable operation and, in the case of off-the-shelf components, will also be influenced by the need to connect to external devices if the I/O ports do not have separate supplies from the core
- P_0 power dissipation at nominal voltage
- P_{p_i} power dissipation of processor p_i
- P_t total power dissipation of the multiprocessor system at reduced operating voltages
- t_d task execution time at nominal voltage level
- t_M target execution time of periodic task
- t_M' the actual execution time of the task on the reduced voltage system

D. Paper Organisation

The remainder of the paper is organised as follows. Section II discusses related work. Section III presents background material on reduced-voltage systems. The power reduction methodology is discussed in Section IV and some experimental results of the application of this methodology are presented in Section V. Some areas of future research are discussed in Section VI and the paper is concluded in Section VII.

II. Related Work

Most power minimisation research has so far been limited to VLSI and ASIC design [2]. Guo and Parameswaran derived new CMOS circuit delay equations [3] which formed the basis for their power minimisation methodology [4] which operates on voltage-reduction in system-level pipelines. Their work is used here as the basis for our power reduction scheme.

Gu and Elmasry [5] have derived equations that consider both static and dynamic power for deep submicron circuits and give upper and lower bounds on the value of V_{dd} corresponding

to the minimum of the power-delay product (PDP). Their work concluded that the minimum power-delay product occurs when V_{dd} is three times the threshold voltage, V_t, and the dynamic power is dominant. Thus V_{dd} should only be reduced as far as $3V_t$.

Techniques for maintaining throughput despite the lower clock rate resulting from reduced operating voltages include architecture-level pipelining and parallelism [6] and the use of variable-threshold CMOS [7] in which the threshold voltage is scaled along with the supply. Application of both techniques however has been limited to the field of digital signal processing [8].

On-chip variable-voltage supplies are becoming available such as the DC-DC switching regulator developed by Namgoong et al [9]. These are allowing dynamically-variable voltage reduction techniques to be developed. Hong et al [10] developed a power minimisation methodology for single processor core systems which also incorporates instruction and data cache selection. Their scheme utilises a variable voltage supply. Multiple independent tasks are scheduled on the single processor core. Supply voltages for each task are allocated to minimise the power consumption while still meeting arrival and deadline times.

Within the HeMP synthesis field power minimisation has thus far consisted of selecting components with lower power requirements that will still meet performance or cost requirements. HeMP Synthesis packages [11], [12], [13], [14] have not actively manipulated the operating conditions of the components they choose in order to minimise power consumption. This paper describes the first use of voltage reduction in the field of HeMP synthesis.

III. Power Minimisation by Voltage Reduction

The power dissipation of an electrical circuit is proportional to the operating voltage squared. Therefore a small reduction in operating voltage can result in large power savings. The work by Guo and Parameswaran [4] derives a set of equations relating power and operating voltage to propagation delay of electronic circuits consisting of a system-level pipeline of length m. We'll begin with a brief introduction to the theory of their work and then derive our own equations which are simplifications of their work.

The power dissipation of an electronic circuit is made up of dynamic or switching power and static power. The static power is due to short-circuit currents and leakage currents. The dynamic power dominates the static power in CMOS technology. The dynamic power dissipated by a circuit is given by

$$P = C_L V_{dd}^2 f$$

where V_{dd} is the supply voltage and C_L the loading capacitance. The voltage is inversely proportional to the critical path delay or execution time, t_d, of the circuit. At the limit of operation the clock frequency of the circuit, f, will be related to $1/t_d$. If the frequency is to be some percentage of $1/t_d$ then $f \propto 1/t_d$. The power can then be said to be proportional to the voltage cubed or inversely proportional to the critical path delay cubed. Therefore, at the limit of operation, power can be evaluated by

$$P = \frac{k}{t_d^3}$$

where k is a constant. Assuming the path execution time, t_d, is limited by the maximal execution time of the stages in a pipeline, t_M, then the difference

$$D = t_M - t_d$$

called *slack time* can be used to reduce the power consumption by lowering the supply voltages since the resources at normal voltage levels are underutilised. The equations derived for the minimum power dissipation of a pipeline, P_{min}, and the operating voltage of the ith pipeline stage, V_{dd_i}, are shown below.

$$V_{dd_i} = V_t + k_{t_i} \frac{k_i^{3/4} (\sum_{j=1}^{m} k_j^{1/4})}{t_M^{3/4} (\sum_{j=1}^{m} t_{d_j} + D)^{1/4}} \quad (1)$$

$$P_{min} = \frac{(\sum_{j=1}^{m} k_j^{1/4})^4}{t_M^3} \quad (2)$$

We have applied these equations to MPU cores which we treat as a pipeline of length one, that is $m = 1$. Thus we can simplify the above equations by substituting the definition of t_M. This substitution reduces equation 1 for V_{dd_i} to

$$V_{dd_i} = V_t + \frac{k_{t_i} k_i}{t_M} \quad (3)$$

and similarly if $t_M = t_d$ (there is no slack time) we can derive an equation for the nominal operating voltage, V_0.

$$V_0 = V_t + \frac{k_{t_i} k_i}{t_d}$$

If we now transform this equation we can get the definition of k_{t_i} as:

$$k_{t_i} = t_d \frac{(V_0 - V_t)}{k_i}$$

when this is substituted back into equation 3 we get:

$$V_{dd_i} = \frac{t_d}{t_M} (V_0 - V_t) + V_t$$

By a similar process of substitution we can determine a reduced equation for the minimum power. When $m = 1$ equation 2 becomes:

$$P_{min} = \frac{k_i}{t_M^3} \quad (4)$$

When $t_M = t_d$ (no slack time) we can derive an equation for the nominal power dissipation P_0:

$$P_0 = \frac{k_i}{t_d^3}$$

which can be transformed into a definition of k_i and substituted back into equation 4 to give:

$$P_{min} = P_0 (\frac{t_d}{t_M})^3$$

IV. Power Reduction Strategy

In this section we will formulate the problem and provide an outline of the proposed power reduction algorithm and a suitable objective function for synthesising a low-power application-specific HeMP system from a given task flow graph and set of design constraints.

A. Problem Formulation

The power reduction process takes as input a static task execution schedule, an allocation of processors and a set of design constraints. The constraints include a library of processors and other devices and their respective nominal voltages, threshold voltages and power consumption and a target execution time t_M.

The power reduction process then reduces the processor supply voltages if possible to slow the execution of the circuits to the target execution time t_M. The output is thus a static voltage assignment for each processor and the total power consumption with that assignment.

B. Objective Function

There are n processors in the system. Each processor p_i has an associated cost C_{p_i} and if a communication link exists between processors p_i and p_j it has a cost $C_{c(i,j)}$. Thus the total implementation cost, C_t, is shown in equation 5. The total execution time is represented by t_d.

$$C_t = \sum_{i=1}^{n} C_{p_i} + \sum_{i=1}^{n} \sum_{j=1}^{n} C_{c(i,j)} \tag{5}$$

Each processor or communication device has an associated power dissipation, P_{p_i}, and the total power, P_t, is thus the sum of these power costs.

$$P_t = \sum_{i=1}^{n} P_{p_i} \tag{6}$$

A target time, t_M, can be set. The synthesis package then attempts to optimise the system to match this target value. Thus the effective objective functions are shown in equation 7 below. This represents searching for the lowest cost, low-power system with the given target performance of t_M.

$$t_d \rightarrow t_M \ , \ C_t \rightarrow 0 \ , \ P_t \rightarrow 0 \tag{7}$$

C. Algorithm Outline

The power minimisation methodology can be incorporated into most existing heuristic-based HeMP synthesis packages which produce static subtask allocations and schedules [11], [12], [14]. The power minimisation strategy is applied after the scheduling has been done at the nominal operating voltage. The augmented output from the scheduler is then used by the search algorithm. A generic evolutionary/genetic algorithm is shown in Fig. 2 to demonstrate how the power reduction strategy can be used. We will concentrate on the power minimisation algorithm which is in Fig. 3.

```
Generate initial population
Get target execution time t_M
repeat
   foreach trial solution
      Allocate and schedule task at nominal
      voltage
      Apply voltage reduction strategy
      Record total cost, power and time
   endfor
   Update record of best solutions
   Generate new population
until convergence or loop limit
```

Fig. 2. Generic Evolutionary Search Algorithm

```
(Allocation and schedule data are passed from
scheduling heuristic)
t_d = total execution time
P_0 = Σ_i P_0i
if t_d < t_M then
      t_M' = t_M
      foreach processor p_i
            if (t_d (V_0i − V_ti)) / (V_mini − V_ti) < t_M' then
                  t_M' = (t_d (V_0i − V_ti)) / (V_mini − V_ti)
            endif
      endfor
      foreach processor p_i
            V_i = V_ti + (t_d / t_M') (V_0i − V_ti)
      endfor
      P_t = P_0 (t_d / t_M')^3
else
      t_M' = t_d
      P_t = P_0
      foreach processor p_i
            V_i = V_0i
      endfor
endif
return (t_M', P_t)
```

Fig. 3. Power minimisation algorithm

If a solution's normal execution time, t_d, is greater than the target execution time, t_M, its power dissipation and cost are still calculated. If no solution is found that can satisfy the performance requirements the next best solution can still be recorded.

If the execution time of the solution, t_d, is less than the target execution time, t_M, we have the opportunity to reduce the supply voltages to the processors and thereby slow their execution to soak up the extra time available. After the allocation and scheduling stage, the power minimisation stage is entered. A check is made to determine if all the processors will still be op-

erational if their supply voltage is reduced to the level required to match t_M. There are two ways to do this. The first way is to test whether the operating voltage at t_M is less than V_{min_i} (the minimum reliable operating voltage). The second way, which we use, is to test whether the execution time when operating at V_{min} is less than t_M. By using this second method we have calculated the longest we can stretch the execution time of a given processor. If this value is less than t_M then we *must* reduce the target execution time, now called $t_M^{'}$, in order to have a workable system from this trial solution. The new value for the target time is then used to determine what the new operating voltage, V_i, of each processor, p_i, should be.

All processors are thereby slowed uniformly so that any interprocessor communication dependencies are still met and don't cause additional delays. The power dissipation is also calculated using this new value of $t_M^{'}$. Note that this means a processor is slowed for its entire operation not for each individual subtask.

As the synthesis package iterates toward a final solution several trial solutions are tested. Faster solutions are slowed to meet the timing requirement, resulting in a power reduction for these solutions. Then the objective function chooses the cheapest, low-power solution that meets the timing requirement.

V. Experimental Results

In order to test our experimental power optimisation scheme four very different problems were selected. All experiments use the same processor types, costs and operating voltages as shown in Table I. *pp2* is Prakash and Parker's [1] second point-to-point interconnection experiment, while *robot* is from the work by Dhodi et al [11]. The remaining two problems are *jpeg*, still-image compression, and *mpeg*, moving picture compression, as described by Guo and Parameswaran [4]. The data in Table III are arranged in pairs of results for each experiment. The first result is the least expensive low-power implementation and the second result is the lowest power implementation irrespective of cost. A bar graph of these results is provided in Fig. 4.

The *pp2* examples have known optima at $t_d = 5, 6, 7, 8, 15$. Testing with t_M equal to those optimum values results in lowest-cost implementations identical to previous literature and lowest-power systems that have $t_d = 5$ and this is demonstrated by the first pair of results for pp2. Testing with values of t_M other than the optima times results in lowest-cost systems based on the $t_d = 8$ solution and lowest-power systems again based on the $t_d = 5$ solution. Similar observations can be drawn for the other test cases.

Thus implementations whose allocation and schedule results in a Pareto point (highest performance for a given cost) in the design space are being chosen by the search algorithm as the best solutions. The lowest-power irrespective-of-cost solution is one of the fastest Pareto point solutions whose reduced voltage levels are still within its reliable operating voltage range. The lowest-cost low-power solution is the least expensive Pareto point solution with a $t_d \leq t_M$.

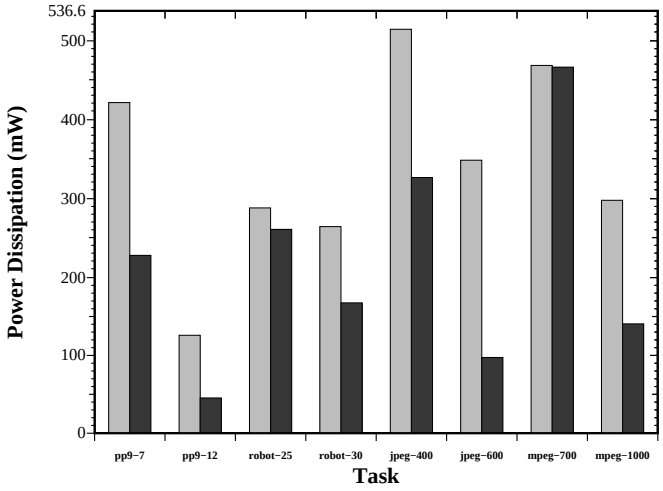


Fig. 4. Power dissipation of minimum cost and minimum power solutions.

The average power reduction of the least-expensive implementations is 24% with a high of 70%. This is a very encouraging result because it shows that sizeable power reductions are achievable for the cost of one fixed supply voltage per processor type. While the lowest-power implementations are achieved with the faster, more expensive systems, they demonstrate the very large power savings that can be achieved by voltage reduction. The average power reduction being 58% with a high of 93%. Their power dissipation is on average 40% lower than the lowest-cost implementations.

VI. Future Work

Our current work has been limited to whole tasks rather than at the subtask level. The main reason for this is that the opportunities for slowing individual subtasks is limited by communication dependencies which would thereby affect the scheduling of other subtasks. There may therefore be periods of processor inactivity that can only be exploited for power reduction by shutting down a processor. Using voltage reduction on a per-subtask basis should provide similar results to those we have obtained by processor-wide voltage reduction. Such a scheme could be applied to a task that has already been allocated and scheduled. Greater flexibility may be achieved by incorporating power-reduction considerations in the scheduling and allocation heuristic. This would also allow for compensation for the quantised voltage levels available from a variable supply or fine-tuning the variable supply's design for the given application.

Further investigation of the combination of voltage reduction and power-saving modes of processor operation is being conducted. Another area of investigation is the estimation of power-supply cost and any additional power-supply power dissipation.

TABLE III
RESULTS ON A SET OF PROBLEMS.

Prob.	subtasks	t_M	t_d	Cost	CPU Types	V_a	V_b	V_c	P_0(mW)	P_t(mW)	%
pp2	9	7	7	8	a, b	5.0	5.0	-	**420**	**420**	**0**
			5	15	a, b, c	3.74	3.77	3.73	**620**	**225.95**	64
		12	8	7	a, b	3.53	3.57	-	**420**	**124.44**	**70**
			5	15	a, b, c	2.43	2.49	2.40	**620**	**44.85**	93
robot	25	25	22	8	a, b	4.47	4.48	-	**420**	**286.22**	**32**
			22	9	a, c	4.47	-	4.47	**380**	**258.96**	32
		30	27	5	a, a	4.56	-	-	**360**	**262.44**	**27**
			22	8	a, b	3.83	3.85	-	**420**	**165.64**	61
jpeg	34	400	384	15	a, c, c	4.82	-	4.82	**580**	**513.15**	**12**
			326	20	c, c, c	-	-	4.18	**600**	**324.81**	46
		600	541	8	a, c	4.87	-	4.87	**380**	**346.82**	**9**
			326	20	c, c, c	-	-	2.97	**600**	**96.24**	84
mpeg	50	700	644	19	c, c, c	-	-	4.64	**600**	**467.21**	**22**
			643	20	c, c, c	-	-	4.64	**600**	**465.04**	22
		1000	920	8	a, c	4.65	-	4.64	**380**	**295.90**	**22**
			703	12	c, c	-	-	3.68	**400**	**138.97**	65

VII. CONCLUSIONS

In this paper, we have presented a strategy for power reduction in the synthesis of application-specific heterogeneous multiprocessors with interdependent subtasks. This simple strategy can be used with heuristic or randomised search methods.

The effectiveness of our approach was demonstrated with least-expensive implementations averaging a 24% reduction in power dissipation. The lowest-power solutions averaged a 58% reduction at a higher implementation cost. Our strategy for voltage reduction can provide significant power savings at the small cost of separate supply voltages for each processor type.

REFERENCES

[1] Shiv Prakash and Alice C. Parker, "SOS: Synthesis of Application-Specific Heterogeneous Multiprocessor Systems," *Journal of Parallel and Distributed Computing*, vol. 16, pp. 338–351, 1992.

[2] Jerry Frenkil, "Tools and Methodologies for Low Power Design," in *34th Design Automation Conference*, June 1997.

[3] Hui Guo and Sri Parameswaran, "New Derivation for Delay in CMOS Circuits," Tech. Rep., Dept. of Elec. and Comp. Eng., University of Queensland, August 1995.

[4] H[ui] Guo and S[ridevan] Parameswaran, "Power Reduction in Pipelines," in *Proceedings of Microelectronics'97*, 1997.

[5] Richard X. Gu and Mohamed I. Elmasry, "Power Dissipation Analysis and Optimisation of Deep Submicron CMOS Digital Circuits," *IEEE Journal of Solid-State Circuits*, vol. 31, no. 5, pp. 707–713, May 1996.

[6] Anantha P. Chandrakasan, Samuel Sheng, and Robert W. Brodersen, "Low-Power CMOS Digital Design," *IEEE Journal of Solid-State Circuits*, vol. 27, no. 4, pp. 473–483, April 1992.

[7] T. Kuroda et al., "A High-Speed Low-Power 0.3um CMOS Gate-Array with Variable Threshold Voltage (VT) Scheme," in *IEEE CICC-96*, May 1996, pp. 53–56.

[8] Kimiyoshi Usami et al., "Design Methodology of Ultra Low-power MPEG4 Codec Core Exploiting Voltage Scaling Techniques," in *35th Design Automation Conference*, June 1998.

[9] W. Namgoong, M. Yu, and T. Meng, "A high-efficiency CMOS dynamic DC-DC switching regulator," in *IEEE International Solid-State Circuits Conference*, 1997, pp. 380–381.

[10] Inki Hong, Darko Kirovski, Gang Qu, Miodrag Potkonjak, and Mani B. Srivasta, "Power Optimisation of Variable Voltage Core-Based Systems," in *35th Design Automation Conference*, June 1998.

[11] Muhammad K. Dhodhi, Imtiaz Ahmad, and Robert Storer, "SHEMUS: synthesis of heterogeneous multiprocessor systems," *Microprocessors and Microsystems*, vol. 19, no. 6, pp. 311–319, August 1995.

[12] Allan R. Rae and Sri Parameswaran, "Application-Specific Heterogeneous Multiprocessor Synthesis Using Differential Evolution," in *ISSS'98*. 1998, IEEE Press.

[13] Yanbing Li and Wayne Wolf, "A Task-Level Hierarchical Memory Model for System Synthesis of Multiprocessors," in *Proceedings of the Design Automation Conference*, 1997.

[14] Wayne H. Wolf, "An Architectural Co-Synthesis Algorithm for Distributed, Embedded Computing Systems," *IEEE Trans. VLSI*, vol. 5, no. 2, pp. 218–229, June 1997.