AN ABSTRACT OF THE THESIS OF

Shyh-Sen Huang for the degree of Master of Science in Electrical and Computer Engineering presented on June 13, 2006.

Title: Low Power Scheduling Schemes that Consider Latency and Resource Constraints at Multiple Voltages

Abstract approved:

Zhongfeng Wang

Power reduction can be achieved at many different levels, such as architecture, algorithm, logic, and transistor levels in circuit design. This thesis focuses on low power scheduling at the algorithm level. We present a latency-constrained scheduling and a latency and resource constrained scheduling, which minimize power consumption for the resources and registers operating at multiple cycles and voltages. The proposed schemes are based on the consideration of assigning most nodes to low voltage and reducing the number of registers simultaneously. We present a comprehensive scheduling methodology to decrease average power, peak power, and the number of registers during the scheduling stage. To target this goal, we propose the efficient algorithms to schedule DFG with the least power dissipation that satisfies the system constraints in timing and resources. Our benchmark shows that our approach has succeeded to reduce power in 12.99~41.97% in latency-constrained scheduling and 20.19~80.64% in latency and resource constrained scheduling for those multimedia kernels.

Low Power Scheduling Schemes that Consider Latency and Resource Constraints at Multiple Voltages

by

Shyh-Sen Huang

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented June 13, 2006

Commencement June 2007

Master of Science thesis of Shyh-Sen Huang
presented on June 13, 2006.

APPROVED:

_____

Major Professor, representing Electrical and Computer Engineering

_____

Director of the School of Electrical Engineering and Computer Science

_____

Dean of the Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

_____

Shyh-Sen Huang, Author

ACKNOWLEDGEMENTS

This thesis is dedicated to my parents, and my girlfriend Jasmine. They always give me their support and I could never finish it without them. In addition, I would like to thank for my advisor, Dr. Zhongfeng Wang. He advised me during my research and gave me advice in my studies. I appreciate it very much. I also would like to thank my friends Yuhao Chang and Pat Meeker for their suggestions and support.

TABLE OF CONTENTS

Page

1. Introduction …… 1

1.1 The General Synthesis Flow……..…….. 3

1.2 Low Power Design at Various Levels of Abstraction…….. 4

2. High Level Synthesis Background…….... 7

2.1 Introduction…….…….. 7

2.2 Scheduling……..……... 7

2.2.1 Transformational Approaches……... 8
2.2.2 Iterative Techniques…….. 8
2.2.2.1 ASAP and ALAP……..……...…... 8
2.2.3 Resource Constrained Scheduling……..…… 9
2.2.4 Latency (Time) Constrained Scheduling…… 9
2.2.5 Operation at Multiple Cycles……..…… 10

2.3 Allocation and Binding…… 10

2.4 Power Dissipation of CMOS Circuit…… 12

2.5 Multiple Voltages Scheduling…….. 15

TABLE OF CONTENTS (Continued)

Page

3. Proposed Low Power Scheduling……..... 17

3.1 Delay and Power Characteristics…… 17

3.2 Lagrange Multiplier Method…….……. 20

3.3 Register Effect……..……... 21

3.4 Branches Determination…….……. 23

3.5 Latency Constrained Scheduling……. 25

3.5.1 Latency Constrained Scheduling Algorithm…….. 26

3.5.2 Latency Constrained Scheduling Example……. 27

3.5.2.1 Determine the Tolerance…….. 27

3.5.2.2 Voltage Assignments for Latency Constraint……..…… 28

3.6 Latency and Resource Constrained Scheduling…… 32

3.6.1 Iterative Method for Latency and Resource Constrained Scheduling… 32

3.6.1.1 First Step Scheduling……..……...…. 33

3.6.1.2 Second Step Scheduling……..……..... 33

3.6.1.3 Hybrid Method……..……...…… 34

3.6.2 Example for Latency and Resource Constrained Scheduling…….. 35

TABLE OF CONTENTS (Continued)

Page

4. Results and Benchmarks …… 40

4.1 Latency Constrained Scheduling…… 40

4.2 Latency and Resource Constrained Scheduling…….. 41

4.2.1 Differential Equation (Diffeq)……..... 42
4.2.2 AR Filter……..... 43
4.2.3 2nd Order Lattice Filter……..... 44
4.2.4 5th Order Elliptic Wave Filter……..... 45
4.2.5 8 Point Fast Fourier Transform (FFT)……..... 46

4.3 Power Consumption by Registers in Multiple Voltages Scheduling…… 47

4.3.1 Power Consumption by Registers in Latency Constrained Scheduling...47
4.3.2 Power Consumption by Registers in Latency and Resource Constrained Scheduling……...…… 47

5. Conclusion…… 50

Bibliography ……..... 51

LIST OF FIGURES

Figure Page

1. Design Flow and the Levels of Abstraction……..... 4

2. Operations at Multiple Cycles……..... 10

3. Delay for a 32-bit Carry-Ripple Adder, Multiplier, and Register……. 17

4. Power for a 32-bit Carry-Ripple Adder, Multiplier, and Register…… 19

5. Power Consumption by Registers……..…..…… 22

6. DFG for example……..…… 23

7. Starting Nodes of the Prospected Branches……..…… 25

8. Latency Constrained Scheduling Algorithm…… 27

9. Latency and Resource Constrained Scheduling Algorithm…….……. 35

10. Illustrative Example DFG……...……... 36

11. Scheduled Nodes after First Step……...…… 37

12. Scheduled Nodes after Second Step……...…… 39

13. Power Reduction for Latency Constrained Scheduling……...…… 41

LIST OF FIGURES (Continued)

Figure Page

14. Power and Peak Power of Diffeq in Latency and Resource Constrained Scheduling…… 43

15. Power and Peak Power of AR Filter in Latency and Resource Constrained Scheduling……. 44

16. Power and Peak Power of 2nd Order Lattice Filter in Latency and Resource Constrained Scheduling…….. 45

17. Power and Peak Power of 5th Order EW Filter in Latency and Resource Constrained Scheduling……..... 46

18. The Ratio of Power Consumption by Registers in Latency Constrained Scheduling……...…… 47

19. Ratio of Power Consumption by Registers in Latency and Resource Constrained Scheduling……...…… 49

LIST OF TABLES

Table Page

1. Delay for a 32-bit Carry-Ripple Adder, Multiplier, and Register…… 17

2. The Number of Required Cycles Used for Different Logics…… 18

3. Power for a 32-bit Carry-Ripple Adder, Multiplier, and Register…… 18

4. Power for a Level Shifter…….. 19

5. Priority for Different Resource by E/D…… 21

6. Ready Sets S, T, M, and A…… 24

7. Tolerance for Each Branch…… 28

8. The Number of Required Cycles for Choosing Resources in Lower Voltages….. 29

9. Tolerance for Multipliers in Each Branch…… 31

10. Latency in Different Voltage Assignments for the Branches……..…… 32

11. Final Voltage Assignment for the Example…… 32

12. The Resource Constraint and the Delay for Each Resource…… 36

13. ASAP and ALAP for Illustrative DFG……. 36

LIST OF TABLES (Continued)

Table Page

14. Power Reduction for Latency Constrained Scheduling…… 40

15. Power Reduction for Diffeq in Different Latency Constraints…… 41

16. Diffeq for Latency and Resource Constrained Scheduling…… 42

17. AR Filter for Latency and Resource Constrained Scheduling…… 43

18. 2nd Order Lattice Filter for Latency and Resource Constrained Scheduling….. 44

19. 5th Order EW Filter for Latency and Resource Constrained Scheduling…… 45

20. 8 Point FFT for Latency and Resource Constrained Scheduling…… 46

21. The Ratio of Power Consumption by Registers in Latency and Resource Constrained Scheduling…… 48

Low Power Scheduling Schemes that Consider Latency and Resource Constraints at Multiple Voltages

1. INTRODUCTION

Power consumption has become an important factor in electronic portable system design, where excess power dissipation may lead to less reliability. Also, the demand for personal computing devices and mobile communication equipment is increasing. More and more consumers use small mobile devices in their daily life. They may talk and surf the Internet using their smart phone, or use a Personal Digital Assistant (PDA) for watching movies or for car Global Positioning System (GPS) guidance. However, these electronic portable devices are extremely limited by battery performance. Unlike the desktop Personal Computer (PC), these handheld devices have limited resources because they are designed to require less power. They are still expected to maintain an acceptable performance. The rapid progress in semiconductor technology has also led to higher chip density and operating frequency, which makes these mobile devices more complex and power-consuming. Therefore, power consumption has become an important issue in circuit design for these mobile devices. Advances in silicon technology also enable entire systems to be integrated on a single chip, which is known as Systems-on-a-Chip (SoC). The development of low power devices, circuits, algorithms, architectures, and CAD tools are fundamental for the successful realization of SoC.

Our work presented in this thesis focuses on the power optimization (minimize average and peak power) at the algorithm level. Transformations at the algorithm level

have been presented in [1]. In addition, research in reducing power consumption of functional units has been proposed in [2]-[5]. There has also been research done in low power design at the behavioral level, such as reducing the number of registers and switching activities in registers, as well as efficient register allocations [3]. The problem of latency-constrained scheduling with multiple voltages has also been addressed [6], which computes the voltage assignment of each node, and is more complex than our latency-constrained algorithm. It does not consider the power consumption of registers and level shifters. An optimal solution has been found for the latency-constrained scheduling problem under variable voltage [7]. However, it did not consider resource constraint. Other techniques do not consider the effect of registers, which may result in an unrealistic scheduling [8][9]. Some attempts to minimize the switching activity for various resources (registers and functional units) have been made, but only considering single voltage [10]-[12].

The proposed latency-constrained scheduling algorithm in this thesis is executed in polynomial time to target resources operating at multiple voltages (5, 3.3, 2.4, 2.2, 1.8, 1.5, 1.2, and 1v). In addition, this thesis considers power consumption by the registers and level shifters operating at multiple cycles. The proposed heuristic latency ***and*** resource constrained scheduling algorithm also considers the power consumption of registers operating at multiple voltages. The concept of our latency ***and*** resource constrained scheduling algorithm is similar to [8]. However the authors of [8] did not consider the effects of registers. Disregarding registers may result in unrealistic low power optimization, since registers actually consume a large amount of power and

cause delay. We also propose a hybrid method which can reduce the number of iterations in latency *and* resource constrained scheduling.

This thesis is organized as follows: In Chapter 1, we present some background about general synthesis flow and low power design at various levels of abstraction. In Chapter 2, we present background on High Level Synthesis, including Scheduling, Allocation, Binding, and power dissipation of CMOS circuits. We also explain multiple voltage scheduling. In Chapter 3, we propose our scheduling algorithms for latency-constrained scheduling and latency *and* resource constrained scheduling problems. We also illustrate our algorithms by simple examples. Experimental results for different multimedia kernels are shown in Chapter 4. Finally, we present the conclusions in Chapter 5.

1.1 The General Synthesis Flow

The hardware design flow can be divided into two typical flows. One is the full-customized design flow, and the other is the cell-based design flow. Full-customized design flow has advantages such as high performance and small area cost compared to cell-based design flow. However, to speed up designs for the market, the cell-based design approach has its significant advantages, especially with the aid of Electronic Design Automation (EDA) tools. The cell-based design flow is described in **Figure 1**, which shows a typical design flow and the levels of abstraction.

The abstract description at the system level specifies the system behavior as a relationship between inputs and outputs in terms of function and timing requirements.

The algorithm level is characterized by a partitioned system description consisting of a netlist of architectural or algorithmic descriptions, each suitable for realization on a single chip. This partitioned description is then refined to an intermediate register transfer level (RTL) design, which describes the chip behavior in terms of Boolean equations and implicit register assignments. At the logic level, the design is represented with a detailed logic gate description and associated gate delays. Finally, the circuit level describes the physical layout-based timing information for gates.

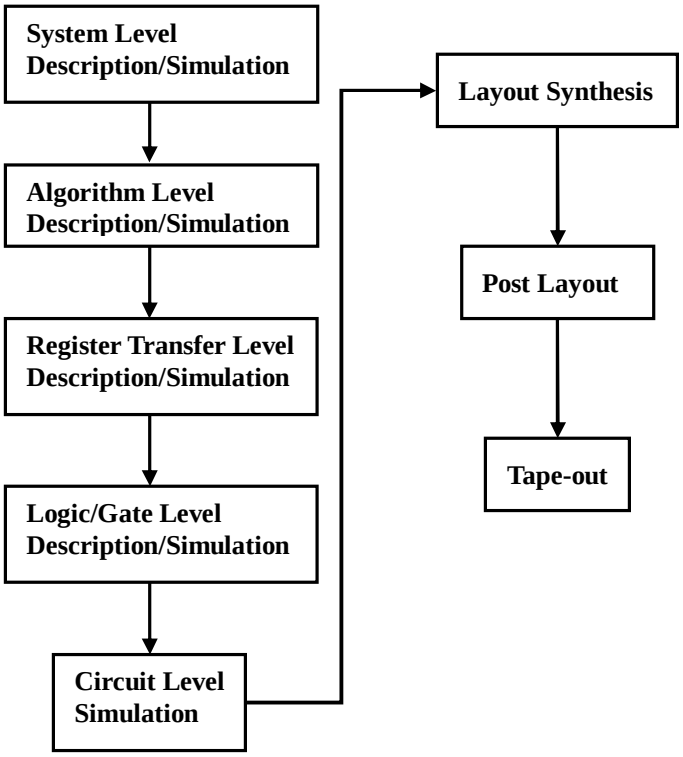


Figure 1. Design Flow and the Levels of Abstraction

1.2 Low Power Design at Various Levels of Abstraction

VLSI design flow takes into account all levels of design: system, behavior (algorithm), logic, circuit, and layout level. The process to design a low-power system has been

established completely, which improves the power in all levels of design process [13]. Typically, low power design can be implemented in the following different levels.

System Level

The system level design deals with connecting the resources, setting up communication, and operating the resources in a functionally correct and efficient fashion. Because digital circuit designers are attuned to the challenges of low power design, software engineers are often less power-conscious.

Behavior Level

Design at the behavior level is made by a hardware description language such as Verilog HDL or VHDL. By the process of behavior synthesis, the design can be transformed into logic level. There are three common approaches to high level power minimization. One approach attempts to minimize the switching activity of the circuit. The second approach tries to minimize the capacitance of the design, but it must rely on behavior power estimation tools that provide accurate information on capacitance. The third approach is to reduce the supply voltage and the threshold voltage.

Logic Level

Logic level power estimation and optimization has been one of the most extensively researched areas in CAD systems. Many techniques are used to deal with the components of logic level descriptions, and these optimization techniques can be seamlessly integrated in a traditional synthesis-based design flow. Because they are

applied late in the design process, they will only reduce power consumption by relatively small amount. One way to eliminate this problem is to optimize power at a higher level of abstraction.

Circuit Level

Except for the technology scaling, reducing the supply voltage for a given technology results in significant power consumption reduction. However, voltage reduction comes at the expense of slower gate speeds. It has become apparent that the voltage scaling approach is insufficient by itself, so we have to focus on advanced design tools and methodologies which address these power issues.

Layout Level

A considerable number of studies have been done on low power CMOS design by adjusting the supply voltage and threshold voltage, which requires a tradeoff between power consumption and delay. The status of silicon-on-insulator approach to scaled CMOS shows potential for 3x saving in power. The performance improvement of silicon-on-insulator compared to bulk CMOS is due to the reduction of parasitic capacitances and body effect.

2. HIGH LEVEL SYNTHESIS BACKGROUND

2.1 Introduction

High Level Synthesis is also called Behavioral Level Synthesis. The benefits of High Level Synthesis to digital system design include:

- Shortens design time to market.
- Significantly lowers design cost.
- Verifies the synthesis process so that fewer errors occur.
- Explores the design space by considering different trade-offs.
- Documents the process of design, and evaluates their effect.

High Level Synthesis consists of the following three interrelated tasks. Because of the complexity of the problem, each task is usually researched separately.

- Scheduling
- Allocation
- Binding

2.2 Scheduling

Scheduling contains information about the exact start time of each operation. This information is added to the initial behavioral description. Scheduling algorithms can be grouped into (1) transformational approaches and (2) iterative techniques. It can be also classified by its goals into (1) latency-constrained scheduling, (2) resource-constrained scheduling or (3) latency *and* resource constrained scheduling.

During scheduling, the operations can be executed in one clock cycle or *n* clock cycles

(i.e., multi-cycle operations), where *n* is any integer number greater than 1.

2.2.1 Transformational Approaches

Transformational algorithms are used to obtain a new schedule from a default schedule by applying transformations. The fundamental transformations consist of moving operations or blocks executed serially into the same execution interval (parallel execution), or moving operations executed concurrently into subsequent time steps (serial execution). These approaches use Branch and Bound algorithms, ILP formulations, Trace and Percolation Algorithms, and Stochastic techniques.

2.2.2 Iterative Techniques

For iterative techniques, each operation is assigned consecutively until all operations are scheduled. The differentiation among these algorithms is the basis for deciding which operation will be scheduled next. The simplest algorithms are the As Soon As Possible (ASAP) algorithm and the As Late As Possible (ALAP) algorithm. Although both of these assume unlimited resources, their significance includes that they determine: (1) the fastest possible implementation, (2) the critical path, and (3) an upper bound on the number of required hardware resources.

2.2.2.1 ASAP and ALAP

ASAP scheduling places all the operations into the earliest possible control step and results in the shortest schedule. For this reason, the ASAP scheduling gives a lower bound on the schedule length. In addition, the ASAP schedule gives an upper bound on the number of required hardware resources. This upper bound is determined with

the maximum number of operations of a given type used in any control step. To obtain the ALAP schedule, we need to be given an execution time constraint. The schedule length is set to this constraint and all operations are assigned to the latest possible control step.

2.2.3 Resource Constrained Scheduling

The aim of resource-constrained scheduling is to assign operations to control steps such that execution time is minimized for a given number of functional units (adders, and multipliers). Some resource-constrained algorithms are:

(1) **List Scheduling**: Places all operations whose inputs are available into a ready list, and then the list is sorted in a priority function. The operations in the sorted list are subsequently assigned until either all operations are scheduled or all hardware resources have been used [14].

(2) **Force Directed List Scheduling**: Similar to list scheduling, with the distinction that the selection of a candidate operation to be scheduled in a given time step is done by using the concept of force [15][16].

(3) **Integer Linear Programming (ILP)**: Provides the optimized solution for the scheduling, but the computation time is high for a large circuit.

2.2.4 Latency (Time) Constrained Scheduling

The aim of latency-constrained scheduling is to assign operations to control steps such that the number of functional modules is minimized for a given execution time. Some of the most important algorithms are:

(1) **Force Directed Scheduling**: It considers the operations one at a time for

scheduling, in contrast to the strategy of considering each schedule step singly as done by list scheduling [15][16].

(2) **Integer Linear Programming (ILP)**: It gives an exact solution but has the same disadvantage as ILP algorithm in resource-constrained scheduling, i.e., the computation time is high.

2.2.5 Operation at Multiple Cycles

In behavioral synthesis, multi-cycle techniques are widely used, which allow an operation to be executed during two or more control cycles. They can increase the performance or minimize the area. In **Figure 2**, we can see that **Op1** is a single cycle operation; **Op2** and **Op3** are executed during two and three control cycles, which are multi-cycle operations. Multi-cycle functional units can also be employed to reduce the power consumption of digital designs in behavioral synthesis approaches [8] [17].

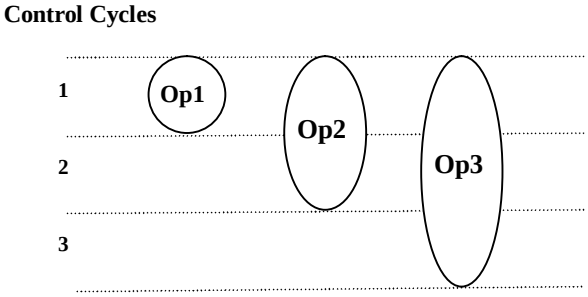


Figure 2. Operations at Multiple Cycles

2.3 Allocation and Binding

Allocation determines the quantity and type of resources used in the chip architecture. It also determines the clocking scheme, memory hierarchy and pipelining scheme [22].

It consists of the following different but interdependent tasks:

- **Module/Functional Unit Allocation**: Determines the number of required arithmetic and logic units
- **Storage Allocation**: Assigns the variables given in the behavioral description to the registers
- **Interconnect Allocation**: Allocates buses and multiplexers in the datapath to connect the functional modules and registers.

This decomposition frequently results in circuits that require more interconnections than necessary when executed independently.

Binding maps operations to functional modules and variables to registers. This allows for the cost of the necessary interconnection structure to be minimal. It can be decomposed into:

- **Functional Unit Binding:** Determines the exact mapping of the operations into the functional units
- **Storage Binding:** Maps data carriers (constants or variable) in the behavioral description to storage elements (registers or memory) in the datapath
- **Interconnection Binding**: Maximizes the sharing of interconnection units (i.e., minimizes the interconnection cost).

The combination of allocation and binding is referred to as datapath allocation.

2.4 Power Dissipation of CMOS Circuit

Because CMOS digital integrated circuits have the advantageous features of low power consumption, large noise margins, and ease of design, they have been widely used in recent years. CMOS can be divided into digital CMOS circuits and analog CMOS circuits. We will focus on digital CMOS in this thesis. The design of portable devices needs to consider the peak power consumption for reliability, and the dynamic power consumption for battery life. The average power consumption is also important for thermal considerations and environmental concerns.

The average power dissipation P of a digital CMOS circuit is composed of two components:

$$P = P_{static} + P_{dynamic} \tag{2.4.1}$$

Static power consumption can be described as the following equation:

$$P_{static} = I_{leakage} \times V \tag{2.4.2}$$

where $I_{leakage}$ is the leakage current, and V is the operating voltage. P_{static} characterizes circuits that have a constant source of current between the power supplies. P_{static} becomes significant when a transistor fault such as “stuck-at on” occurs in CMOS circuits. However, this power contribution is not an issue for properly designed CMOS circuits.

$P_{dynamic}$ is the dominant part of the power dissipation in CMOS circuits, which can be decomposed into the following three terms:

$$P_{dynamic} = P_{switching} + P_{short\text{-}circuit} + P_{leakage} \quad (2.4.3)$$

$P_{switching}$ is called switching power, which is a result of the charge and discharge of the capacitances associated with each node of the circuit. The power consumption of a CMOS gate is given as equation (2.4.4) while ignoring the internal capacitances.

$$P = \frac{1}{2} a C_L V_{dd}^2 f \quad (2.4.4)$$

where α is the switching activity (i.e., the sum of the probabilities that a rising or falling transition occurs on the output in each clock cycle), C_L is the load capacitance, V_{dd} is the supply voltage, and f is the operating frequency. The switching power dominates most of the total power consumption. Therefore, it is important to estimate and minimize this component for the electronic components.

$P_{short\text{-}circuit}$ is called short-circuit power, which derives the short-circuit current from the supply voltage to the ground during output transitions. Finally, $P_{leakage}$ is called leakage power, which is due to the leakage current.

From equation (2.4.4), we know that power savings can be achieved by reducing the following parameters:

- Switching activity
- Load capacitance
- Supply voltage
- Clock frequency

These parameters are not mutually independent. It is essential in low power design to analyze the interactions and trade-offs among these parameters. For dynamic power consumption, we can lower the power consumption greatly by scaling down the operating voltage, and this will also reduce the leakage power consumption. However, scaling down operating voltage may also increase the delay time for operations. The following equations show the relationship between energy dissipation $E(V_{dd})$, execution time $d(V_{dd})$, and supply voltage V_{dd}.

$$E(V_{dd}) = \left(\frac{V_{dd}}{V_{max}} \right)^2 E(V_{max}) \qquad (2.4.5)$$

and

$$d(V_{dd}) = \frac{V_{dd}(V_{max} - V_t)^2}{V_{max}(V_{dd} - V_t)^2} d(V_{max}) \qquad (2.4.6)$$

where V_{max} is the maximum supply voltage, V_t is the threshold voltage, $E(V_{max})$ and $d(V_{max})$ are the energy dissipation and execution time at V_{max}. By equation (2.4.5) and (2.4.6), we find it will reduce the power dissipation by lowering the supply voltage at the expense of increasing the delay time.

2.5 Multiple Voltages Scheduling

Multiple voltage scheduling can be described as follows: assume that a designer knows the availability of low power functional units such as adders, multipliers, etc., and these functional units can be operated at different voltages. Supply voltages, delay, and power dissipation of these functional units are known. The designer can find out how to use the components from the given library, such that the whole design consumes the least power under a latency or resource constraint.

For dual supply voltage, V_L/V_H can be used to minimize power dissipation of circuits. One theory to deal with the optimal V_L/V_H is described in [18]. The power reduction ratio R becomes a minimum value when V_L is between $0.6V_H$ and $0.7V_L$. In multiple power supplies $\{V_1>V_2>\ldots V_n\}$, power dissipation is given by:

$$P_n = f\{(C_1 - \sum_{i=2}^{n} C_i)V_1^2 + \sum_{i=2}^{n} C_i V_i^2\} \qquad (2.5.1)$$

where C_i is total capacitance of circuits and interconnections while operating at V_i, and f is the operating frequency. The ratio of power dissipation in the multiple power supplies, compared to that in a single power supply, is given by:

$$\frac{P_n}{P_1} = 1 - \sum_{i=2}^{n} [\frac{C_i}{C_1}\{1 - (\frac{V_i}{V_1})^2\}] \qquad (2.5.2)$$

From equation (2.5.2), we see that multiple voltages scheduling techniques can obtain higher power efficiency than using single supply voltage by using the slack time. Nevertheless, they may affect the IC layout in the following ways [19]:

- An area increase due to the routing of the supplies will occur. There will be a trade-off between lower energy dissipation and higher routing cost.
- It may be necessary to partition the chip into separate regions, where operations in the same region operate at the same voltage.
- Isolation will be necessary between regions operated at different voltages.
- New design rules for routing may be needed to deal with signals at one voltage passing through a region of another voltage.

Another problem will be the area and delay overhead of the required level shifters.

3. PROPOSED LOW POWER SCHEDULING

3.1 Delay and Power Characteristics

The delay and power characteristics of different functional units operating at different voltages are obtained from [4], where the author used Mentor Graphic's Design Architect and Accusim to simulate these various units at different voltages (5v, 3.3v, 2.4v, 2.2v, 1.8v, 1.5v, 1.2v, 1.0v). We use the results of power and delay from a 32-bit carry-ripple adder, a 32-bit carry-ripple multiplier, and a 32-bit register, which are laid out by TSMC 0.35 μ m technology for our simulation. **Table 1** shows the delay values for resources operating at different voltages.

Table 1. Delay for a 32-bit Carry-Ripple Adder, Multiplier, and Register [4]

Volt (v)	5	3.3	2.4	2.2	1.8	1.5	1.2	1
Adder	**13.51**	**16.50**	**21.49**	**23.66**	**31.20**	**43.71**	**78.81**	**170.64**
Multiplier	**33.22**	**40.57**	**52.85**	**58.19**	**76.72**	**107.47**	**193.79**	**419.59**
Register	**3.67**	**4.50**	**5.63**	**6.44**	**8.12**	**11.27**	**20.01**	**43.95**

Delay (ns)

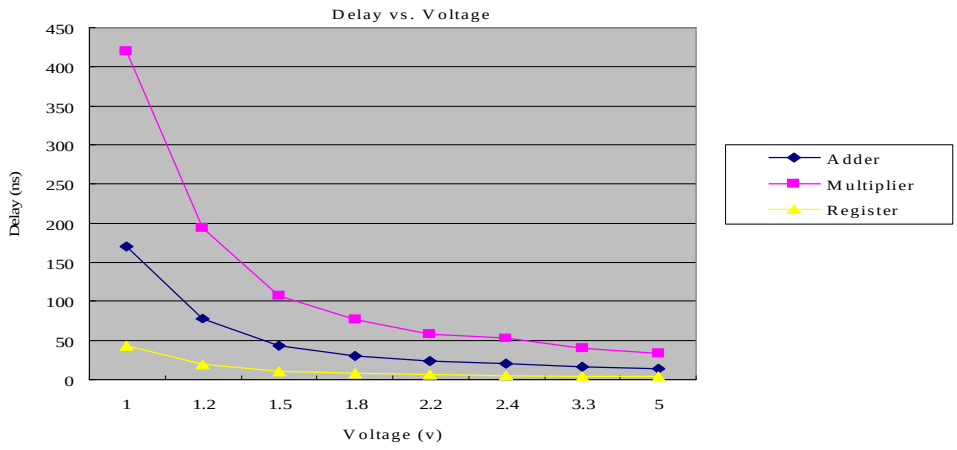


Figure 3. Delay for a 32-bit Carry-Ripple Adder, Multiplier, and Register

Table 2 shows the number of required cycles for resources operating at different voltages. Here we assume one cycle time is 10 ns.

Table 2. The Number of Required Cycles Used for Different Logics

Volt (v)	5	3.3	2.4	2.2	1.8	1.5	1.2	1
Adder	2	2	3	3	4	5	8	18
Multiplier	4	5	6	6	8	11	20	42
Register	1	1	1	1	1	2	3	5

Delay (cycle)

Table 3 shows the power dissipation for resources operating at different voltages. In this table, we can find that the power dissipation of a register cannot be ignored because its power value is approximately equal to the power value of an adder. We should note that the multiplier is approximately three times slower than the adder (**Figure 3)**, and consumes much more power compared to adders and registers (**Figure 4)**.

Table 3. Power for a 32-bit Carry-Ripple Adder, Multiplier, and Register [4]

Volt (v)	5	3.3	2.4	2.2	1.8	1.5	1.2	1
Adder	9335.60	3984.48	2068.21	1835.20	986.25	402.57	121.41	15.54
Multiplier	28431.00	12134.50	6298.60	5588.95	3003.56	1226.01	369.75	47.32
Register	8390.60	3558.90	1897.33	778.81	526.89	358.80	73.19	11.41

Power Characteristic (μ W)

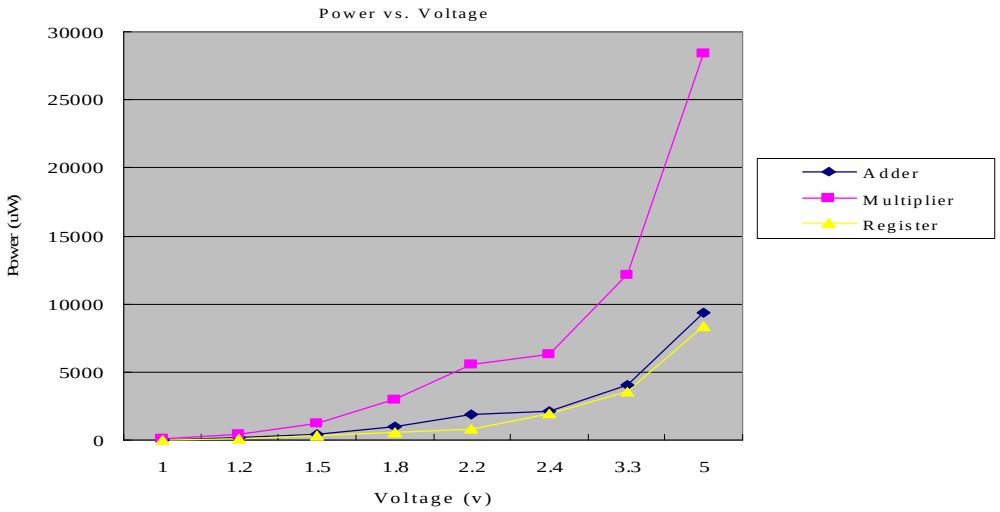


Figure 4. Power for a 32-bit Carry-Ripple Adder, Multiplier, and Register

We also get the power values for level shifters, which are used for data flow at different operating voltages. They are taken from [20], where the values are expanded to support more voltages from a curve-fitting algorithm by the author in [4]. In **Table 4**, we find that the power consumption of $V_{low\text{-}to\text{-}high}$ level shifter is less than $V_{high\text{-}to\text{-}low}$ level shifter in two different stages. We ignore the delay of level shifters in this thesis because they are negligible, compared to that of other functional units.

Table 4. Power for a Level Shifter (μ W)

To\From	1.0	1.2	1.5	1.8	2.2	2.4	3.3	5
1.0	0	45	55	66	80	87	118	177
1.2	32	0	59	70	95	92	125	187
1.5	39	44	0	76	92	100	136	203
1.8	46	52	62	0	96	108	146	220
2.2	55	63	74	85	0	118	160	320
2.4	60	68	80	92	108	0	167	253
3.3	81	92	108	124	145	156	0	356
5	121	137	160	184	220	234	260	0

3.2 Lagrange Multiplier Method

From [8], we see that we can distribute the slack among the nodes in the critical path by using the Lagrange multiplier method. The delay of a resource is determined by the delay of the gates on the critical path. In order to minimize the total energy subject to the time constraint, we use the Lagrange multiplier method to determine the supply voltage of each node, and we can find the minimum total energy while the specific condition is satisfied in the critical path. The relationship between the voltages of the nodes in the critical path is derived in the descriptions that follow.

We find that E_{total} is minimized while the following condition is satisfied among the nodes in the critical path from [8]:

$$\frac{a_1 C_1 V_1 (V_1 - V_t)^3}{C_{C_1}(V_1 + V_t)} = \text{.....} = \frac{a_n C_n V_n (V_n - V_t)^3}{C_{C_n}(V_n + V_t)} \qquad (3.2.1)$$

where C_n is the total load capacitance of the resource n, and α_n is its average switching activity. C_{C_n} represents the sum of the capacitances of the gates on the critical path of resource n.

The above equation can be simplified while $V_j > 3V_t$, and we can get equation (3.2.2):

$$\frac{a_1 C_1 (V_1 - V_t)^3}{C_{C_1}} = \text{.....} = \frac{a_n C_n (V_n - V_t)^3}{C_{C_n}} \qquad (3.2.2)$$

The $\frac{a_j C_j}{C_{C_j}}$ ratio is a constant for each resource, which is equal to $\frac{E_j}{D_j V_j (V_j - V_t)^2}$.

We let V_{ref} be the reference voltage and we can get:

$$\left[\frac{E_1}{D_1}\right]_{V_{ref}} (V_1 - V_t)^3 = \ldots = \left[\frac{E_n}{D_n}\right]_{V_{ref}} (V_n - V_t)^3 \qquad (3.2.3)$$

where E_n is the energy of resource n operating at V_n, and D_n is the corresponding delay.

In order to minimize the total energy, we see that we should assign lower voltage resources to the nodes with high $\left[\frac{E}{D}\right]$ from equation (3.2.3). From **Table 1** and **Table 3**, we conclude the following priority table (**Table 5**). We should note that *the higher priority means the resource should be disabled earlier in low power scheduling*.

Table 5. Priority for Different Resources by E/D

Priority	Resource	E/D value
1	**5v multiplier**	**861**
2	**5v adder**	**691**
3	**3.3v multiplier**	**299**
4	**3.3v adder**	**241**
5	**2.4v multiplier**	**119**
6	**2.4v adder**	**96**
7	**2.2v multiplier**	**96**
8	**2.2v adder**	**77**

3.3 Register Effect

Registers play an important role in all kinds of computer systems. They are necessary because they hold the values of variables, which are generated and consumed during

the execution of an operation. For mobile devices, registers are even more significant because of the power and timing requirements. In **Table 3**, we see that a single 32-bit register operating at 5v consumes power around 8,390 μW, and its delay is around 3.67 ns. Although both power usage and delay of a single register is negligible, the amount of power they consume is still significant, and the operating time of the entire system is increased because they are constantly used. In **Figure 5**, we can see the role that registers play, and find surprise in the ratio of power consumption by registers in the operation. For the simple circuit, in which the OP is operating at 5v, the corresponding registers consume around 47% or 73% of the total power depending on the operating resource. Therefore, it is essential to consider the power consumption by registers in low power design. With multiple voltages and multi-cycle techniques, we can reduce the average power and peak power for functional units and registers. The significance of registers is also addressed in [21], which discusses the latency-constrained scheduling problems.

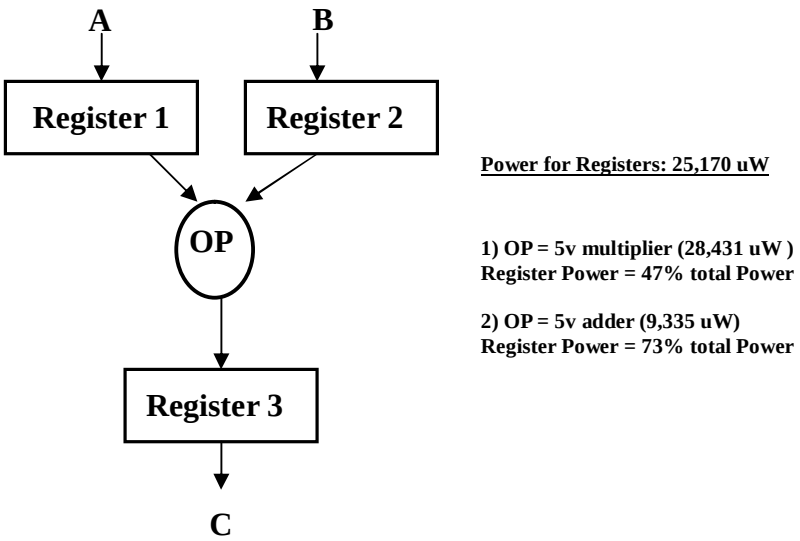


Figure 5. Power Consumption by Registers

3.4 Branches Determination

For the Data Flow Graph (DFG) in **Figure 6**, the input edges and node categories are depicted in **Table 6**, and stored in the ready sets S, T, M, and A, respectively. We present a heuristic algorithm which is efficient in determining the required branches. The input edges of DFG are stored in the ready sets S and T, where a DFG is a directed acyclic graph whose nodes represent operations, and edges represent dependencies between the operations. We then distinguish the nodes (i.e., remove the redundant nodes) for both sets and put the results in Set(S1) and Set(T1), respectively. When comparing Set(S1) and Set(T1), the duplicate values are excluded in the updated starting node set.

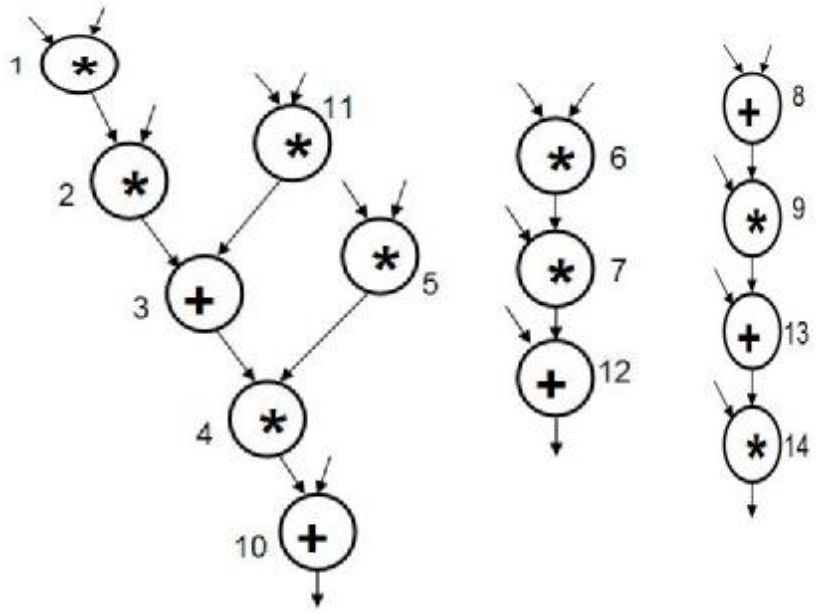


Figure 6. DFG for example

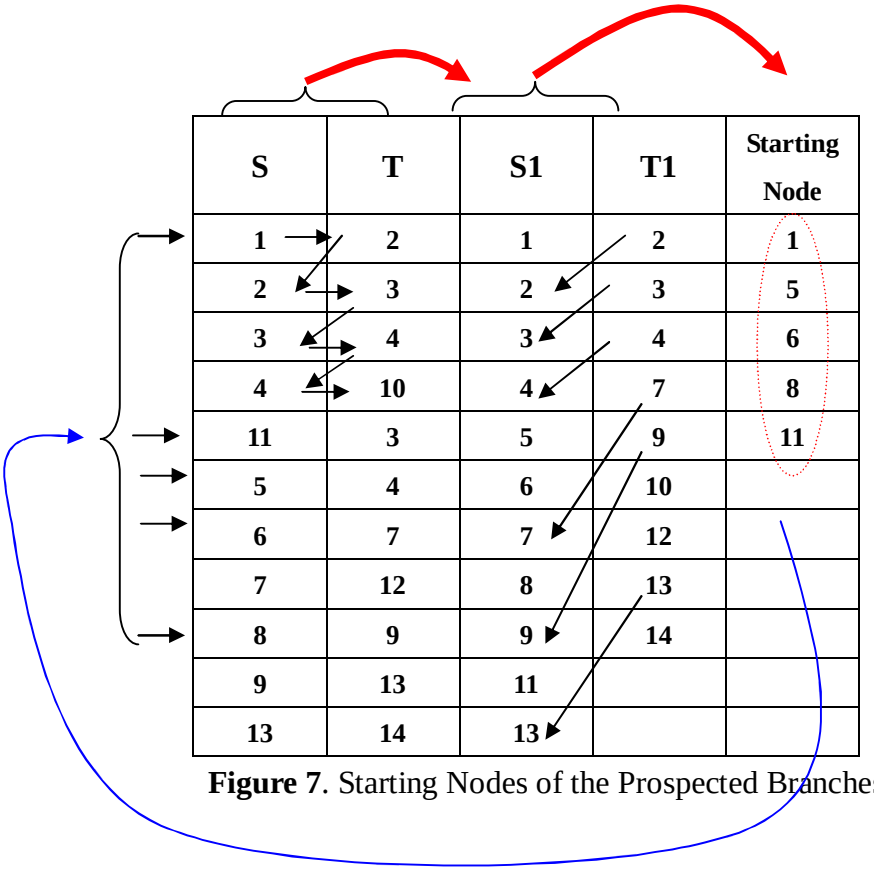
Table 6. Ready Sets S, T, M, and A

S	**1**	**2**	**3**	**4**	**11**	**5**	**6**	**7**	**8**	**9**	**13**
T	**2**	**3**	**4**	**10**	**3**	**4**	**7**	**12**	**9**	**13**	**14**

M	**1**	**2**	**4**	**5**	**6**	**7**	**9**	**11**	**14**
A	**3**	**8**	**10**	**12**	**13**				

Node Categories: {M: multiplier, A: adder}

For example, at the beginning, the Set(S) contains {1, 2, 3, 4, 11, 5, 6, 7, 8, 9, 13}, and Set(T) contains {2, 3, 4, 10, 3, 4, 7, 12, 9, 13, 14}, which refers to the corresponding edge (*node1→node2*), (*node2→node3*) ….. (*node13→node14*). After comparing Set(S1) and Set(T1), these nodes (2, 3, 4, 7, 9, 13) should be deleted in Set(S1). The remaining values in Set(S1) are updated to {1, 5, 6, 8, 11}, which are the starting nodes of these branches in the DFG. Then, we can find all branches by finding the sequenced nodes in Set(S) and Set(T). For instance, the starting node ‘1’ in Set(S) corresponds to ‘2’ in Set(T), backwards to ‘2’ in Set(S), and so on (i.e., the arrows in Set(S) and Set(T) in **Figure 7**). The first group in Set(S) and Set(T) consists of the nodes of 1, 2, 3, 4, 10, which makes up the branch P1.



S	T	S1	T1	Starting Node
1	2	1	2	1
2	3	2	3	5
3	4	3	4	6
4	10	4	7	8
11	3	5	9	11
5	4	6	10	
6	7	7	12	
7	12	8	13	
8	9	9	14	
9	13	11		
13	14	13		

Figure 7. Starting Nodes of the Prospected Branches.

By this method, all the branches are finally obtained as follows:

P1: {1, 2, 3, 4, 10};
P2: {11, 3, 4, 10};
P3: {5, 4, 10};
P4: {6, 7, 12};
P5: {8, 9, 13, 14}

3.5 Latency Constrained Scheduling

The input to a latency-constrained scheduling scheme is a DFG and a latency constraint. A latency constraint is a restriction for the maximum operating time in the scheduling of DFG.

3.5.1 Latency Constrained Scheduling Algorithm

After we obtain all the branches in the DFG, we assign the highest voltage to all nodes in the DFG to determine the minimum latency. We compare the minimum latency of each branch to determine their priority. Then, we assume the adders and registers are operating at the highest voltages in order to determine the largest tolerance for the multipliers in the branch. To minimize the power consumption by observing **Table 3**, we should assign a voltage, which is as close to the desired voltage as possible for the nodes which are ready to be scheduled. For example, if we can assign 3.3v for two multipliers, it will be better than assigning 5v and 2.4v individually for them. Not only will the power consumption be less, but we will use less registers and resources. Additionally, because the power consumption of multipliers is much larger than that of adders, we assign low voltage for multipliers as often as we can (multipliers have higher priority than adders in latency-constrained scheduling). We should note that if some multipliers only connect to each other and are not interconnected with adders, we assign a lower voltage to the multiplier in the top and a higher voltage to the one in the bottom. This is because the power consumption of the low-to-high level shifter is less than the high-to-low in two different voltage stages from **Table 4**.

After all the multipliers have been assigned their voltage, we update the tolerance for the adders in the branch. We use similar steps to assign the voltage for the adders. After all the adders in the branch have been assigned, we update the priority of branches and repeat the scheduling for the next branch. We show the latency-constrained scheduling algorithm in **Figure 8**, where the input is DFG, latency constraint L, and clock cycle time.

1. *Determine the number of cycles used for different logics by the given clock cycle time.*
2. *Find all branches in the DFG.*
3. *Determine minimum required cycles (P) for every branch, where all nodes are operating at 5v.*
4. *Computer the maximum tolerance(T) for every branch, where T=L(latency constraint)-P*
5. *Determine the priority by the minimum required cycles (P) of every branch.*
6. *For a branch, check the position for each multiplier (k is the # of multipliers in the branch)*

   For i=0, i++, i< (k+1) {

   If (the multiplier is the first or the last node of the branch)

   {T=T-1}

   If (the multiplier is connected with two other nodes, which they are not multipliers, in the branch)

   {T=T-2} }
7. *Determine the lowest voltage for multipliers by [T/ # of multipliers], and assign the voltage for multipliers by their tolerance.*

   If (the required cycles P'< L){

   Update T=L-P' & go to Step8 }

   Else { Repeat Step 7}
8. *Determine the lowest voltage for adders by [T/ # of adders], assign the voltage for adders, and update P'.*

   If (the required cycles P'< L) {

   Go to Step 9 }

   Else { Repeat Step 8 }
9. *Assign voltage for the next branch by their priority.*

Figure 8. Latency Constrained Scheduling Algorithm

3.5.2 Latency Constrained Scheduling Example

3.5.2.1 Determine the Tolerance

After we determine the branches in the previous example (**Figure 6**), the next step is to calculate the minimum required cycles for each branch, while the nodes thereof are assigned to 5v. For instance, the minimum required cycles for branch P1: 4 cycles

(*node1* is a multiplier) + 4 cycles (*node2* is a multiplier) + 2 cycles (*node3* is an adder) + 4 cycles (*node4* is a multiplier) + 2 cycles (*node10* is an adder) + (5+1) (registers) = 22 cycles. The cycles for the registers are the number of nodes in the branch plus one because the register operating at 5v is one cycle. Here we assume the latency constraint L is equivalent to the maximum cycles among these branches in this example, hence L = 22 cycles. The tolerable cycles are calculated by the subtraction of the minimum required cycles and L. All the tolerable cycles of branches in this example are described in **Table 7**.

Table 7. Tolerance for Each Branch (cycle)

Branch	Nodes	Min Required Time	Max Tolerance
P1	1, 2, 3, 4, 10	22	0
P2	11, 3, 4, 10	17	5
P3	5, 4, 10	14	8
P4	6, 7, 12	14	8
P5	8, 9, 13, 14	17	5

3.5.2.2 Voltage Assignments for Latency Constraint

The next step is to assign the voltages based on the tolerable cycles of each branch. The following rules of our latency-constrained scheduling algorithm have to be known. The multipliers have higher priority than adders during the voltage assignments. Additionally, the position of the nodes has an effect on the voltage assignments. If the node is located on the top, on the bottom, or two like nodes are connected in a branch, the tolerable cycles are reduced by one, this is because the number of registers assigned to different voltages is increased by one when the voltage of registers is

higher than 1.5v. Note that the number of registers is increased by two or more if the voltage assigned on the node is equal to or less than 1.5v. On the other hand, if the node is located in the middle, or two like nodes are *not* connected in a branch, the tolerable cycles are reduced by two, this is because the number of registers assigned to different voltages is increased by two when the voltage of the registers is higher than 1.5v. Note that the number of registers is increased by four or more if the voltage assigned on the node is equal to or less than 1.5v for the new configuration. The peak power minimization is also taken into account by reducing the number of required resources at the scheduling assignment.

Table 8. The Number of Required Cycles for Choosing Resources in Lower Voltages

Voltage (v)	**5**	**3.3**	**2.4**	**2.2**	**1.8**	**1.5**	**1.2**	**1**
$\mathbf{D_{adder}}$	**0**	**0**	**1**	**1**	**2**	**3**	**6**	**16**
$\mathbf{D_{multiplier}}$	**0**	**1**	**2**	**2**	**4**	**7**	**16**	**38**

For P1{1,2,3,4,10}, we assign 5v for each node in the branch because the tolerable cycle is zero in this example. Now, 5v node:{1,2,3,4,10} and the remaining nodes {5,6,7,8,9,11,12,13,14} are stored in Set(R). For P2{11,3,4,10}, the tolerable cycles are 5. *Node3, node4,* and *node10* have been addressed, so these nodes should be ignored. The only node that needs to be processed is *node11*. Since *node11* is located on the top of this branch, the tolerable cycles (slack time) for this operation are 4 (tolerable cycles 5 – 1) as seen in **Table 9**. Because *node11* has a 4 cycle tolerance, the next step is to find the desirable voltage in **Table 8**. *Node11* is assigned to 1.8v, now R={5,6,7,8,9,12,13,14}. For P3{5,4,10}, the tolerable cycles are 8. *Node4* and *node10*

have been addressed, so these nodes can be disregarded. The only node that has to be processed is *node5*. Since *node5* is located on the top of this branch, the available tolerable cycles for this operation are 7 (tolerable cycles 8 – 1). We find the desirable voltage, 1.5v, which is fit to be assigned to *node5*. However, if *node5* is assigned to 1.5v, it will need two more cycles for registers, and this will cause the overall latency to be larger than the latency constraint. Therefore, we assign 1.8v to *node5*, so that R={6,7,8,9,12,13,14}. We should note that branch P3 can tolerate 3 more cycles, which can be used for the consideration on the reduction of the peak power and number of registers.

For P4{6,7,12}, 8 cycles are tolerable. All nodes have to be processed because none have been addressed. The multiplier nodes have higher priority than the adders at the voltage assignment stage. *Node6* and *node7* are directly connected, so the number of tolerable cycles is reduced by one, and now the number of tolerable cycles is 7. The number of tolerable cycles for each multiplier is 7 divided by 2 (2 multipliers), and each multiplier has the slack (available) time for 3.5 cycles. The lowest voltage assignment on the two multipliers is 2.2v. After this assignment, the tolerable cycles are updated to leave 3 cycles (i.e., 7-2*2), which allows us to assign 1.5v on *node12* from **Table 8**. However, an overhead of assigning 1.5v to *node12* will be that the time required for the register will be increased by 1 or more cycles, in which the overall cycles are larger than L. Therefore, we adjust this assignment and choose 1.8v for *node12*, and now R={8,9,13,14}.

Table 9. Tolerance for Multipliers in Each Branch (cycle)

Path	Node	Latency	Tolerance	Minimal Interval
P1	**1,2,3,4,10**	**22**	**0**	**Tolerance=0 (the longest path.)**
P2	**11,3,4,10**	**17**	**5**	**5 – 1(register cycle for *node11*)=4**
P3	**5,4,10**	**14**	**8**	**8-1(register for *node5,4*) =7**
P4	**6,7,12**	**14**	**8**	**8-1(register for *node6,7*) =7**
P5	**8,9,13,14**	**17**	**5**	**5-(1+2)(register for *node9,14*)=2**

For the last branch P5{8,9,13,14}, 5 cycles are tolerable. All the nodes have to be processed because none have yet been addressed. The multipliers have higher priority than the adders at the voltage assignment stage. *Node9* and *node14* are not directly connected, so the tolerable cycles are reduced by two for *node9* and by one for *node14*, and the number of tolerable cycles becomes 2 (i.e., 5-3). The number of tolerable cycles for each multiplier is 2 divided by 2 because there are two multipliers. The lowest voltage assignment for the two multipliers is 3.3v by their 1 cycle slack time. After this assignment, the tolerable cycles of the branch are updated to leave 0 cycles, which leaves *node8* and *node13* assigned at 5v. **Table 10** shows the progress of voltage assignment, and **Table 11** shows the final voltage assignment for the nodes in the DFG.

Table 10. Latency in Different Voltage Assignments for the Branches; L=22 cycles

Path	Node Assignment	Latency
P1	All the nodes are 5v	L= (4*3) + (2*2) + 6 = 22 (=L)
P2	node 3,4,10 = 5v, node 11 = 1.8v	L= 8 + (2*2) + 4 + 6= 22 (=L)
P3	node 4, 10 = 5v, node 5 = 1.5v Node 4, 10 = 5v, Node 5 = 1.8v	L = 11 + 4 + 2 + 7= 24 (> L) L = 8 + 4 + 2 + 5 = 19 (< L)
P4	Node 6, 7 = 2.2v, Node 12 = 5v 7-(2*2) = 3(assign them to Node 12) Node 6, 7 = 2.2v, Node 12 = 1.5v Node 6, 7 = 2.2v, Node 12 =1.8v	L = (6*2) + 2 + 5 = 19 (< L) L = (6*2) + 5 + 7= 24 (> L) L = (6*2) + 5 + 5 = 21 (< L)
P5	Node 8, 13 = 5v, Node 9,14 = 3.3v	L = (5*2) + (2*2) + 8 = 22 (=L)

Table 11. Final Voltage Assignment for the Example

Node	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Property	*	*	+	*	*	*	*	+	*	+	*	+	+	*
Volt (v)	5	5	5	5	1.8	2.2	2.2	5	3.3	5	1.8	1.8	5	3.3

3.6 Latency and Resource Constrained Scheduling

The input to a latency *and* resource constrained scheduling scheme is the DFG, the resource constraint, and the latency constraint. That is, we have the restriction on the number of each type resource such as adders and multipliers. Not only do we have the limit imposed on the resources, we also have the restriction in the latency (operating time).

3.6.1 Iterative Method for Latency and Resource Constrained Scheduling

In this section, we address the problem of scheduling under resource and latency constraints where the resources are operating at multiple voltages. The proposed

heuristic algorithm operates in two steps. At the first step, we consider the resource-constrained scheduling by using the highest available voltage resources to the nodes, which means it will take the minimum amount of time. At the second step, we reduce the power consumption by disabling the resources according to their priority (**Table 5**). The procedure is operated by an iterative method. For each iterative action, the resource with the highest E/D ratio is disabled. These iterations continue until there is a timing violation. We also propose a hybrid method, which combines the first and the second steps at the beginning of scheduling. It reduces the number of iterations and the computing time.

3.6.1.1 First Step Scheduling

In the first step scheduling, we only consider the resource constraint. The nodes in a ready set are prioritized based on their freedom, and the nodes with the lowest freedom are chosen among the ready nodes. The node with lower freedom should be assigned to a higher voltage, and the node with higher freedom should be assigned to a lower voltage. Finally, if the freedom of a node is greater than the delay of the resource to which it is assigned, the node can be re-scheduled to a lower voltage resource if available.

3.6.1.2 Second Step Scheduling

At the end of the first step scheduling, if the timing tolerance is larger than 0 (i.e., there is no timing violation), we can re-schedule the nodes by disabling the resources in the order of priority at this moment (the 5v multiplier will be disabled at first if

possible, then the 5v adder, then the 3.3v multiplier and so on). There will probably be a timing violation after some resources have been disabled. When the conflict occurs at the assignment for the last one or more nodes, we should re-assign the higher voltage to these nodes. For instance, if we have disabled the 5v adder resource, now the maximum voltage resource for the adder should be 3.3v. However, there is a timing violation now, which means there is not enough time for all the adders operating at 3.3v. Therefore, we assign 5v to the last adder to prevent the timing violation, even when the resource of 5v adder is disabled. We will discuss this in an illustrative example.

3.6.1.3 Hybrid Method

If the latency constraint is much larger than the minimum required operating time (i.e., all nodes are assigned to the highest voltage possible), it will take some time to finish the scheduling for large circuits. For example, we have to schedule all nodes from the resource with the highest voltage in the first step scheduling. Then, we disable the resource with the highest priority (i.e., 5v multiplier), and schedule them by the second step scheduling algorithm. We continue to do the procedure again after we disable the second highest priority resource (i.e., 5v adder). It will continue until the timing conflict occurs. To avoid this, we propose the hybrid method by implementing both the first step and the second step at the beginning of scheduling. We compute the average timing tolerance for each node in the critical branch (the branch with the lowest freedom). By the average timing tolerance, we can determine how many resources we can disable at the beginning of scheduling. We show the latency *and* resource constrained scheduling algorithm in **Figure 9**.

First step (resource-constrained scheduling part)

1. *Make or update the ready set, ASAP, and ALAP table for the minimum operating time without constraints.*
2. *Start scheduling by assigning the lowest freedom node in ready sets to the available resource with the highest voltage.*
3. *If the freedom of the scheduled node is larger than the delay of the resource, assign the node to the available lowest voltage resource.*

Second step (power reduction scheduling part)

4. *Disable the resource with the highest priority, and schedule all nodes in the DFG from Step (1).*

If (there is no timing violation){
repeat Step 4}
else {Assign the last one or more nodes to higher voltage resource (i.e., the resource has just been disabled), and complete the low power scheduling.}

Hybrid method

Computing the average timing tolerance for each node in the critical branch (the branch with the lowest freedom)

If (the average timing tolerance>1){
Determine what resources can be disabled. Disable them, and schedules the nodes in resource-constrained scheduling}
else { Schedule the nodes by regular procedures from first step.}

Figure 9. Latency and Resource Constrained Scheduling Algorithm

3.6.2 Example for Latency and Resource Constrained Scheduling

We use the DFG in **Figure 10** to explain our latency *and* resource constrained scheduling algorithm. The resource constraint and the delay for each resource are described in **Table 12**. For the DFG in **Figure 10**, we assume the latency constraint is L=18 (cycles).

Table 12. The Resource Constraint and the Delay for Each Resource

	5v [Mult]	**3.3v** [Mult]	**2.4v [**Mult]	5v [Adder]	3.3v [Adder]	2.4v [Adder]
# of resource	**1**	**1**	**1**	**1**	**1**	**1**
# of delay (cycle)	**4**	**5**	**6**	**2**	**2**	**3**

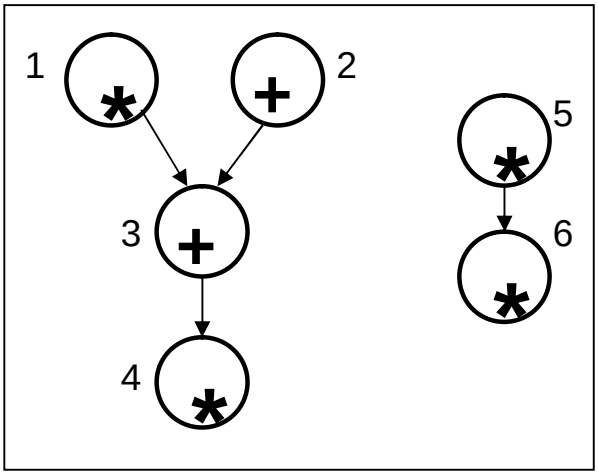


Figure 10. Illustrative Example DFG

Table 13. ASAP and ALAP for illustrative DFG

	ASAP	**ALAP**
Node1	**2**	**2**
Node2	**2**	**4**
Node3	**7**	**7**
Node4	**10**	**10**
Node5	**2**	**5**
Node6	**7**	**10**

First Step (resource-constrained scheduling part):

1. In control cycle 2, the ready sets are M{*node1*, *node5*} and A{*node2*}. For ready set M, because *node1* has lower freedom compared to *node5*, we assign it to 5v multiplier. As for *node5*, 3.3v multiplier is available now; therefore we assign it

to 3.3v multiplier.

2. For *node2*, we should assign it to 5v adder at this moment. However, *node3* will be available at control cycle 7 because the operation of *node1*; therefore we can assign *node2* to 2.4v adder (2.4v adder is the available lowest voltage resource now)
3. In control cycle 7, the ready set is A{*node3*}; we assign it to 5v adder.
4. In control cycle 8, the ready set is M{*node6*}; we assign it to 5v multiplier.
5. In control cycle 10, the ready set is M{*node4*}; we should assign it to 5v multiplier. However, the 5v multiplier resource is not available at this moment so we assign *node4* to 3.3v multiplier.

Cycle	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
5v multiplier	®	***node1***				®		®	***node6***				®			
3.3v multiplier	®	***node5***					®			®	***node4***					®
2.4v multiplier																
5v adder							***node3***		®							
3.3v adder																
2.4v adder	®	***node2***			®											

Figure 11. Scheduled Nodes after First Step

For the first step scheduling, it takes 16 cycles to finish the minimum time resource-constrained scheduling. We denote T_{low} the minimum time to finish the first step scheduling. Because the latency constraint is L=18, which is larger than T_{low}, some resources can be disabled for power reduction in the second step scheduling. In **Figure 11**, the symbol "R" denotes the register for operations. In control cycle 6, there

is no symbol R in front of *node3* because *node1* and *node3* are operating at the same voltage.

Second Step (power reduction scheduling part):

From the result of the first step, the 5v multiplier resource can be disabled because the timing tolerance is larger than 0, and 5v multiplier has the highest priority now. We re-schedule the DFG as follows.

1. In control cycle 2, the ready sets are M{*node1, node5*} and A{*node2*}. For *node1* and *node5*, because *node1* has lower freedom compared to *node5*, we assign *node1* to 3.3v multiplier (the 5v multiplier resources have already been disabled). As for *node5*, now the 2.4v multiplier resource is available, so we assign *node5* to 2.4v.
2. For *node2*, we should assign it 5v at this moment. However, *node3* will be available at control cycle 8; therefore we assign *node2* to 2.4v adder (2.4v adder is the available lowest voltage resource now).
3. In control cycle 8, the ready set is A{*node3*}; we assign *node3* to 5v adder.
4. In control cycle 9, the ready set is M{*node6*}; we assign *node6* to 3.3v multiplier.
5. In control cycle 12, the ready set is M{*node4*}; we should assign *node4* to 2.4v multiplier because 3.3v multiplier is not available now. However, if we assign *node4* to 2.4v, it will violate the latency constraint (*node3* is ending at control cycle 11, and if *node4* starts at control cycle 12, 11+6(2.4v multiplier) +2(register)=19>L=18). Therefore, *node4* should be assigned back to 5v, even the 5v multiplier resource has been disabled at this moment.

Cycle	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
5v multiplier	*Disabled*											*node4*				®
3.3v multiplier	®	*node1*					®		®	*node6*					®	
2.4v multiplier	®	*node5*						®								
5v adder								®	*node3*		®					
3.3v adder																
2.4v adder	®	*node2*			®											

Figure 12. Scheduled Nodes after Second Step

Hybrid Method

For the hybrid method, we can reduce the number of iteration and computing time by determining what resources can be disabled at the beginning of the scheduling. For example, the branch with the lowest freedom in **Figure 10** is branch [*node1-node3-node4*]. We assume all nodes of this branch are operating at 5v in order to determine the minimum operating time, T_{crit}=14 cycles [4*2+2+4(registers)=14]. The latency constraint is 18 cycles, so we know there are additional $(\frac{18-14}{3}) = 1.33$ tolerable cycles available for each node in the branch. Because the average timing tolerance cycle is larger than one, we know some resources can be disabled from **Table 8**. In this case, the 5v multiplier resource can be disabled. We assign *node1* to 3.3v multiplier from the beginning of the resource-constrained scheduling, and we do not have to use all resources as with the previous algorithm. The other scheduling steps of the hybrid method in this case are similar to the second step scheduling in the previous example. Finally, we get the same scheduling result and voltage assignment as described in **Figure 12**.

4. RESULTS AND BENCHMARKS

We present the results obtained by using our algorithms on some high-level synthesis benchmarks [4]. We compute several DFGs for benchmark, including the Differential Equation, AR Filter, 2^{nd} Order Lattice Filter, 5^{th} Order Elliptic Wave Filter, 8 Point FFT, etc. In each case, we tabulate the results for various factors and calculate the percentage of power reduction.

4.1 Latency Constrained Scheduling

The results of latency-constrained scheduling are summarized in **Table 14** and **Figure 13**. For these different DFGs, we get power reduction rate from 12.99% to 41.97% after considering the power consumption of level-shifters and registers. We should note that we use the $T=T_{crit}$ for these DFGs in this simulation (i.e., the latency constraint is equal to the minimum operating time, while all nodes of the critical branch are operating at 5v). If we increase the latency constraint, the power reduction rate will increase enormously. We can see the example from Diffeq in **Table 15**.

Table 14. Power Reduction for Latency Constrained Scheduling

Benchmark	Power at 5V	Power at Multi-Voltage	Power Reduction Rate
Diffeq	**427029.0**	**247807.7**	**41.97 %**
AR Filter	**1053578.0**	**814446.7**	**22.69 %**
EW Filter	**1082687.4**	**900952.9**	**16.79 %**
Lattice Filter	**443386.0**	**385772.9**	**12.99 %**
5^{th} EW	**1044496.6**	**836721.8**	**19.89 %**
8-Point FFT	**1863277.6**	**1239272.6**	**33.49 %**
Fast DCT	**1530780.6**	**924541.9**	**39.60 %**

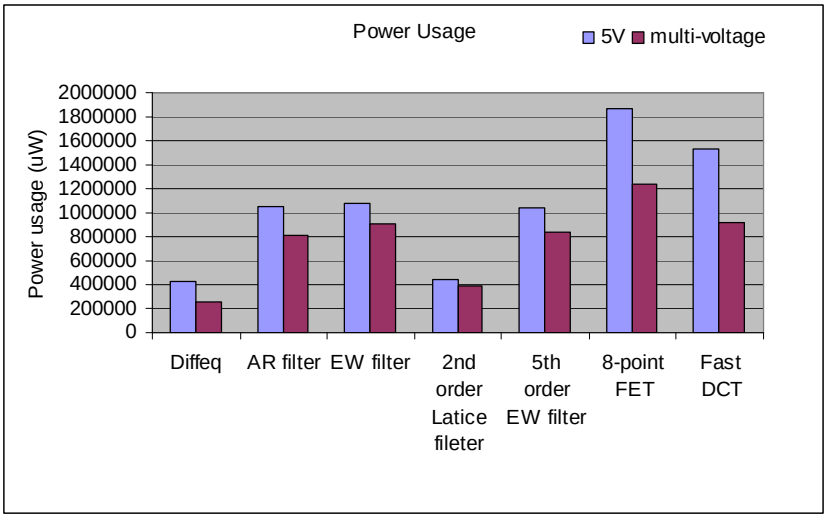


Figure 13. Power Reduction for Latency Constrained Scheduling

Table 15. Power Reduction for Diffeq in Different Latency Constraints

Latency Constraint	Power (uW)	Power Reduction Rate
17	**247,807**	**41.97 %**
20	**107,884**	**74.74 %**
30	**39,151**	**90.83 %**
40	**16,632**	**96.11 %**

4.2 Latency and Resource Constrained Scheduling

We present the results for the latency *and* resource constrained scheduling in this section. The latency constraint is equal to T_{low}, $1.2T_{crit}$, $1.3T_{crit}$, and $1.4T_{crit}$, where T_{low} represents the minimum operating time for DFG in resource-constrained scheduling, and T_{crit} represents the time that all nodes in the critical branch are operating at 5v. The resource constraints of the multiplier and adder resources are operating at 5v, 3.3v, 2.4v, 2.2v, and 1.8v for each. By using multiple voltages and multiple cycles for the operations, we can see the power reduction rate from 20.19~49.39% after the first step

scheduling. Some of our simulations in AR filter and 8 point FFT can not be scheduled because their latency constraint is less than T_{low}.

4.2.1 Differential Equation (Diffeq)

The DFG of Diffeq consists of six multiplications, three subtractions, one addition and one comparison. In this simulation, we use the power and delay characteristics of an adder for the subtraction and comparison because of their similarity. **Table 16** shows the power reduction rate, peak power, etc. We should note that T_{crit} of Diffeq is equal to 17 cycles, and the power consumption of Diffeq is 427,029 uW while all components are operating at 5v.

Table 16. Diffeq for Latency and Resource Constrained Scheduling

Latency Constraint	Power Reduction (%)	Total Power (uW)	Peak Power (uW)	FU Power (uW)	Register Power (uW)	Latency (# of cycles)
T_{low}	31.94%	290,598	61,796	133,865	156,733	19
$1.2T_{crit}$	46.83%	227,048	37,030	91,684	135,364	20
$1.3T_{crit}$	63.90%	154,120	36,368	64,685	89,435	22
$1.4T_{crit}$	75.57%	104,300	18,883	47,802	56,498	23

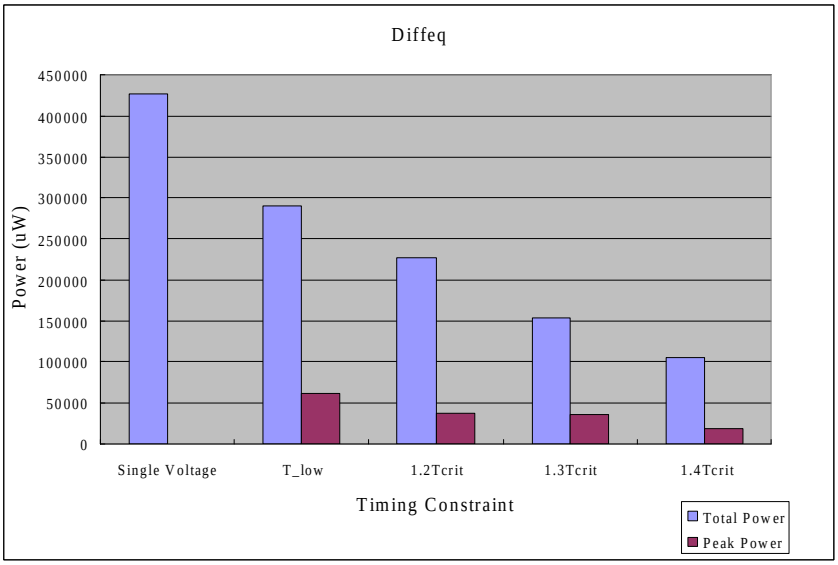


Figure 14. Power and Peak Power of Diffeq in Latency and Resource Constrained Scheduling

4.2.2 AR Filter

T_{crit} of AR filter is equal to 31 cycles, and the power consumption is 1,053,578 uW while all nodes are operating at 5v. The results of AR filter are listed in **Table 17**. We should note that some results are unobtainable because their latency is less than $T_{low.}$ (i.e., $1.2T_{crit}$= 37; $1.3T_{crit}$=40, they are both less than T_{low}=42).

Table 17. AR Filter for Latency and Resource Constrained Scheduling

Latency Constraint	Power Reduction (%)	Total Power (uW)	Peak Power (uW)	FU Power (uW)	Register Power (uW)	Latency (# of cycles)
T_{low}	48.48%	542,749	60,708	277,747	265,002	42
$1.2T_{crit}$	n/a	n/a	n/a	n/a	n/a	37
$1.3T_{crit}$	n/a	n/a	n/a	n/a	n/a	40
$1.4T_{crit}$	56.38%	459,595	57,532	229,486	230,109	43

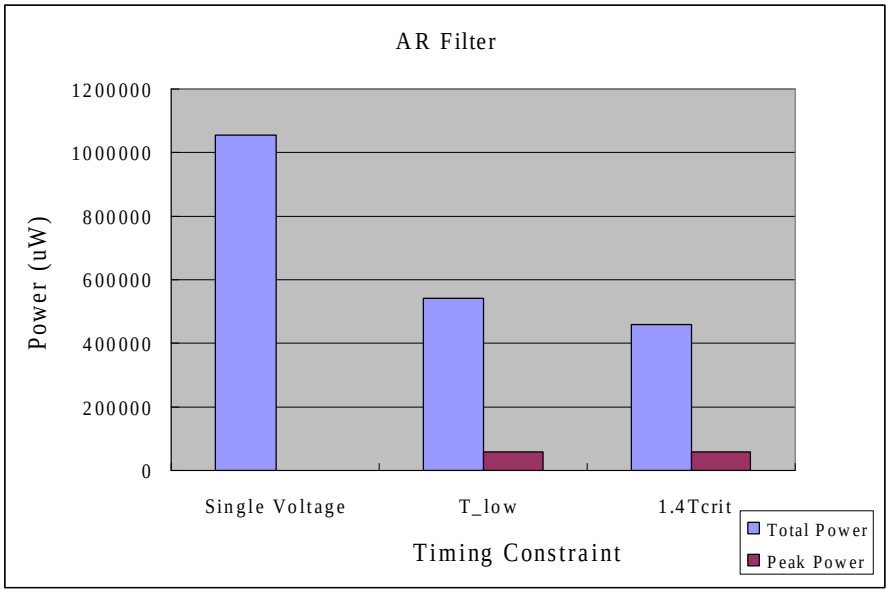


Figure 15. Power and Peak Power of AR Filter in Latency and Resource Constrained Scheduling

4.2.3 2nd Order Lattice Filter

The 2nd Order Lattice filter is usually used in data sampling. T_{crit} of 2nd Order Lattice Filter is equal to 34 cycles, and the power consumption is 443,386 uW while all nodes are operating at 5v. We can observe that when the latency constraint increases, not only does the total power decrease, but the peak power decreases rapidly also.

Table 18. 2nd Order Lattice Filter for Latency and Resource Constrained Scheduling

Latency Constraint	Power Reduction (%)	Total Power (uW)	Peak Power (uW)	FU Power (uW)	Register Power (uW)	Latency (# of cycles)
T_{low}	20.19%	353,860	40,565	164,408	189,452	38
$1.2T_{crit}$	52.58%	210,222	40,565	131,814	78,408	40
$1.3T_{crit}$	71.40%	126,785	15,137	53,619	73,166	44
$1.4T_{crit}$	79.21%	92,178	11,896	44,868	47,310	47

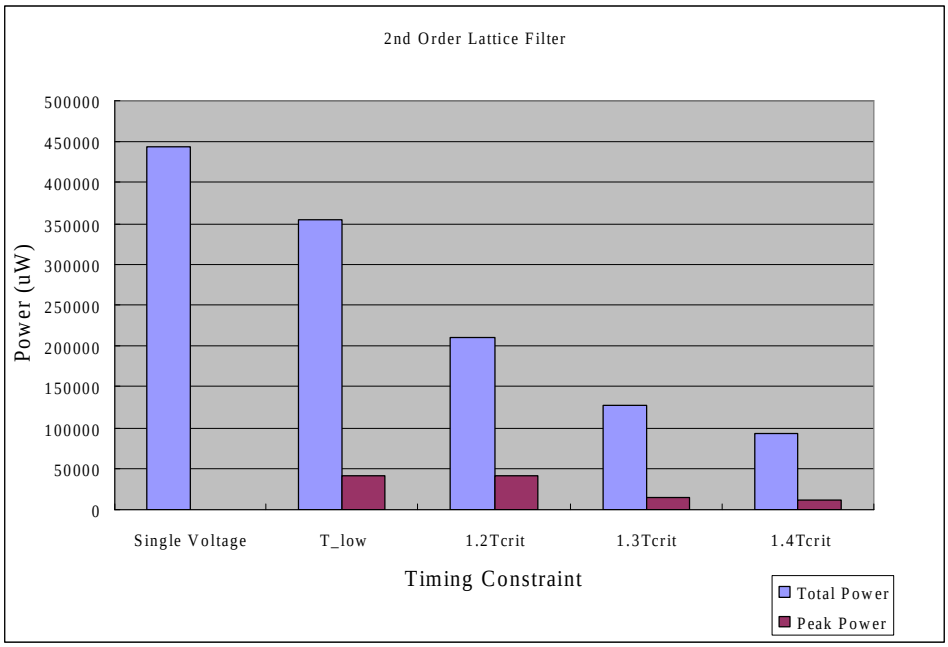


Figure 16. Power and Peak Power of 2nd Order Lattice Filter in Latency and Resource Constrained Scheduling

4.2.4 5th Order Elliptic Wave Filter

The 5th Order Elliptic Wave Filter is also widely used in data sampling. In this simulation, T_{crit} of 5th Order Elliptic Wave Filter is equal to 47 cycles, and the power consumption is 1,044,496 uW while all nodes are operating at 5v.

Table 19. 5th Order EW Filter for Latency and Resource Constrained Scheduling

Latency Constraint	Power Reduction (%)	Total Power (uW)	Peak Power (uW)	FU Power (uW)	Register Power (uW)	Latency (# of cycles)
T_{low}	44.92%	575,302	52,461	249,750	325,552	50
$1.2T_{crit}$	57.80%	440,804	31,751	174,096	266,708	56
$1.3T_{crit}$	71.02%	302,630	27,588	127,082	175,548	61
$1.4T_{crit}$	80.64%	202,250	27,033	95,307	106,943	65

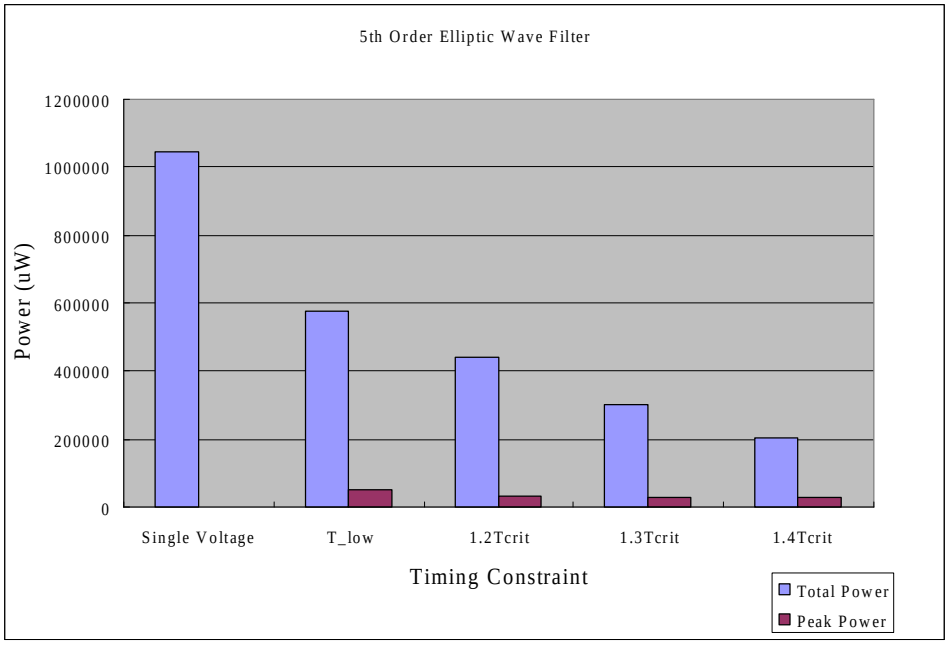


Figure 17. Power and Peak Power of 5th Order EW Filter in Latency and Resource Constrained Scheduling

4.2.5 8 Point Fast Fourier Transform (FFT)

Fast Fourier Transform (FFT) is widely used for digital audio. The T_{crit} of 8 Point Fast Fourier Transform is equal to 34 cycles, and the power consumption is 1,057,732 uW while all nodes are operating at 5v. Because their latency constraints ($1.2T_{crit}$, $1.3T_{crit}$, and $1.4T_{crit}$) are less than T_{low}, none of them can be scheduled.

Table 20. 8 Point FFT for Latency and Resource Constrained Scheduling

Latency Constraint	Power Reduction (%)	Total Power (uW)	Peak Power (uW)	FU Power (uW)	Register Power (uW)	Latency (# of cycles)
T_{low}	49.39%	942,857	69,783	378,089	564,768	50
$1.2T_{crit}$	n/a	n/a	n/a	n/a	n/a	40
$1.3T_{crit}$	n/a	n/a	n/a	n/a	n/a	44
$1.4T_{crit}$	n/a	n/a	n/a	n/a	n/a	47

4.3 Power Consumption by Registers in Multiple Voltages Scheduling

In this section, we show the ratio of power consumption by registers in latency-constrained scheduling, and the latency *and* resource constrained scheduling.

4.3.1 Power Consumption by Registers in Latency Constrained Scheduling

We show the ratio of power consumption by registers in latency-constrained scheduling in **Figure 18**. We can find that registers consume more than half of the total power in latency-constrained scheduling, whether they are operating at a single voltage (5v) or multiple voltages.

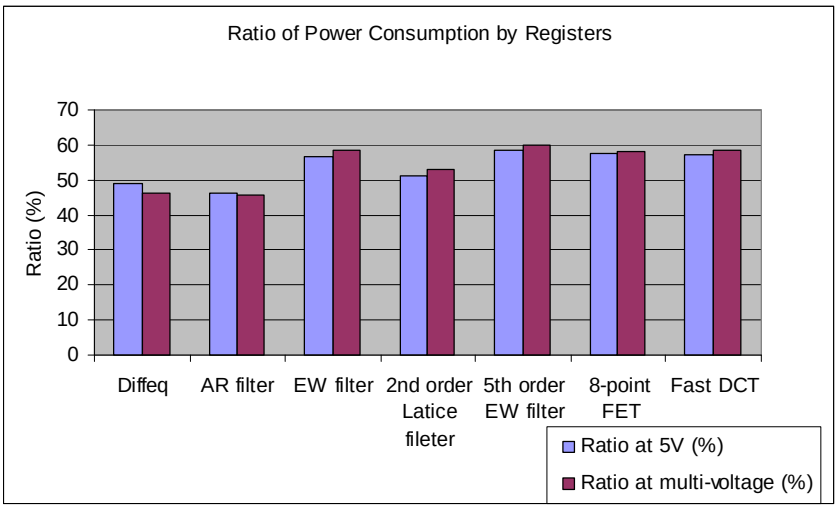


Figure 18. The Ratio of Power Consumption by Registers in Latency Constrained Scheduling

4.3.2 Power Consumption by Registers in Latency and Resource Constrained Scheduling

In **Table 21**, we tabulate the results for power consumption by registers in latency *and*

resource constrained scheduling. We find that registers consume over 50% total power in most benchmarks of our simulation. The ratio of power consumption by registers in latency *and* resource constrained scheduling is similar to that in latency-constrained scheduling. Both of the results imply that it is important to consider the power consumption of registers for low power design because they enormously affect the total power consumption. In **Figure 19**, we show the relationship between latency constraint and the ratio of power consumption by registers for three different DFGs. From our simulation results, the ratio is not correlated closely with the latency constraint.

Table 21. The Ratio of Power Consumption by Registers in Latency and Resource Constrained Scheduling

		Diffeq		AR Filter		2nd lattice Filter		5th Ew Filter		8 Point FFT	
		Power	Reg %	Power	Reg %	Power	Reg %	Power	Reg %	Power	Reg %
T_{low}	P_{total}	290,598	53.93%	542,749	48.83%	353,860	53.54%	575,302	56.59%	942,857	59.90%
	P_{reg}	156,733		265,002		189,452		325,552		564,768	
$1.2T_{crit}$	P_{total}	227,048	59.62%	n/a	n/a	210,222	37.30%	440,804	60.50%	n/a	n/a
	P_{reg}	135,364		n/a		78,408		266,708		n/a	
$1.3T_{crit}$	P_{total}	154,120	58.03%	n/a	n/a	126,785	57.71%	302,630	58.01%	n/a	n/a
	P_{reg}	89,435		n/a		73,166		175,548		n/a	
$1.4T_{crit}$	P_{total}	104,300	54.17%	459,595	50.07%	92,178	51.32%	202,250	52.88%	n/a	n/a
	P_{reg}	56,498		230,109		47,310		106,943		n/a	

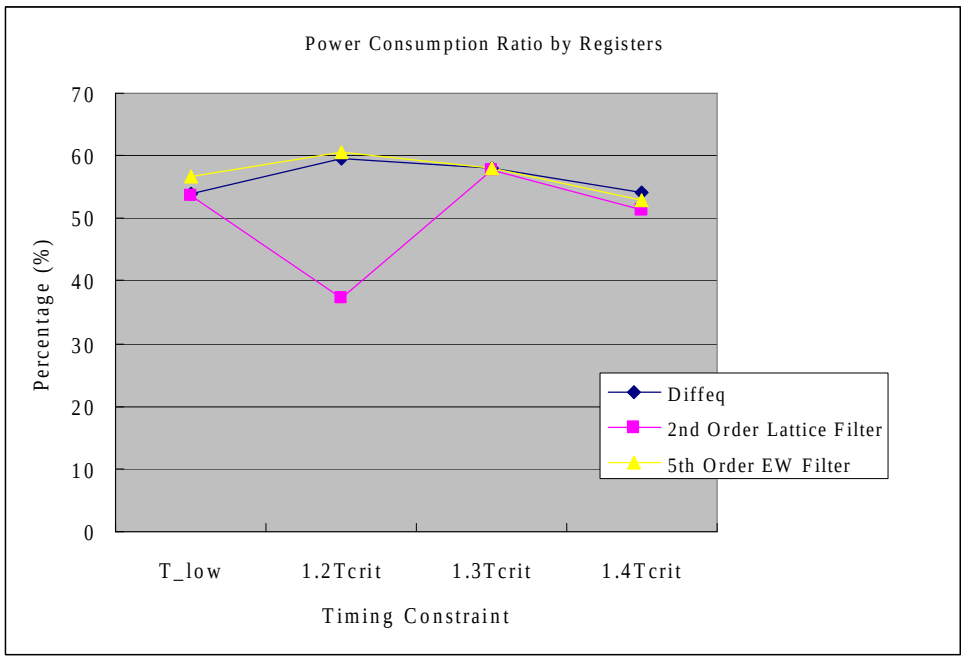


Figure 19. Ratio of Power Consumption by Registers in Latency and Resource Constrained Scheduling

5. CONCLUSION

We present the latency-constrained scheduling and latency *and* resource constrained scheduling when the resources and registers are operating at multiple voltages and multi-cycles. Our algorithms reduce the overall power consumption of the components, including the resources, registers, and level shifters. Using the schemes discussed, we assign as many nodes as possible to operate at lower voltages without violating the latency or resource constraints. We reduce the average power and peak power for these constrained schedulings. We also consider the power consumption of registers. This consideration provides more practical methods to achieve the low power design because registers enormously affect the power consumption in the various voltage supplies.

In addition, our heuristic algorithms only have polynomial time complexity compared to ILP scheduling algorithms, which have exponential time complexity in the worst case. Some further work will be necessary in the future. For example, the number of register resource constraints has not yet been considered.

BIBLIOGRAPHY

[1] A. Chandrakasan, M. Potkonjak, R. Mehra, J. Rabaey, and Brodersen, "Optimizing Power Using Transformations," IEEE Trans. Computer-Aided Design, vol. 14, Jan. 1995, pp. 12-31.

[2] A. Dasgupta and R. Karri, "Simultaneous Scheduling and Binding for Low Power Minimization during Microarchitecture Synthesis," in Proc. Int. Symp. Low-Power Design, Apr. 1995, pp. 69-74.

[3] A. Raghunathan and N. K. Jha, "Behavioral Synthesis for Low Power," in Proc. IEEE Design Automation Conf., 1995.

[4] C. Lee, "A Resource-Constrained Scheduling Scheme that Considers Resources Operating at Multiple Voltages and Register Assignment," Thesis on School of Electrical Engineering and Computer Science, Oregon State University, 2003.

[5] E. Musoll and J. Cortadella, "High-level Synthesis Techniques for Reducing the Activity of Functional Units," in Proc. Int. Symp. Low Power Design, 1995, pp. 99-104

[6] "Scheduling with Multiple Voltages," Integration: The VLSI J., vol. 23, 1997, pp. 37-59.

[7] S. Raje and M. Sarrafzaden, "Variable Voltage Scheduling", Intl. Sym on Lowe Power Design, 1995.

[8] A Manzak and C. Chakrabarti, "A Low Power Scheduling scheme With Resources Operating at Multiple Voltages", IEEE Transactions on Very Large Scale Integration Systems, Vol 10., Issue 1, pp. 6-14, 2002

[9] W.-T. Shiue and C. Chakrabarti, "Low-Power Scheduling with Resources Operating at Multiple Voltages," IEEE International Symposium on Circuits and Systems, vol. 2, no. 6, Jun. 1998, pp. 437-440.

[10] J.M. Chang and M. Pedram, "Register Allocation and Binding for Low Power," Design Automation Conf. 1995

[11] J.M. Chang and M. Pedram, "Module Assignment for Low Power", EURO-Design Automation Conf, 1996.

[12] C.G. Lyuh, and K, Taewhan, "High-level Synthesis for Low-Power Based in Network Flow Method," Trans on VLSI System, 2003.

[13] A. Chandrakasan and R. Brodersen, "Minimizing power consumption in digital CMOS circuits," in Proc. IEEE, vol.83, Apr. 1995, pp. 498-523.

[14] Nestor J. A., Thomas D. E., "Behavioral Synthesis with Interfaces", ICCAD, Proceedings on the International Conference on Computer Aided Design, pp. 112-115, 1986

[15] P. Paulin and J. Knight, "Force-Directed Scheduling for the Behavioral Synthesis of ASICs", IEEE Transactions on CAD/ICAS, Vol. CAD-8, No. 6, Page(s): 661 - 679, July 1989

[16] P. Kollig, B.M. Al-Hashimi and K.M. Abbott, "Efficient Scheduling of Behavioral Descriptions in High-level Synthesis", IEE Proceedings Computer Digital Techniques, Vol. 144, No. 2, March 1997

[17] W. T. Shiue and C. Chakrabarti, "Low Power Scheduling with Resources Operating at Multiple Voltages", IEEE Transactions on Circuits and Systems II. Analog and Digital Signal Processing, 2000

[18] M. Kamble, K. Ghose, "Analytical Energy Dissipation Model for Low-power Caches," ACM/IEEE International Symposium on Low Power Design, April 1995, Pages:63-68.

[19] M.C. Johnson and K. Roy, "Datapath Scheduling with Multiple Supply Voltages and Level Converters", ACM Transactions on Design Automation of Electronics Systems, 1997

[20] J. M. Chang and M. Pedram, “Energy Minimization Using Multiple Supply Voltages”, IEEE Transactions on Very Large Scale Integration Systems, Volume: 5 , Issue: 4 , Dec. 1997 Pages:436 – 443

[21] Wong, Jennifer, Megerian, Seapahn, and Potkonjak, “Forwarding-Looking Objective Functions: Concept & Application in High Level Synpaper. DAC 2002, June 10-14, 2002.

[22] D. D. Gajski and L. Ramachandran, “Introduction to High-Level Synthesis”, IEEE Design & Test of Computers, Vol. 11, Issue: 4, pp. 44-54, Winter 1994