行政院國家科學委員會專題研究計畫　成果報告

硬體描述語言低功率架構設計(3/3)

計畫類別：個別型計畫
計畫編號：NSC92-2213-E-002-004-
執行期間：92 年 08 月 01 日至 93 年 07 月 31 日
執行單位：國立臺灣大學電機工程學系暨研究所

計畫主持人：賴飛羆

計畫參與人員：蔡坤霖，張四維，張馨怡，鄭世揚，黃俊銘，陳立偉

報告類型：完整報告
報告附件：出席國際會議研究心得報告及發表論文
處理方式：本計畫可公開查詢

中　華　民　國 93 年 10 月 12 日

行政院國家科學委員會補助專題研究計畫 **☑ 成 果 報 告**
□期中進度報告

硬體描述語言低功率架構設計(3/3)

計畫類別：☑ 個別型計畫　　□ 整合型計畫
計畫編號：NSC 92－2213－E－002－004－
執行期間： 92 年 8 月 1 日至 93 年 7 月 31 日

計畫主持人： 賴 飛 羆　　教授
共同主持人：
計畫參與人員： 蔡坤霖、張四維、張馨怡、鄭世揚、黃俊銘、陳立偉

成果報告類型(依經費核定清單規定繳交)：□精簡報告　☑完整報告

本成果報告包括以下應繳交之附件：
□赴國外出差或研習心得報告一份
□赴大陸地區出差或研習心得報告一份
□出席國際學術會議心得報告及發表之論文各一份
□國際合作研究計畫國外研究報告書一份

處理方式：除產學合作研究計畫、提升產業技術及人才培育研究計畫、列管計畫及下列情形者外，得立即公開查詢
□涉及專利或其他智慧財產權，□一年□二年後可公開查詢

執行單位：國立台灣大學電機工程學系

中 華 民 國 93 年 10 月 12 日

摘要

本計畫的主要目的是在行為層上建立一個低功率電路的合成流程。隨著科技日新月異，製程已經進步到了奈米的時代，靜態功率消耗也已經變得跟動態功率消耗一樣地重要。為了管理功率消耗的情形，在這個計畫當中，我們提出了一個同時考量雙供應電壓與雙臨界電壓的低功率設計方法，來解決行為層合成的排程問題。當使用我們的方法時，可以獲得一個在功率消耗設計具有彈性設計空間。我們結合了基因演算法與模擬退火法來解決這樣的排程問題。實驗的結果證實了約 41.6%的功率可以被節省。

關鍵詞：低功率、高階合成、基因演算法、模擬退火法、雙供應電壓、雙臨界電壓

Abstract

The goal of this project is to design a synthesis flow for the low power circuit at behavioral level. As the technology scales down to the nanometer dimensions, the static power consumption has become as important as dynamic power consumption. To manage the power consumption, in this project, 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 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.

Keywords: low-power, high-level synthesis, genetic algorithm, simulated annealing, dual supply voltage, dual threshold voltage.

Contents

1. 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.

2. 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 $e^{\frac{E_i - E_j}{T}}$. 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.

3. 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 Vdd and dual Vth 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.

3.1 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 $\frac{1}{1+e^{\frac{E_i - E_j}{T}}}$. 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 power-delay 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} / 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.

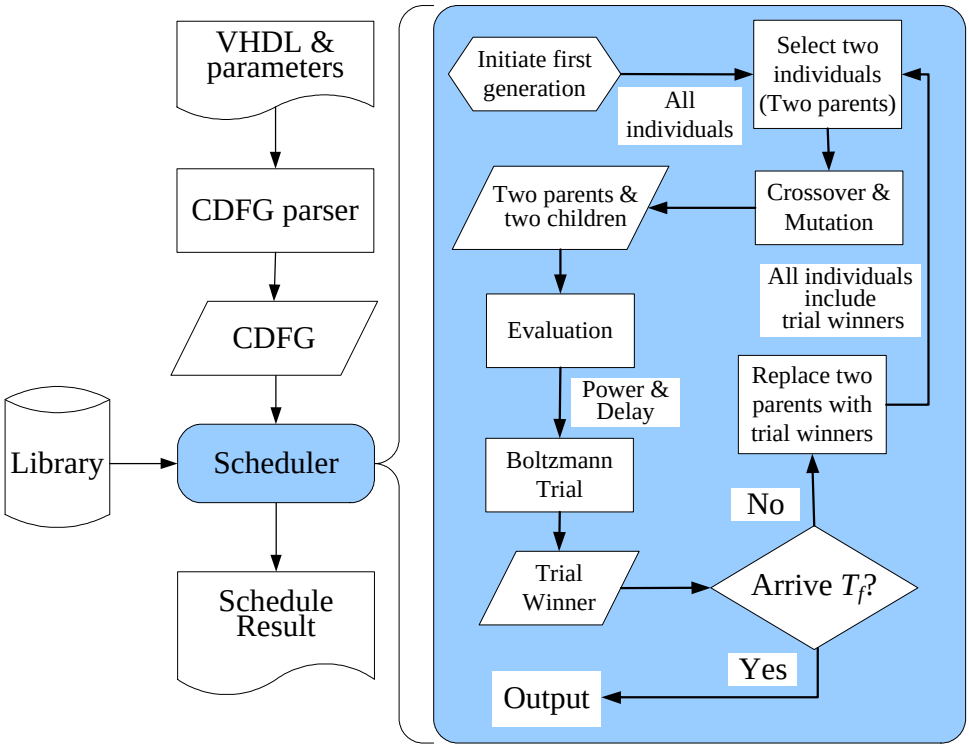


Fig. 1. The GASA scheduling algorithm flowchart.

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

3.2 Dual Vdd / Vth library

An essential component of the GASA algorithm is the cell library, in which each cell has four instance types, as shown in Table 1. Table 1 shows the power and delay of a multiplier with different Vdd and Vth. In Table 1, Vdd_H represents a cell with high supply voltage. Similarly, Vth_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.

Table 1. An example of technology library with dual Vdd and dual Vth.

Multiplier				
	V_{dd}_H/V_{th}_H	V_{dd}_H/V_{th}_L	V_{dd}_L/V_{th}_H	V_{dd}_L/V_{th}_L
Power (mW)	561.6	754.7	273.5	365.3
Delay(Control cycle time)	18	17	25	24

3.3 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 2. shows the chromosome representation in our GASA algorithm. In Table 2, 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.

Table 2. Chromosome representation.

One individual				
	n_1	n_2	…	n_k
Relative CS	0	1	…	1
Instance (Vdd / Vth)	L / H	H / L	…	L / L

3.4 GASA Operations and Parameters

3.4.1 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

3.4.2 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.

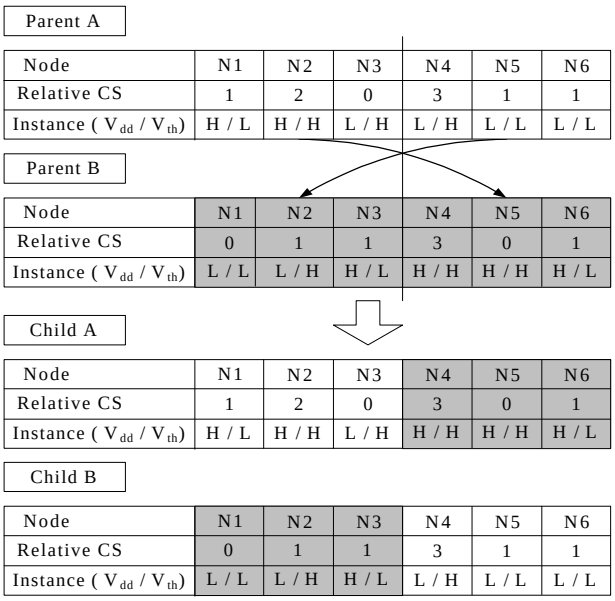


Fig. 2. An example of one point crossover operation.

3.4.3 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.

3.4.4 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.

3.4.5 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.

3.4.6 Temperature

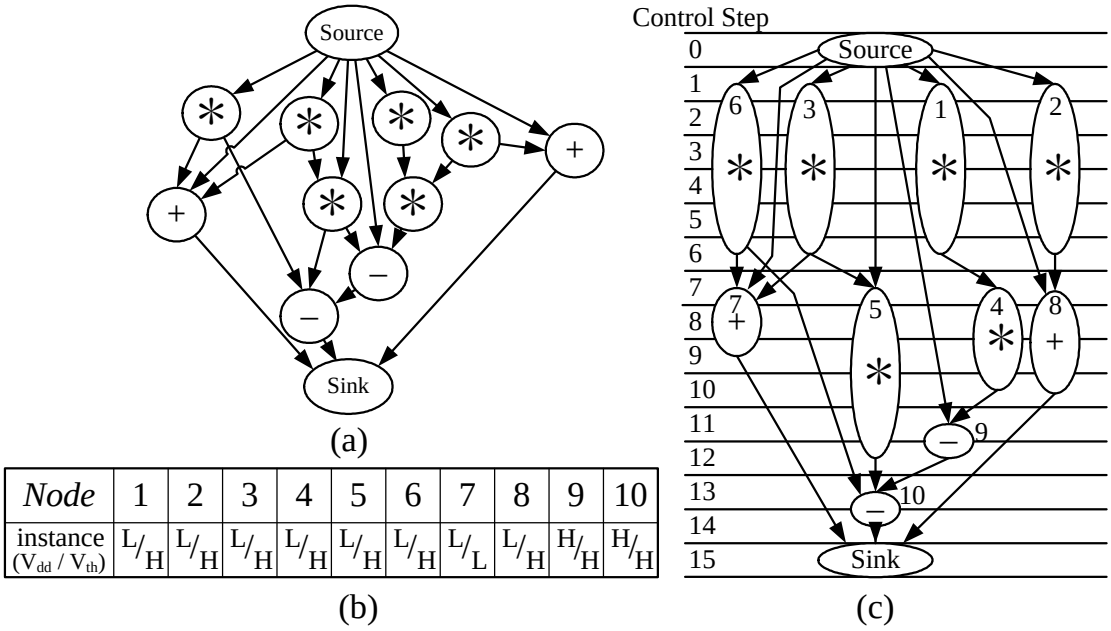
The last two parameters are the starting temperature T_s and terminating temperature T_f. We set the terminating temperature as *0.1*. The starting temperature T_s is set by the mathematical method. $T_s = \frac{E_j - E_i}{-\ln\frac{1}{k} - 1}$, where E_j and E_i is the initial energy of state *i* and *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.

3.5 Example of GASA Scheduling

Suppose we have a library which consists of several components. Each component was implemented with four different kinds of the Vdd / Vth combination, as shown in Table 1. The input file is the DFG, as shown in Fig. 3.(a). At the beginning of the scheduling, each node was assigned with different Vdd and Vth. After many loops, the scheduling result was shown in Fig. 3.(c), and the Vdd / Vth 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.



(a)

Node	1	2	3	4	5	6	7	8	9	10
instance (V_{dd} / V_{th})	L/H	L/H	L/H	L/H	L/H	L/H	L/L	L/H	H/H	H/H

(b)

(c)

Fig. 3. An example of GASA scheduling. (a) Original DFG. (b) Vdd / Vth of each node. (c) Scheduling result.

4. 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 Vdd only scheduling method, and the third is the proposed dual Vdd and dual Vth scheduling method. A dual Vdd / Vth 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 0.18μm process. The Vdd_H was 1.8 V, Vdd_L was 1.26 V, Vth_H was 0.558 V, and Vth_L was 0.458 V.

The experimental result is shown in Table 3. Six CDFG benchmarks are used to examine the scheduling algorithms. It is clearly that the proposed dual Vdd / Vth can obtain the 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 Vdd / Vth scheduling methods. In this figure, the solid lines represent the dual Vdd / Vth scheduling relative to the ASAP method.

Table 3. Experimental result

	As Soon As Possible		Dual Vdd only		Dual Vdd / Vth	
Benchmark	Power	Delay	Power	Delay	Power	Delay
diffeq	5199.4	54	3079.0	62	2300.2	64
fir11	9979.5	109	5435.7	130	5797.9	112
dct	16436.8	72	110941.0	83	9332	84
iir7	13669.3	144	8519.8	158	7656.9	156
wdf7	17023.8	125	10576.0	150	9485.1	150
nc	28177.3	135	17834.9	151	14814.2	145

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

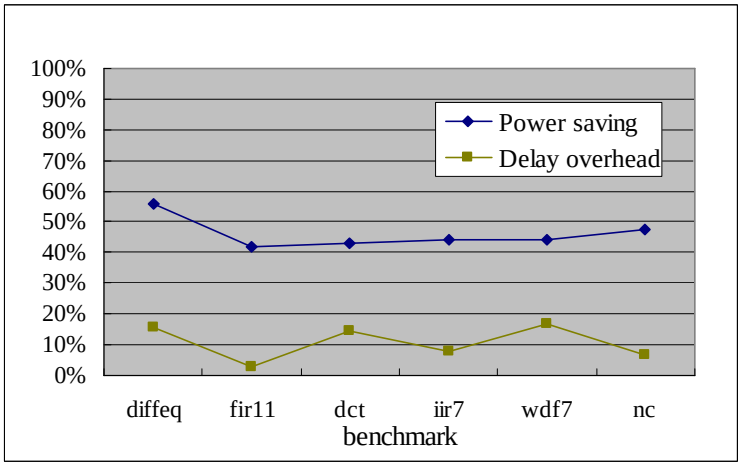


Fig. 4. A comparison of power saving and delay overhead between ASAP scheduling and dual Vdd / Vth scheduling.

The experimental result shows that we can get further power reduction when using a dual Vdd and dual Vth 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.

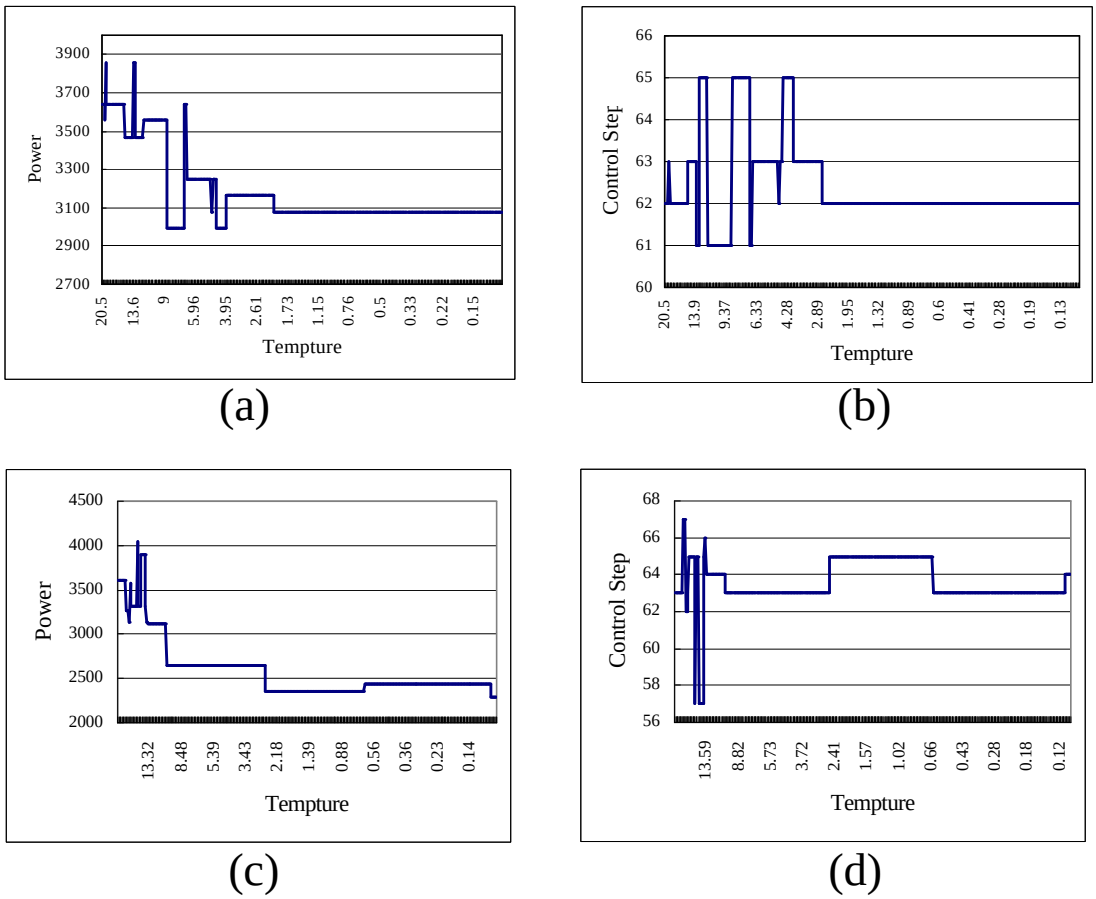


Fig. 5. Convergence result of diffeq benchmark. (a) power result with dual Vdd. (b) delay result with dual Vdd. (c) power result with dual Vdd / Vth. (d) delay result with dual Vdd / Vth.

5. Conclusions

In this paper, a dual Vdd / Vth 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 Vdd / Vth to achieve the low power goal and to control the computing time. The experimental result shows that our method is feasible. The contribution of this paper is that the GASA method can be used on multiple Vdd / Vth scheduling and takes both power and performance into account at the same time.

參考文獻

[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 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. Nikolic, “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_{dd}/V_{th} 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.

計畫成果自評

I. 原訂計畫目標：

本計畫的目的在於建立一個在行為層對低功率電路的自動化合成環境。藉此，電路設計者能快速、方便地設計出一套低耗電、高效能的電路架構。此一行為描述合成系統可讓低功率 IC 設計者在設計程序的早期很快速地加入低功率的特性。本計畫的原訂目標包含 1) 硬體描述語言與功率消耗關係分析 2) 針對電路特性發展低功率電路架構 3) 行為層硬體語言低功率電路轉換 4) 低功率電路合成。

II. 研究內容與原計畫相符程度

完全符合

III. 預期目標達成情況與綜合自評

本計畫的研究成果主要針對行為層的硬體描述語言提出一個低功率的架構。對於高階合成(high-level synthesis)而言，主要包含 3 個階段，及排程(scheduling)、資源配置(resource allociation)與資源繫結(resource binding)。而在計畫當中，我們對於高階排程提出了一個高度可用性的方法。在實做上，我們首先將硬體描述語言轉換成控制資料流程圖(CDFG)，再藉著雙供應電壓與雙臨界電壓的元件庫，對 CDFG 進行排程，以元件庫當中的多型態元件來替換某些特定的運算單元。在賴飛羆教授的帶領與督導下，本研究進行的進度與原先計畫的進度完全一致，研究成果已投稿至 IEEE ISCAS 2005。綜觀本計畫執行情形與研究成果，堪稱非常滿意。