# A Low Power Scheduling Method using Dual $V_{dd}$ and Dual $V_{th}$

Kun-Lin Tsai<sup>1</sup> Department of EE<sup>1</sup> National Taiwan University Taipei 106, Taiwan. Email: kunlin@orchid.ee.ntu.edu.tw Szu-Wei Chang<sup>2</sup> and Feipei Lai<sup>1,2</sup> Department of CSIE<sup>2</sup> National Taiwan University Taipei 106, Taiwan. Email: flai@ntu.edu.tw Shanq-Jang Ruan<sup>3</sup> Department of ET<sup>3</sup> National Taiwan University of Science and Technology Taipei 106, Taiwan. Email: sjruan@et.ntust.edu.tw

Abstract—As the technology scales down to the nanometer dimensions, the static power consumption has become more and more important. To manage the power consumption, in this paper, we propose a low power method, which considers the dual supply voltage  $(V_{dd})$  and the dual threshold voltage  $(V_{th})$  at the same time, to deal with the scheduling problem in the behavioral synthesis stage. A flexible design space of power, and a better performance can be achieved when we use the proposed method. A combined algorithm of GA (Genetic Algorithm) and SA (Simulated Annealing) is used to solve the scheduling problem. Experimental results illustrate 41.6% power reduction on average.

### I. INTRODUCTION

In recent years, the power consumption of a chip has become a very important issue, especially for SoC design. It is obvious that in the next decade low power design would be a big challenge for the IC design companies [1]. Among lots of design methods, the most effective way to reduce power consumption is to lower the supply voltage  $(V_{dd})$  of a circuit. Reducing the supply voltage, however, increases the circuit delay. A solution method is using the dual or the multiple supply voltages.

The dual  $V_{dd}$  method was used on every level of low power circuit design, such as behavioral level [2], [3] and gate level [4]. However, taking only the supply voltage into account is not enough. In deep sub-micron design, the leakage power consumption is also a very important issue. In [5], [6], they proposed the dual threshold voltage ( $V_{th}$ ) to tackle with the leakage power optimization problem. Although the leakage power is greatly reduced, the total power consumption still hang in the balance. Hence, some papers proposed the design method of using dual  $V_{dd}$  and dual  $V_{th}$  at the same time on gate level to further reduce power dissipation [7] and circuit level [8].

The work presented in this paper focuses on high level power optimization. We address the problem of scheduling a data-flow graph, for the case when the resources operate at dual supply voltage and dual threshold voltage. An algorithm which combines genetic algorithm with simulated annealing was used to assign the voltage for each node of the CDFG. The contributions of this paper are 1) take both dynamic power and static power consumption into account; 2) a novel application of *genetic algorithm based simulated annealing algorithm* is used in high level synthesis.

# II. SIMULATED ANNEALING AND GENETIC ALGORITHM

The goal of high level synthesis is to map the high level descriptions to hardware structures that meet the design constraints such as area, latency, and power consumption. There are many algorithms available in the high level synthesis, and the simulated annealing (SA) and genetic algorithm (GA) are two of them.

Simulated annealing [9], SA in short, is an optimization technique which is naturally motivated by the process of annealing. Simulated annealing starts with a high temperature T. By applying a *neighborhood* operation, a current state i (with energy  $E_i$ ) may change to the state j (with energy  $E_j$ ), when  $E_j < E_i$ . If  $E_j > E_i$ , the state i is replaced by the state j with probability  $e^{(E_i - E_j)/T}$ . The process is repeated with a new state, and a lower temperature comes from the cooling function until the temperature is smaller than the termination temperature  $T_f$ .

The genetic algorithm (GA) is a method which explores the design space to find a local optimal solution. The genetic algorithm consists of four steps: crossover, mutation, natural selection, and survival of the fitness. The detailed descriptions can be found in [10].

Both of the above algorithms can be used to solve the optimization problem. Generally, the process of simulated annealing is hard to parallelize, but the genetic algorithm is a naturally parallel algorithm. However, the genetic algorithm is hard to converge to a good result. The proposed GASA (Genetic Algorithm based Simulated Annealing) algorithm inherits strengths from both GA and SA, and gets rid of the disadvantages of them. The GASA can be easily implemented in parallel. By parallelizing the algorithm, several machines can be gathered to speed up the computing time. Besides, the design space can be explored by performing neighborhood operation from SA.

#### III. LOW POWER SCHEDULING WITH GASA

In this section, we show our GASA (Genetic Algorithm based Simulated Annealing) scheduling method. First, we introduce the data flow of the GASA algorithm. Secondly, we talk about the dual  $V_{dd}$  and dual  $V_{th}$  library. And thirdly, the chromosome representation and some GASA operations and parameters are introduced in detail. Finally, an example of the GASA scheduling is illustrated.



Fig. 1. The GASA scheduling algorithm flowchart.

#### A. Data flow of GASA

The GASA algorithm runs with several simulated annealing processes in parallel. The *mutation* operation in GA is analogical to the *neighborhood* operation in SA, and *crossover* operation represents the role of recombining independent solutions. Before we come to the GASA flow, one term must be defined first.

**Definition:** A *Boltzmann trial* is defined as a competition between states i and j, and the probability of state i wins the competition is  $1/(1 + e^{(E_i - E_j)/T})$ . Here, e is the natural constant, and  $E_i$  and  $E_j$  denote as the energy of state i and state j respectively. T represents the temperature in the SA algorithm.

By the definition, the energy  $E_i$  and  $E_j$  represent the powerdelay products of the scheduling results. If the power-delay product of state *i* is smaller than that of state *j*, then we define  $E_i < E_j$ . What should be noted is that if temperature *T* is large enough, then the next state *j* will be accepted even the energy of *j* is larger than the energy of *i*. By using the *Boltzmann trial*, the SA *uphill* operation can be presented in our scheduling algorithm.

The data flow of the GASA algorithm is shown in Fig. 1. In this flow, the inputs are the CDFG (Control/Data Flow Graph) and some parameters. The main scheduler assigns different  $V_{dd}$  and  $V_{th}$  to each node in the CDFG. Thus, a library which consists of several dual  $V_{dd}/V_{th}$  components is necessary to the scheduler. At the beginning of the scheduler, it brings out the first generation, and generates many individuals. Then, it randomly selects two individuals as the parents, and performs the *crossover* and *mutation* operations to generate two children. After that, the scheduler evaluates the power and delay of two parents and two children, and decides the *Boltzmann trial* winner by Definition 1. If the temperature cools down to the  $T_f$  (terminated temperature), it will output the trial winner, else it will continue the GASA loop.

In our scheduling algorithm, we try to recombine the results of each individual rather than just randomly generate a new

TABLE I An example of technology library with dual  $V_{dd}$  and dual  $V_{th}$ .

Multiplier						
	$V_{dd_H} / V_{th_H}$	$V_{dd_H} / V_{th_L}$				
Power	561.6	754.7				
Delay	18	17				
	V <sub>dd_L</sub> / V <sub>th_H</sub>	V <sub>dd_L</sub> / V <sub>th_L</sub>				
Power	273.5	365.3				
Delay	25	24				

(Power: *mW*) (*Delay: control cycle time*)

TABLE II Chromosome representation

One individual						
	$n_1$	$n_2$	• • •	$n_k$		
Relative CS	0	1	• • •	1		
instance $(V_{dd}/V_{th})$	L / H	H / L	•••	L/L		

individual. By combining the results of each individual, we can improve the convergence speed. The recombining phase will select two parents from the selection pool, and produce two children. The two children may have some essential parts of genes that make the fitness of the children better or worse than that of their parents. Here, the "essential" parts of genes represent those nodes belonging to the critical paths, or reducing large amount of power if we choose another  $V_{dd}$  or  $V_{th}$ .

# B. Dual $V_{dd}$ / $V_{th}$ library

An essential component of the GASA algorithm is the cell library, in which each cell has four instance types, as shown in Table I. Table I shows the power and delay of a multiplier with different  $V_{dd}$  and  $V_{th}$ . In Table I,  $V_{dd\_H}$  represents a cell with high supply voltage. Similarly,  $V_{th\_L}$  means a cell with low threshold voltage. In order to simplify the calculation, the delay of each instance type is set as the number of control cycles rather than the actual delay time.

## C. Individual and Chromosome Representation

A suitable chromosome representation is needed to represent the individual in the GASA scheduling algorithm, since it affects the running time of the algorithm. The chromosome representation must include the information of supply voltage, threshold voltage and control cycles. Table II shows the chromosome representation in our GASA algorithm. In Table II,  $n_i$  means the  $i^{th}$  node in the data flow graph, the *relative cs* of  $n_i$  shows the number of control steps between the starting control step of  $n_i$  and the maximum occupied control step of all preceding nodes of  $n_i$ . The *instance* records what kind of resources allocated to this operation; H indicates the high voltage and L indicates the low voltage. By checking the *instance* field of one node, we can look up the power consumption, area cost, and delay information from the library. Each column in the table represents a *chromosome*. If there are k nodes in the CDFG, this individual needs k chromosomes to represent itself.

Parent A				I			
Node	N1	N2	N3	N4	N5	N6	
Relative CS	1	2	0	3	1	1	
Instance ( V <sub>dd</sub> / V <sub>th</sub> )	H/L	H/H	L/H	L/H	L/L	L/L	
Parent B			$\geq$	$\leq$			
Node	N1	N2	N3	N4	N5	N6	
Relative CS	0	1	1	3	0	1	
Instance ( V <sub>dd</sub> / V <sub>th</sub> )	L/L	L/H	H/L	H/H	H/H	H/L	
Child A			$\bigcirc$				
Node	N1	N2	N3	N4	N5	N6	
Relative CS	1	2	0	3	0	1	
Instance ( V <sub>dd</sub> / V <sub>th</sub> )	H/L	H/H	L/H	H/H	H/H	H/L	
Child B					-	-	
Node	N1	N2	N3	N4	N5	N6	
Relative CS	0	1	1	3	1	1	
Instance (V <sub>44</sub> /V <sub>4</sub> )	L/L	L/H	H/L	L/H	L/L	L/L	

Fig. 2. An example of one point crossover operation.

#### D. GASA Operations and Parameters

1) Mutation operation: While performing the mutation operation on one individual, we will choose some chromosomes to mutate their values by a specific mutation rate. For example, if there are k nodes in one individual, and the mutation rate is  $P_m$ . We will choose  $k \times P_m$  chromosomes to mutate, while randomly changing the genes (Relative CS and instance). Through *mutation* operation, some variants of one individual can be produced to explore the neighbors of the current position in design space. Later we shall give a discussion on the mutation rate  $P_m$ .

2) Crossover operation: Crossover is a kind of recombination operation. Fig. 2 shows an example of the crossover operation. In the GASA scheduling, we adopt *one point crossover* operation which will exchange the right half part of two individuals to each other.

3) Population size: The population size is one of the major control parameters of the GASA. Generally, the larger of the population size, the better result we can obtain. However, the larger size of the population also requires the larger amount of memory and takes the longer operation time.

4) cooling procedure: The cooling procedure in our GASA scheduling is to multiply the current temperature by a cooling constant CC (0 < CC < 1). If CC is set with a large value, the temperature would reduce slowly and it would produce large generations. In generally, the population size and the cooling constant both affect the optimization gain and the computing time. The designer should tune both of these parameters to meet the design constraints.

5) Mutation rate: The value of mutation rate will affect the *difference* between parents and children. If the mutation rate is too large, some *good* chromosomes will be annihilated. If the mutation rate is too small, the resemblance between parents and children will be too close. Therefore, it will result in a local minimal solution. In our algorithm, we set the mutation rate as 20%. A larger mutation rate is set if the result seems to fall in a local minimal value.



Fig. 3. An example of GASA scheduling. (a) Original DFG. (b)  $V_{dd}/V_{th}$  of each node. (c) Scheduling result.

6) Temperature: The last two parameters are the starting temperature  $T_s$  and terminating temperature  $T_f$ . We set the terminating temperature as 0.1. The starting temperature  $T_s$  is set by the mathematical method.  $T_s = \frac{E_j - E_i}{-ln(\frac{1}{k} - 1)}$ , where  $E_j$  and  $E_i$  is the initial energy of state i and j, and k is the probability of  $E_i$  larger than  $E_j$ . The starting temperature influences the convergence speed and the accepting rate of the parent individuals at the beginning of the optimization process.

#### E. Example of GASA Scheduling

Suppose we have a library which consists of several components. Each component was implemented with four different kinds of the  $V_{dd}$  /  $V_{th}$  combination, as shown in Table I. The input file is the DFG, as shown in Fig. 3.(a). At the beginning of the scheduling, each node was assigned with different  $V_{dd}$  and  $V_{th}$ . After many loops, the scheduling result was shown in Fig. 3.(c), and the  $V_{dd}$  /  $V_{th}$  assignment was shown in Fig. 3.(b). Note that the different instances used in the scheduling process influence the final delay and the power consumption of whole design.

The goal of GASA is to minimize the power and delay penalty of a system. Through our GA-based simulated annealing approach, the nearly optimal solution can be achieved in the tolerable processing time.

## IV. EXPERIMENTAL RESULT

To show the effectiveness of our method, we compare the power consumption and delay overhead among three scheduling algorithms. The first is the ASAP (As Soon As Possible) scheduling method. The second is the dual  $V_{dd}$  only scheduling method, and the third is the proposed dual  $V_{dd}$  and dual  $V_{th}$  scheduling method. A dual  $V_{dd}/V_{th}$  library, which contains several components such as multiplier and adder, is used for low power scheduling. In this library, each component was designed by TSMC  $0.18\mu m$  process. The  $V_{dd_{-H}}$  was 1.8V,  $V_{dd_{-L}}$  was 1.26 V,  $V_{th_{-H}}$  was 0.558 V, and  $V_{th_{-L}}$  was 0.458V.

The experimental result is shown in Table III. Six CDFG benchmarks are used to examine the scheduling algorithms. It is clearly that the proposed dual  $V_{dd}/V_{th}$  can obtain the

	As Soon As Possible (ASAP) Scheduling			Dual $V_{dd}$ Scheduling			Proposed Dual $V_{dd}$ / $V_{th}$ Scheduling		
Benchmark	Power	Delay	Power * Delay	Power	Delay	Power * Delay	Power	Delay	Power * Delay
diffeq	5199.4	54	2807677.6	3079.0	62	190898	2300.2	64	147212.8
fir11	9979.5	109	1087765.5	5435.7	130	706641	5797.9	112	649364.8
dct	16436.8	72	1183449.6	10941.0	83	908103	9332	84	783888
iir7	13669.3	144	1968379.2	8519.8	158	1346128.4	7656.9	156	1194476.4
wdf7	17023.8	125	2127975	10576.0	150	1586400	9485.1	150	1422765
nc	28177.3	135	3803935.5	17834.9	151	2693069.9	14814.2	145	2148059

(Power: mW) (Delay: number of control cycles)



Fig. 4. A comparison of power saving and delay overhead between ASAP scheduling and dual  $v_{dd}/v_{th}$  scheduling.

best power-delay product, and it also shows the proposed method can obtain about 46.1% power saving and 12.3% delay overhead when compare with the ASAP scheduling method. Fig. 4 shows the comparison of power saving and delay overhead between ASAP scheduling and dual  $v_{dd}/v_{th}$ scheduling methods. In this figure, the solid lines represent the dual  $v_{dd}/v_{th}$  scheduling relative to the ASAP method.

We also show the convergence result of *diffeq* benchmark. Fig. 5.(a) and (b) are the power and delay convergence result by using dual  $V_{dd}$  only scheduling method. Fig. 5.(c) and (d) indicate the power and delay convergence result by using dual  $V_{dd}$  and dual  $V_{th}$  scheduling method.

The experimental result shows that we can get further power reduction when using a dual  $V_{dd}$  and dual  $V_{th}$  library with limited delay overhead. It means that the proposed method has a tradeoff between power consumption and delay. In this algorithm, most controlling parameters can be set automatically to reduce the designers' load. More constraints can be added to this algorithm such as area constraints, and the additional penalty so that it can provide more flexible design space.

## V. CONCLUSIONS

In this paper, a dual  $V_{dd}$  /  $V_{th}$  scheduling method is proposed for low power high level synthesis. In the proposed method, the dynamic and static power consumption are considered simultaneously. By using the GASA (genetic algorithm based simulated annealing) algorithm, each node in the CDFG (Control Data Flow Graph) is assigned with either a high or low  $V_{dd}$  /  $V_{th}$  to achieve the low power goal and to control the computing time. The experimental result shows that our



Fig. 5. Convergence result of *diffeq* benchmark. (a) power result with dual  $V_{dd}$ . (b) delay result with dual  $V_{dd}$ . (c) power result with dual  $V_{dd}/V_{th}$ . (d) delay result with dual  $V_{dd}/V_{th}$ .

method is feasible. The contribution of this paper is that the GASA method can be used on multiple  $V_{dd}$  /  $V_{th}$  scheduling and takes both power and performance into account at the same time.

#### References

- [1] http://public.itrs.net/
- [2] M. A. Elgamel, and M. A. Bayoumi, "On low power high level synthesis using genetic algorithms," *in IEEE Proc. of ICECS 2002*, vol. 2, pp. 725–728, Sept. 2002.
- [3] S. P. Mohanty, and N. Ranganathan, "A framework for energy and transient power reduction during behavioral synthesis," *IEEE Trans. on VLSI system*, vol. 12, No. 6, pp. 562–572, June 2004.
  [4] K. Usami, and M. Igarashi, "Low-power design methodology and
- [4] K. Usami, and M. Igarashi, "Low-power design methodology and application utilizing dual supply voltage," in IEEE Proc. of ASP-DAC, pp. 123–128, Jan. 2000.
- [5] D. Samanta, and A. Pal, "Synthesis of dual-V/sub T/ dynamic CMOS circuits," in Proc. of VLSI Design 2003, pp. 303–308, Jan. 2003.
- [6] K. S. Khouri, and N.K.Jha, "Leakage Power Analysis and Reduction During Behavioral Synthesis," *IEEE Trans. on VLSI Systems*, vol. 10, No. 6, pp. 876–885, Dec. 2002.
- [7] S. Augsburger, and B. Nikolić, "Combing dual-supply, dual threshold and transistor sizing for power reduction," *in IEEE Proc. of ICCD'02*, pp. 316–321, Sept. 2002.
- [8] A. Srivastava, and D. Sylvester, "Minimizing total power by simulaneous V<sub>dd</sub>/V<sub>th</sub> assignment," *IEEE Trans. on CAD*, vol. 23, No. 5, pp. 665– 677, May 2004.
- [9] P.J.M. van Laarhoven and E.H.L. Aarts, *Simulated annealing : theory and applications*, Kluwer Academic Publishers, 1987.
- [10] D. E. Goldberg, Genetic algorithm in search, optimization, and machine learning, Addison-Wesley, 1989.