受理
06.2.-3
電気・電子情報・
電子工学専攻

平成１７年度修士論文

電　子　1243

Design of Microcontroller with Completion Detection Capability by using Dual-Rail Domino Circuit

２線式ドミノ回路による終了検出型マイクロコントローラの設計

2006年2月3日提出

指導教官

浅田邦博 教授

東京大学大学院　工学系研究科　電子工学専攻

学籍番号 46419

氏名 ディア キン フイ (DIA KIN HOOI)

Abstract

A 8-bit non-pipeline microcontroller equipped with completion detection capability is designed by using dual-rail domino circuit. The microcontroller is designed based on the instruction set of Z80 microcontroller. It is implemented with Rohm 0.35μm CMOS technology with chip size of 4.9$\times$4.9mm^2, and the measurement results reveal that it could functionally works correctly regardless of the variations due to the instruction dependency, data dependency, and the inter-chip variability. The microcontroller achieves an average speed performance of 23.3ns for evaluation time, and it needs 2.2ns for precharge time at nominal supply voltage of 3.3V. It also exhibits an automatic performance adaptation to the physical properties such as power supply voltage.

Along with these, this paper presents a new footless dual-rail domino circuit that efficiently combines a footless dynamic circuit technique with a robust self-timed precharge scheme for high performance VLSI circuit design. Besides, the proposed circuit achieves a whole footless dual-rail domino circuit with the use of the proposed separator. A 20-stage NAND chains are implemented both in 0.15μm SOI CMOS technology and 90nm bulk CMOS technology for performance evaluation. Measurement results reveal that the proposed circuit achieves speed improvement over the circuit implemented with the conventional static CMOS, CPL, dynamic DCVSL, D^{4}L, and DR-domino.

List of Content

List of Figure

List of Table

Chapter 1

Introduction

1.1 Background

Recent advances in VLSI have continued to shrink device geometries at a steady rate in accordance with Moore's Law. However, this advancement has also been accompanied by increasing variations in the performance of fabricated circuits as shown in Fig. 1.1 [1]. As we scale down technology to the sub-100nm feature size, both intrinsic device variations and process lithography control issues are increasing the statistical variability of each gate in a circuit [2]. This delay variation causes the expected delay for a circuit, which is the expected value of the maximum of all the path delays, to grow larger as the wall of critical paths gets taller. The increasing performance variation results in even higher timing margin cost issues associated with global, periodic, and common clock that is the temporal basis for synchronous circuits. In synchronous circuits, a significant percentage of the clock period is dedicated to margin during which there is no useful logical computation. The speed at which integrated circuit operate varies with the circuit fabrication process, and fluctuations in operating temperature and supply voltage. In order to achieve a reasonable shield against these variables, the clock period is extended by a certain margin. The importance of accurately estimating the margin is directly related to a company's overall revenue. An overestimation increases the design complexity, possibly leading to an increase in design time complexity, an increase in die size, rejection of otherwise good designs and even missed market windows. Conversely, an underestimation can compromise the product's performance and overall yield, as well as increase the silicon debug time [3]. In current practice, these margins are often 100% or more in high-speed circuits [4]. We are expecting, as the CMOS scaling trends going on, the timing margin of the clock is becoming a high performance design obstacles for synchronous chips in the future gigascale

integration circuits.

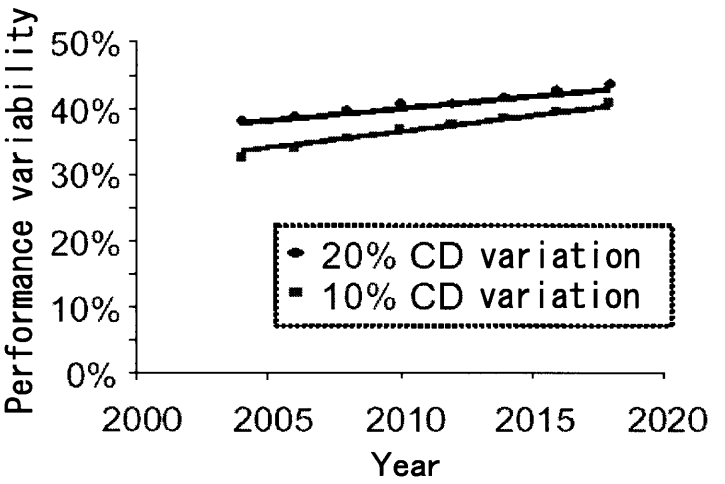


Fig. 1.1: Performance variability [1].

Therefore, one must find out an exit for new paradigm by using other circuit design. The obvious way to avoid these problems is to throw out the clock. In this research, hereby, a design scheme which takes advantage of dual-rail domino circuit is used to design a micro-controller with completion detection capability. With the completion detection capability, the microcontroller could eliminate the necessity of the use of globally distributed clock; hence offers the following benefits:

- Avoidance of clock skew - Synchronous design methodologies use a global clock signal to regulate operation, with all state changes in the circuit occurring when the clock signal changes level. As the feature size decrease and the integration levels increase, the physical delays along wires in a chip are becoming more significant, causing different parts of the circuit to observe the same signal transition at different times. If the affected signal is a clock, then this time difference, known as clock skew, limits the maximum frequency of operation of a synchronous circuit [5]. Through careful engineering of the clock distribution network, it is possible to mitigate the clock skew problem, but solutions such as balanced clock trees [6] are expensive in silicon area and power consumption and require extensive delay modeling and simulation. The absence of a global clock in the proposed microcontroller design avoids the problems of clock skew and the complexity of the clock distribution network.

- Average- instead of worst-case performance - The fixed clock period of synchronous circuits is chosen as a result of worst-case timing analysis. It is not adaptive and

therefore does not take advantage of average- or even the best-case computational situations. In contrast, the proposed microcontroller senses when a computation has completed, allowing it to exhibit average-case performance. Therefore, there is an opportunity to achieve increased performance where the worst-case situation is rare, and when the difference between the worst-case and average-case latencies is significant. Arithmetic circuits provide good examples. Arithmetic circuit performances are typically dominated by the propagation delay of carry or borrow signals. The worst-case propagation situation rarely occurs, yet synchronous arithmetic circuits must be clocked in a manner that accommodates this rare worst-case condition.

- Automatic adaptation to physical properties - The delay through a circuit can change with process-voltage-temperature variations in fabrication. Also, adaptive power supply voltage can be lowered when speed is not required. Since power depends quadratically on voltage, the combination of slow-down and adaptive supply yields a cubic power saving with the reduction of speed. In addition, leakage power, which becomes more significant in newer process technology, can also be managed by reducing the supply voltage [7]. Synchronous circuits must assume that the worst possible combination of factors is present and clock the system accordingly. It is easier to vary supply voltage in the proposed microcontroller since there is no need to coordinate simultaneous variation of the clock frequency.

- Low power - Power consumption is important in many embedded systems where battery life is at a premium. In larger, higher performance, systems power consumption affects the packaging cost of the system due to the need both to supply the energy onto the chip and to remove the heat generated. Clockless design can reduce power consumption by avoiding two of the problems of synchronous design: (i) All parts of a synchronous designs are clocked, even if they perform no useful function; (ii) the clock line itself is a heavy load, requiring large drivers, and a significant amount of power is wasted just in driving the clock line. There are synchronous solutions to these problems, such as clock-gating, but the solutions are complex [8]. In the proposed microcontroller design, it often requires more transitions on the computation path than synchronous circuits due to the use of dual-rail domino circuit. However, low power design techniques targeting dual-rail domino circuit have been proposed and can be implemented with ease, resulting in transitions only in areas involved in the current computation.

- Improved electro-magnetic compatibility (EMC) - The global synchronization of a clocked design caused much of the activity in the circuit to occur at the same instant time. This concentrates the radiated energy emissions of the circuit at the harmonic frequencies of the clock. Synchronous design approaches to spreading this radiated energy across the spectrum, such as varying the clock period, are complex to implement and affect the performance of the systems since the clock period can only be made longer (not shorter) than the minimum for safe operation of the circuit. In contrast, the activities in the proposed microcontroller are uncorrelated, producing broadband distributed interference spread across the entire spectrum. This can be significant advantage in systems which use radio communication where interference must be minimized.

To design the proposed microcontroller, we are looking for one complete CAD system suitable for all design tasks including specification, design, simulation, validation, verification, debugging and synthesis. Naturally, since the CAD system is the tool rather the research, and since such a grand CAD system requires immense resources to develop and maintain, we have turned to the domain of commercial CAD products in our quest. Unfortunately, no large scale commercial CAD systems are available for such design. Therefore, a design methodology for the design of microcontroller with completion detection capability based on the commercial synchronous CAD system is proposed. A design discipline is developed by which any explicit dependence on the clock is carefully avoided. The circuit synthesized by the tool into a synchronous structure, but subsequently it is converted into dual-rail domino circuits that could yield the completion detection capability.

1.2 Dual-Rail Domino Circuit

1.2.1 Dual-Rail Logic

In this study, a design scheme which takes advantages of dual-rail logic is used to eliminate the necessity of global clock. In dual-rail circuits, 1-bit signal is encoded into 2-bit status, i.e. status 01 and status 10, to express the valid state of signal 0 and signal 1. Status 00 is used as a spacer to express that no valid data is being transferred on the data line. The dual-rail signaling encoding is summarized in Table 1.1 The built-in redundancy in dual-rail logic can be effectively exploited for completion detection and error detection, as shown in Fig. 1.2. Also, in such microcontroller design, hazards must be removed from the circuit, or not introduced in the first place, to avoid incorrect results. Dual-rail logic avoids

the problem as it guarantees hazard-free operation [9]. By using dual-rail communicating, the proposed microcontroller is delay-insensitive in the sense that its functionality is unaffected by any wire or buffer delays that are inserted. This allows it to exhibit average- instead of worst-case performance as in synchronous circuits.

$d_x^a d_x^b$	Meaning
00	Initial Status
01	Data: 0
10	Data: 1
11	Illegal Status

Table 1.1: Dual-rail signaling encoding.

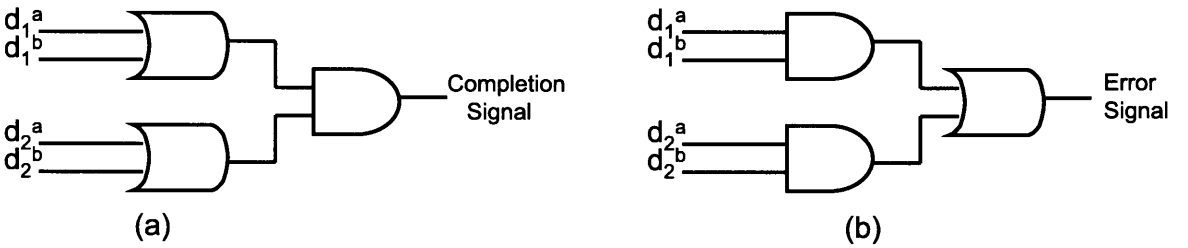


Fig. 1.2: (a) Completion detection circuit for 2-bit signal (b) error detection circuit for 2-bit signal.

1.2.2 DCVSL Cell Library

The standard cell library is specially designed in DCVSL (Differential Cascode Voltage Switch Logic) [10] circuit style which suited the dual-rail logic principle. DCVSL is a dynamic logic with precharge, and can generate a completion signal from its differential property.

DCVSL circuit have two behavior that lead to it's usage in dual-rail logic. (i) During the precharge phase, input data has no effect on the output values. That means no matter what value of data appears at the input lines, the outputs will be precharged to low. (ii) During the evaluation phase, the DCVSL gate begins evaluation as soon as the input data are valid. When the input data are not valid, the DCVSL gate remains at the precharge state. These two behavior lead to latch-free and simple handshake protocol. Two basic DCVSL circuit designs, NAND gate and NOR gate, are shown in Fig. 1.3. DCVSL circuit cannot operate with traditional flip-flop because the outputs of flip-flop must remain its previous state or

to keep 00 state during one instruction execution. To solve this problem, dedicated flip-flop are designed, which is shown in Fig. 1.4.

The penalty of using DCVSL logic is the increased power dissipation compared to static CMOS as well as dynamic circuit techniques which use single-ended logic gates. In order to minimize the power consumption, the microcontroller is designed so that only the requested blocks can be activated and consumes power. Moreover, there is no need for clock distribution network, thereby, further contributing in saving power.

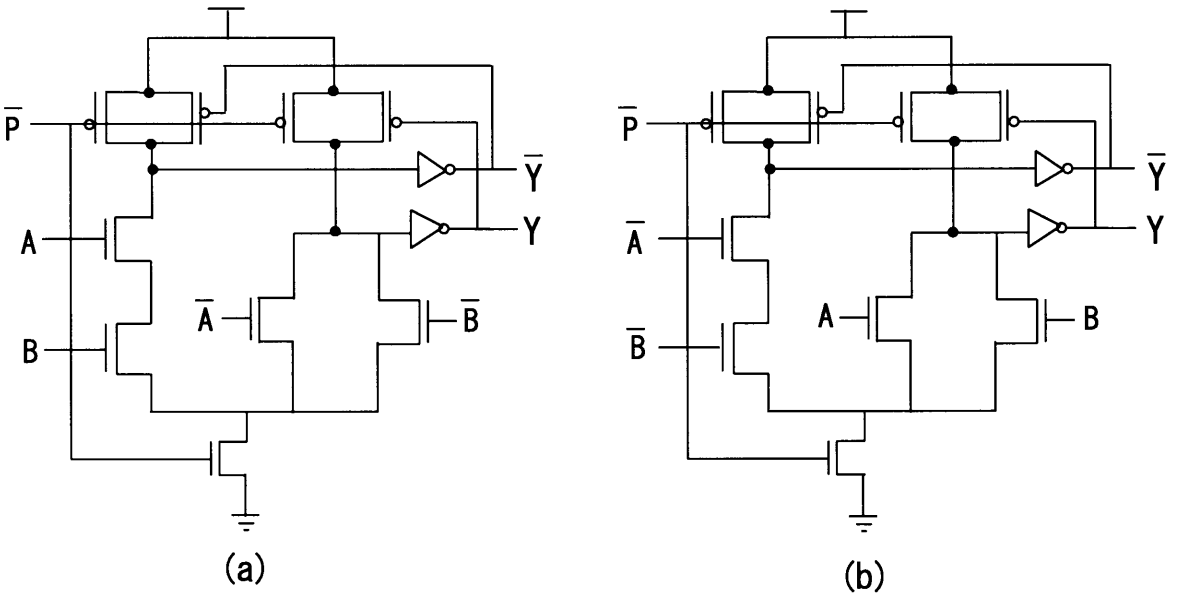


Fig. 1.3: DCVSL: (a) NAND gate (b) NOR gate.

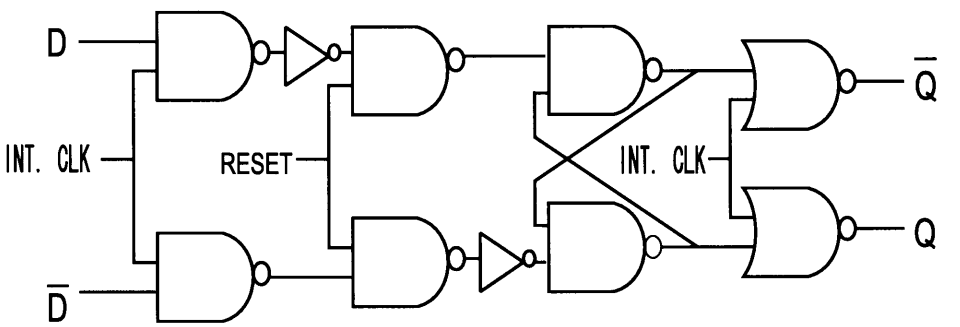


Fig. 1.4: Flip-flop design for DCVSL circuit system.

1.2.3 Completion Detection Circuit

Since the microcontroller does not have a global clock to act as a timing reference, it must generate completion signal by itself. In order to do so, its data paths are implemented

with DCVSL. DCVSL logic uses redundant logic encoding that can provide completion information as part of the logic signals. During the evaluation phase, the DCVSL gate begins evaluation as soon as the input data are valid, and the outputs of the DCVSL gate become 01 or 10. When the input data are not valid, the DCVSL gate remains at the precharge state with outputs remain as 00. Thus, by adding a static CMOS OR logic at the outputs of the DCVSL logic can act to sense when a computation is completed. This approach eliminates the need for a delay matching circuit; hence average- instead of worst-case performance is achievable.

Since the microcontroller is designed based on the instruction set of Z80 microcontroller by adopting non-pipeline architecture, completion detection is needed only at the end parts of the combination circuits. The completion of all the instructions of Z80 occurs only when one or more of the registers, memory and I/0 are written. Thus, by placing completion detection circuits at the inputs of the data paths to the registers, memory and I/O, and apply a logical AND to the outputs of the completion detection circuits can act to detect when an instruction is completed. To which completion detection circuits should function differed from each of the instructions, hence control signal generated at instruction decoder are needed to assign to which completion detection circuits are to be checked for functionality. The completion detection circuit is shown in Fig. 1.5.

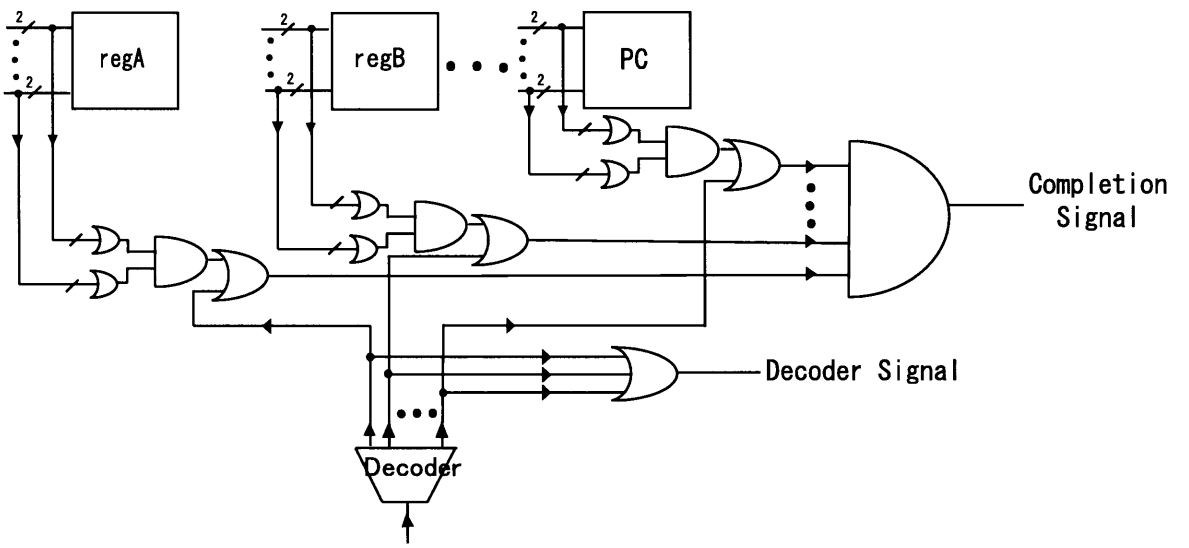


Fig. 1.5: Completion detection circuit.

1.2.4 Error Detection Circuit

One of the primary motivations for this study is the need for low power in many fault-tolerant systems. The current way of providing high-coverage error detection with conventional synchronous circuits is to run two in lock-step and compare them. Its power is thus doubled in a duplex configuration. Completion detection style designs are expected to require considerably less power than two synchronous chips.

DCVSL signaling redundancy can be used to provide error detection. Consider a DCVSL circuit in Fig. 1.3. The circuit is designed so that a single transistor fault or error will only affect one of the two sides (true or complement), and thus will only affect one output. The fault will cause an error output pair of 00 or an illegal output pair of 11. Failure to precharge will eventually result in a 11 output while failure to pull down through the common pull down transistor will results in 00 outputs. Precharging while evaluating or pulling down while precharging cause degraded signals that will either result in the correct output, 11 or 00 depending upon transistor sizing. Given that faults in single-level DCVSL circuits produce 11 or 00 outputs, it is easy to show that a multi-level network behaves similarly. When a good circuit receives incorrect signal input pairs from a faulty circuit (i.e. 00 or 11), it will produces either the correct output or in an output of 00 or 11. A 00 input can only prevent a side from being pulled down, producing an error output of 00. Similarly, an 11 on an input pair can only produce an error output of 11. These error will be masked or propagated through multiple levels of DCVSL circuits and be at the output [11].

Error detection circuit that functioning to detect the illegal output pair of 11 is designed, as shown in Fig. 1.6. If one of the input circuits fails to complete and generates a 00 signal pair, then a completion signal would never be generated due to no completion, and the error can be detected by using a timer.

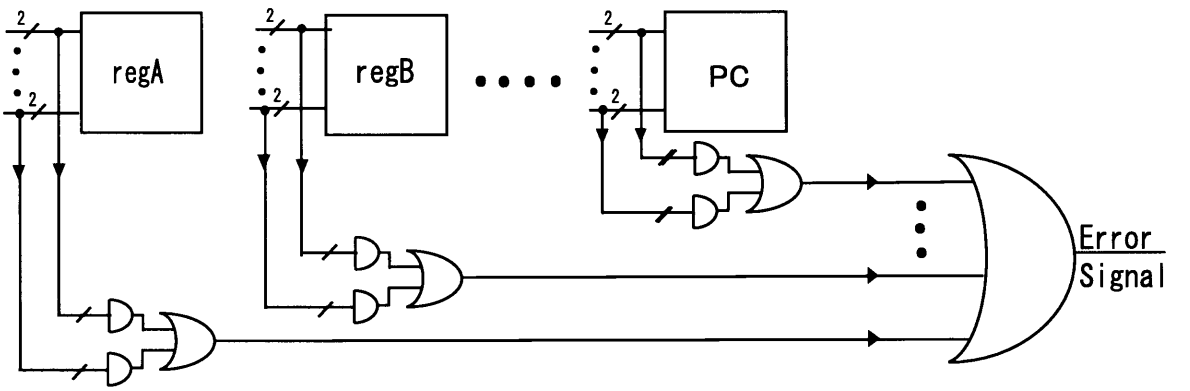


Fig. 1.6: Error detection circuit.

Chapter 2

Architecture

2.1 Non-pipeline Architecture

One of the fundamental decisions to be made in the design of a microcontroller is the choice of the structure of the pipeline. The pipeline is a design technique that allows an increased data flow throughput to pass through a combinational block. It consists of dividing the combinatory logic blocks into several stages. For each stage, a task performed which yields a partial result. As a result, the performance gain is an increase in the throughput of the results, which is related to balanced divisions of the pipeline stages. Pipeline registers are inserted between stages to ensure data flow synchronization. For some time, the VLSI community has been dramatically improving the performance of synchronous designs thanks to the use of global clocks to synchronize the switching activity between pipeline stages.

Recently, ample evidence has emerged showing that this global synchronous approach has started to encounter major difficulties [12]. Among these difficulties are (i) circuitry has to operate constantly at a frequency determined by the longest path in the combinational logic even it is not necessary to do so, (ii) registers are triggered by the clock at each cycle, and subsequently dissipates energy whether the state has changed or not, (iii) numerous clocked blocks in a large design must be synchronized in order to insure correct operation, a requirement that is increasingly difficult to satisfy when faced with the growing dominance of interconnect delays, and (iv) the constant switching of the clock can cause surges in power-supply noise and electromagnetic emissions.

In view of the imminent difficulties caused by the pipeline architecture; hence non-pipeline architecture is adopted for the proposed microcontroller design (Fig. 2.1). Without the use of clock for synchronization between the pipeline registers helps to avoid the

problems related to clock. In addition, dual-rail domino circuit is utilized so that signal transitions are limited to one within one cycle of instruction execution. This should efficiently decrease the switching noise. As the result, the waveform of the internal signals is expected to be relatively stable in the microcontroller.

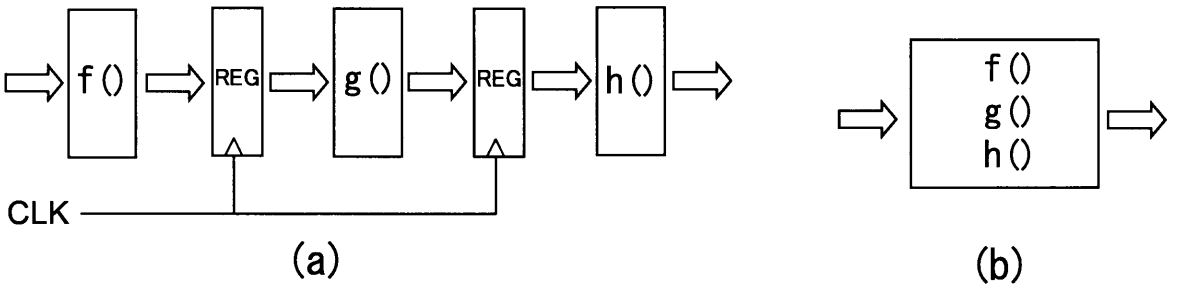


Fig. 2.1: (a) Pipeline architecture (b) non-pipeline architecture.

2.2 Multiplexer Architecture

In the proposed microcontroller, rather than using bus architecture, the multiplexer architecture is preferred to ease the completion detection. In the bus architecture, the high impedance state of inout signals cannot be used to sense when a computation is completed by using dual-rail domino circuit. In contrast, in the microcontroller based on multiplexer architecture, completion detection can be easily sensed due to the non-existence of high impedance state.

In the bus architecture (Fig. 2.2(a)), the inout signals could be realized easily through the implementation of tri-state cell and tri-state bus. Nevertheless, the high impedance state of inout signals cannot be used to sense when a computation is completed by using dual-rail domino circuit. This results the existence of uncertainty state in completion signal. The uncertainty state in completion signal could stall the operation due to no completion of the instruction. In contrast, in the microcontroller based on multiplexer architecture (Fig. 2.2(b)), the uncertainty state in completion signal is disappeared due to the non-existence of high impedance state. Furthermore, data are transferred using dedicated lines, so that the bus drivers need not to drive the extra capacitance related to unnecessary drivers and receivers, which results in a reduction of power dissipation. The multiplexer architecture is able to reduce the power dissipation by about 30% compared to similar bus architecture [13]. By adopting multiplexer architecture, the number of transistors used in the proposed processor is decreased by 9,916, which is about 7.4% if compared to bus architecture as shown in Table 2.1. However, the inout signals could not be realized in the multiplexer

architecture. Instead of using inout signals directly, there is a need to represent the inout signals by using input and output signals separately.

Fig. 2.3 shows the structure of the proposed microcontroller built on the conventional multiplexer circuits. Noted that the data flow is controlled dedicatedly by the control signals generated from the instruction decoder, and registers are only used in the final stage of the structure to store the data.

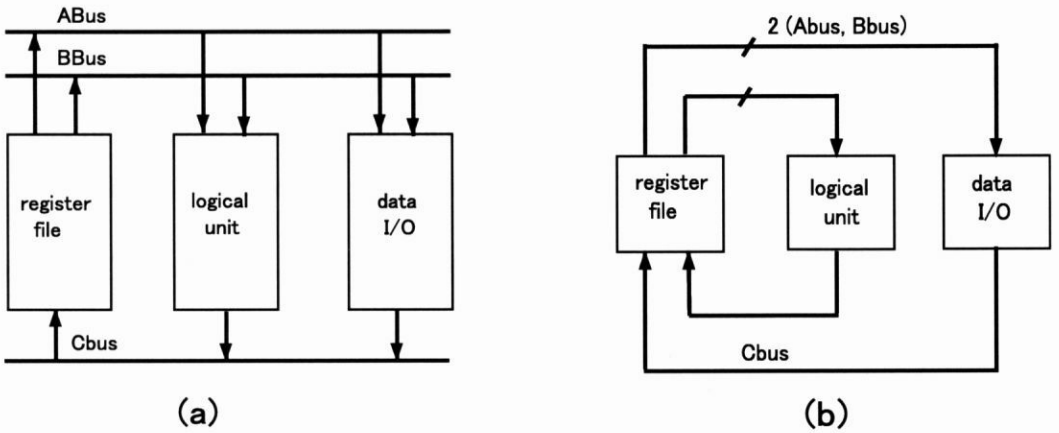


Fig. 2.2: (a) Bus architecture (b) multiplexer architecture.

Type	No. of transistor
Bus architecture	134,659
Multiplexer architecture	124,743

Table 2.1: Transistors used in different architectures.

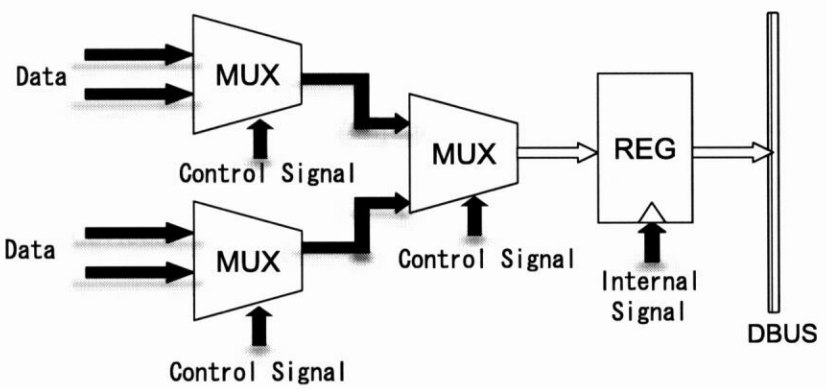


Fig. 2.3: Conventional multiplexer architecture.

2.3 Instruction Set

The proposed microcontroller is designed based on the instruction set of Z80 microcontroller [14]. Although Z80, which was first developed in 1976 by Zilog, Inc., might be an out-of-date microcontroller, it is still widely used in embedded applications and can be a good comparison standard for our new design. The Z80 instructions fall into theses major groups:

- Load and Exchange - The load instructions move data internally among CPU registers or between CPU registers and external memory. The exchange instructions can trade the contents of two registers.
- Block Transfer and Search - The block transfer instructions allow a block of memory of any size can be moved to any other location in memory.
- Arithmetic and Logical - The arithmetic and logical instructions operate on data stored in the accumulator and other general-purpose CPU registers or external memory locations.
- Rotate and Shift - The rotate and shift instructions allows any register or any memory location to be rotated right or left, with or without carry either arithmetic or logical.
- Bit Manipulation (Set, Reset, Test) - The bit manipulation instructions allow any bit in the accumulator, any general-purpose register, or any external memory location to be set, reset or tested with a single instruction.
- Jump, Call and Return - The JUMP, CALL and RETURN instructions are used to transfer between various locations in the user's program.
- Input/Output - The input/output group of instructions allow for a wide range of transfer between external memory locations or the general-purpose CPU registers, and the external I/O devices.
- Basic CPU Control - The basic CPU control instructions allow various options and modes such as setting or resetting the interrupt enable flip-flop or setting the mode of interrupt response.

The microcontroller supports 157 instructions from the 198 instructions possessed by Z80. This includes the load instructions, block transfer and search instructions, arithmetic

and logical instructions, rotate and shift instructions, bit manipulation (test) instructions, JUMP, CALL and RETURN instructions, and input/output group of instructions. However, the microcontroller does not support the exchange instructions, bit manipulation (set, reset) instructions, and the basic CPU control instructions due to the restrictions imposed by its architecture.

2.4 Structure of Microcontroller

The architecture of the microcontroller is redesigned so that the proposed 8-bit non-pipeline microcontroller built on dual-rail domino circuit could be realized. As the result, it is designed with 6 operation units, 12 CPU registers, and 168-bit control signals. The control signals are used to control 76 selectors in the microcontroller due to its multiplexer architecture. Also, 6 operation units need to be used as the reuse of the operation units are prohibited due the DCVSL's latch-free behavior of the microcontroller. In some of the Z80 instructions, for instance the 16-bit load instructions, the upper and lower 8-bits of the 16-bit CPU register needs to access memory twice during a cycle of operation. Since DCVSL behaves in such a way that no changes in inputs is allowed before the next precharge phase, two data and two address buses need to be used to ensure that the buses work properly [15]. The structure of the microcontroller is featured in Fig. 2.5, and the original Z80 structure is shown in Fig. 2.4.

2.5 Self-Recovery

A self-recovering circuit can detect and recover from a transient fault. Conventional method to synthesize the self-recovering circuit is to duplicate the computation. The two copies of the computation then execute on disjoin hardware, both copies taking the same input. A voting circuit checks the result, detecting a fault if the outputs disagree. Once a fault is detected, the computation then is restarted again [16].

In our study, we have proposed a self-recovery mechanism that takes advantages of signaling redundancy of the dual-rail domino circuit. This eliminates the need of using the duplicate circuit as in the conventional method. The self-recovery mechanism is shown in Fig. 2.6. The fault can be detected by using an error detection circuit and a timer for illegal output pair of 11 and 00, respectively. Once a fault is detected, the computation then is restarted again. To enhance the success rate of re-computation, the concept of self-recovery mechanism involving "Smart Clock Driver" is proposed. The "Smart Clock Driver" is

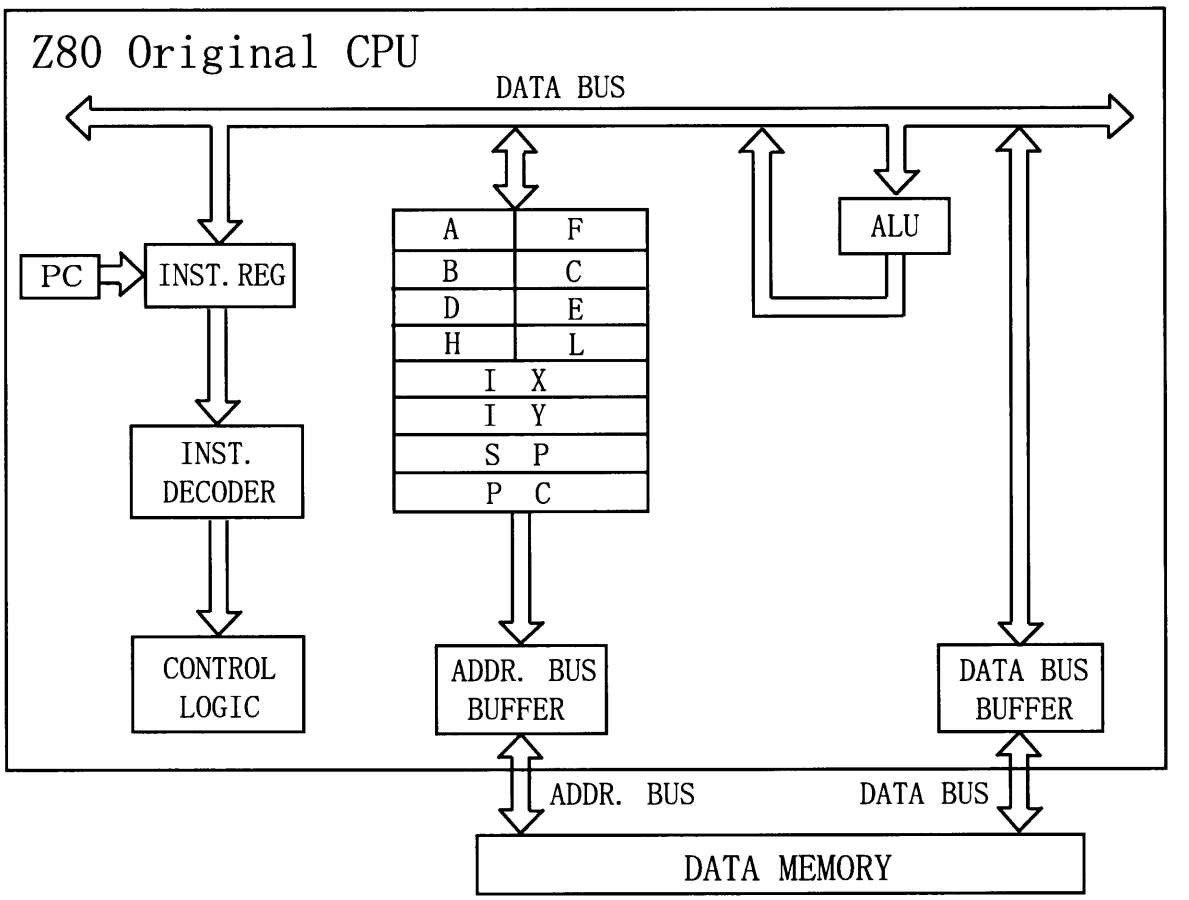


Fig. 2.4: Z80 original structure.

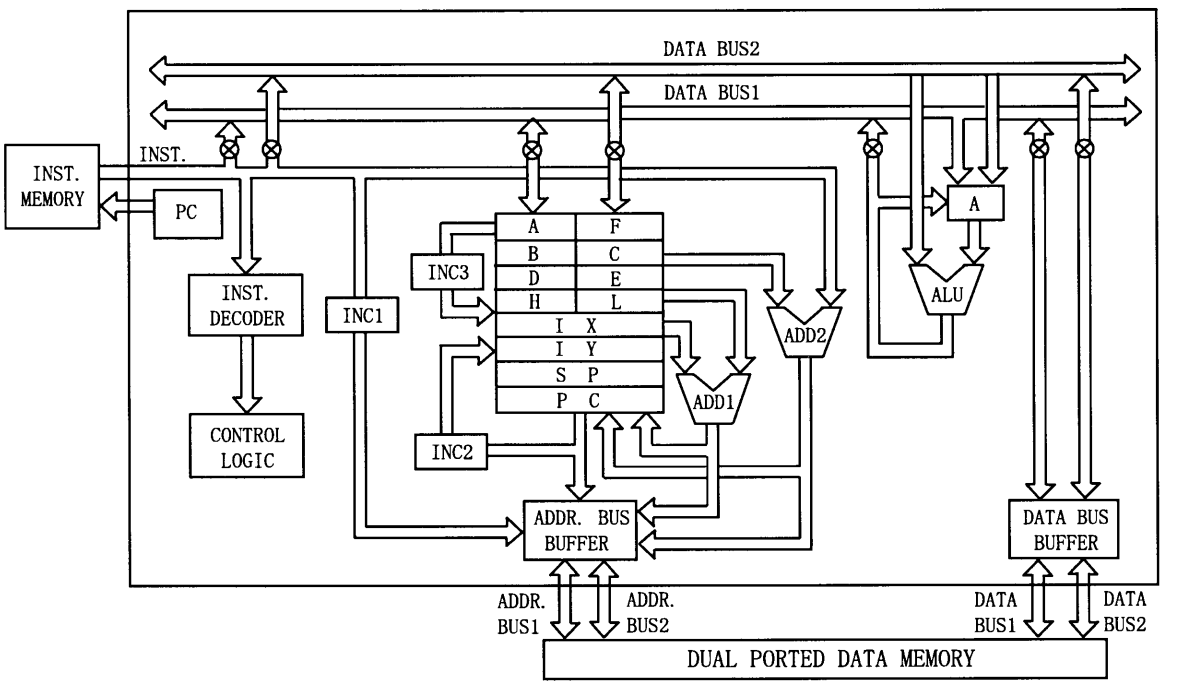


Fig. 2.5: Structure of proposed microcontroller.

functioning as a timing generator to vary the parameters and conditions of the circuit before the next trial of re-computation. The parameters and conditions of the circuit that are considered to be varied are those closely related with the occurrence of the faults. This includes the precharge time, supply voltage (Vdd), and the transistor threshold voltage (Vt). If the precharge time of the microcontroller is too short, some parts of its domino circuits might not be precharged during the precharge phase. To effectively ensure that the whole circuits are completely precharged is to increase the precharge time. The sources of noise for dynamics circuits include charge leakage and charge sharing [17, 18]. The noise margin for a domino gate is the threshold voltage of the NMOS transistor. Any input noise above threshold voltage can turn on the NMOS and discharge the evaluation node capacitance. As the threshold voltage is scaled with the decreasing power supply voltage, the subthreshold leakage current become significantly higher which increases the rate of charge loss through an OFF transistor. One obvious way to combat the problem is to increase the threshold voltage of the NMOS transistor. On the other hand, the charge sharing occurs when the charge which is stored at the output node in the precharge phase is shared among the junction capacitance of transistors in the evaluation phase. Charge sharing may degrade the output voltage level or even cause erroneous output value. The effective way to combat the problem is to increase the power supply voltage. Thus, by increasing precharge time, power supply voltage, or transistor threshold voltage could increase the noise tolerance of domino circuits. With such adaptive self-recovery mechanism, we are expecting that the reliability of the proposed microcontroller would be increased.

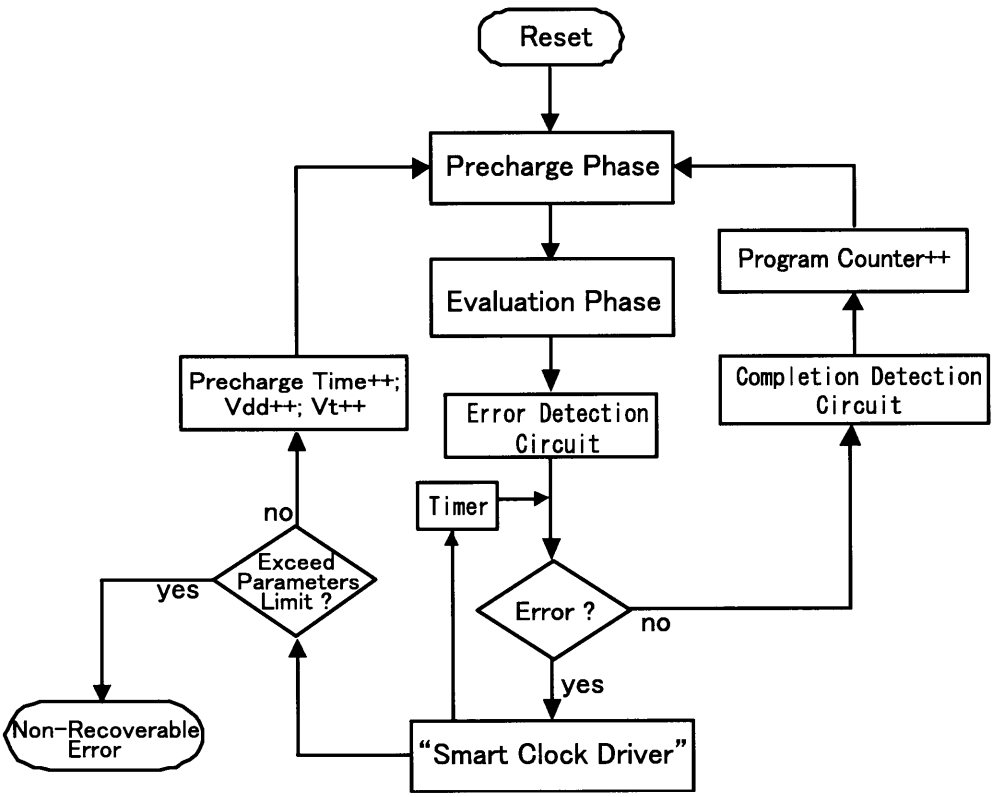


Fig. 2.6: Self-recovery mechanism involving "Smart Clock Driver".

Chapter 3

Design Methodology

3.1 CAD Methodology

In this study, we have proposed a design methodology based on the commercial synchronous CAD systems for the design of the microcontroller. Modifications are made so that the commercial CAD tools can be applicable for the design. A design discipline is developed by which any explicit dependence on the clock is carefully avoided. The circuit synthesized by the tool into a synchronous structure, but subsequently it is converted into dual-rail domino circuits that could yield the completion detection capability.

The overview of the design flow is illustrated in Fig. 3.2. Verilog-HDL description is used to produce RTL (Register Transfer Level) code that clearly exhibits the functionality prescribed in the specification. The Verilog-HDL description is then checked to verify the correctness of the description. The RTL code is used to create an optimized gate-level single-rail netlist (synchronous style) through logic synthesis. The complementary effect of dual-rail logic is manipulated to reduce the number of transistors used during the logic synthesis. For instance, the DCVSL OR gate can be realized by simply interchanging the output wires of its complementary cell, which is DCVSL NOR cell. The intermediate single-rail netlist then undergoes gate-level simulation before transferring it to a dual-rail netlist by using the translation tool that is developed in Perl language. The single-rail netlist is transferred into a fully dual-rail netlist by overloading all the single-rail assignments as dual-rail assignments. Precharge signal and buffer insertions are also performed during the translation. The Verilog-HDL description and its generated netlists are shown in Fig. 3.1. The generated dual-rail netlist is then checked for fanout analysis, and gate-level simulation to ensure the correctness of the translation into the dual-rail netlist. Then, place and route tool is used for placing and routing of the layout from the dual-rail netlist. Finally, the

layout is checked for DRC verification and transistor-level simulation before submitting it for chip fabrication through VDEC (VLSI Design and Education Center).

```
module flag (
 reset, we_F, regF_sel, regF_alu_sel, regF_add2_sel,
 regF_alu, regF_bit, dbus1_regF, regF_o, precharge );
  input   reset;
  input  [3:0]  regF_sel;
  input   we_F;
  assign regF_alu = (regF_alu_sel == 2'b11) ? regF[0] : 1'bz;
  assign regF_add2 = (regF_add2_sel == 2'b11) ? regF[0] : 1'bz;
```

(a) Verilog-HDL description

```
module flag (
reset, we_F, ¥regF_sel[3] , ¥regF_sel[2] , ¥regF_sel[1] ,
¥regF_o[2] , ¥regF_o[1] , ¥regF_o[0] , precharge );
input  reset, we_F, ¥regF_sel[3] , ¥regF_sel[2] ,
output regF_alu, ¥regF_dbus1[7] , ¥regF_dbus1[6] ,
   NOR2 U57 ( .A(n106), .B(¥regF_sel[2] ), .Y(n108) );
   NAND2 U58 ( .A(n110), .B(n111), .Y(n109) );
   NAND2 U59 ( .A(n112), .B(n113), .Y(n67) );
```

(b) Single-rail netlist

```
module flag  (reset_1 , reset_2 , we_F_1 , we_F_2 ,
¥regF_o_1[0] , ¥regF_o_2[0] , precharge_1 );
 input reset_1 , reset_2 , we_F_1 , we_F_2 , precharge_1 ;
 output regF_alu_1 , regF_alu_2 , ¥regF_o_1[0] , ¥regF_o_2[0] ;
 INV1 U3 ( .P(precharge_1), .A_1(n215_1 ), .A_2(n215_2 ), .Y_1(n42_1 ), .Y_2(n42_2 ) );
 NAND2 U19 ( .P(precharge_1), .A_1(n57_1 ), .A_2(n57_2 ), .B_1(n58_1 ), .B_2(n58_2 ) );
 DF     ¥regF_reg[1] ( .C(n51_1 ), .D_1(N20_1 ), .D_2(N20_2 ), .Q_1(N211_1 ), .Q_2(N211_2 ) );
```

(c) Dual-rail netlist

Fig. 3.1: Verilog-HDL description and its netlists.

3.2 Fanout Analysis

The fanout denotes the number of load gates N that are connected to the output of the driving gate (Fig. 3.3). Increasing the fanout of a gate can affect its logic output levels. When the fanout is large, the added load can deteriorate the dynamic performance of the driving gate. For these reasons, the dual-rail netlist is checked for fanout analysis to guarantee that the static and dynamic performance of the element meet specification. In this study, we use the width of the transistor to serve as the parameter for the fanout analysis (Fig. 3.4). The specification is defined as the ratio between the total width of the load gates and the driving gates.

$$Width\ Ratio = \frac{\sum Width(load\ gate)}{\sum Width(driving\ gate)}$$

The flow of the fanout analysis can be referred in Fig. 3.2. Buffers would be inserted for the gates which found to be failed meeting the predetermined specification. In this study, we have set the specification (i.e. width ratio) as 8. Width ratio of 8 is chosen to ensure that none of the dynamic performance of the driving gates is deteriorated due to the extra large load resulting from the improper translation from single-rail netlist to dual-rail netlist. The analysis is repeated until whole the driving gates inside the dual-rail netlist meet the

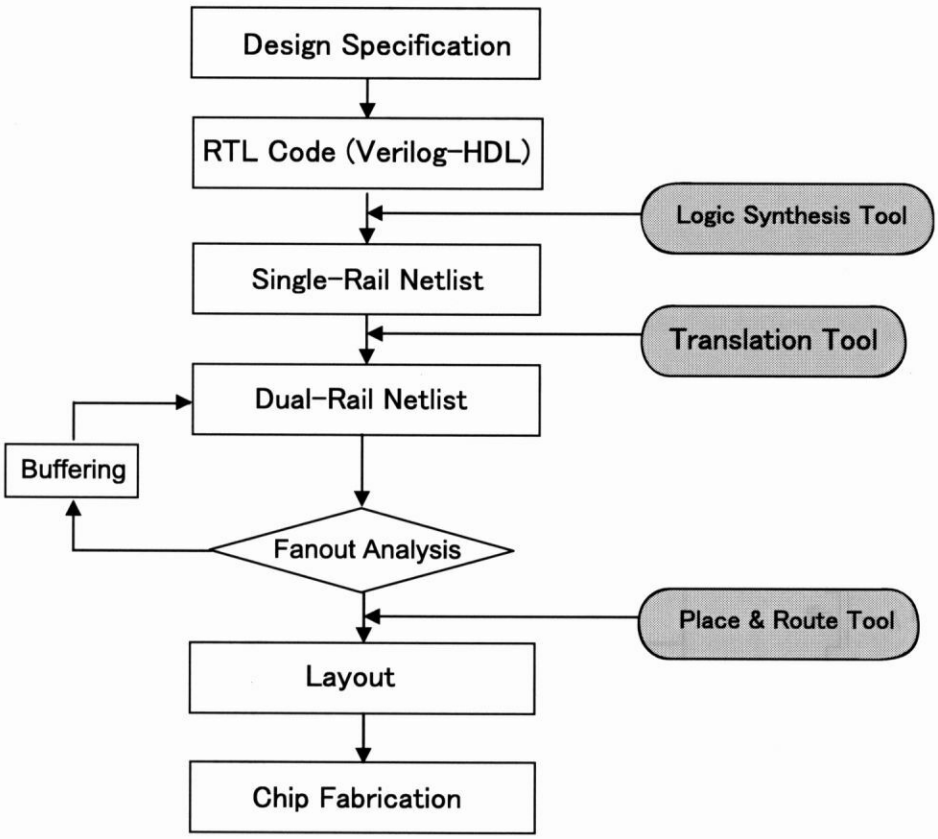


Fig. 3.2: Design flow of microcontroller.

specification. Fig. 3.5 shows the result of the fanout analysis for the dual-rail netlist which meet the specification, and ready to be used for placing and routing of the layout. From the figure, about 60% of the driving gates is comprised of width ratio equals to 1. This is due to the one-to-one connection only between the DCVSL gates.

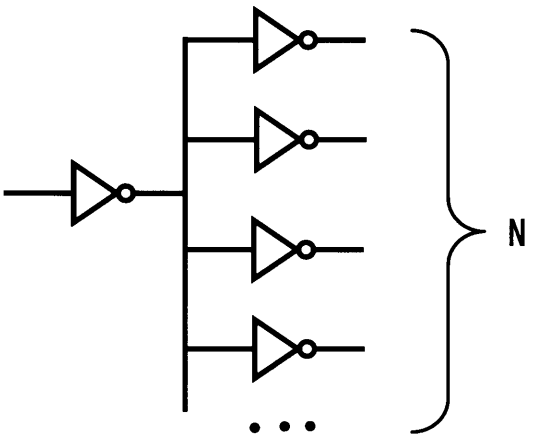
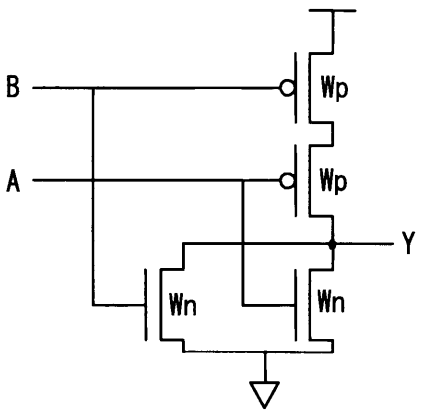
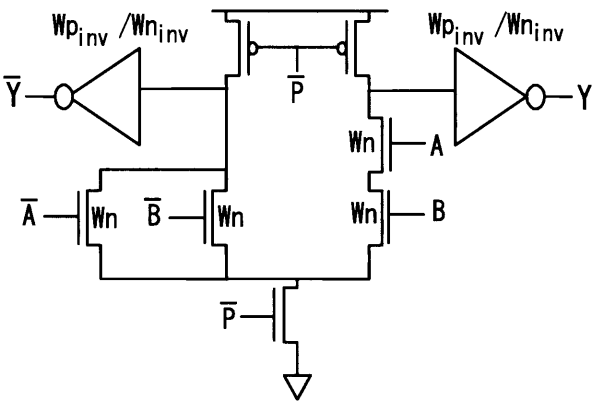


Fig. 3.3: Fanout N.



Width(load gate) : W(A)=W(B)=Wp+Wn
Width(driving gate) : W(Y)=((Wp/2)+Wn)/2

(a) Static CMOS gate



Width(load gate) : $W(A) = W(\overline{A}) = W(B) = W(\overline{B}) = Wn$
Width(driving gate) $_{\text{Evaluation Phase}}$: $W(Y) = W(\overline{Y}) = Wp_{inv}$
Width(driving gate) $_{\text{Precharge Phase}}$: $W(Y) = W(\overline{Y}) = Wn_{inv}$

(b) DCVSL gate

Fig. 3.4: Parameters used in the fanout analysis.

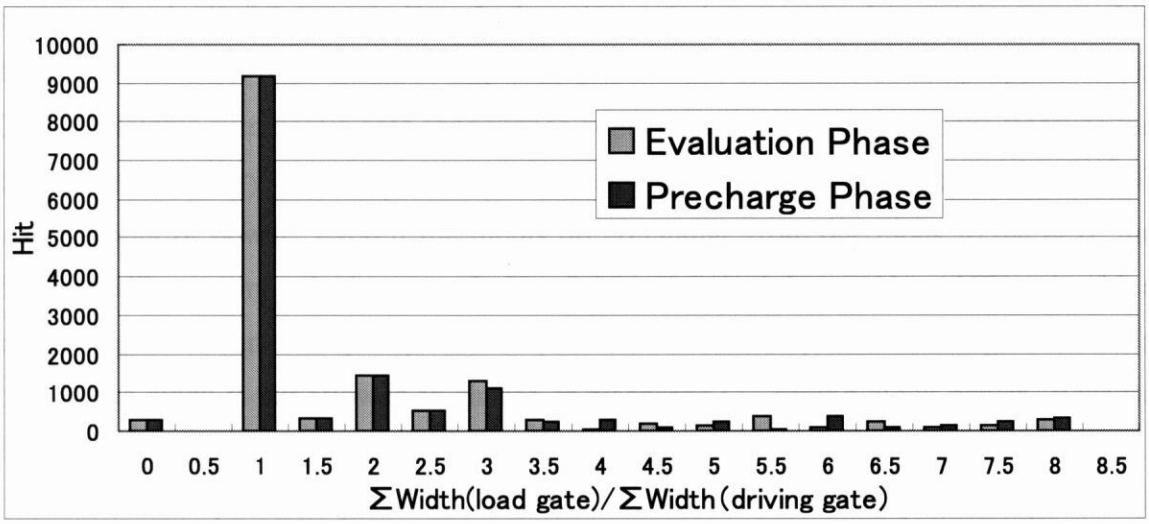


Fig. 3.5: Result of fanout analysis.

3.3 Low Power Design

As mentioned above, this study proposes to employ standard commercial logic synthesis tools to synthesize a large digital system into single-ended synchronous circuits, and to convert the result into a corresponding dual-rail domino circuits. However, the conversion process only able to realize the valid status (i.e. 01 or 10), as the single-rail signaling could at most represents two status (i.e. 0 or 1). Therefore, it suffered from not able to realize the status 00, which acts as initial status in dual-rail signaling protocol. As the result, one of the outputs of all the dual-rail domino circuits would always switch during the evaluation phase. This results in large power consumption, and also other undesirable effects due to the frequent switching activities. To deal with the problem mentioned above, low power design techniques based on clock-gating, selective-evaluation, and new multiplexer circuit have been implemented. The aim is to assure that current only flows along the useful computation paths, and no unnecessary switching activities along the idle paths.

3.3.1 Clock-Gating

Clock-gating has shown to be an efficient technique to significantly reduce dynamic power dissipation [8]. Because individual circuit usage varies within and across applications, not all the circuits are used all the time, giving rise to power reduction opportunity. By ANDing the clock with a control signal, clock-gating essentially disables the clock to a circuit whenever the circuit is not used, avoiding power dissipation due to unnecessary charging and discharging of the unused circuit. In a similar way, to reduce the power

consumption in the microcontroller, clock-gating technique targets the power consumed in flip-flops have been implemented. Fig. 3.6(a) shows the schematic of a flip-flop element. C_g is the latch's cumulative gate capacitance connected to the internal clock. Because the clock switches every cycle, C_g charges and discharges every cycle and consumes significant amount of power. Even if the inputs do no change from one clock to the next, the flip-flop still consumes clock power. In Fig. 3.6(b), the clock is gated by ANDing it with a control signal. When the flip-flop is not required to switch state, the control signal is turned off and the clock is not allowed to charge or discharge C_g, saving clock power. Because the AND gate's capacitance itself is much smaller than C_g, there is a net power saving.

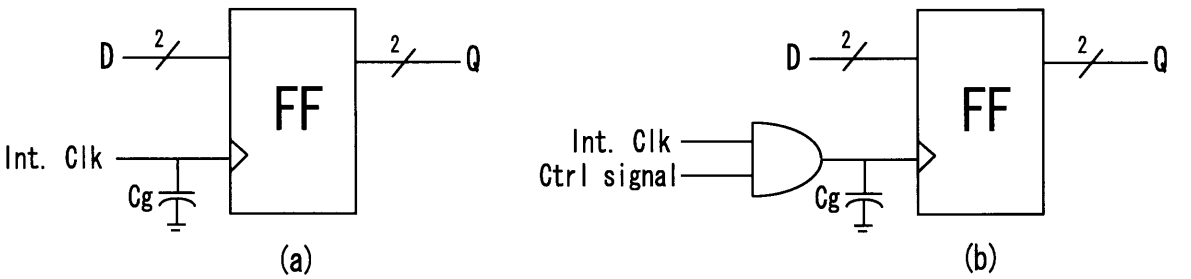


Fig. 3.6: Clock-gating a flip-flop.

3.3.2 Selective-Evaluation

To effectively implement the power-saving technique to the dual-rail domino circuit, module-level implementation of selective-evaluation technique is proposed. Module-level implementation is preferred to avoid the large overhead incurred by the control circuitry if implemented in gate-level. The proposed selective-evaluation technique is illustrated in Fig. 3.7(a). The precharge signals (p1~p2), are generated from the decoder in cases of the corresponding modules are used in the cycle (Fig. 3.7(b)). Otherwise, the precharge signals would remain low, preventing all the domino circuits inside the modules from switching in the cycle.

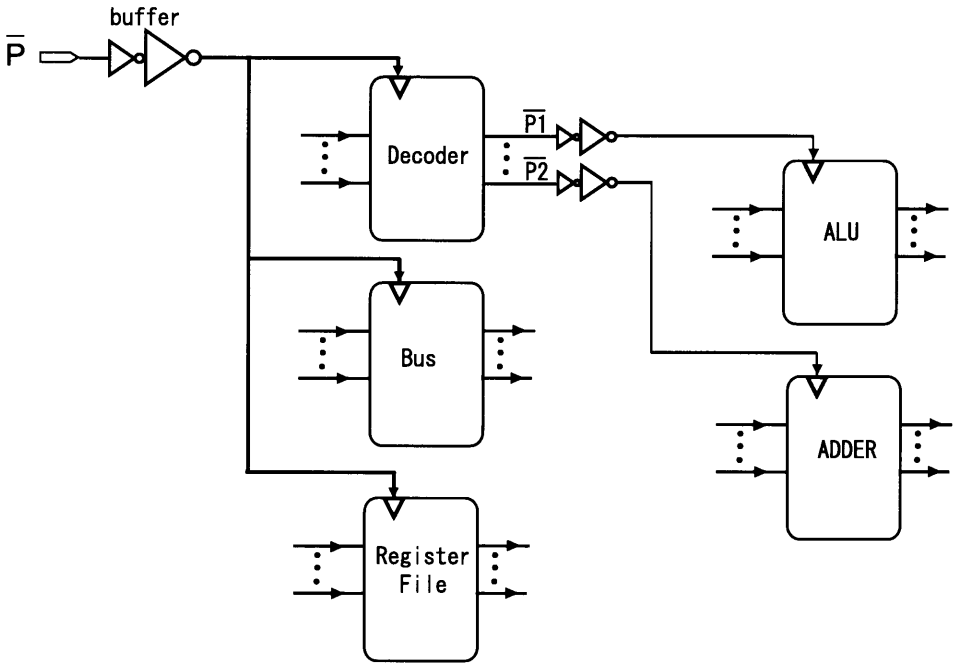
The selective-evaluation technique is used to activate or deactivate the operation units existed in the microcontroller. As many as 6 operation units are used as the reuse of the operation unit is prohibited in every cycle of instruction execution due to the dual-rail domino circuit behavior. The operation units, thus, constitute as much as 21% of all the transistors in the microcontroller. Because individual operation unit usage varies with instructions, not all the operation units are used all the time, giving rise to power reduction opportunity.

Not only the power is significantly reduced, the peak precharge current is also considerably reduced by using the selective-evaluation technique. Fig. 3.8 illustrates the precharge scheme in the conventional microcontroller design. The simultaneous precharge of all the domino circuits resulting in large peak precharge current, which might causes an unacceptable IR-drop noise. In contrast, the precharge sequence is delayed during the precharge phase for the module that was used in the previous evaluation phase by using selective-evaluation technique. For the module that was idle in the previous evaluation phase, precharging an already-charged module does not consume power unless there are leakage losses (which we do not consider in this paper). Both the cases contribute to reduce the peak precharge current in the microcontroller. The precharge timing sequence of the selective-evaluation technique is illustrated in Fig. 3.7(b).

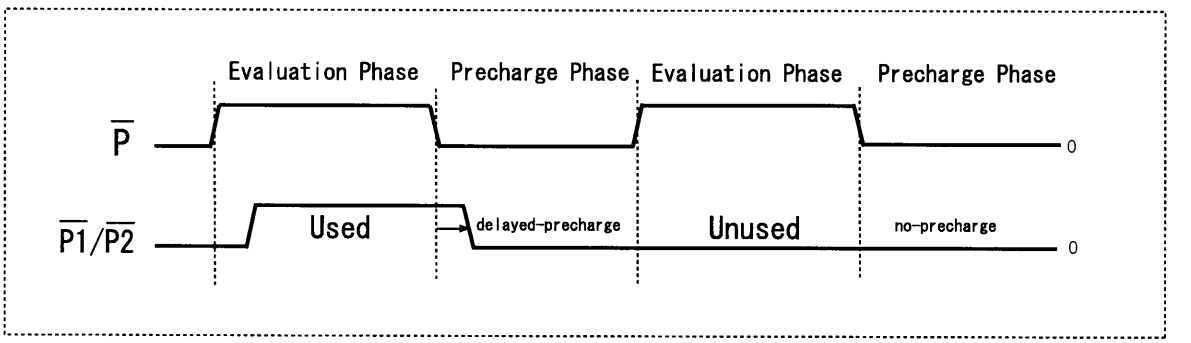
3.3.3 New Multiplexer Circuit based on Bulb and Junction Structure

Besides implementing selective-evaluation technique in the module-level targeting operation units, we have been proposing new low power technique in circuit-level specifically targeting the multiplexers. Multiplexers have been widely used in the microcontroller due to the multiplexer architecture which has been adopted. The conventional 2-input multiplexer is shown in Fig. 3.9. The conventional multiplexer would always switch during the evaluation phase even it is not used in the cycle. To automatic shut-off the idle parts efficiently, we have proposed a new multiplexer circuit built on the bulb and junction structure. To utilize the new multiplexer circuit, the selector signals are described in the one-hot encoding. Fig. 3.10 shows the illustration of the new multiplexer circuit. Also, the 4-input new multiplexer circuit built on 2-input JUNCTION (JUNC2) gate is illustrated in Fig. 3.11. The BULB and JUNC2 gates used for the construction of bulk and junction structure are shown in Fig. 3.12. The new multiplexer circuit effectively ensures that for every instruction executed, current only flows through a particular channel of the multiplexer circuit selected by the one-hot encoding. For the channel selected by the one-hot encoding, the precharge signal to its bulb gate would turn high. Consequently, the bulb gate would enter the evaluation phase. For the rest of the channels, the precharge signals would remain low, preventing the bulb gates and also the nodes along the channels from switching in the cycle; hence contributing to power-saving. Also, the delayed-precharge scheme of the bulb gate, and the effective prevention of the unnecessary switching of the new multiplexer could help to reduce the peak precharge current.

Fig. 3.13 shows the comparison between the conventional multiplexer and the new mul-



(a) Selective-evalutation implementation in microcontroller.



(b) Precharge Timing Sequence.

Fig. 3.7: Implementation of selective-evaluation in dual-rail domino circuit.

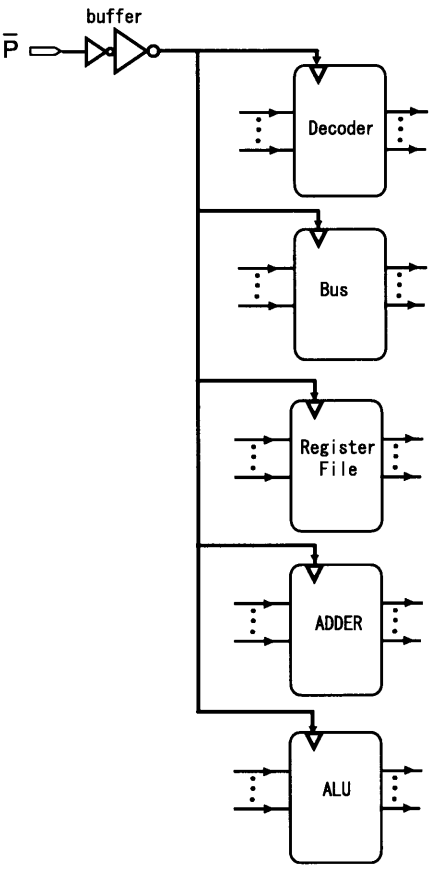


Fig. 3.8: Precharge scheme in conventional microcontroller design.

tiplexer in terms of number of transistors used in one-hot encoding. The data used is based on the results on logic synthesis using gates limited to INV1, NAND2, NOR2, BULB, and JUNC2. The results show that the new multiplexer uses less transistors than the conventional multiplexer if implemented in DCVSL circuit.

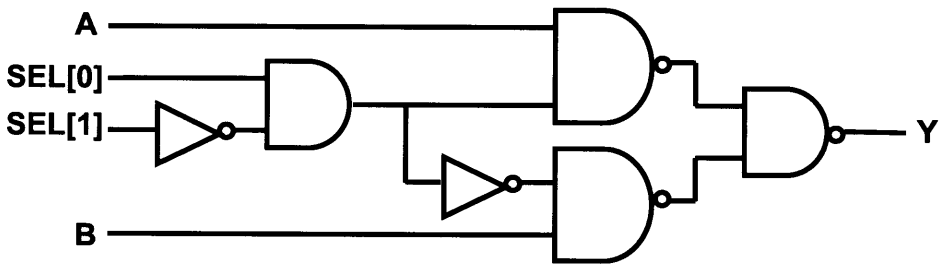


Fig. 3.9: Conventional 2-input multiplexer.

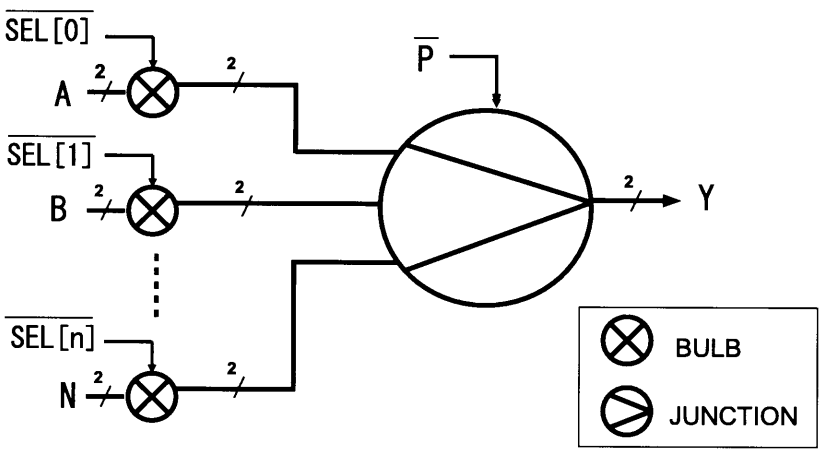


Fig. 3.10: Illustration of the new multiplexer circuit.

3.4 Layout of Microcontroller

The layout of the microcontroller is implemented by using Rohm 0.35μm CMOS technology with chip size of 4.9×4.9mm^2 (Fig. 3.14). The total number of transistors used in layout is 93,975 with core size of 2.2×2.2mm^2.

The total number of transistors that is used in the layout is notably high if compared to 8,200 transistors that are used in the original Z80 microcontroller. For DCVSL circuit, no inputs change is allowed before the next precharge phase. This results in a design that for every cycle of operation, at most only one transition is allowed to occur for each of the inputs of the logic circuits, which contributes to the increased number of transistors. In

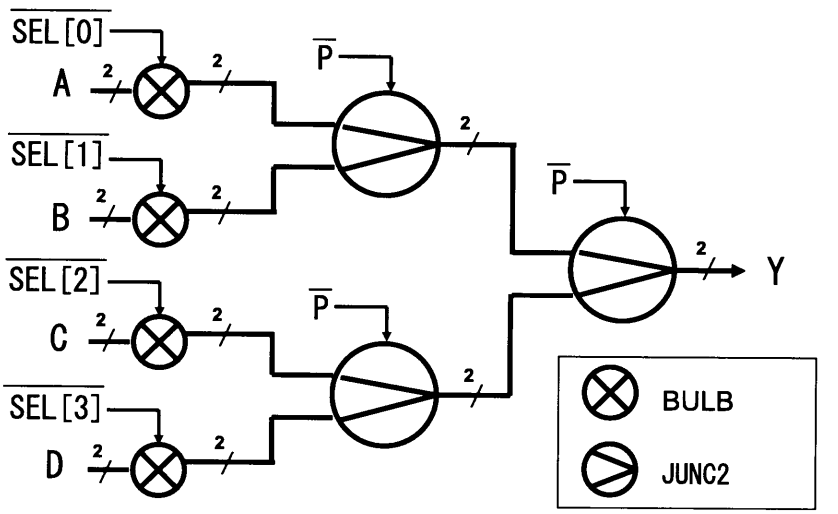


Fig. 3.11: Block diagram of the 4-input new multiplexer circuit.

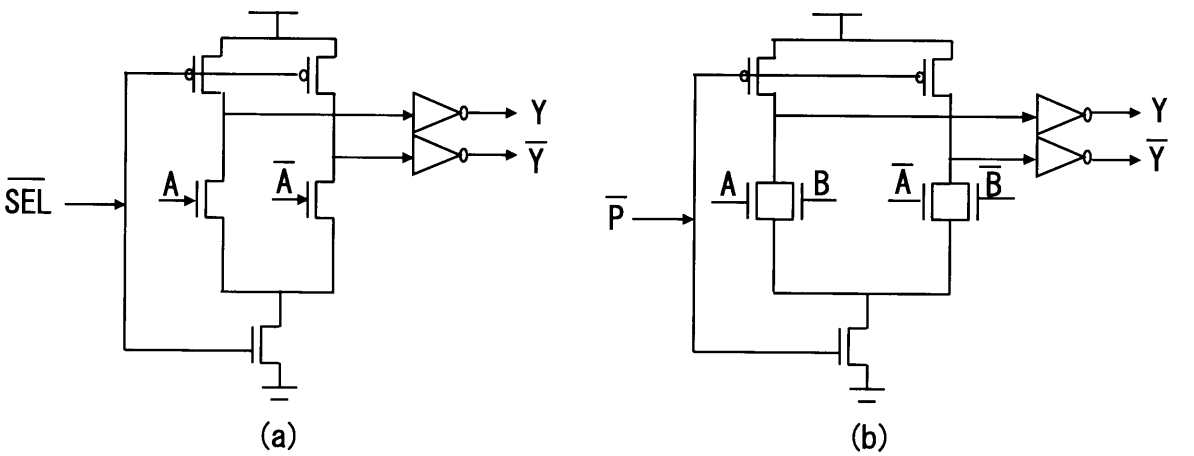


Fig. 3.12: (a) BULB gate (b) JUNC2 gate.

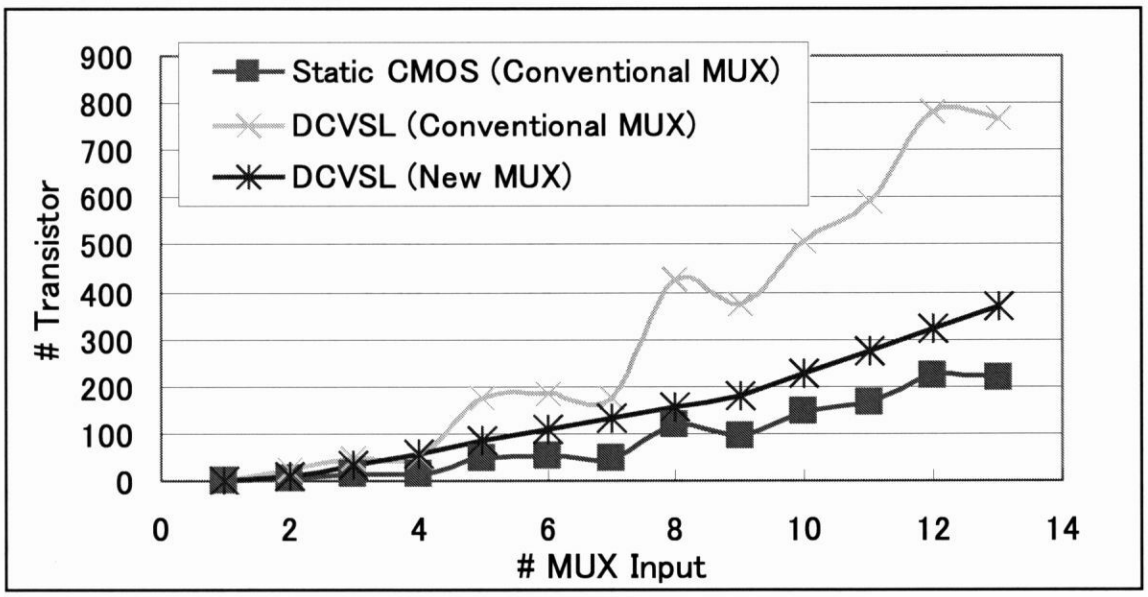


Fig. 3.13: Transistors used in multiplexer using one-hot encoding.

addition, DCVSL circuit used in the microcontroller consumes more transistors than static CMOS circuit as shown in Table 3.1.

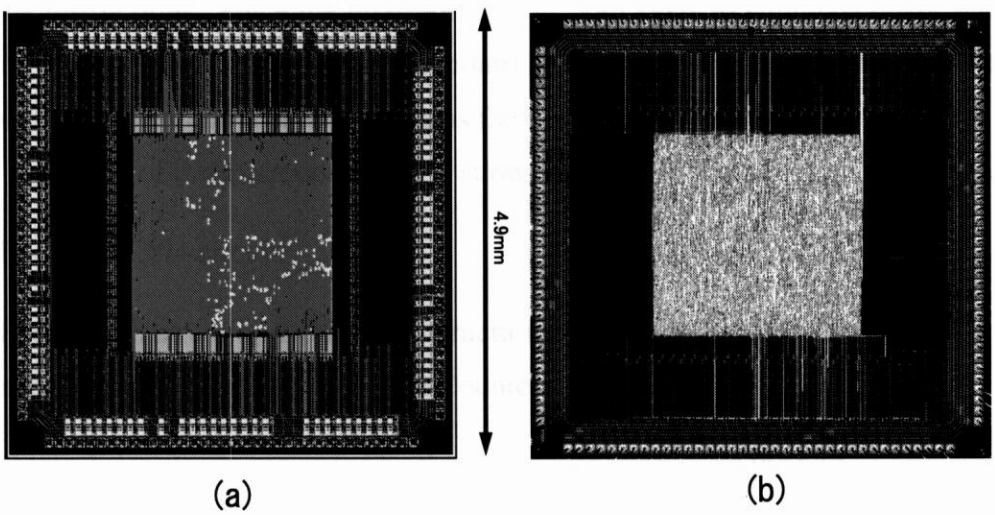


Fig. 3.14: (a) Layout of microcontroller (b) overview of chip fabricated.

3.5 Operation of Microcontroller

The proposed microcontroller is simulated by using the model which is featured in Fig. 3.15. The microcontroller is composed of three main parts, namely CPU (Central Processor Unit) circuit, completion detection circuit and error detection circuit. The CPU circuit

Type of gate	No. of transistor per gate	
	DCVSL	CMOS
INV1	11	2
NOR2	13	4
NAND2	13	4
XOR2	15	10

Table 3.1: Number of transistor in DCVSL and static CMOS gate

is composed of dual-rail domino logic circuit, while the completion detection and error detection circuits are composed of static CMOS circuit. The virtual peripheral units needed for simulation are timing generator, dual-ported memory, instruction memory and I/O device. The timing generator is employed to control and adjust the timing related items such as precharge time, register write time, and to serve as a timer. A 64K x 8 dual-ported RAM (Random Access Memory) memory is used to serve as the data memory of the microcontroller. The dual-ported memory is used as there is a need to access memory twice per cycle for some instructions. Since the dual-port allows both sides to simultaneously access the memory, it could also enhance the microcontroller performance. A 64K x 32 ROM (Read Only Memory) memory is used to serve as the instruction memory of the microcontroller.

The operation of the microcontroller is summarized as below.

1. Reset the CPU circuit.
2. Fetch the instruction from the instruction memory directed by instruction memory address saved inside the Program Counter.
3. Control signals are generated by the instruction decoder in the CPU circuit according to the instruction being fetched.
4. CPU circuit is executed according to the control signals.
5. Completion signal is generated after the instruction is completely executed in the completion detection circuit.
6. Internal clock is generated according to the completion signal in timing generator.
   (a) Internal clock is synchronized with the external clock.
   (b) Computation result is written into registers according to the internal clock.

(c) Next instruction is fetched according to the internal clock.

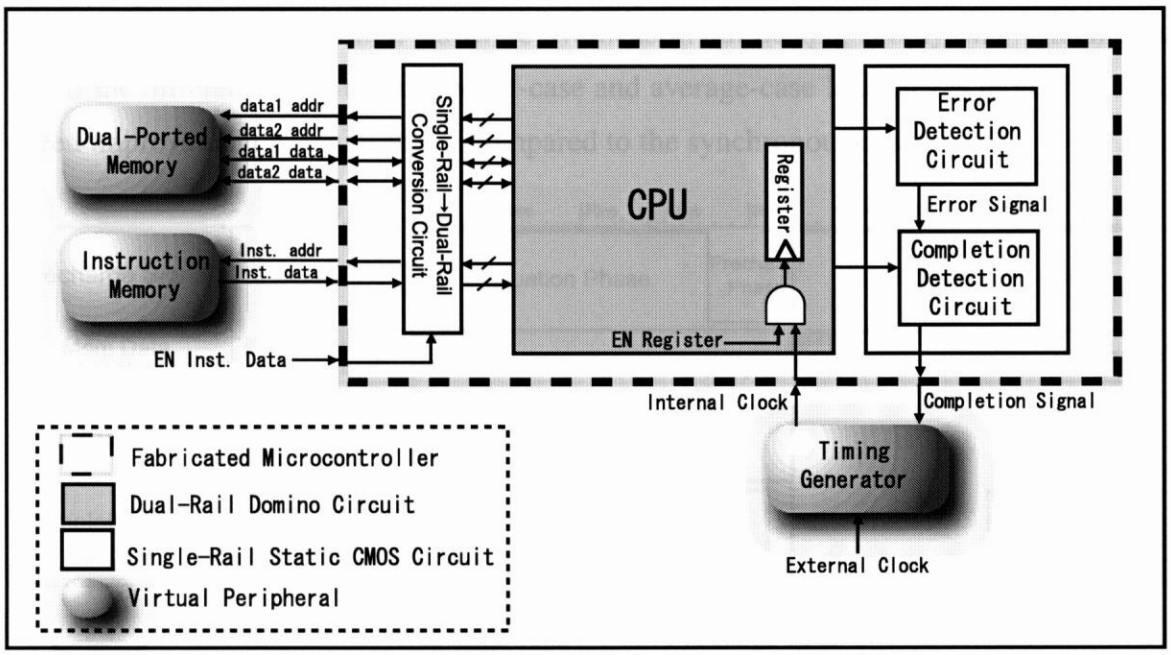


Fig. 3.15: Simulation model.

3.6 Simulation Results

The simulation is done using transistor-level circuit simulator, NanoSim, in full-chip simulation. Fig. 3.16 shows the waveform of the simulation. The operation of the microcontroller is divided into two phase, namely precharge phase and evaluation phase. The evaluation time and the precharge time are also shown in the figure. The evaluation time is defined as the time lapse between the precharge signal and the completion signal. From the simulation, the minimum time required for the precharge time is 1.8ns.

The full-chip simulation is also done to estimate the ratio of the evaluation time consumed in different circuit components of the microcontroller. The microcontroller is divided into 4 main circuits, namely single-rail to dual-rail conversion circuit, instruction decoder, computation circuit, and completion detection circuit (Fig. 3.17). Fig. 3.18 shows the results of the simulation for two different instructions, which are load (LD A,n) and subtraction (SUB n) instructions. The result reveals that for arithmetic and logical instructions such as subtraction instruction, the major propagation delay time occurs in the computation circuits. In contrast, for others such as load instruction, the time consumed in instruction decoder becomes the main component for the evaluation time. The instruction

decoder is considerable large as it is used to generate 168-bit control signals that are used to control the 76 selectors in the microcontroller. Please note that the time consumed in completion detection circuit, a hardware overhead in our design, is only 4ns. This means that if the difference between the worst-case and average-case latencies is more than 4ns, speed improvement is achievable if compared to the synchronous design.

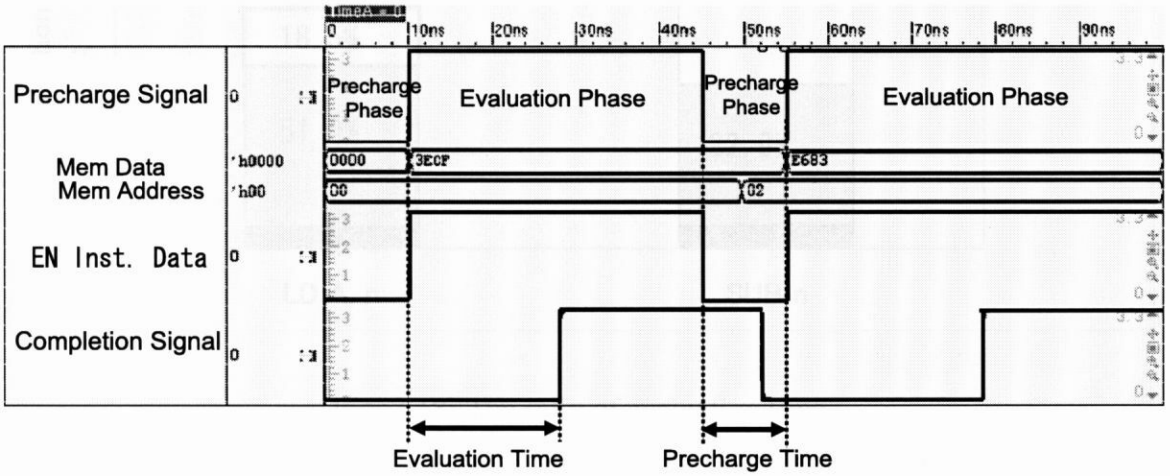


Fig. 3.16: Waveform of full-chip simulation.

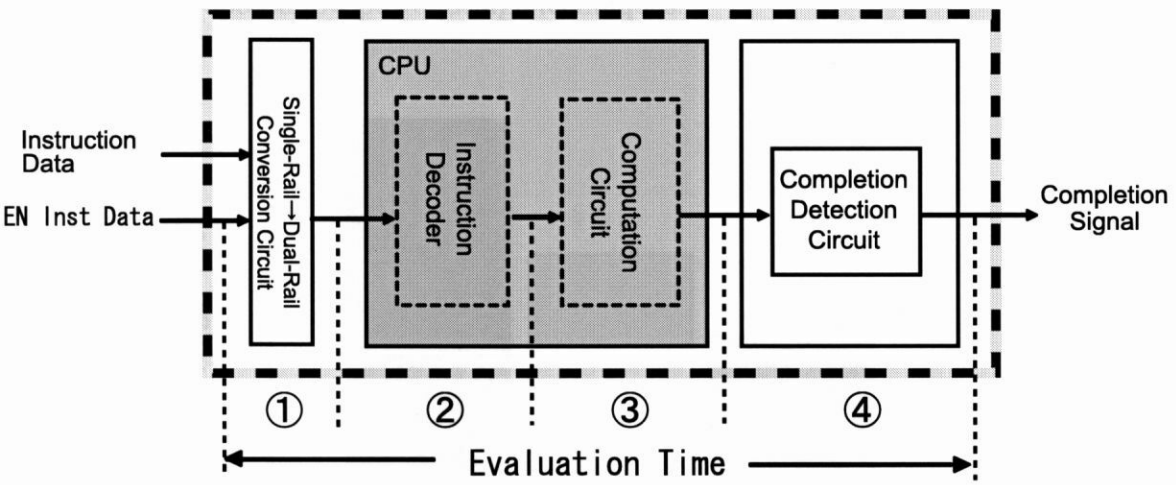


Fig. 3.17: Circuit components related to evaluation time.

The simulation is also done to evaluate the power-saving efficiency of the proposed low power design techniques. The power-saving efficiency is measured using two parameters, namely the energy consumption (Fig. 3.19) and the peak precharge current (Fig. 3.20). The microcontroller achieves a total energy reduction of 53% from the low power design techniques that have been implemented. From the total energy reduction, the new multiplexer circuit and the selective-evaluation technique constitute 33% and 20%, respectively. A total of 69% peak precharge current reduction is achieved by implementing the low power

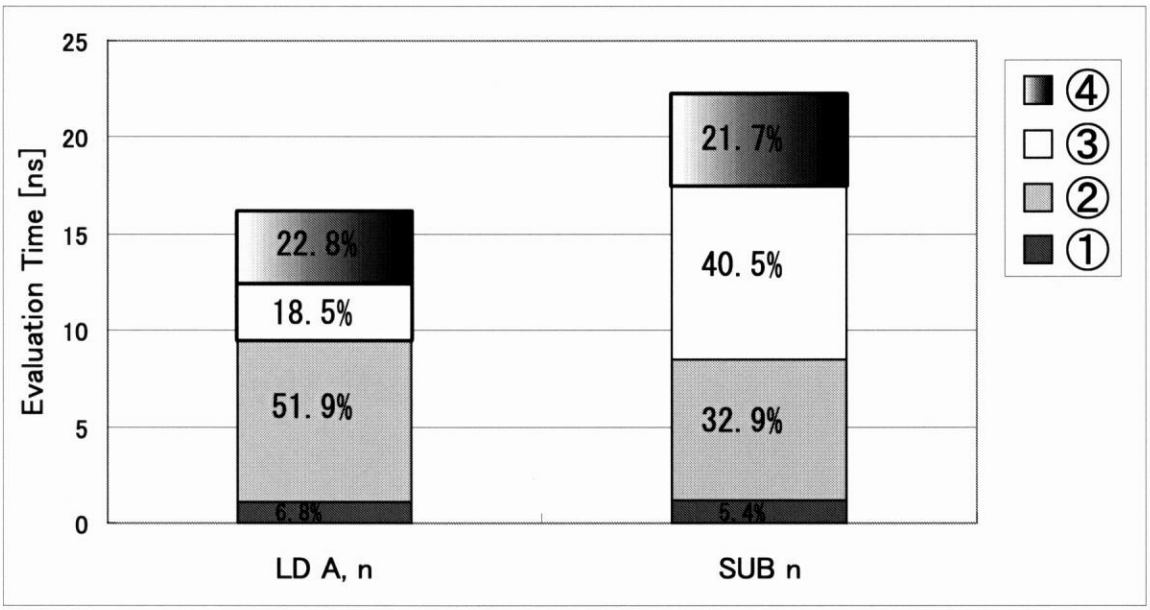


Fig. 3.18: The ratio of evaluation time.

design techniques. Specifically, the new multiplexer circuit and the selective-evaluation technique help to reduce 48% and 21%, respectively. This proves the power-saving efficiency of the proposed low power design techniques.

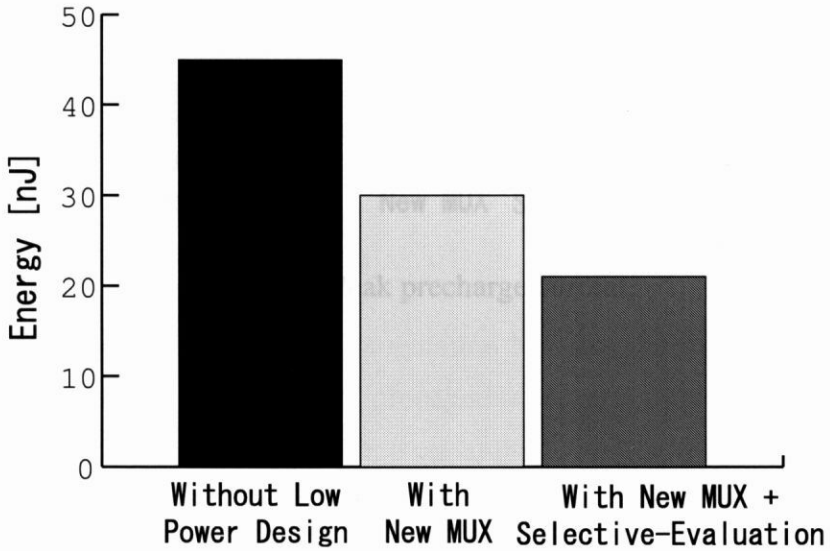


Fig. 3.19: Energy consumption.

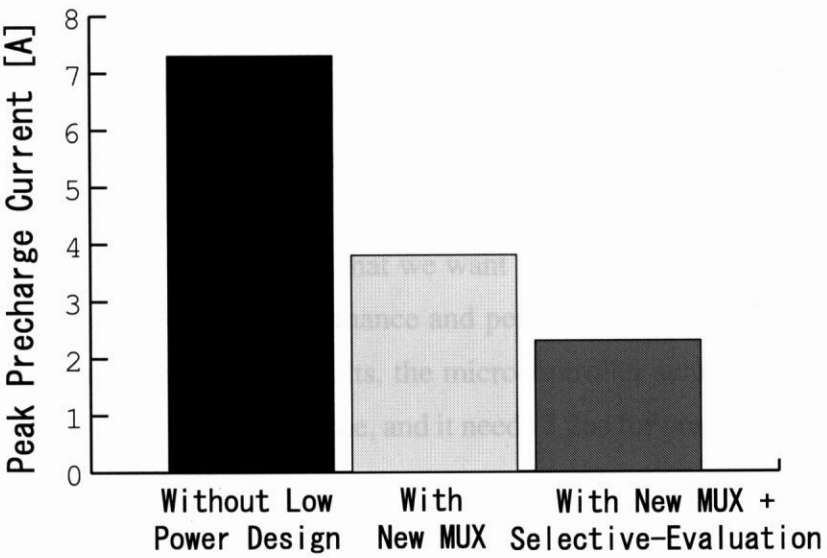


Fig. 3.20: Peak precharge current.