J Sign Process Syst (2010) 58:145–160
DOI 10.1007/s11265-009-0338-3

Total Power Optimization for Combinational Logic Using Genetic Algorithms

Wei-lun Hung · Yuan Xie · Narayanan Vijaykrishnan · Mahmut Kandemir · Mary Jane Irwin

Received: 29 December 2006 / Revised: 25 December 2008 / Accepted: 8 January 2009 / Published online: 19 February 2009

Abstract Power consumption is a top priority in high performance circuit design today. Many low power techniques have been proposed to tackle the ever serious, highly pressing power consumption problem, which is composed of both dynamic and static power in the nanometer era. The static power consumption nowadays receives even more attention than that of dynamic power consumption when technology scales below 100 nm. In order to mitigate the aggressive power consumption, various existing low power techniques are often used; however, they are often applied independently or combined with two or at most three different techniques together, and that is not sufficient to address the escalating power issue. In this paper, we present a power optimization framework for the minimization of total power consumption in combinational logic through multiple V_{dd} assignment, multiple V_{th} assignment, device sizing, and stack forcing, while maintaining performance requirements. These four power reduction techniques are properly encoded into the genetic algorithm and evaluated simultaneously. The overhead imposed by the insertion of level converters is also taken into account. The effectiveness of each power reduction mechanism is verified, as are the combinations of different approaches. Experimental results are presented for a number of 65 nm benchmark circuits that span typical circuit topologies, including inverter chains, SRAM decoders, multiplier, and a 32 bit carry adder. Our experiments show that the combination of four low power techniques is the effective way to achieve low power budget. The framework is general and can be easily extended to include other design-time low power techniques, such as multiple gate length or multiple gate oxide thickness.

Keywords Power optimization · Genetic algorithm · VLSI circuits

W.-l. Hung · Y. Xie (✉) · N. Vijaykrishnan · M. Kandemir · M. J. Irwin
Computer Science and Engineering,
The Pennsylvania State University,
University Park, PA 16802, USA
e-mail: yuanxie@cse.psu.edu

W.-l. Hung
e-mail: whung@cse.psu.edu

N. Vijaykrishnan
e-mail: vijay@cse.psu.edu

M. Kandemir
e-mail: kandemir@cse.psu.edu

M. J. Irwin
e-mail: mji@cse.psu.edu

1 Introduction

Process scaling and aggressive performance improvements have resulted in power consumption becoming a first-order design criterion. For example, Intel Pentium 4 processor (Prescott, 2004) has a power consumption of 103 W, almost four times larger than that of the Pentium III (1999). Consequently, it has led to the advent of Chip Multiprocessors (CMP) as a viable alternative to the complex and power hungry superscalar architecture. CMPs are simple, compact processing cores forming a decentralized micro-architecture which scales more efficiently with increased integration densities, since each core itself is simple and more power efficient. Nevertheless, a 4-core Intel Core 2 Extreme processor can still have a high power consumption as much as 136 W. The concern of power for portable devices and battery-powered applications is even evident since these applications are normally staying longer in stand-by mode instead of in active mode. Thus, a big portion of power, stand-by power (leakage), is wasted, and

thus largely affects the available operating time. In addition to its clear impact on battery lifetime in portable embedded systems, processor power consumption has also become a primary constraint on workstation performance due to cooling and heat dissipation issues. This situation is even worse in high performance computing where a large number of processors are arranged and ran in parallel to gain high computing capability; however, such arrangement also has huge adverse effects on power. Therefore, reducing power dissipation is a top priority in modern VLSI design.

Power dissipation in CMOS digital circuits consists of dynamic power, short circuit power, and static power. Short circuit power consumption can be kept within bounds by careful design and tuning the switching characteristics of complementary logic (slope engineering). When the rise and fall times of inputs and outputs are equalized, most power dissipation is associated with the dynamic power, and only a minor fraction (<10%) is devoted to short circuit currents [4]. Thus, it is becoming less important in deep submicron technologies, compared to dynamic power and leakage power. Therefore, we will focus on the latter two sources of power consumption, as indicated by Eq. 1.

$$P = A \cdot V_{dd} \cdot I_{peak} \cdot t_s + A \cdot C \cdot {V_{dd}}^2 \cdot f + I_{leak} \cdot V_{dd} \qquad (1)$$

In this equation, the second term is the dynamic power dissipation and the last term models the static power dissipation due to leakage current I_{leak}. (A is the switching activity factor for dynamic power, C is the switched capacitance, and V_{dd} is the supply voltage). Dynamic power was once the dominant power consumption term. However, as the result of technology scaling and V_{th} (threshold voltage) decreasing, leakage power is now accounting for a large portion of total power consumption. Although there are many techniques to reduce power dissipation, most existing efforts focus on one technique in isolation instead of concurrently applying a number of power minimization techniques. In this paper, we propose a power optimization framework [21] based on the genetic algorithm. Our optimization strategy combines four power reduction techniques: multiple V_{dd} assignment, multiple V_{th} assignment, gate sizing, and stack forcing. It simultaneously applies and evaluates the effects of these techniques to achieve maximum power saving under a hard timing constraint. The framework is targeted at combinational logic, and it can be easily extended to include other power reduction techniques. To the best of our knowledge, we are not aware of any other power optimization work that can make use of these four power reduction techniques simultaneously.

The paper is organized as follows. Section 2 reviews different power reduction techniques and previous works. Section 3 describes the delay model used in this work. Section 4 gives a brief introduction to the genetic algorithm. Section 5 describes our power optimization framework based on the genetic algorithm. Experimental results are presented and explained in Section 6 and we conclude this paper in the last section.

2 Power Reduction Techniques and Related Work

Due to the quadratic relationship between dynamic power consumption and V_{dd}, reducing the supply voltage is the most effective way to lower the dynamic power, at the expense of increasing gate delay. In order to prevent the negative effect on performance, the threshold voltage (V_{th}) must be reduced proportionally with the supply voltage so that a sufficient driving current is maintained. This reduction in the threshold voltage causes an exponential increases in leakage power, which in turn can raise the static power of the device to unacceptable levels.

To counter the loss in performance while improving the power efficiency, multiple V_{dd} [1] and multiple V_{th} [2] techniques have been proposed. The gates on critical paths operate at the higher V_{dd} or lower V_{th}, while those on non-critical paths operate at the lower V_{dd} or higher V_{th}, thereby reducing overall power consumption without performance degradation. These techniques have been successfully implemented. For example, IBM's ASIC design flow can fully take advantage of the power-performance tradeoff by using their voltage island concept and multiple-V_{th} standard cell library [3]. Gate sizing [4] is another powerful method of power optimization. Logic gates on critical paths may be sized up to meet timing requirement, at the expense of higher power consumption; while those on non-critical paths can be sized down to reduce the power consumption. Hamada et al. [5] examined multiple supply voltages, multiple threshold voltages, and transistor sizing individually and derived a set of rules of thumb for optimal supply voltages, threshold voltages, and transistor sizing. A good summary of these three techniques is presented by Brodersen et al. [6].

Besides multiple V_{th} technique, another solution, stack forcing, is being used to tackle the ever-increasing waste of leakage power. It has been shown that the stacking of two off transistors can significantly reduce leakage power than a single off transistor [7]. Therefore, we can force a non-stacking device to a stack of N devices without affecting the input load. Figure 1 shows a stacking force example of an inverter with N=2. By ensuring the input load unchanged, we guarantee that the previous stages' delay and dynamic power consumption are not affected. The logic gate with stack forcing has much lower leakage power, however, at the expense of a delay penalty, because the effective device width W_{eff} becomes W/N^2 after stack forcing. The idea of stacking force is thus similar to replacing a low-V_{th} device with a high-V_{th} device in a multiple-V_{th} design.

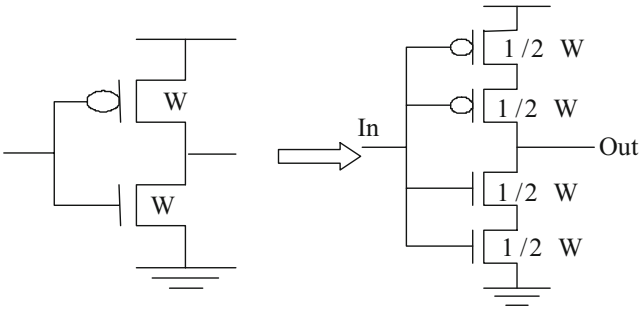

Figure 1 Stacking force example for an inverter.

To achieve the most power efficient design, all these power reduction techniques have to be complementary used. Stojanovic et al. [8] combined gate sizing and supply voltage optimization to minimize power consumption under a delay constraint. Roy et al. [9] presented a heuristic algorithm to combine dual-V_{dd} and dual-V_{th} techniques. Augsburger and Nikolic [10] evaluated the effectiveness of multiple supply voltage, transistor sizing, and multiple thresholds independently and in conjunction with each other, showing that the order of application of these techniques determines the final savings in active and leakage power.

Recently, researchers have been looking at the joint optimization of these techniques [30, 32, 33], since such joint optimization can help to achieve maximum power savings compared to a sequential application of a single variable optimization. Sirichotiyakul et al. [11] presented an algorithm for joint optimization of dual-V_{th} and sizing to reduce leakage power. Karnik et al. [12] developed a heuristic iterative algorithm to do device sizing and dual-V_{th} allocation simultaneously to exploit the timing slack for reduction of total power consumption. They found that joint dual-V_{th} and sizing can reduce the power by 10% and 25% compared with pure V_{th} allocation or pure sizing method, respectively. Srivastava et al. [13] were the first to investigate the effectiveness of simultaneously multiple supply and threshold voltage assignment for total power saving. Their algorithm is based on a linear programming approach. Nguyen et al. [14] developed another linear programming algorithm that can simultaneously perform the threshold voltage assignment and sizing optimization, and then apply the supply voltage optimization as a sequential step. Lee et al. [19] proposed heuristic algorithms for simultaneous state, V_{th}, and gate oxide assignment. Srivastava et al. [20] proposed a sensitivity-based algorithm to perform concurrent sizing, V_{dd}, and V_{th} assignment. Recently, work [31] started investigating the integration of retiming and simultaneous supply/threshold voltage assignment. Their three steps approach showed promising results even when the Flip-Flop delay/power was considered.

Figures 2 and 3 demonstrate how the proposed approach works. For a given circuit, the critical path of the circuit is first identified and is shown as the darkened gates in Fig. 2.

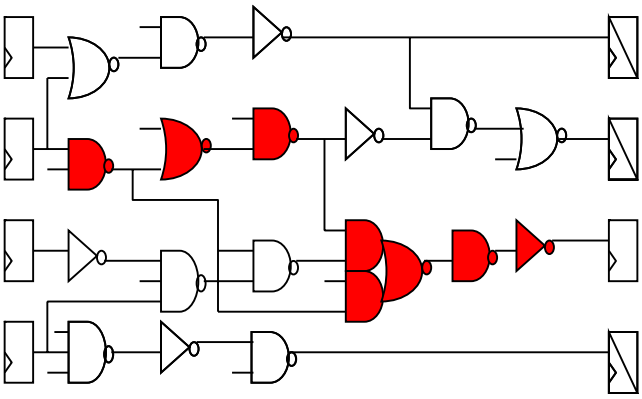

Figure 2 An example circuit with a critical path of darkened gates.

In Fig. 3, the gates in the critical path are assigned with high supply voltage while the others gates are assigned with low supply voltage. For gate A in the non-critical path, the stack forcing is applied to reduce the static power consumption. One level converter is inserted between the output of gate B and the input of gate C to eliminate the undesirable static current. This current flows since the logic *High* signal of the low supply voltage driven cell cannot completely turn off the PMOS pull-up network of the following high supply voltage driven cell. Since gate E is assigned with high threshold voltage, it needs to be sized-up to keep the speed in order to meet the performance requirement. All these low power techniques should be carefully applied; otherwise, the critical path may change due to the complexity of a circuit, thereby increasing the critical path delay.

In this paper, we present a GA-based power optimization framework for combinational logics, and the optimization approach can simultaneously exploit four power optimization techniques: multiple supply voltage assignment, multiple threshold voltage assignment, gate sizing, and force stacking. To the best of our knowledge, the use of all four low power techniques has not been attempted before and none of the previous works can do simultaneously

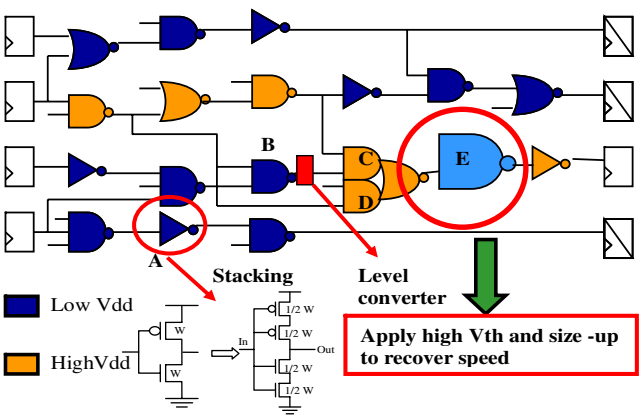

Figure 3 An example circuit with applying four low power techniques.

joint optimization of more than three techniques. The framework is built on top of the genetic algorithm and the power optimization problem is encoded into chromosome representations.

Note that there exists other power optimization framework which used different optimization algorithms. For example, Dutta et al. have proposed a power optimization framework for combinational logic circuits, which is called ASAP and only focused on gate sizing technique, and the approach is based on simulated annealing algorithm [38]; Dabiri et al. also present a convex programming algorithm to study the gate sizing approach for power optimization [37]; and Chi et al. proposed a greedy heuristics to apply multiple supply voltage assignment [36]; Various heuristics have also been proposed to perform power reduction using joint-optimization of gate sizing, multi-Vdd, and multi-Vth [11–14]. However, the drawback of these methods are: 1) Some could only handle one power optimization technique (for example, multiple Vdd only [36] or gate sizing only [37–38]) without considering other techniques; 2) Heuristics such as those proposed in [11–14] could consider joint-optimization of different techniques, but these techniques are usually applied sequentially, and it is difficult to extend the heuristics if extra techniques (such as multiple gate oxide thickness) are to be integrated into the framework. Compared to those methodologies, a GA-based power optimization framework could handle multiple power optimization techniques (in our experiments, four techniques are included in the framework); and more importantly, the interaction among these techniques can be studied because all techniques are applied simultaneously, rather than sequentially. In addition, the framework can be easily extended to include other techniques such as multiple gate oxide assignment. Because of such unique advantages, we have seen recent work that adopted genetic algorithms in power optimization frameworks which used multiple power reduction techniques, for example, the power optimization for asynchronous circuits [34] and FinFET-based combinational logics [35].

3 Delay Model

Since our goal is to reduce the power consumption under the performance constraints in association with critical paths, one kind of delay calculations needs to be assumed. The *logical effort* based delay model [16] is adopted to perform the calculation of delay time. The derived technique, gain-based synthesis, has been realized its effectiveness in reducing synthesis-placement iterations and has already been incorporated in design automation flow in many industry companies, IBM and Magma, for instance [25, 26]. The advantage of this approach resides in the use of a delay-centric approach to logic optimization which naturally targets timing optimization and the use of gain rather than size, and thus obfuscates the need for a pre-placement wire load model. Since the critical paths in the optimized circuit remain critical in the placement phase, the resultant synthesized circuits are in good starting points for subsequent physical design.

The logic effort model is essentially a reformulation of the conventional RC model of CMOS gate delay and it can be simply stated as follows:

$$d = \tau * (p + gh) \tag{2}$$

In Eq. 2, g stands for logical effort which is the ratio of input capacitance of a gate to the input capacitance of an inverter. The logical effort represents the fact that, for a given load, complex gates have to work harder than an inverter to produce a similar response. In other words, the logical effort tells how much worse it is at producing output current than an inverter. The h is the electrical effort which is defined as the ratio between the external load and the input capacitance of the gate. The p represents the intrinsic delay and depends solely on diffusion capacitance of a gate. It can be seen from this equation that the delay of a gate is independent of p and g when h is fixed. Since the logical effort of an inverter is defined as 1 in the original definition [16], we assume that the smallest size inverter will be given logical effort of 1, and for the rest of other gates, the logical efforts are assigned to be the ratio of input capacitances of these gates to the input capacitance of the smallest inverter. An example is shown with two inverters in Fig. 4. Although both of the inverters have the logical effort of 1, the delay of the left inverter is larger than the right inverter with the reason of higher electrical effort assuming p is the same for both inverters. The τ in Eq. 2 is a scaling parameter that characterizes the semiconductor process being used in order to quantify the gate delay. That is, the term, $(p+gh)$ will then be multiplied by a process speed factor to obtain the actual delay time. For 0.25 μm technology, the process speed is 20 ps, while 15 ps for 0.18 μm. It approximates 8–11 ps for 65 nm technology which is our target technology. One thing to keep in mind is that we do not use the logic effort to calculate the delay of

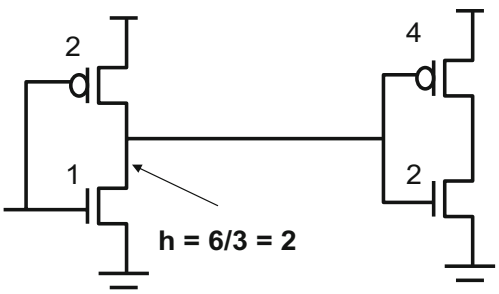

Figure 4 The example of two inverters.

the level converter. Because the spice simulation result for one specific implementation of the level converter is sufficient when we try to calculate the delay of a critical path. Furthermore, the level converter can be viewed as a separation point where the path delay calculation before and after this point are independent.

With the facility of the logical effort, the total delay of a critical path can be defined as the summation of the delays of all gates along the path. In addition to the gate delay, the delay of the level converter also needs to be considered. Thus, the total path delay D should also take the number of level converters on the critical path into consideration and is thus defined as follows:

$$D = \sum_{i=1}^{n} d_i + \#LC * delay\_LC \tag{3}$$

By using the logic effort based delay calculation and BFS STA-like approach, we can easily identify the criticalities of the paths. Note that, although more sophisticated timing analysis approaches can be used to accelerate the path identifying process, it is not the main concern and out of scope of this paper. The estimated delay time will then be compared with the performance requirement we give as the input to the algorithm. When there is a discrepancy as the algorithm proceeds, a certain amount of penalties will be added to the fitness score of a chromosome. Thus degrading the chance a chromosome can survive till the final round.

4 Genetic Algorithms

Genetic algorithms (GA) [15] are a class of search and optimization methods that mimic the evolutionary principles in natural selection. The evolution process generally eliminates the bad genes and maintains the good genes to generate better solutions. This concept has been recently applied to solve a range of complex VLSI combinatorial optimization problems [23, 24, 27, 28]. In this kind of approach, the feasible solution is usually encoded into a binary string called *chromosome*. Instead of working with a single solution, the search begins with a random set of chromosomes called *initial population*. Specifically, the more populations you have, the quicker you will converge or the more generations are needed with a small initial population. This characteristic can be used for run time controlling in some works. Each chromosome is assigned a *fitness* score that is directly related to the objective function of the optimization problem. In general, one most critical aspect of GA is represented by the computation of the fitness function which is the key factor to the successful evolution in GA. The population of chromosomes is modified to a new generation by applying three operators similar to natural selection operators—*reproduction*, *crossover*, and *mutation*. Reproduction selects good chromosomes based on the fitness score and duplicates them. Crossover picks two chromosomes randomly and some portions of the chromosomes are exchanged with a probability P_c. Finally, mutation operator changes a 1 to a 0 and vice versa with a small mutation probability P_m. A genetic algorithm successively applies these three operators in each generation until a termination criterion is met. It can very effectively search a large solution space while ignoring regions of the space that are not useful. That is, the solution space exploration is directed. This algorithmic methodology leads to very time-efficient searches. In general, a genetic algorithm has the following steps:

(1) Generation of initial population,
(2) Fitness function evaluation,
(3) Selection of chromosome,
(4) Reproduction, Crossover and Mutation operations.

The selection function tries to select the parents' chromosome with probabilities proportional to their fitness. In this way, the more highly fit chromosome will have higher number of children in the succeeding generation. The crossover and mutation operations will modify the patterns of chromosomes in order to generate better survivable solutions, so it will be passed on to future generations. Like most classical research and optimization methods, GA also faces difficulty in handling constraints. To handle this, the most commonly-used strategy is the penalty function method. In our problem, the constraint is the smallest delay time with the lowest power consumption. When the delay time is obtained in each generation, we compare it with the expected delay time. If this constraint is violated, an extra penalty is added to the fitness score of a chromosome. Through this way, the reproduction operator discourages the propagation of the inferior solutions to future generations.

5 Power Optimization Framework

Our power optimization flow is based on the genetic algorithm and is shown in Fig. 5. The circuit configuration information, such as supply voltage assignment and gate sizing, are encoded into a binary string called chromosome. The optimization flow begins with a random generated initial population, which consists of many randomly generated circuit configurations. The optimization flow is an iterative procedure. The chromosomes with better fitness will survive at each generation and are applied three different operations (reproduction, crossover, and mutation) to be a new set of chromosomes—or new circuit

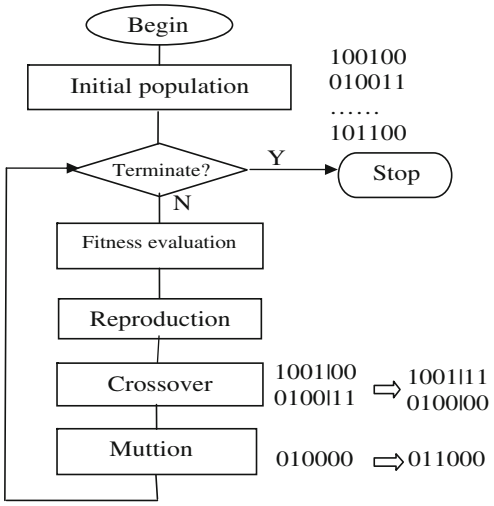

Figure 5 The typical optimization flow of a genetic algorithm.

configurations. The iteration continues until the termination criterion is met.

5.1 Chromosome Encoding

Given different power reduction techniques, we can encode all the tuning variables into a binary chromosome string. Figure 6 shows both the structure of a chromosome and an encoding example, representing N gates in a circuit. This encoding example is based on the assumption that we use a dual-V_{dd}, dual-V_{th} library with four discrete sizes for each type of gate, and the gate has one forced stacking version. For example, a 0 in voltage means using high V_{dd}, a 1 in threshold tells this gate to use low V_{th}, and a 0 in stacking effect stands not to choose the stacking version of a gate. For transistor sizing, we have used little endian notation; that is, 00 is the minimum size and 11 is the maximum size. While the V_{th} allocation and force stacking can be done in transistor level, for simplicity we assume the granularity is at the gate level. In Fig. 6, the chromosome shows that Gate 1 is assigned to use lower V_{dd}, higher V_{th}, size 1 of the gate and with no force stacking. Note that only the tuning variables are encoded into the chromosome. The type of each logic gate and the circuit topology information are known *apriori* to calculate the power and delay based on the chromosome configuration.

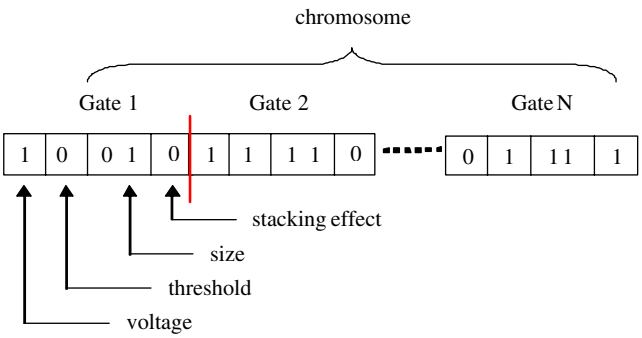

Figure 6 The structure of a chromosome.

This encoding scheme is easy to extend for more complicated standard cell library. For example, if a standard cell library has more than two supply voltage choices or more than two threshold voltage choices, we just need more bits in the chromosome. Increasing the flexibility of the library simply increases the bits required for each gate. Since the chromosomes are randomly generated and we use the binary bit string representation for chromosomes, in some cases, the chromosome configurations may not be valid. For example, for a library with three threshold voltage choices, we have to use 2 bits for the V_{th} configuration and there will be one invalid configuration. We resolve this problem by adding penalties to fitness score and that is described in the next section.

5.2 Fitness Function

The fitness function, which decides the surviving chance for a specific chromosome, is related to the power consumption and the delay of the circuit, as well as the validity of the chromosome. In the following subsections, we will give detail explanation for each factor in the objective fitness function.

5.2.1 Power

In our power optimization framework, the goal is to find a configuration such that the power consumption for the circuit is as low as possible and the timing requirement is met at the same time. Therefore the fitness of a chromosome should be related to the power consumption of that particular configuration, which can be calculated using Eq. 1 discussed in Section 1. For the dynamic power of the gate, the switching activity is obtained by exhaustive simulation with the assumption that the input probabilities of being high or low are equal and independent. The gate leakage and sub-threshold leakage were also characterized by similar simulation. Our method is general enough and more accurate power estimates can be used, if more information on probabilities is available.

When using multiple supply voltages in a circuit, level converters are required whenever a logic gate at the lower supply has to drive a gate at the higher voltage [4]. Note that, the overall power consumption of the circuit also includes the power consumptions from the level converters.

5.2.2 Delay

It should be noted that the power optimization is under a specified timing constraint. If the critical path delay in a circuit is longer than the timing requirement, the configuration is not desirable and the corresponding chromosome should have

little chance to survive. The delay calculation of the circuit is based on the logical effort [16] mentioned in Section 3. The delay from level converters is also taken into account.

5.2.3 Penalty Function

As we have discussed in Section 5.1, since we use the binary string to represent a chromosome, when the number of choices for a tuning variable is not 2^N (for example, a gate has six discrete sizes, or three threshold voltage choices), we may end up with an invalid configuration during the population initialization or chromosome operations. For those chromosomes representing invalid configurations, the chance of surviving should be set smaller than those of valid configurations. Based on the above argument, the fitness function can be defined as

$$Fitness = \frac{1}{Total\_power} - Penalty \quad (4)$$

where the penalty is a big number if timing violates or the chromosome is invalid, such that those valid chromosomes with lower power consumption while meeting timing requirement have better chances to remain alive.

6 Experimental Results

Previous work [13, 22] have shown that using two supply and threshold voltages provides substantial improvement in power dissipation and the use of additional voltages results in small power improvements, which is hard to justify the additional costs associated with multiple supply and threshold voltages. Thus, to validate our proposed framework, we have constructed a dual-V_{dd} and dual-V_th standard cell library in 65 nm process technology. The logic gates in the library are inverter, NAND2, NOR2, XOR2, and a level converter. The framework was implemented in C. Although V_{th} allocation and force stacking can be done either at transistor level or at gate level, and each type of the gates may have several discrete sizes. For simplicity, we assume the granularity of both V_{th} assignment and forced stacking are at the gate level and the gates have only two discrete sizes, such that we need only 4 bits to encode the tuning variables. Note that, the flexibility of our approach is that only the length of bits will be affected to reflect the encoding change of gate's configuration.

Figure 7 shows the adopted level converter implementation [4]. As many researchers predict that it is not practical to provide multiple level converters within a combinational block, integrating level converters into flip-flops and providing only supply voltage transition is more feasible. In our framework, the use of the level converters can be constrained and enforced; moreover, the use of the supply voltage transition can also be confined. However, the purpose of our approach is to achieve maximum power savings while to satisfy performance demands. Hence, we assume the possibility of arbitrary placement of level converters in our approach. Note that, the delay and the power costs incurred from the level converter are cumulated and considered in calculating the path delays and the total power consumptions of circuits.

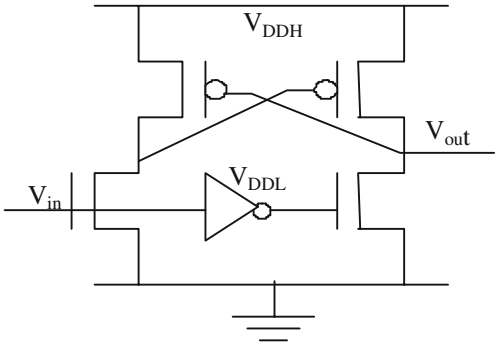

Figure 7 The implementation of the level converter.

The benchmark circuits we choose to map to our library span typical circuit topologies, including circuits from ISCAS 85 benchmark, an inverter chain, a 32-bit carry-ripple adder, an 8×8 carry-save multiplier, an 8-to-256 SRAM decoder, and three manually generated circuits. The circuit sizes range from 8 gates to 1,050 gates. The circuits were first optimized for maximum speed (i.e. using all high V_{dd} and low V_{th} on the critical path with the larger transistor size and no stacking enforced) and then were optimized for lowest power consumption. Note that, if the demanded performance cannot be met with the above circuit configuration, our framework stops without evaluation. Since such rigid performance requirement simply cannot be satisfied with the fastest circuit setting obtainable from our circuit technique combinations. We then perform power optimization with the timing constraint relaxation. The experiment was carried out on an Intel Pentium 4 processor (2.8 GHz, 512 MB RAM).

6.1 Control Parameters

Based on the conclusion in [13], the optimal second V_{dd} in a dual-V_{dd} system should be around 50% of the higher supply voltage. The supply voltages for our library are defined as 1 V and 0.5 V. For NMOS (PMOS) transistors, the high threshold voltage and the low threshold voltage are 0.22 V (−0.22 V) and 0.12 V (−0.12 V) respectively. The library was characterized by using Berkeley 65 nm BSIM predictive model [17]. The gate leakage and sub-threshold leakage were pre-characterized.

While generating the initial population, we have to set an appropriate population size, and the crossover probability P_c,

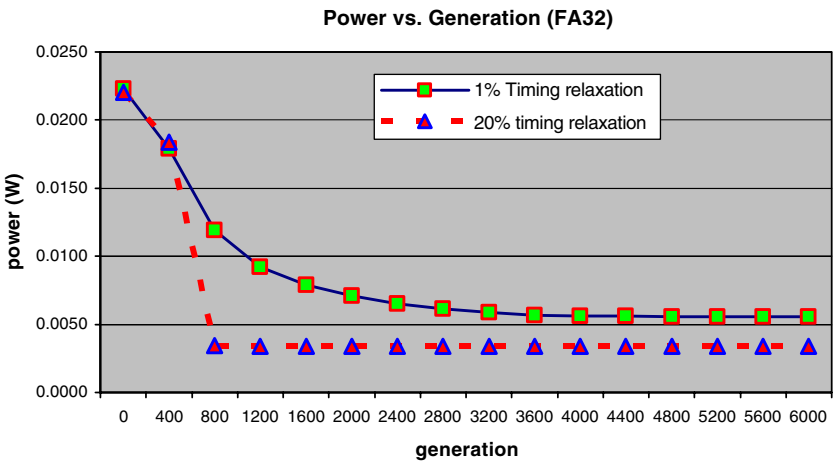

Figure 8 Solution convergences under different timing relaxations.

as well as the mutation probability P_m. If the population size is too small, the genetic diversity within the population may not increase for many generations. On the other hand, a large population size increases the computation time for each generation but it may take fewer generations to find the best solution. Schaffer et al. [18] have conducted extensive simulation on a wide range of functions and concluded that a small population of size 20 to 30, a crossover probability in the range of 0.75 to 0.95, and a mutation probability in the range of 0.005 to 0.01 perform very well. In our implementation, we set the population size to be 100, crossover probability P_c to be 0.9 and the mutation probability P_m to be 0.01. The termination criterion of the iterative evolution can be user defined. We specify that if the power reduction is less than 0.001% during the last 100 generations, the evolution stops without going through all remaining generations.

6.2 Generation and Performance

In this subsection, we conducted experiments to explore the goodness of generated results from different aspects. As mentioned in the previous section, the number of generations used in the genetic algorithm has strong impacts on the solution. Figure 8 displays this phenomenon. The power consumption of the 32-bit carry-ripple adder (FA32) circuit decreases while the generation increases along 1% timing relaxation curve. We can also see that there is little improvement around 4,000 generation and more generations will not help reduce power consumption.

Figure 8 also shows the power reduction process for the FA32 circuit under different timing relaxations. The timing relaxation is done through mitigating the performance requirement. That is, the delay of the critical path is enlarged to lessen the performance demand. As shown in the figure, we can see that for a very tight timing constraint (1% timing relaxation), the algorithm achieves a maximum power saving within 3,600–4,000 generation region under this rigid timing constraint. Running more generations will not help much. If we relax the timing constraint to 20% timing relaxation, the convergence can be achieved much faster (within 1,200 generation). With this performance degradation, the power saving is larger than that of 1%

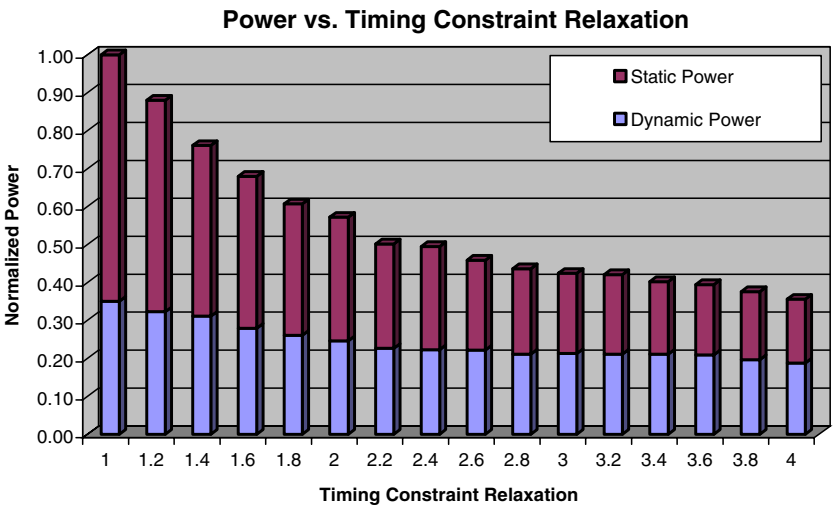

Figure 9 Power reductions as timing constraint relaxed.

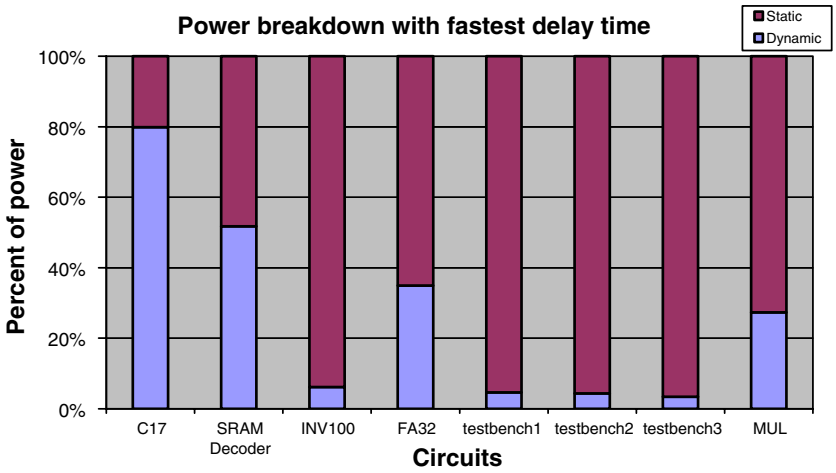

Figure 10 Power breakdown for speed-optimized circuits.

timing relaxation. Thus, we know that performance and power consumption of a design need to be tradeoff cautiously in order to reach the point of design equilibrium. We also performed sensitivity analysis on parameters that are used to drive the GA-algorithm: 1) the population size changes from 100 to 1,000, with an incremental step of 100; 2) the crossover probability changes from 0.75 to 0.95, with an interval of 0.05; 3) and the mutation probability changes from 0.005 to 0.01, with an interval of 0.001. Our experiment results show that when the population size increases, the solution convergence could be slightly improved, achieving the same maximum power savings within the region of 3,300–3,600 generation region. However, the improvement is small, but the run-time is increased due to a larger size of the population. In addition, the sensitivity analysis on crossover probability and mutation probability also demonstrated that Schaffer's conclusion [18] on setting P_c=0.9 and P_m=0.01 could indeed achieve the maximum power saving.

Figure 9 shows the changes in power composition with the application of our algorithm on FA32 benchmark. The timing constraint (cycle time) and power consumption are both normalized to the fastest speed setting. We can see that when the timing constraint is relaxed to be twice larger than the fastest speed, the power reduction we can achieve is about 45%. Further relaxation contributes to 65% power saving when the timing constraint is four times larger than that of the fastest performance speed. We can also see that most of the power reduction comes from the static power reduction. The implication behind this result is that with the less aggressive performance requirement, the algorithm is more effective in delivering low power budget solution by applying four low power techniques. For the targeted 65 nm process technology, the static power is thus reduced more obvious than the dynamic power as shown in the figure. One more observation from this result is that the dynamic power does not decrease as linearly as Eq. 1 indicates when the timing constraint keeps relaxing. That is, the dynamic power should decrease at a per 1/T timing relaxation. The reason for this phenomenon is that the goal of our fitness function tries to minimize the total power consumption instead of individual static power or dynamic power. As a result, the reductions of the total power are more obvious from the figure.

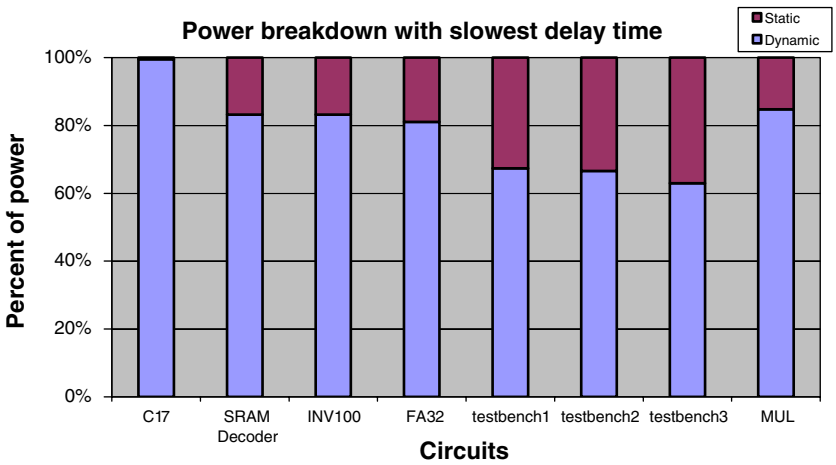

Figure 11 Power breakdown for power-optimized circuits.

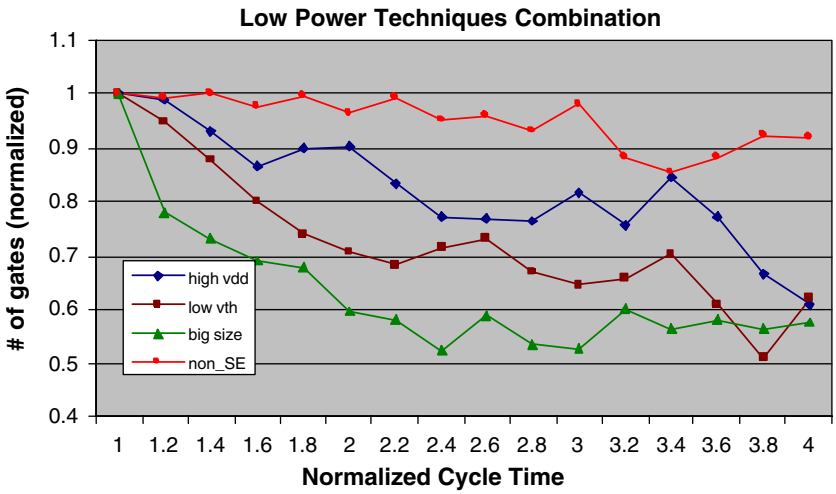

Figure 12 The profile of used low power techniques.

6.3 Applying Four Low Power Techniques

Figures 10 and 11 present the static power and dynamic power breakdown for maximal speed optimized circuit and minimal power optimized circuit, respectively. With our 65 nm library, the static power accounts for average 74% of the total power while the dynamic power accounts for 26% of the overall power. After applying all four power reduction techniques without any timing constraints, the static power accounts for only about 20% of the overall power while dynamic power accounts for 80%. A significant part of overall power reduction is from the static power reduction by using these four techniques altogether. Overall 84% of the power reduction is from the static power reduction.

Figure 12 shows the profile of power reduction techniques that are used in the FA32 circuit when the timing constraint is relaxed iteratively. The number of gates used for a specific configuration is normalized to the fastest-speed optimized case. It is obvious that as the timing constraint relaxed from 0% to 400%, the gates tend to use the less power consuming configurations, i.e., low V_{dd}, high V_{th}, small gate size, and stacking force. The interaction of these four tuning variables (V_{dd}, V_{th}, sizing, and stacking force) can also be discovered from our experiment, which is represented by the fluctuations of four curves.

Figure 13 shows the power consumptions between configurations with stacking force and without stacking force. The comparison is conducted with all four low power techniques available and all techniques except stacking force. The power saving is 20% under the fastest-speed performance demand between stacking and non-stacking approaches. As the timing constraint keeps relaxing, the stacking approach still maintains a reasonable level of power savings, for example, 17.8% in 2X and 17.4% in 3X timing relaxations. The overall saving is 18.6% for all timing relaxations. From this result, we observe that stacking force is an effective mechanism to reduce power consumption.

Based on the previous result, we know that stacking force mechanism can be employed to lower the power

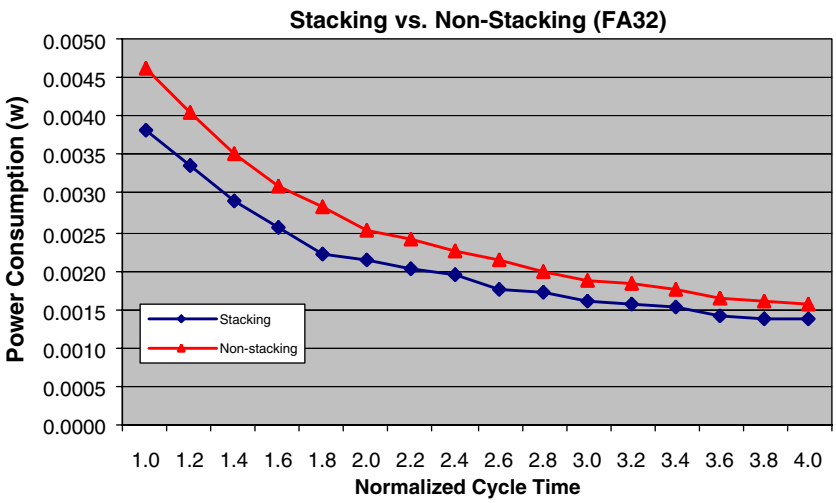

Figure 13 Comparison of power consumption with/without stacking force.

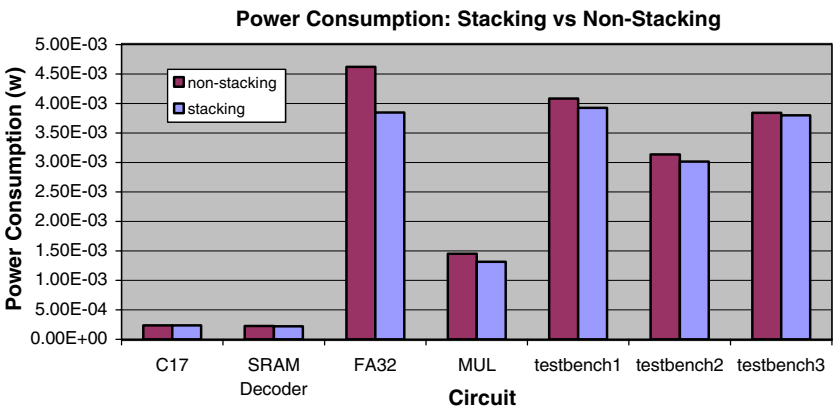

Figure 14 Comparison of circuits' power consumptions with/without stacking force.

consumption further; thus, we adopt this technique on all of the benchmarks. Figure 14 shows the comparison between power consumptions with and without stacking effect for all circuits. As can be seen from the figure, stacking effect is effective in lowering the power consumptions under the same rigid performance demands. The power reduction can be as large as 20% for circuit FA32 and is 11% for circuit MUL, while the overall average power reduction is 6% for all seven circuits. This reduction is not as surprising as we anticipate. We have thought of two reasons for this if we exclude two smallest benchmarks. First, benchmarks are an important factor in deciding the length of the critical path. Unlike FA32 and MUL circuits, testbenchs 1 to 3 have fairly large proportion of gates lying on the critical paths. Since the gates on the critical paths are normally enforced not to use stacking effect so as to maintain their driving strength and the rest of the gates are free to use this facility. However, as a large number of gates constitute the critical paths, only a small number of gates can fully take advantage of using stacking force to save power. Second, although stacking force can lower the static power, it requires other compensating techniques (high V_{dd} or bigger transistor size) to recover the loss of the driving strength, which brings the dynamic power consumption higher. As the total power consumption is set for fitness function, the power of stacking effect is thus canceled out with the power consumption.

The reason why circuit INV100 is not included is that with the same rigorous performance requisite, this circuit simply cannot use stacking force along the chain; otherwise, the performance demands cannot be met. Thus, under the same performance requirement, two configurations will result in the same power consumption. From this result, we know that the force stacking effect can lower power consumption even further when applied to the circuits with more than one path.

Figure 15 shows the comparison of power consumption with only one low power technique available at a time to one equipped with four low power techniques. The reason for some curves not showing the point from the beginning is that they simply cannot meet timing considered at that point. We can easily observe from the figure that the dual-V_{dd} is the most effective way to reduce power while the dual-V_{th} is the less effective one. But dual-V_{th} can still maintain good timing constraint achievements compared to the other three techniques. The curve of all available techniques can easily

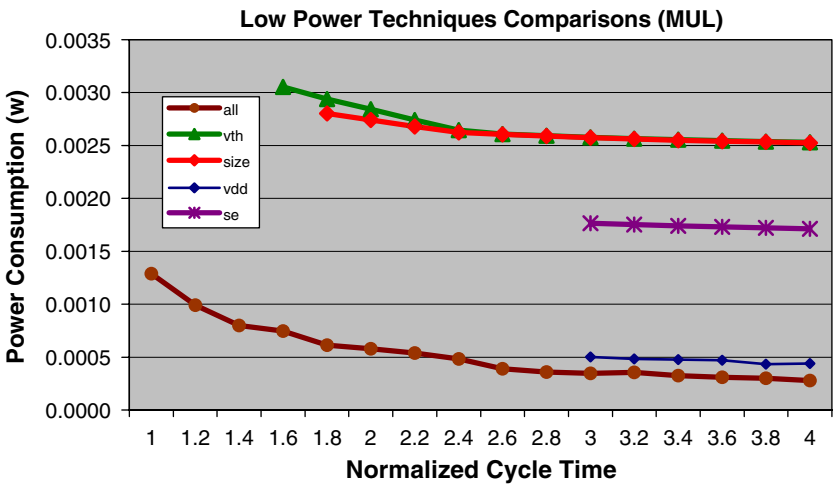

Figure 15 Comparison of different low power techniques.

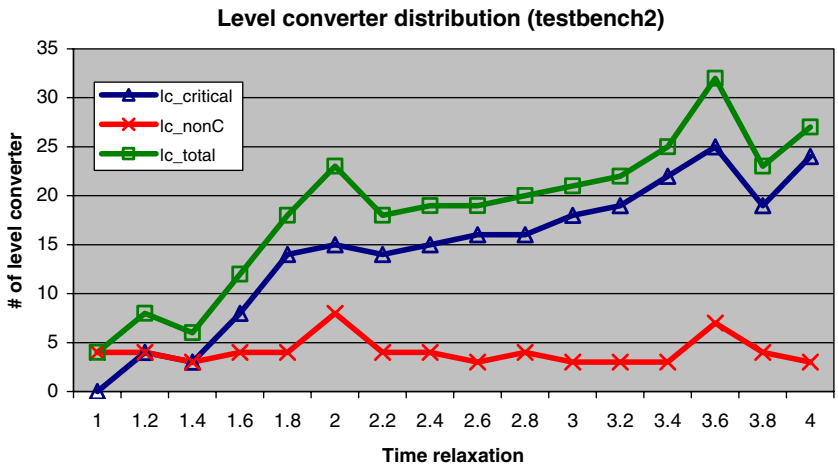

Figure 16 The distribution of level converters of testbench2 circuit.

maintain solutions with low power consumption and without incurring performance degradation.

The distribution of the level converter of testbench2 circuit is shown in Fig. 16. With the fastest-speed performance requirement, the gates in the critical path are all assigned with high supply voltage; therefore, there are no needs for the level converter under this circumstance. In contrast, as the timing constraint keeps being relaxed, the use of the level converter increases. The reason behind this is that only some gates in the critical path need to be assigned with high supply voltage while the rest of them are driven with low supply voltage with the purpose of saving power. As can be observed in the figure, there are two sharp increases of the level converter, which are at 2X and 3.6X timing relaxation points. The reasoning for the first steep increase is that the increasing use of level converter on the non-critical paths may contribute to this phenomenon. For the second sharp increase of level converter, the increasing use of level converter on the non-critical paths is the first reason which is the same as the previous explanation, while the second reason is the relaxing timing. With less aggressive performance requirements, some gates of a circuit may be assigned to be driven by low-supply voltage, and then drive the following high-supply voltage gates, which require the level converter between them.

6.4 Comparison with Other Exiting Combined Low Power Techniques

In this subsection, we conducted experiments to compare the existing proposed approaches with our proposed techniques in terms of exploring power saving issue. The best known method to save power is the use of multiple supply and threshold voltages for a design. Thus, experiments are performed and both total power and leakage power saving are recorded to demonstrate the effectiveness of our approach.

Total power is an important concern for high performance systems and portable devices. Based on this concern, the comparison of power consumptions of different combined low-power techniques is performed. Figure 17 shows the results of total power consumptions from different combined low power techniques, which are dual-V_{dd}+V_{th}, dual-V_{dd}+size, dual-V_{dd}+V_{th}+size, and all four techniques, respectively. From the figure, the combined all four low power techniques outperforms the other

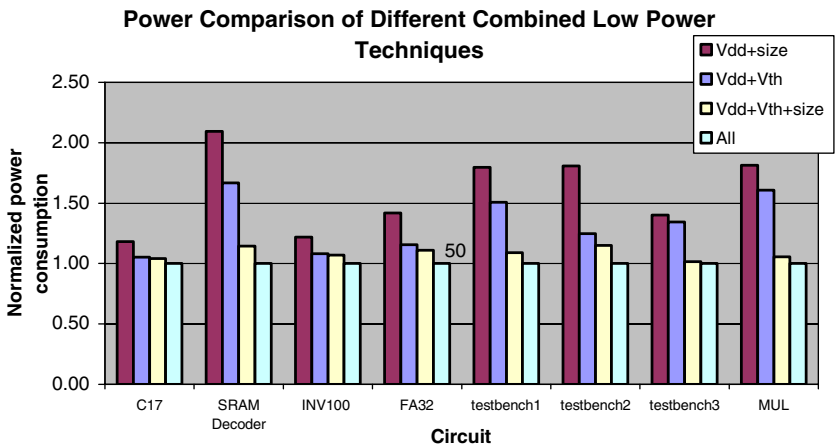

Figure 17 Power comparison of V_{dd}+V_{th}, V_{dd}+size, V_{dd}+V_{th}+size, and all four low power techniques.

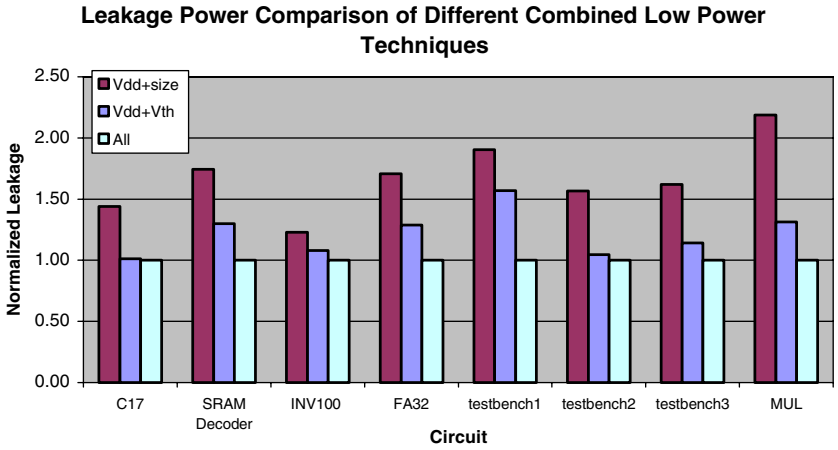

Figure 18 Comparison of leakage power from different combined low power techniques.

three combined approaches in terms of total power savings. The dual-V_{dd}+V_{th} also functions better than dual-V_{dd}+size one. The rationale behind this is that the leakage power is sensitive to the threshold voltage and the leakage power is the preeminent power in 65 nm process technology. With the help of dual threshold voltages, the dual-V_{dd}+V_{th} approach can save more power from the leakage power aspect.

As we mentioned before, the leakage is a dominant portion of the total power consumption; thus the comparison of leakage power on different combined techniques is evaluated. The experimental result is presented in Fig. 18. As shown in the figure, the dual-V_{dd}+V_{th} has the lower leakage power than that of dual-V_{dd}+size approach, but applying all four low power techniques can result in the lowest leakage power consumption when compared with its counterparts. The implication from this result states that although the leakage power can be effectively reduced by applying dual supply and dual threshold voltages with the strong relationship between them, applying stacking force, in fact, can further improve the leakage power by 16% of all benchmarks, in average.

6.5 Exploiting Parallelism in Genetic Algorithms

Compared to the ILP approach that were used by [13, 14], one advantage of our GA approach is that the parallel nature of genetic algorithms suggests parallel processing as the natural route to explore. We implemented a parallel version of our algorithm on a dual Intel Xeon processors (3.2 GHz, 2 GB RAM) machine running Linux, by dividing the population processing between dual processors. In Fig. 19, we can see that using dual processors for parallel version of GA can effectively reduce the amount of running time needed in single processor by almost 50%. We noticed that an average of 1.97X run-time speed-up on a 2-processor workstation against the single-processor version of our algorithm (The reason that we cannot achieve a 2X speedup is the interaction overhead between parallel processes). Note that, the small circuits (C17, SRAM Decoder, INV100) cannot make use of the parallel advantage because the overhead incurred from the dual processors dissipates the speedup of parallelism.

Although it is beneficial to adopt the parallelized version of GA's to improve run-time efficiency, other techniques can also

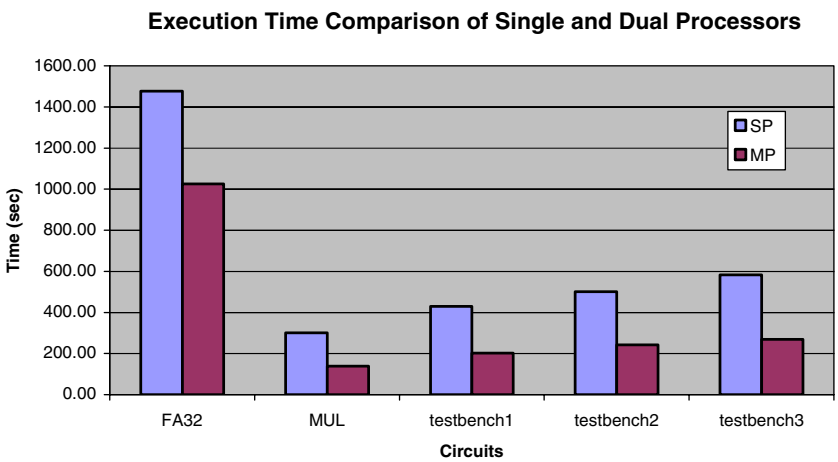

Figure 19 Run time comparison of single processor and dual processors.

be incorporated. For instance, intelligence can be employed to identify critical paths of the circuit at the phase of generating the initial population. This way, the feasibility of convergence to the global optimal solution is guaranteed and the efficiency of the algorithm is also effectively improved.

Another thing also needs to keep in mind is that the computation complexity seems to be a little bit high from the above figure. As we mentioned in Section 6.1, the best population to be used should be in the range of 20 to 30 from the previous study [18]; however, the population of 100 was used in conducting the experiments. Moreover, we have used the number of 0.001% as the improvement indicator, which instructs the algorithm for early termination if the improvement does not exceed this number over a pre-defined generation. This improvement indicator is a relatively small number and thus has little possibility of early algorithm termination to save the computational time. Thus, we believe that the run-time of our framework can be effectively reduced by carefully choosing these two numbers.

The last thing we want to emphasize is the scaling ability of the genetic algorithm. As far as scaling is concerned, it is usually of the top priority to find out how the GA behaves on problems where the problem size is exponentially scaled. Lobo et al. [29] conducted a theoretical and empirical analysis of the time complexity of genetic algorithms. As indicated in that work, genetic algorithms with perfect mixing have time complexity of $O(m^2)$ for cases of exponential scaling. To that end, although we did not perform a thorough theoretical and empirical analysis on our framework, we believe that the formation of power optimization problem in GA is still suitably efficient as the problem size grows.

7 Concluding Remarks

Power is the current challenge for high performance computer system designs and the widespread use of portable and wireless electronic systems. With the technology continues scaling down, the power issue is even more prominent than before with the highly integration capability on a single chip. Power dissipation affects battery life and performance, and greatly affects reliability and heat removing costs, and thus reduction in power has become more and more important in the coming nanometer era.

In this paper, we present a power optimization framework based on the genetic algorithm. The optimization strategy can simultaneously perform multiple-V_{dd} assignment, multiple-V_{th} assignment, gate sizing in conjunction with stacking force technique to minimize total power consumption, while maintaining performance requirements. We demonstrate the effectiveness of our total power optimization framework by conducting various experiments. The comparisons to the different combined low power techniques have also been conducted and the results confirm that our approach is valid and is comparable to other approaches.

References

1. Usami, K., & Horowitz, M. (1995). Clustered voltage scaling techniques for low-power design. In *Proceedings of the International Symposium on Low power design* (pp. 3–8).
2. Wang, Q., & Vrudhula, S. (2002). Algorithms for minimizing standby power in deep submicron, dual-Vt CMOS circuits. *IEEE Transactions on Computer-Aided Design*, *21*, 306–318.
3. Puri, R., Stok, L., Cohn, J., Kung, D., Pan, D., Sylvester, D., et al. (2003). Pushing ASIC performance in a power envelope. In *Proceedings of Design Automation Conference* (pp. 788–793).
4. Rabaey, J., Chandrakasan, A., & Nikolic, B. (2003). *Digital integrated circuits: A design perspective*. Second edition. NJ: Prentice Hall.
5. Hamada, M., & Ootaguro, Y. (2001). Utilizing surplus timing for power reduction. In *Proceedings of the IEEE Custom Integrated Circuits Conference* (pp. 89–92).
6. Brodersen, R., Horowitz, M., Markovic, D., Nikolic, B., & Stojanovic, V. (2002). Methods for true power minimization. In *Proceedings of Computer-Aided Design* (pp. 35–42).
7. Narendra, S., Borkar, S., De, V., Antoniadis, D., & Chandrakasan, A. (2001). Scaling of stack effect and its application for leakage reduction. In *Proceedings of the International Symposium on Low Power Electronics and Design* (pp. 195–200).
8. Stojanovic, V., Markovic, D., Nikolic, B., Horowitz, M., & Brodersen, R. (2002). Energy-delay tradeoffs in combinational logic using gate sizing and supply voltage optimization. In *Proceedings of European Solid-State Circuits Conference* (pp. 211–214).
9. Roy, K., Wei, L., & Chen, Z. (1999). Multiple Vdd Multiple Vth (MVCMOS) for lower power applications. *International Symposium on Circuits and Systems* (pp. 366–370).
10. Augsburger, S., & Nikolic, B. (2002). Reducing power with dual supply, dual threshold and transistor sizing. *International Conference on Computer Design* (pp. 316–321).
11. Sirichotiyakul, S., Edwards, T., Oh, C., Zuo, J., Dharchoudhury, A., Panda, R., et al. (1999). Stand-by power minimization through simultaneous threshold voltage selection and circuit sizing. In *Proceedings of Design Automation Conference* (pp. 436–441).
12. Karnik, T., Ye, Y., Tschanz, J., Wei, L., Burns, S., Govindarajulu, V., et al. (2002). Total power optimization by simultaneous Dual-Vt allocation and device sizing in high performance microprocessors. In *Proceedings of Design Automation Conference* (pp. 486–491).
13. Srivastava, A., & Sylvester, D. (2003). Minimizing total power by simultaneous Vdd/Vth assignment. In *Proceedings of Asia-South Pacific Design Automation Conference* (pp. 400–403).
14. Nguyen, D., Davare, A., Orshansky, M., Chinnery, D., Thompson, B., & Keutzer, K. (2003). Minimization of dynamic and static power through joint assignment of threshold voltages and sizing optimization. In *Proceedings of the International Symposium on Low Power Electronics and Design* (pp. 158–163).
15. Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning*. New York: Addison-Wesley.
16. Sutherland, I. V., Sproull, R. F. & Harris, D. F. (1999). *Logical effort*, M. Kaufmann.
17. http://www.device.eecs.berkeley.edu, Berkeley predictive model.
18. Schaffer, J., Caruana, J., Eshelman, L., & Das, R. (1989). A study of control parameters affecting online performance of genetic

algorithms for function optimization. In *Proceedings of International Conference on Genetic Algorithms* (pp. 51–60).
19. Lee, D., Deogun, H., Blaauw, D., & Sylvester, D. (2004). Simultaneous state, Vt and Tox assignment for total standby power minimization. In *Proceedings of Design, Automation and Test in Europe* (pp. 494–499).
20. Srivastava, A., Sylvester, D., & Blaauw, D. (2004). Concurrent sizing, Vdd and Vth assignment for low-power design. In *Proceedings of Design, Automation and Test in Europe* (pp. 718–719).
21. Hung, W., Xie, Y., Vijaykrishnan, N., Kandemire, M., Irwin, M. J., & Tsai, Y. (2004). Total power optimization through simultaneously Multiple-VDD, Multiple-VTH assignment and device sizing with stack forcing. In *Proceedings of the International Symposium on Low Power Electronics and Design* (pp. 144–149).
22. Usami, K., Igarashi, M., Minami, F., Ishikawa, T., Kanzawa, M., Ichida, M., et al. (1997). Automated low-power techniques exploiting multiple supply voltage applied to a media processor. In *Proceedings of Custom Integrated Circuits Conference* (pp. 131–134).
23. Saab, Y. G., & Rao, V. B. (1989). An evolution-based approach to partitioning ASIC systems. In *Proceedings of Design Automation Conference* (pp. 767–770).
24. Cohoon, J. P., & Paris, W. (1996). Genetic placement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, *6*(6), 956–964.
25. http://www.research.ibm.com/da/logic.html.
26. http://www.magma-da.com/articles/Magma_GBS_White_Paper.pdf.
27. Chan, H., Mazumder, P., & Shahookar, K. (1991). Macro-cell and module placement by genetic adaptive search with bitmap-represented chromosome. *Integration: The VLSI Journal*, *12*(1), 49–77.
28. Bright, M. S., & Arslan, T. (1996). A genetic framework for the high-level optimization of low power VLSI DSP systems. *IEEE Electron Letters*, *32*(13), 1150–1151.
29. Lobo, F. G., Goldberg, D. E., & Pelikan, M. (2000). Time complexity of genetic algorithms on exponentially scaled problems. In *Proceedings of the Genetic and Evolutionary Computation Conference*.
30. Chinnery, D. G., & Keutzer, K. (2005). Linear programming for sizing, Vth and Vdd assignment. In *Proceedings of the International Symposium on Low Power Electronics and Design* (pp. 149–154).
31. Ekpanyapong, M., & Lim, S. K. (2006). Integrated retiming and simultaneous Vdd/Vth scaling for total power minimization. In *ACM International Symposium on Physical Design* (pp. 142–148).
32. Diril, A., Dhillon, Y., Chatterjee, A., & Singh, A. (2005). Level-shifter free design of low power dual supply voltage CMOS circuits using dual threshold voltages. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 13(9), 1103–1107.
33. Srivastava, A., & Sylvester, D. (2004). Minimizing total power by simultaneous Vdd/Vth assignments. In *Proceedings of the Asia and South Pacific Design Automation Conference* (pp. 400–403).
34. Ghavami, B., Khosraviani, M., & Pedram, H. (2008). Power optimization of asynchronous circuits through simultaneous Vdd and Vth assignment and template sizing. In *Proceedings of the 11th EUROMICRO conference on Digital System Design Architectures, Methods and Tools* (pp. 274–281).
35. Ouyang, J., & Xie, Y. (2008). Power optimization for FinFET-based circuits using genetic algorithms. In *Proceedings of IEEE International SOC Conference* (pp. 211–214).
36. Chi, J., Lee, H., Tsai, S., & Chi, M. (2007). Gate level multiple supply voltage assignment algorithm for power optimization under timing constraint. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, *15*(6), 637–648.
37. Morifuji, E., Patil, D., Horowitz, M., & Nishi, Y. (2007). Power Q2 optimization for SRAM and its scaling. *IEEE Transactions on Electron Devices*, 54(4), 715–722.
38. Dabiri, F., Nahapetian, A., Massey, T., Potkonjak, M., & Sarrafzadeh, M. (2008). General methodology for soft-error-aware power optimization using gate sizing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, *27* (10), 1788–1797.
39. Dutta, S., Nag, S., & Roy, K. (1994). ASAP: a transistor sizing tool for speed, area, and power optimization of static CMOS circuits. In *Proceedings of IEEE International Symposium on Circuits and Systems* (pp. 61–64).

Wei-lun Hung received the B.S. degree in Information Management from National Taiwan University of Science and Technology, Taiwan, in 2000, and the M.S. degree in Computer Science from National Tsinghua University, Taiwan in 2002, and the Ph.D. degree in Computer Science and Engineering from Pennsylvania State University in 2006. He was with Lattice Semiconductor as a FPGA CAD engineer. He is currently with Sun Microsystems. His research interests include VLSI Design, computer arithmetic, and electronics design automation. He is a member of IEEE.

Yuan Xie is Associate Professor in Computer Science and Engineering department at the Pennsylvania State University. He received the B.S. degree in electronic engineering from Tsinghua University in Beijing, the M.S. and Ph.D. degrees in electrical engineering from Princeton University. He was a recipient of the SRC Inventor Recognition Award in 2002, NSF CAREER award in 2006, and IBM Faculty Award in 2008. He also received Best Paper Award in ASP-DAC 2008 and ASICON 2001. He is currently Associate Editor for IEEE Transaction on VLSI and IET Computers and Digital Techniques.

Narayanan Vijaykrishnan is a Professor of Computer Science and Engineering at The Pennsylvania State University. His research interests are in Computer Architecture, Embedded Systems and Nanoarchitectures.

Mahmut Kandemir is an associate professor in the Computer Science and Engineering Department at the Pennsylvania State University. He is a member of the Microsystems Design Lab. His research interests are in optimizing compilers, runtime systems, embedded systems, I/O and high performance storage, and power-aware computing. He is the author of more than 300 papers in these areas. His research is funded by NSF, DARPA, and SRC. He is a recipient of NSF Career Award and the Penn State Engineering Society Outstanding Research Award. He is a member of ACM and IEEE.

Mary Jane Irwin received her Ph.D. degree in computer science from the University of Illinois in 1977. Dr. Irwin has been on the faculty at Penn State since 1977, and was named the A. Robert Noll chair of Engineering in 2003 and an Evan Pugh Professor in 2006 in the Department of Computer Science and Engineering. Her research and teaching interests include computer architecture, embedded and mobile computing systems design, power and reliability aware design, and emerging technologies in computing systems. She has more than 300 refereed conference and journal publications and has supervised more than twenty PhD's. Dr. Irwin co-leads (with Drs. Kandemir, Narayanan, and Xie) the Microsystems Design Lab with a focus on power and reliability aware design, embedded and mobile computing systems design, and emerging technologies in computing. Their research is supported by the National Science Foundation, the FCRP Gigascale Systems Research Center, the Semiconductor Research Corporation, Intel Corporation, and Microsoft. Dr. Irwin is also collaborating with Dr. Raghavan in the development of adaptive software tools to co-manage quality-performance-power tradeoffs in large-scale scientific simulations.

Dr. Irwin received an Honorary Doctorate from Chalmers University, Sweden in 1997 and the Penn State Engineering Society's Premier Research Award in 2001. She was awarded the ACM Distinguished Service Award and the CRA Distinguished Service Award, both in 2006. She was named a Fellow of IEEE in 1995, a Fellow of ACM in 1996, was elected to the National Academy of Engineering in 2003. Dr. Irwin is currently serving as a member of the US Board on Army Science and Technology, as a member of Microsoft Research's ER&P Advisory Board, and as a member of the Computing Research Association's Committee on Women Steering Committee. In the past she has served as an elected member of the IEEE Computer Society's Board of Governors, of ACM's Council, of the Computing Research Association's Board of Directors, as Vice President of ACM, as the Editor-in-Chief of ACN's Transactions on Design of Electronic Systems, and as chair of the NSF's Computer Information Sciences and Engineering Directorate's Advisory Committee.