VLSI DESIGN
1998, Vol. 7, No. 3, pp. 303–320
Reprints available directly from the publisher
Photocopying permitted by license only

Optimizing Energy During Systems Synthesis of Computer Intensive Realtime Applications

CATHERINE H. GEBOTYS

Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, Ontario, Canada

Optimizing energy during the synthesis of VLSI systems for realtime-constrained embedded applications is an important new problem. This paper presents a new methodology for simultaneous scheduling and allocation of VLSI systems which minimize estimated energy for large realtime compute intensive applications. Minimization of estimated energy and VLSI chip area using hierarchical decomposition, bin packing algorithms and integer linear programming techniques along with voltage scaling is performed. Common subexpression elimination, precomputation, data regeneration, and loop merging transformations are supported. A large complex real industrial application, audio compression, donated by Motorola, is used to study the energy savings using different single and multichip system implementations. Results of synthesizing this complex application show that up to 10 times improvement in estimated energy are attainable for only 2.7 times increase in estimated chip area. Precomputation and other low energy transformations provided on average over 1.6 times savings in energy respectively. This research is important for industry since energy dissipation consideration at the early stages of design is crucial for mapping high performance applications into cost-efficient and reliable systems.

Keywords: Low energy, low power, systems, synthesis, integer programming, data regeneration, heterogeneous, VLSI

1. INTRODUCTION

Recently low power systems design has gained significant attention largely due to demands from the portable electronics industry. However system design for low power is also very important for other industries such as automotive, telecommunications, information technology, etc.. This is due to the fact that low power designs can offer significant reductions in system packaging costs and improvements in system reliability [1]. For example in multimedia applications low power is believed to be crucial [2]. This includes audio and video algorithms which process large amounts of data, performing computations in real time. Although video has higher speed requirements than audio, the audio applications remain a very difficult problem for low power design [3]. Multiple chips are required to provide sufficient parallelism to meet the real time constraints. In order to be cost effective the system is typically heterogeneous [2], where high speed chips are

necessary for time critical computations and slower speed chips can implement the non critical computations. High level design issues, such as determining the number of chips, the heterogeneity nature of the system, the scheduling of computations to meet real time constraints, the partitioning of computations among chips, are seen as having a significant impact on the final system's cost, performance and power dissipation. Clearly there is a need to study high level techniques for designing low power, high performance systems.

Tools for low energy system design need to be developed to support the mapping applications to more than one chip and explore energy versus cost tradeoffs to meet the performance of the application. The low energy system design problem involves several interdependent subproblems including scheduling and allocation. For example given an application consisting of a number of tasks, each task must be assigned to a functional unit (allocation subproblem), and a clock cycle during which its starts its computation (scheduling subproblem). It is also critical to consider memory requirements, since external memory will dissipate significantly more power than using on-chip memory. It is known that these early design decisions during scheduling and allocation have a significant impact on the final VLSI implementation. However scheduling and allocation are highly interdependent and thus must be solved simultaneously in order to synthesize optimal systems. The low energy system synthesis problem is most likely NP-hard, since many of its subproblems, such as simultaneous scheduling and allocation, have been defined as NP-complete [7].

2. PROBLEM DESCRIPTION AND PREVIOUS RESEARCH

The following problems, problem 1 and 2 given below, are important parts of the low energy system design problem that will be studied in this paper. For simplicity let us assume that an algorithm to implement the application has already been assigned based upon accuracy required, low power implementation, etc.. The algorithm is given as a partially ordered list of tasks, and each task is represented by a partially ordered list of code operations.

Problem 1 Schedule the algorithm, by mapping each code operation within each task to a time when it starts execution, maintaining the partial order among code operations.

The objective is to minimize the estimated energy dissipation given constraints on the throughput and the estimated chip area.

Problem 2 Schedule and partition the application, by mapping each task to a chip and to a time when it starts execution, maintaining the partial order among tasks.

Sending a word from one chip to a different chip requires a given number of csteps.

The objective is to minimize the estimated energy dissipation given constraints on the timing, and the number and capability of each chip (given as the speed of the chip and the parallelism that it can support).

The total energy per computation, for example in an adder (based upon CMOS VLSI technology [5]), is given in the equation (1) below,

$$\begin{aligned} \textit{Energy per computation} &= P_{\text{total}}/f_{\text{clk}} \\ &= C_{\text{effective}} V_{\text{dd}}^2 \end{aligned} \tag{1}$$

where $C_{\text{effective}}$ is the average effective capacitance being switched per clock cycle (clock cycle = $1/(f_{\text{clk}})$). The term P_{total} is the total energy dissipated in the system over the interval of time being one clock cycle. For applications with fixed throughput requirements (or with a fixed number of data samples which must be processed per second), it is generally agreed that energy should be used to compare design techniques and VLSI architectures as opposed to power [11]. If one were to use power as a measure, the frequency component of the equation would be equal to

the throughput of the application and hence would not change. From now on we will refer to energy savings.

According to equation (1), in order to reduce energy one can reduce the average capacitance of the system being switched. Some researchers have studied choices of algorithms [12] or modified the applications [13, 5] to reduce the number of high capacitance computations or total number of computations being performed, thus reducing the capacitance being switched. General techniques for reducing the number of computations are given in [5]. In other cases, these types of techniques may be closely tied to the application [11] and unfortunately may not be that useful in general. A second alternative to reducing energy is to lower the supply voltage, even though the speed of the chip decreases. A very popular technique, called voltage scaling, VS, [19, 17, 1] is to increase the parallelism in the system so that the time to perform the computations is very short. Then lowering the supply voltage will reduce the energy, and slow the system down until the timing constraints are met. The linear increase in capacitance (from added parallelism) is offset by the quadratic decrease in voltage, thus reducing the energy dissipated. More recently dynamic voltage scaling [14], DVS, where supply voltage is modified during the application processing was used to reduce energy. A list scheduler [15] was developed for DVS, however the constraint on area or hardware was not incorporated and the voltage supply values had to be selected by the designer. New processors being developed can operate at any supply voltage within a certain range [16], can operate at different clock frequencies, set dynamically [17], and support energy management modes which can be used to energy down parts of the processor when they're not being used [11 18].

Although most research in low energy design has remained at the gate level, a few researchers have studied low energy for higher levels of design. It is believed that good relative energy measures are useful for high level design decisions [1]. High level capacitance models for various functional units (i.e., multipliers, adders, etc), register files, interconnections, and controllers were developed for 1.2 micron CMOS process in [5] and verified with several real examples. Other capacitance models for example for RAMs and bus capacitance were also presented in [12, 10]. In [19] an energy model for digital signal processing systems was developed. Multiplications were used as a measure of computation (since additions and register accesses were negligible compared to the multiplications). Other studies identified that the energy dissipation required to transfer words off of a chip can be very significant [13]. By reducing the number of computations performed on one chip and allowing a smaller data path to be used, one may have more space on the chip to put more on-chip memory, thus reducing the number of external memory accesses, and overall reducing the energy dissipation significantly [13]. For example on-chip cache is very good for low energy design because it minimizes the number of off-chip memory accesses (read or write accesses to external memory) which is much larger than the energy dissipated for on-chip memory accesses [2]. Other researchers have discussed data coding strictly for off-chip transfers to reduce the total number of bits which switch [2]. Power dissipation is not only significant during external memory accesses but also approximately 40% of total chip power has been found to be due to the clock lines [9] and in other examples even at a low power chip design has 30% of total chip power dissipated in the data-path [23]. Therefore it is important to minimize power throughout the design where ever possible.

In this manuscript a new methodology is presented to solve problem 1 and 2, low energy VLSI systems design. Unlike previous research, an integer programming (IP) approach to finding minimum energy VLSI systems is introduced. Also low energy transformation techniques such as loop merging, and data regeneration are studied in conjunction with voltage scaling to study energy savings in systems. Hierarchy is introduced into the data flow graph in order to synthesize large complex applications. Precomputation and com-

mon subexpression elimination are also supported. Bin packing is introduced for scheduling loop structures within large applications efficiently. Energy savings are estimated for a large real industrial audio compression algorithm, VSELP, donated by Motorola [20].

3. ASSUMPTIONS, NOTATION, AND MODELS

The following terminology will be used in this paper: k = a task of the application (or input algorithm). o = a code operation in a task of the application. Let us assume that the application can be represented by a partially ordered list of tasks or a task flow graph (TFG), where tasks of the application are nodes and the partial order between tasks are arcs. TFG is an acyclic directed graph. The data flow graph, DFG, (also an acyclic directed graph) is used to represent each task. Let an arc of TFG from $k1$ to $k2$ be represented by $k1 \rightarrow k2$, which means that task $k1$ produces data needed by task $k2$. Similarly for the DFG where $o1 \rightarrow o2$ represents the partial order and data transfer. We will use x MACs to refer to the x number of multiply-accumulate operations that have to be performed.

The following parameters will be used:

$Energy_{\text{on-chip}}$ = energy dissipated by the chip.
E_f= energy dissipated by functional units (i.e., multiplier-accumulators, alus, etc.) on the chip.
E_m = energy dissipated by the registers, register files, RAMs and ROMs on the chip.
E_w = energy dissipated by the interconnections (i.e. busses and wires) on the chip.
E_c = energy dissipated by the controller on the chip.
$Vsc\ (T)$ = the value of the scaled voltage where T is the execution time or latency of the scheduled task flow graph and R is the required time to execute the application or the sample period (inverse of throughput) specified by the real time constraint.

Note $T < R$ and we can obtain the scaled voltage by solving $Vsc\ (T)$ below [11]:

$$R/T = [Vsc\,(T)\,[5 - Vth]^2] \,/\, [5\,[Vsc\,(T) - Vth]^2] \tag{2}$$

For results presented in this paper we use $Vth = 0.7$ V.

For illustration purposes the following variable will be used to define the single chip energy model below:

$y_{T,f,r,o} = 1$ if the application requires T control steps or clock cycles to complete, and f number of functional units, and r number of register files or RAMs and o number of ROMs in the VLSI architecture being synthesized for the application. Otherwise the variable is zero. The operand bitwidths of the functional unit and size of each memory unit must also be specified. This variable is easily extended for multiple types of functional units and multiple register files, RAMs and ROMs of different sizes. Also it can be extended to incorporate energy dissipation for external memory chips.

The single chip energy model is estimated as follows

$$Energy_{\text{on-chip}} = E_f + E_m + E_w + E_c \tag{3}$$

$$= \left[\sum_{T,f,r,o} totalCap(f, r, o, T) \; Vsc\,(T)^2\, y_{T,f,r,o} \right] \tag{4}$$

The total effective capacitance of the chip is modeled using the capacitance models of the functional units (multipliers, adders, etc), register files, RAMs, ROMs, and controller given in [5] for a 1.2 micron library, summed together with an interconnect capacitance model. The interconnection capacitance model used is taken from [10] and takes the square root of the estimated chip area as the average wire length, from which is calculates

average capacitance switched [5]. The chip area estimation is taken from [6] which is double the size of the total sum of the areas of all the functional units, with the exception that we also sum the areas of the RAMs, ROMs, and register files. The area of each of these units is obtained from COMPASS Design Automation Tools [24] for a 0.8 micron CMOS standard cell library.

The following parameters will be used in the multichip model of energy:

taskCap (k, p) = the average switched capacitance of task k executed by chip p.
capOff = the average switched capacitance per word transferred off of a chip.

The following binary variables will be used to define the energy model below:

$y_{k,p,T} = 1$ if task k is executed by chip p and the TFG can be completed by T clock cycles, otherwise it equals zero.
$b_{d,T} = 1$ if the TFG can be completed by time T and the total number of words transferred between chips is d, otherwise it equals zero.

The system energy model is estimated as follows

$$Energy_{\text{system}} = Energy_{\text{onChip}} + Energy_{\text{offChip}} \quad (5)$$

$$= \sum_{k,p,T} taskCap(k, p)\, y_{k,p,T}\, Vsc(T)^2 + \sum_{d,T} capOff\, |d|\, b_{d,T}\, Vsc(T)^2$$

The task capacitance model includes on-chip memory for the multichip case. In both the single chip and multichip cases we have presented the exact formulation for energy as it is used during synthesis. The energy model presented above will be used to estimate the relative power savings in the rest of this paper.

A ratio of average switched capacitance per cycle per computation on chip to average switched capacitance per word transferred off chip can be used for a relative energy comparison. For example consider the schedule in Figure 1b) where

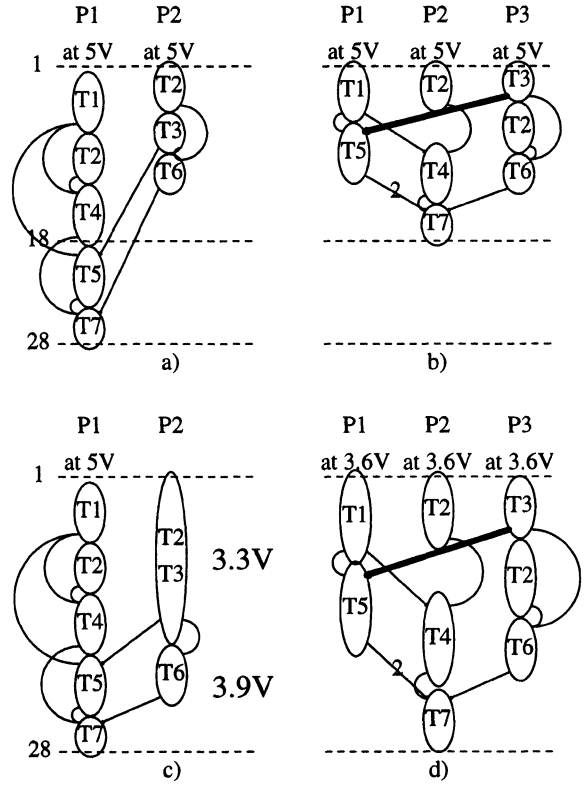

FIGURE 1 Two schedules for the application taken from [26] are shown in a) and b). One has minimum *ipc* a) and the other has minimum latency b). For a timing requirement of 28 cycles the schedule in a) can employ dynamic voltage scaling in chip P2 c) and the schedule in b) can voltage scaled, see d). (All edges in the Figure represent arcs of the TFG).

$T = 18$, $R = 28$, using equation (2) the scaled supply voltage is 3.6 volts (3.6V), shown in Figure 1d) (where all edges in the figure represent arcs of the task flow graph). Now let us assume for now that $taskCap(k, p) = 28\,(cyc(k, p))$ pF $\forall k, p$, $capOff = 80$ pF, and $\Sigma_k cyc(k, p) = 40$, $\forall p$. Note that in Figure 1b) there are three edges between chips each transferring one word, and one arc represents two words being transferred (indicated by a 2 on the edge in the figure), therefore $ipc = 5$. The bold arc indicates that the transferring of data from chip P3 to chip P1 is slower (every word requires double the amount of time compared to word transfers between all other chips). Solving for the system energy using equation (5) we have

$Energy_{system1d} = (28(40) + 80(5))(3.6^2)$ for Figure 1d) since there are 5 words being transferred off of the chips. If we then calculate this for Figure 1a) which only transfers two words off of the chip then we get $Energy_{system1a} = (28(40) + 80(2))5^2$. Note that the schedule requires 28 cycles so that we could not apply voltage scaling techniques. We could now say that Figure 1d) has a factor of 1.62 times energy savings (or power savings) at the expense of an extra chip. Note that we could also take the schedule in Figure 1a) and apply dynamic voltage scaling techniques to obtain the schedule in Figure 1c).

4. METHODOLOGY

This section will briefly describe the representation of the application and each stage of the low energy synthesis methodology. This methodology presented in this paper maps an application into a low energy architecture. The first stage of the methodology performs transformations on the application with the objective of minimizing the average switched capacitance through reduction of the total number of code operations in each task. These transformations involve precomputation (section 4.1) and common subexpression elimination (section 4.2). The second stage of the methodology tries to reduce the memory requirements of the application by performing loop merging (section 4.3) and data regeneration (section 4.4). It also reduces the complexity of scheduling the application, by performing the loop scheduling independently of the rest of the task flow graph, using bin packing algorithms. In this paper we use the term data regeneration to refer to a method which may increase the number of computations being performed but decrease the overall memory requirements. The final stage of the methodology (section 4.5) uses an integer programming approach to simultaneously schedule and allocate an architecture for the application from a hierarchical task flow graph. It searches for an optimal energy architecture over a given estimated area range with fixed throughput constraints on the application.

4.0. Algorithm Representation

Many large embedded applications typically are represented by a block diagram accompanied with detailed descriptions of the function of each block. We refer to each block as a task and the detailed function of each task is represented by the code operations. The task flow graph is created from this block diagram except the data transfers are detailed as arcs and extra task may be generated representing the block being computed over time. For large compute intensive applications with fixed throughput requirements, we partition the types of computations in each task into three types: nested dependent loops, independent nested loops, and random data flow graphs. This hierarchial representation helps to reduce the complexity of large applications. The processing of loops and the random data flow graphs will be described below. The nested dependent loops are scheduled separately using fast bin packing algorithms to determine the execution times versus the number of MAC functional units. The second independent loop constructs are loop unrolled and scheduled simultaneously with scheduling the random data flow graphs explained in section 4.5 in more detail. Since many applications contain conditional statements, we assume worst case conditions. In other words that the largest number of computations will be performed, so that when we scale down the voltage the throughput conditions will be met under all circumstances.

4.1. Precomputation

One approach to reducing the number of computations performed are to precompute as many values where ever possible. By identifying the data which is constant in the application (i.e., that would be stored in ROM), and checking their computation to see if precomputed values can be stored in ROM, often very significant compute

time and energy can be saved. For example in the audio compression algorithm vector quantization [20] is performed. During vector quantization we search for different codes to minimize some function. Our codebook consists of gs, $p0$, $p1$ three codes and we are given $C_i, i=1,\ldots,6$. The algorithm must find the three codes to minimize the following function:

$$C_1 gs(1 - p0 - p1) + C_2 gs p0 + C_3 gs p1 + C_4 \sqrt{gs(1 - p0 - p1)} + C_5 \sqrt{gs p0} + C_6 \sqrt{gs p1}$$

We would require two subtractions, three multiplications, and three square roots just to compute the coefficients. An alternative would be to store $gs(1-p0-p1)$, $gs\,p0$, $gs\,p1$ and their square roots $\sqrt{gs(1 - p0 - p1)}$, $\sqrt{gs\,p0}$, $\sqrt{gs\,p1}$ in the codebook. As long as the capacitance switched to access the wider codebook (ROM) is less than the capacitance switched to access gs, $p0$, $p1$ and perform the computations to produce the coefficients, this would be a good alternative.

4.2. Common Subexpression Elimination

Many optimizing compilers perform common subexpression elimination as a means of reducing the execution time of programs. Here we use it to reduce the total average switched capacitance [5]. The method used here is to first make as many substitutions as possible, multiply out the expression, then perform common subexpression elimination, and check to see if the total average switched capacitance has been reduced. Typically we would examine the high switched capacitance operations first such as squareroots, divides, and multiply-accumulates.

In some cases this technique has been shown to minimize energy by minimizing the size of memory. For example consider the following code taken from a radar signal processing application [29].

$$b_n = b_{n-1} + a_{(n-(K+1)/2)} - a_{(n-(M+K+1)/2)} - a_{(n+(K-1)/2)} + a_{(n+(M+K-1)/2)}$$

We can represent the subtraction of the second and third terms on the right hand side by $s(n-(K+1)/2)$, where $s(n-(K+1)/2) = a_{(n-(K+1)/2)} - a_{(n-(M+K+1)/2)}$. Similarly we can also represent the subtraction of the last two terms in the equation for b_n by $s(n+(M+K-1)/2)$. The equation now looks like

$$b_n = b_{n-1} + s(n - (K+1)/2) - s(n + (M+K-1)/2)$$

Thus in each iteration in which we compute b_n we do not need to access 5 values from memory, only 3 values from memory (b_{n-1}, $s(n-(K+1)/2)$ and $s(n-(M+K-1)/2)$). Instead of performing four addition/subtraction operations we only need to perform three addition/subtraction operations. Using common subexpression elimination we need to store only $M/2 + K + 1$ values (the difference of the indices of the two $s()$ terms in equation for b_n, plus one for the storage of b_{n-1}) instead of $M+K+1$ (the difference of the indices of the fourth or last and the second $a()$ terms in the equation for bn, plus one for storage of b_{n-1}). In this application $M \gg K$, so the energy savings is mainly due to the capacitance of the smaller size of memory and fewer memory accesses.

However in the following example taken from the audio compression algorithm we can use this technique to reduce energy by reducing the number of computations (as opposed to mainly memory size alone). Consider the example below where we have to perform the computations in (6.1) and (6.2) in order to produce the variable R_m^k.

$$q1_{k,m}(n) = q_{k,m}(n) - K\gamma_m^k b(n), \forall n, m, k. \quad (6.1)$$

$$R_m^k = \sum_{n=0}^{n=39} q1_{k,m}(n)\,p(n), \ \forall m, k. \quad (6.2)$$

First we substitute $q1_{k,m}(n)$ into R_m^k and multiply out as shown below in (7).

$$R_m^k = \sum_{n=0}^{n=39} [q_{k,m}(n)p(n)] - \sum_{n=0}^{n=39} [K\gamma_m^k b(n)p(n)], \ \forall m, k. \quad (7)$$

In this example, earlier in our application the last expression was already computed as C as shown below in (8).

$$\textit{Where this was precomputed } C = \sum_{n=0}^{n=39} b(n)\,p(n) \tag{8}$$

Therefore we can substitute to obtain equation (9).

$$R_m^k = \sum_{n=0}^{n=39} [q_{k,m}(n)\,p(n)] + K\gamma_m^k\,C,\ \forall m, k. \tag{9}$$

Equation (6.1) requires $2mnk$ MACs (multiply-accumulate operations) and (6.2) requires n MACs for a total of $2mnk + n$ MACs. However equation (9) requires only $mk + 1 + n$ MACs.

4.3. Loop Merging and Bin Packing

To avoid the use of off-chip memory in the design of architectures for embedded applications, scheduling techniques must be extremely memory efficient. This is especially true for single chip systems since the added parallelism in the data-paths (used to obtain significant voltage scaling) will increase the area of the chip by adding more functional units, leaving less space on the chip for on-chip memory.

Consider the example in Figure 2. In Figure 2a) mathematical equations representing part of the application are given. We assume that the synthesized architecture will have a number of multiply-accumulate functional units. These are transformed directly into loops shown in Figure 2b). The three loops are merged in Figure 2c). This loop merging has reduced the memory requirements and number of memory accesses (by using multiply-accumulate functional units). We refer to this type of loop structure as a nested dependent loop since the number of loops performed by the i loop depends on n which is controlled by the outer loop. If this were not the case we would say that the nested loops would be independent. The loop

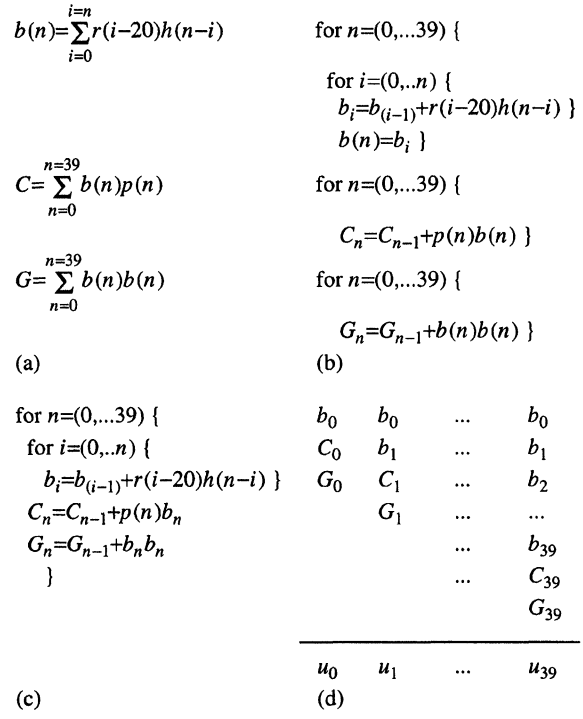

FIGURE 2 In (a) the mathematical equations are transformed directly into loops in (b), which are merged in (c), and then unrolled in (d).

structure in Figure 2c) could be executed by one MAC unit which would perform the b_i calculation for all i and then the result could be broadcast and stored in the accumulator of two other MAC units which would then in one clock cycle update C_n and G_n.

In order to extract more parallelism from this nested dependent loop structure we unroll the loops into strands of multiply accumulates as shown in Figure 2d). For example b_0 to b_1 in the second column represents the computation of the i loops for $n = 1$. The scheduling of this type of loop structure is equivalent to the bin packing problem. We refer to these columns of computations in Figure 2d) as integers where $I(u_n) = 3 + n$ for $n = 0$ to 39. These integers must be assigned to a bin such that each bin has the sum of it's integers $\leq B$. The bin packing problem [7] is to assign all the integer to bins and minimize the total number of bins required. Here the number of bins is the

number of MAC units in the architecture and B is the time to execute the loop structure.

There are very fast and efficient, pseudo-polynomial time algorithms for solving the bin packing problem which are sufficient for our synthesis problem. This preprocessing, scheduling the loops for a range of bins, is performed to characterize the computation over a range of 2 MACs to 8 MACs. If the computations are a nested independent loop structure, then the loop is unrolled a number of times specified in [25]. More sophisticated loop optimization techniques can be found in [30] [31].

4.4. Data Regeneration

Data regeneration is a technique introduced in this paper for both single chip and multichip systems. For single chip synthesis it is used to reduce the size of the memory requirements so that external memory can be avoided. Multichip systems use this technique to minimize the number of off-chip transfers. It is somewhat tied to an application however once it is discovered in the application it may be automatically used during scheduling.

For example for single chip systems consider the task and operation flow graphs in Figure 3, and defined below. The purpose of each set of operations, denoted by a circle, is given below in equations (10–13) labeled by the name in the circle: In Figure 3a) task T1 is composed of operations *od* and *o*1, task T2 is composed of operation *o*2 and task T3 is composed of *og* and *o*3. Task *o*1 generates i^* for operation *og* or *regen* (this arc is not shown in Fig. 3). *og* generates $f_{i^*}(n)$ for operation *o*3. The two arrays, $\theta_{i,m}$ and $v_m(n)$, are stored in ROM on the chip.

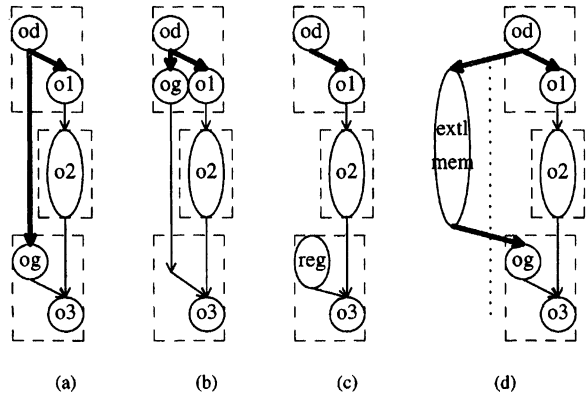

FIGURE 3 Bold arrows indicate large amount of data to be stored between tasks (identified by dashed boxes). Each task has circles representing groups of operations. Original task flow graph in (a) is modified in (b). Alternatively data can be regenerated by *regen* in (c) or stored in external memory in (d).

$$od: \quad q_m(n) = G(v_m(n), h(n)), \tag{10}$$

$$o1: \text{Find } i = i^*; \text{which minimizes } F(\theta_{i,m}, q_m(n)) \tag{11}$$

$$og: \text{Given } i^*, f_{i^*}(n) = \sum_m \theta_{i^*,m} q_m(n) \tag{12}$$

$$reg: \text{Given } i^*, f_{i^*}(n) = G(u_{i^*}(n), h(n)), \tag{13}$$

$$\text{where } u_{i^*}(n) = \sum_m \theta_{i,m} v_m(n)$$

In general tasks *od* and *regen* use the $G()$ function to represent the convolution with $h(n)$. Equation (13) is derived by changing the order of operations in the definition of $f_{i^*}(n)$ so that only i^* requires storage (since $\theta_{i,m}$ and $v_m(n)$ are stored in ROM). For example we can write $f_{i^*}(n) = \sum_m \theta_{i^*,m} q_m(n) = \sum_m \theta_{i^*,m}[G(v_m(n), h(n))]$ which can then be transformed to $G([\sum_m \theta_{i^*,m} v_m(n)], h(n))$. We use the last form to calculate $f_{i^*}(n)$, since it avoids performing m convolutions (first equality), or avoids storing $q_m(n)$, or storing $f_{i^*}(n)$ until it"s later use in task T3 (as in Figure 3a) and 3b) respectively). Note that we assume that there is no significant loss of accuracy (although in practice this would be verified or the word length would be increased to avoid accuracy problems). This new representation avoids performing m convolutions to obtain $q_m(n)$ (using equation (1)), and then summing these results to obtain $f_{i^*}(n)$ (as in task *og*). It instead first sums the values (obtaining $u_{i^*}(n)$) and then performs one convolution on the summation to obtain $f_{i^*}(n)$. In Figure 3a) $nm+1$

($q_m(n)$ and i^*) values must be stored during the execution of task T2, indicated by bold arrow and equation (*od*). In Figure 3b) if we move a code operation *og* from task T3 to T1 then we only need to store $n(f_{i^*}(n))$ values. Finally in Figure 3c) it may be possible, as in this example, to regenerate the data, $f_{i^*k}(n)$ by storing only one data value (i^*). Alternatively one may choose to store the larger array in external memory. The decision of which of Figure 3a) through d) to choose, would depend upon how much memory could fit on one chip. Given this constraint, in order to minimize the estimated energy dissipated the remaining problem would be to tradeoff the capacitance switched by performing the extra computations detailed in operation *regen* over those in operation *og* versus the capacitance switched by reading and writing the data from/to the on-chip or external memory. For example Figure 3d) may be advantageous over Figure 3a) if the on-chip memory size is so large that fabricating the chip for Figure 3a) would be too expensive.

Data regeneration in multichip systems is used to minimize the total number of data words transferred off of the chip, d (from equation (3)). For example in Figure 4, if the task T1 must transmit data to a different chip then we do not want to transfer q (representing $q_m(n)$) words as shown in Figure 4b). In this case we would prefer to transfer fewer words by regenerating the data on the other chip as shown in Figure 4c or by performing task duplication as in Figure 4a).

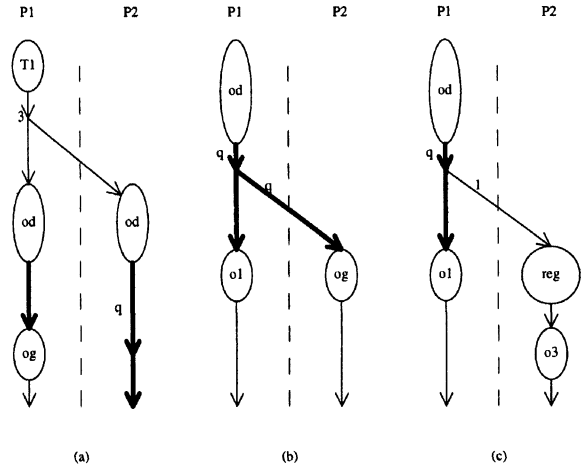

FIGURE 4 Task Duplication for multichip systems in (a) requires only 3 units of data to be transferred from one chip (P1) to the other (P2). The scheduled tasks in (b) indicate that a large array of words (q) is transferred from P1 to P2. Data regeneration is illustrated in (d), requiring only one word to be transferred. Task *reg* regenerates the required array using a different calculation than *od* on P1.

4.5. Scheduler in this Paper

The scheduler used in this paper is based upon mathematical programming and is used for both the single and multichip system. Although this typically takes more cpu time than list schedulers the results are optimal. Since the paper presents a new methodology and an initial analysis of power in multichip systems it was appropriate to use optimal schedules so that the results would not be a function of how good the scheduler was. Due to space limitations the model is not presented in this paper but it is an extension of previously published models [21, 27], for modeling energy, and simultaneously deciding whether to perform task duplication and can be found in [22]. Loop unrolling or loop unfolding is used along with loop winding or software pipelining to minimize latency.

The scheduler performs simultaneous scheduling and allocation of an energy optimized architecture. The input to the scheduler is the transformed TFG after precomputation, common subexpression elimination, and data regeneration decisions have been made by the user. Since it is often not clear how to make these decisions the methodology is currently an iterative procedure. In single chip systems the IP model allocates functional units and memory on the chip and energy is minimized using voltage scaling techniques, see equation (4). For multichip systems, the IP model optimally decides whether and where it should use task duplication to minimize energy simultaneously with scheduling, allocation and partitioning. The number of inter-chip word transfers, the total average capacitance switched on each chip, and the voltage scaling is used to formulate energy minimization as pre-

viously given in equation (5). This model is an extension of the integer programming models in [27] which optimized multichip architectures for area or interchip busses. The simultaneous scheduling and allocation is performed simultaneously on the computations within each task. Specifically only the unrolled nested independent loops and the random data flow graphs are scheduled. Several loops in the application with the same amount of nesting, same number of MACs, and number of indices are only scheduled once and their computation time is multiplied by how many instances of this loop structure is in the application. The precomputed times for the nested dependent loops are used. The constraints involve scheduling and allocation constraints based upon variables for the set of operations in each task and new constraints (15 – 17) given in appendix A are used for modeling energy. The extensions to the multichip IP model are given in appendix B to support task duplication.

4.6. Power Management and Dynamic Voltage Scaling

The previous transformations and scheduling and allocation were performed under worst case application conditions. In other words that the largest number of computations will be performed so that when we scale down the voltage, the throughput conditions will be net under all circumstances. However there still may be some idle time during which a task is not being executed on the chip when certain parts of the application are deactivated or fewer code operations are performed. To save energy it may be possible to shutdown the chip using energy management unit [28]. However this may not be cost effective or possible unless it is idle for a very long duration of time (due to time required to energy down and then restart the processor). Alternatively the clock could be dynamically switched to a lower frequency, thereby saving on capacitance (where the average estimated switched capacitance term in (equation (1) would be reduced) being switched by the clock [17]. However if it is possible to use this idle time by dynamically lowering the supply voltage and executing tasks slowly at this lower supply voltage [14], this would save more energy than only lowering the clock frequency (since here the supply would not be lowered).

In multichip systems given a voltage-scaled schedule that meets our timing constraints we first merge tasks where possible, and then determine which merged tasks can operate at even lower supply voltages. Merging of tasks is performed to try to minimize the number of times one must dynamically switch the voltage. Tasks are merged in the schedule whenever there are no off-chip or on-chip transfers inbetween adjacent tasks of the same chip. The capacitances and cycles of execution of the tasks are summed together. The new task-merged TFG is then further modified. The order of tasks in each chip in the schedule can be represented by adding new arcs in the task flow graph. For example if the following tasks are executed by chip 1 in order of earliest to latest time, ta, tb, tc, td then the following three arcs ta $\rightarrow$tb$\rightarrow$tc$\rightarrow$td must be added to the TFG. The critical path method [11] is performed on the TFG to determine the earliest time that a task may start its execution (asap) and the latest time (alap). Tasks whose asap times are less than their alap times are placed in a task list. For each task in the list the maximum energy savings is calculated (by slowing the task down to asap$-$alap $+1$-$cyc(t,p)$). The task with the largest energy savings is chosen for DVS and removed from the list. Then the asap/alap values and energy savings in the TFG are updated and the process is repeated. For example in Figure 1c) we merge T2 and T3, T2-T3, but we cannot merge T6 because it is separated from T2-T3 by an off-chip data transfer. The new merged task T2-T3 is dynamically scaled first to it's maximum amount, followed by T6.

5. EXPERIMENTAL RESULTS

A video compression and audio compression algorithm is used to illustrate the low energy

methodology. The video compression algorithm, DCT-II [8], is represented as a random data flow graph of over 43 operations. A large real complex industrial audio compression algorithm [20], VSELP, donated by Motorola, is composed of over 123,000 multiply accumulate operations which dominated the application. These computations compress one frame of speech (160 speech samples). The VSELP application was partitioned into tasks and the computations in the tasks were partitioned into random data flow graphs and independent and dependent nested loops. Unfortunately there is limited research on energy savings in systems so it is not possible to make a comparison with previous research. However we have used a realistic industrial example and optimal schedules to explore the new methodology and initial energy savings. In both examples the energy models described in previous research with modification described in this paper were used. All IP problems were solved on a IBM RS/6000 workstation, and the plots show estimated energy (plotted in terms of E-x Vsq.F representing ten to the power of negative x volts squared times farads).

The results of simultaneously scheduling and allocating a low energy architecture for the DCT-II application are shown in Figure 5. The points in the graph were obtained by varying area of the architecture and minimizing latency. For each point the estimated energy was calculated and plotted. This was done only for illustration purposes. However the IP model was only solved once to find the architecture with a minimum estimated energy (see equation (4)) over the complete design space (spanned by area and latency). The DFG was scheduled using the integer programming model which performed loop winding. The energy optimal architecture was obtained by solving over the entire design space with fixed throughput constraint where only $y_{T,f,r,o} \varepsilon \{0,1\}$. The scheduling variables $x_{j,k}$ were continuous. After the architecture was determined by execution time and number of functional units a detailed scheduled was then obtained by solving the reduced IP problem. Two energy models were used to illustrate the efficiency of the search over different energy spaces. The circle and square points in Figure 5 show the design space given by capacitance model 1 and model 2. Capacitance model 1, circle points, uses a RAM capacitance model for a centralized controller, whereas model 2 used the distributed controller capacitance model in [10]. The IP problem had over 3000 variables and over 3000 constraints. Note that in the first design space, circle points, there is more than one local optimal point, yet our IP approach was able to find the globally optimal solution in only 654 cpu seconds. The reason for the local optimum can be seen if we examine the plotted points in terms of (total architecture capacitance, scaled operating voltage). The last four points representing the larger chip areas are detailed as (0.7945 E-9,2.3 V), (0.862 E-9,2.0 V), (1.38 E-9,1.8 V), (1.85 E-9,1.5 V) showing the chip capacitance increasing and the scaled voltage decreasing. The second capacitance model created a design space with only one local optima. The IP problem found this architecture in 25 cpu seconds. The final architecture was obtained in an additional 3 cpu sec.

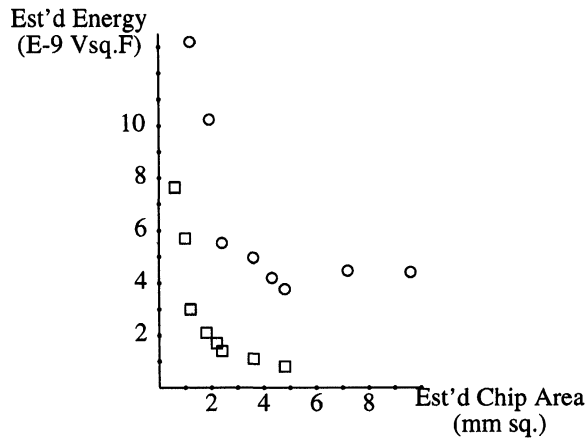

FIGURE 5 Estimated Energy Model 1, circles, and Model 2, squares, for DCT application plotted versus estimated chip area, using IP Synthesized Architectures and Voltage Scaling (4.3V to 1.5V).

Figures 6 and 7 illustrate the energy (using voltage scaling techniques) versus single chip area design space of the VSELP application. In Figure

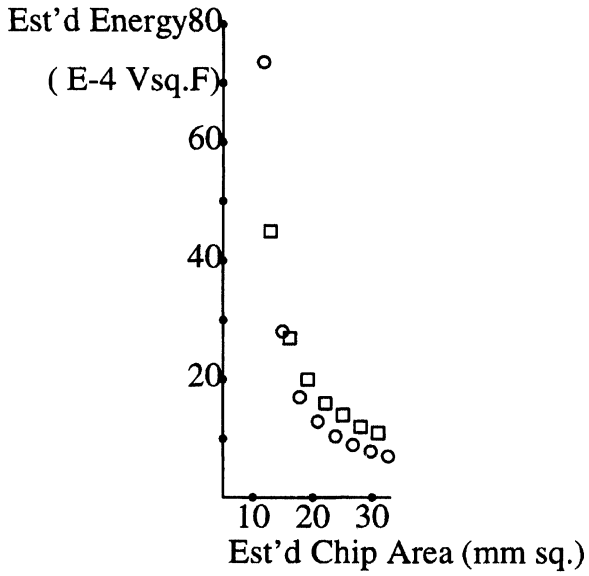

FIGURE 6 Estimated Energies of VLSI architectures for loop-transformed VSELP, square points, and lopp-energy-transformed VSELP, circle points, plotted versus estimated Area. Not shown is square at (10,117).

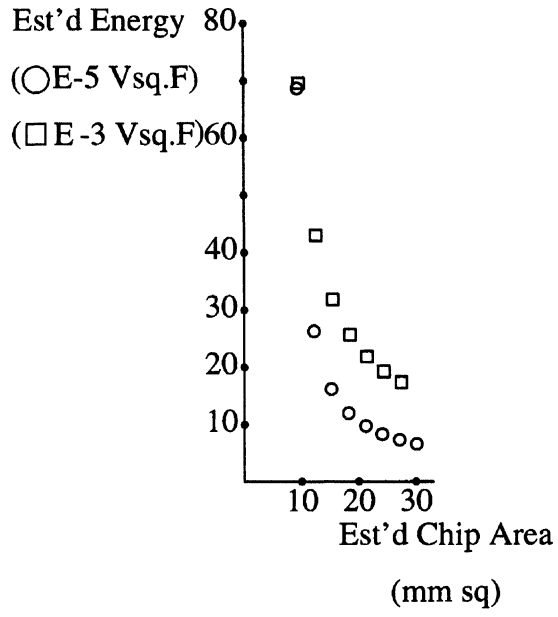

FIGURE 7 Energies for VSELP using data regeneration, circle points, and external memory, square points, plotting estimated Energy versus estimated Area. Not shown is square at (6.6,182.E-3).

6 the circle points represent the alogrithm after the energy transformations were performed. Specifically precomputation saved 3972 MACs which did not have to be performed and the approach to common subexpression elimination saved approximately 69,632 MACs. Both savings are on a per frame basis. The savings in energy is shown in Figure 6 since the square points represent the original algorithm without these savings. On average the energy savings was 1.6 times. The energy savings provided in total by these transformations and voltage scaling is 10 times at a cost of 2.7 times due to the increase in the chip area. Figure 7 illustrates the savings based upon the data regeneration technique presented in this paper. The data regeneration added 4260 MACs to the application but allowed the architecture to use on-chip memory. The alternative system used external memory had fewer MACs and on-chip RAM but had larger energy dissipation. We assumed in as in [4] that the energy dissipation of an off-chip external memory read/write is ten times that of an internal memory read/write. Also the energy computed for the external memory system is the energy for the VLSI chip along with the energy of the external memory.

The architectures were also completed using the COMPASS design automation assistant [24]. The area estimate based upon the model in [6] which we used in this methodology was shown to be within 36% of the area estimates in the COMPASS tools. The synthesized architecture with two MAC functional units and the one with eight MAC units, circle points in Figure 8 were also studied in the COMPASS Design Automation system. The activities of the MAC units and memory units were set in COMPASS to represent the amount of work done over the sample period. For example for the 8 MAC unit architecture the switching activity of each MAC was equivalent to one quarter the activity of a MAC unit in the 2 MAC architecture. The power measurement taken from the COMPASS tools was then transformed into energy and scaled voltages were used. The energy improvement was the same as in the model

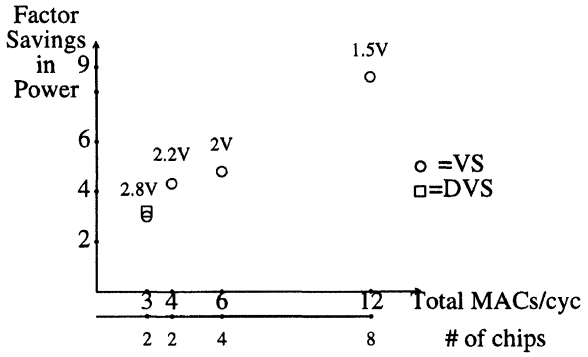

FIGURE 8 Graph of energy savings, for VSELP #1 algorithm, versus MACs/cyc and number of chips in system, using (dynamic) voltage scaling.

used in this paper which was an energy improvement of about four times. Thus the area and relative estimates used in this paper were verified to be practical.

For multichip systems analysis the VSELP application was transformed manually into a task flow graph of 68 tasks, which represented the computations performed on one frame of speech (160 speech samples). A second TFG was also extracted manually which processed a subframe of speech (40 speech samples) and it had 16 tasks. Each task was defined to have parameter *cyc* (*t*,*p*) equivalent to the number of multiply accumulate operations that needed to be performed (MACs), since these dominate for time and power [19]. These TFGs were then used in the methodology to study the energy savings possible in a multichip system. The energy savings were computed relative to a single chip implementation (which is currently how the VSELP is implemented [20]). The audio compression applications task flow graphs were further transformed into two algorithms, VSELP #1 and VSELP #2. Algorithm #1 uses the actual filtered speech of each subframe to update certain parameters of the algorithm for the next subframe of processing, where as algorithm #2 uses the synthesized speech for updating as specified in [26]. Apart from this both algorithms are identical. There is significantly more parallelism in algorithm #1 than algorithm #2. The factor of energy savings is calculated with respect to a single chip implementation of the two algorithms operating at 5V which just meets the real time constraint. Data regeneration was determined manually and the IP model performed simultaneous scheduling, allocation, partitioning, task duplication, and loopwinding. The graphs in Figure 8 and 9 show the factor of energy savings. The voltages shown in Figure 8 (1.5 V–2.8 V) are the system supply voltage for all chips. Task duplication provided on average 5–6% savings in energy (or 1.05 to 1.06 factor of savings in energy). DVS added at most two other supply voltages to the system provided 5–6% extra savings in energy. The loopwinding schedules obtained 2.5 times and 2.8 times savings in energy compared to 3 and 3.2 times savings obtained from the loop unrolling schedules (shown for two chip systems in Fig. 8). The other points in the Figure 8 use the loop winding technique. Figure 9 shows the energy savings for two chip systems. We refer to one chip as *sw*, capable of 1MAC per clock cycle and the other chip as *hw* which can perform in parallel 2MAC per clock cycle. The data regeneration presented in this paper provided up to 51% savings in energy for the results in Figure 9. The speed ratios illustrate the energy savings if we could speed up one or both of the chips by 1,2,4 or 6 times (perhaps using a more expensive VLSI technology, with similar average switched capaci-

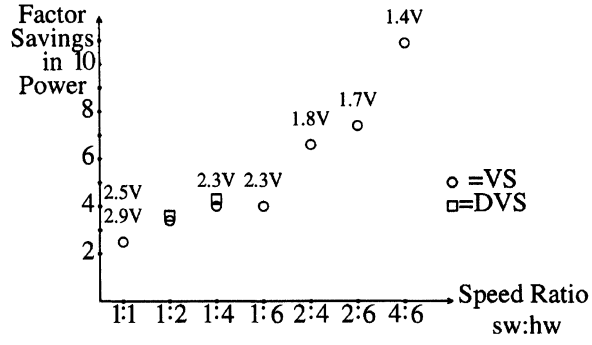

FIGURE 9 Power savings for two chip (*sw*, *hw*) system using VSELP #2 algorithm. The clock speed varies as shown on *x* axis, for example 2:4 means that the clock speeds are increased by 2 times on the *sw* chip and 4 times on the *hw* chip.

tances and supply requirements). This approach was studied for algorithm #2 due to it's lack of parallelism in the computations.

6. DISCUSSIONS AND CONCLUSIONS

In summary energy savings of up to a factor of 10 (see Fig. 6) was attained with an area increase of only 2.7 times. The data regeneration technique along with the loop merging techniques were found to have an effect on memory size requirements and a very significant impact on energy dissipation. The precomputation and common subexpression elimination techniques were found to provide 1.6 times savings in energy. Two of the synthesized VSELP architectures were mapped to a 0.8 micron CMOS standard cell library and area and power were measured using the COMPASS design automation tools. By mapping the power measures from the COMPASS design automation assistant into energy with scaled voltages, an improvement of 4 times in energy was obtained which was equivalent to that measured with the theoretical models [5, 10, 6] of this methodology.

Dynamic voltage scaling was experimentally used in this paper for multichip systems only. We could have performed dynamic voltage scaling using a project scheduling algorithm(chapter 13 in [28]), however it requires constant power savings per unit time a task is slowed down. Unfortunately the power savings is not linear. Nevertheless this algorithm may be useful to study estimated power costs versus time for non critical applications. It is interesting to note that the dynamic voltage scaling did not provide significant power savings. Since it comes at the expense of extra circuitry to raise and lower the supply voltage of each chip it may not be cost effective according to the results in this paper. Although we do consider memory during the partitioning for multichips systems, it did not have a significant impact on the chip area since typically the tasks using the common data structures ended up on the same chip. As found elsewhere in the literature [19] there was more area on-chip for memory to avoid use of off-chip external memory transfers.

Using multichip systems for implementation of the VSELP application, powr savings of up to a factor of 8 (see Fig. 8) to 10 (see Fig. 9) were calculated using system in conjunction with voltage scaling. Task duplication and dynamic voltage scaling techniques each contributed approximately 5–6% power savings. Loop unrolling provided 14–20% power savings over loop winding scheduling techniques. The data regeneration technique introduced in this paper contributed up to 51% savings in power. The contributions of this paper include presenting a new extension for energy minimization to IP approaches to synthesis and the new application of task duplication and data regeneration to synthesizing energy minimized multichip systems.

We have introduced a methodology for energy minimization in large realtime compute intensive applications. The methodology incorporates a hierarchical approach to representing large applications and uses bin packing algorithms to preprocess loop computations. The IP approach finds optimal architectures in reasonable amounts of cpu time due to the hierarchical approach for task flow representation and tight design space of the scheduling subproblem [21] in IP model itself. A combination of transformations to reduce the average switched capacitance by reducing the number of operations, and reducing the memory requirements was shown to provide over 1.6 times improvements in energy. Using a large complex real industrial example, audio compression algorithm donated by Motorola [20], it was shown that systems could provided up to 10 times savings in energy at only an increase in chip area of 2.7 times. This is especially exciting since this type of application is very complex and has been described as very difficult to optimize for energy [3]. Secondly this application is targeted for portable digital cellular phones so a implementation should provide a significant decrease in battery size, packaging costs and perhaps reliability. This research is important for industry since energy

dissipation consideration during these early stages of design are critical to ensuring that the final product will be cost effective, competitive, and meet high performance requirements.

Future research involves extending the optimization models to simultaneously decide whether to use data regeneration, and other higher level transformations such as precomputation during the low energy architecture search and to incorporate minimization of the total number of memory accesses. This research was supported in part by grants from NSERC. The author would like to thank COMPASS Design Automation Inc, San Jose, CA, for their generous donation of the COMPASS Design Automation tools. Thanks also goes to Motorola Inc, DSP Operations, Austin, Texas, for their donation of the VSELP application. In addition the author would like to thank the reviewers and guest editor, Dr. F. Najm, for their useful suggestions.

References

[1] Keutzer, K. (1994). "The Impact of CAD on the Design of Low Power Digital Circuits", *IEEE Symposium on Low Power Electronics*, pp. 42–45.

[2] Wuytack, S. *et al.* (1994). "Global Communication and Memory Optimizing Transformations for Low Power Systems", *International Workshop on Low Power Design*, pp. 203–208.

[3] DeMan, H., Discussion at Panel on "Where Does the Power Go?", at *International Symposium on Low Power Design*, April 1995.

[4] Bunda, J., Athas, W. and Fussell, D. (1994). "Evaluating energy implications of CMOS microprocessor design decisions", *International Workshop on Low Power Design*, pp. 147–152.

[5] Chandrakasan, A. *et al.*, "Optimizing Power Using Transformations", *IEEE Transactions on CAD.*, Jan 1995, **4**(1), 12–31.

[6] Conte, T. *et al.*, "A Technique to Determine Power-Efficient High Performance Superscalar Processors", HICSS, Jan 1995, pp. 324–333.

[7] Garey and Johnson (1979). **Computers and Intractability**, New York Freeman and Co.

[8] Rao and Yip (1990). **Discrete Cosine Transform** Academic Press.

[9] Landman, P. and Rabaey, J. (1994). "Black-Box Capacitance Models for Architectural Power Analysis", *Int''l Workshop on Low Power Design*, pp. 165–170.

[10] Mehra, R. and Rabaey, J. (1994). "Behavioral Level Power Estimation and Exploration", *Int''l Workshop on Low Power Design*, pp. 197–202.

[11] Horowitz, M., Indermaur, T. and Gonzalez, R. (1994). "Low-Power Digital Design" *IEEE Symposium on Low Power Electronics*, pp. 8–11.

[12] Tiwari, V., Malik, S. and Wolfe, A., "Power Analysis of Embedd Software; A First Step Towards Software Power Minimization", *IEEE Trans on VLSI*, **2**(4), Dec 1994, 437–445.

[13] Chandrakasan, A., Burstein, A. and Brodersen, R., "A Low-Power Chipset for a Portable Multimedia I/O Terminal", *IEEE Journal of Solid State Circuits*, Dec 1994. pp. 1415–1428.

[14] Macken, P. *et al.* (1990). "A voltage reduction technique for digital systems", *ISSCC*, pp. 238–9.

[15] Raje, S. and Sarrafzadeh, M. (1995). "Variable Voltage Scheduling", *International Symposium on Low Power Design*, pp. 9–13.

[16] Argade, P. *et al.* (1993). "Hobbit: a high performance low energy microprocessor", COMPCON, pp. 88–95.

[17] Bechade, R. *et al.* (1994). "a 32b 66 MHz 1.8W Microprocessor" *ISSCC*, pp. 208–209.

[18] Biggs, T. *et al.* (1994). "A 1Watt 68040-compatible microprocessor", *International Symposium on Low Power Electronics.*, pp. 12–13.

[19] Powell, S. and Chau, P. "A Model for Estimating Power Dissipation in a Class of DSP VLSI Chips", *IEEE Trans on Circuits and Systems*, **38**(6), June 1991.

[20] Sunwoo, M. and Park, S. "Principles of VSELP Speechcoder and its implementation on the DSP56156", Motorola Inc, *DSP Operations*, Austin, Texas.

[21] Gebotys, C., "Throughput Optimized Architectural Synthesis", *IEEE Trans on VLSI*, **11**(3) Sept 1993, 254–261.

[22] Gebotys, C., "ILP model for Simulataneous Scheduling and Partitioning for Low Power System Mapping", *Dept. of ECE, VLSI group, Tech. rept.*, April 1995.

[23] Beerel, P. A., USC, Panel Presentation on "Where Does the Power Go?" at *International Symposium on Low Power Design*, April 1995.

[24] The Design Automation Assistant, COMPASS Design Automation Inc, San Jose, CA. 1994.

[25] Parhi, K. K. and Messerschmitt, D. G., "Static Rate-Optimal Scheduling of Iterative Data-Flow Programs via Optimum unfolding", *IEEE Trans on Computers*, **40**(2), February 1991.

[26] Hwang, J.-J., Chow, Y.-C., Anger, F. and Lee, C.-Y., "Scheduling Precedence Graphs in Systems with Interprocessor Communication Times", *SIAM J. COMPUT.*, **18**(2), 244–257, April 1989.

[27] Gebotys, C., "An Optimization Approach to the Synthesis of Multichip Architectures", *IEEE Trans on VLSI*, **2**(1) March 1994, 11–20.

[28] Lawler, E. (1976). **Combinatorial Optimization Networks and Matroids.**, Holt, Rinehart and Winston.

[29] CFAR Radar Signal Processing Application Discussions with Stopford, R. Litton Systems Canada Ltd, 1993.

[30] Wolfe, M. (1989). "Optimizing Supercompilers for Supercomputers", *Research Monograph in Parallel and Distributed Computing*, The MIT Press.

[31] Moldovan, D. I. (1993). "Parallel Processing from Applications to Systems", Morgan Kaufmann.

APPENDIX A

New constraints and variables which are used in the energy-minimized IP model [22] for single chip systems [21] are defined below. The new binary variable introduced is: the $y_{\text{Texec,numfu,numram,}}$

$_{\text{numrom}}$ variable (or $y_{T,f,r,o}$). The single chip parameters are used:

N_{numfu} = variable used in the scheduler to represent the number of functional units in the architecture

$N_{\text{numram,numrom}}$ = variable used in the scheduler to represent the number of RAMs and ROMs in the architecture

TotalTime is the execution time of the application in the synthesized architecture which is defined as the time to execute each task multiplied by the number of times that task is computed over the sample period.

TimetoCompute(t) is the time to perform the computations of each task. It is determined by the time for each schedule of the random DFGs and each set of unrolled independent/dependent loops. The following constraints define the y variable which models energy.

$$\sum_{\text{Texec,numram,numrom}} numfu^{*} y_{\text{Texec,numfu,numram,numrom}} = N_{\text{numfu}} \tag{14}$$

$$\sum_{\text{Texec,numfu}} mem^{*} y_{\text{Texec,numfu,numram,numrom}} = N_{\text{numram,numrom}} \tag{15}$$

$$\sum_{\text{Texec,numram,numrom,numfu}} (Texec) y_{\text{Texec,numfu,numram,numrom}} = \sum_{\text{Texec,numram,numrom,mumfu}} (TotalTime(numfu)) y_{\text{Texec,numfu,numram,numrom}} \tag{16}$$

The time for each task is determined by the time for each schedule of the random DFG and the unrolled independent loops and the time obtained from the number of functional units in architecture.

$$TotalTime(numfu) = \sum_{t} Num\,of\,Times(t)\,Time\,to\,Compute(t, numfu) \tag{17}$$

$$Time\,to\,Compute\,(t, numfu) = timeDFG(t) + timeloop(t) + timedloop(t, numfu)$$

More details can be found in [22].

APPENDIX B

New constraints and variables which are used in the energy-minimized IP model from [27] for multichip systems are defined below. Specifically the task duplication modifications are supported. Previously in this model the binary variables $x_{j,k,p,c}$ equaled one if operation k started execution at time j on-chip p and sent data off-chip $c = off\text{-}chip$ or kept it"s data on-chip $c = on\text{-}chip$. The parameters to be used to illustrate the constraints are $outdegree(k)$ = the number of arcs which originate from operation k and go to another operation in the DFG. The following modification of the variable assignment constraint (18–19) and addition of the task duplication constraint is performed (20–21).

$$\sum_{p,c,j} x_{j,k,p,c} \geq 1, \forall k \tag{18}$$

$$\sum_{p,c,j} x_{j,k,p,c} \leq outdegree(k), \forall k \tag{19}$$

$$\sum_{p1,j1} x_{j1,k,p1,\text{off-chip}} + \sum_{j} x_{j,k,l,p,\text{on-chip}} \leq 1, \quad \forall outdegree(k) = 2, p \tag{20}$$

Constraint (20) says that if data is transferred off of one chip then there is no need to also duplicate the task on a different chip.

$$\sum_{p1,j1} x_{j1,k1,p1,\text{off-chip}} + \sum_{j} x_{j,k1,p,\text{on-chip}} + \left(1 - \sum_{j,c} x_{j,k,2,p,c}\right) \geq 1 \forall k1 \rightarrow k2, \quad outdegree(k1) \geq 2, p \tag{21}$$

Constraint (21) says that if $k1$ does not transfer data off of any chip and onto chip p then either $k1$ is executed on-chip p or $k2$ is not executed on-chip p.

Authors' Biography

Catherine H. Gebotys received the B.A.Sc., degree in engineering science in 1982 and M.A.Sc., degree in electrical engineering in 1984 both from University of Toronto, Toronto, Ont., Canada. She received the Ph.D., degree in electrical engineering in 1991 from the University of Waterloo, Waterloo, Ontario Canada.

She worked at Litton Systems Canada Ltd., Display Systems Engineering, from January 1985 to December 1986 in the area of CAD for VLSI and chip design. From January 1987 to August 1989 she was a Research Associate in the VLSI Group, Department of Electrical Engineering, University of Waterloo, Waterloo, Ontario, Canada. She is an Associate Professor in the Department of Electric and Computer Engineering, University of Waterloo, Waterloo, Ontario, Canada. She has published a number of research papers in the high-level synthesis area: global optimization approaches to CAD for VLSI, high-level architectural synthesis, low energy system synthesis, and retargetable code generation for DSP cores; and is coauthor of the book Optimal VLSI Architectural Synthesis.