Design of 3.33GHz CML Processor Datapath

by

Abdullah Al Owahid

A thesis submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Auburn, Alabama
May 7, 2012

Keywords: CML, CMOS, Processor

Approved by

Fa Foster Dai, Chair, Professor of Electrical and Computer Engineering
Vishwani D. Agrawal, James J. Danaher Professor of Electrical and Computer Engineering
Victor P. Nelson, Professor of Electrical and Computer Engineering

Abstract

Almost a decade processor speed has been stuck at operating frequency 2-3GHz due to excessive power consumption of CMOS logic gate at higher frequency whereas predicted speed at present was 10-15GHz. This leads the idea of multi-core design in today's processor architecture. However it increases the communication overhead β and there exist data dependency which cannot fully exploit the advantage of many-core design. Further many core design is increasing number of dark silicon and number of core cannot be increased after certain limit. Therefore a novel approaches in processor design using CML logic gate has been proposed.

Handcrafted 16-bit CML microprocessor datapath has been developed at operating frequency 3.33GHz using 130nm CMOS technology. With the same feature size, CMOS gate is incapable to operate beyond 1GHz whereas CML logic gates were optimized for 12GHz using bias current of 70% of peak f_t current with a logic swing of 600mV. Considering critical path delay, circuit has been slowed down to operate at 3.33GHz.

All the processor components - decoder, mux, register file, ALU was deliberately handcrafted due to lack of analog synthesizer tool. Reported static power consumption of multicycle CML processor datapath is 41.264W. However it is not the best case and could have been reduced to 50% by implementing multi-input CML logic. Expected chip area is 2.2mm x 3.45mm and power density per unit area is $5.44\mu W/\mu m^2$. Estimated performance evaluated is 892 MIPS. Supply voltage used is 2.8V. CML logic was defined as, logic-1 = 2.8V and logic-0 = 2.2V. 1V reference voltage was used to constant bias the current source and reset signal uses 1.3V and 0.7V for high and low logics respectively. It has been observed that it is possible to realize ultra-high speed processor using existing technology with minimum power consumption in CML logic.

Acknowledgments

I would like to acknowledge the continuous support and guidance of Dr. Fa Foster Dai. Without his suggestion and direction it would have been impossible to complete this thesis work. I would also like to thank my committee members Dr. Vishwani D. Agrawal for his meaningful suggestions regarding processor architecture and Dr. Victor P. Nelson.

I thank my friends and colleagues - James Clark, Shannon Price, Xin Jin and Baohu Li for being with me and making life at Auburn enjoyable.

Last but not the least, I would like to thank my family members - my parents whose love brought me so far, my brother and sister, and especially my wife for her patience.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

ALU Arithmetic Logic Unit

BiCMOS Bipolar and CMOS in same integrated chip

BJT Bipolar Junction Transistor

CML Current Mode Logic

CMOS Complementary Metal Oxide Semi-conductor

ECL Emitter Coupled Logic

ISA Instruction Set Architecture

MCML Mos Current Mode Logic

MIPS Million Instructions Per Second

nMOS n-type Metal Oxide Semi-conductor

RISC Reduced Instruction Set Computing

SoC System on Chip

SPICE Simulation Program with Integrated Circuit Emphasis

SRAM Static Random Access Memory

Chapter 1

Introduction

This thesis presents the design of a handcrafted 3.33GHz CML processor datapath. RISC architecture has been adopted in designing the multi-cycle processor datapath and the ISA is 16-bits long. It is the first ever MCML processor that requires constant power dissipation unlike CMOS processors. Also, once optimized, power dissipation does not increase with increasing operating frequency. Optimizing CML logic for higher operating frequency requires higher power than optimizing for lower frequency. Therefore, once CML logic has been optimized for a targeted maximum frequency, operating the circuit at lower frequency will lose the benefits in terms of power.

Due to larger switching noise associated with CMOS circuits, CML is a better choice for high speed circuit realization [1]. BJT based CML gates are faster than MOSFET CML due to higher g_m and lower power but have been avoided for process difficulty and uncommonness in designing digital circuits. Therefore MOSFET-CML (MCML) logic has been used.

1.1 Problem Statement

The problem solved in this thesis: *Design a high speed low power processor datapath.*

1.2 Background and Motivation

Processor speed has been stuck at 2-3GHz due to excessive power consumption, as indicated in Figure 1.1 for the last 10 years [2]. Therefore multi-core design has evolved to increase performance. But the number of cores cannot be increased after a certain limit and there exists communication overhead β. Further, due to data dependency, programs cannot be fully parallelized, which inhibits proper exploitation of multi-core design.

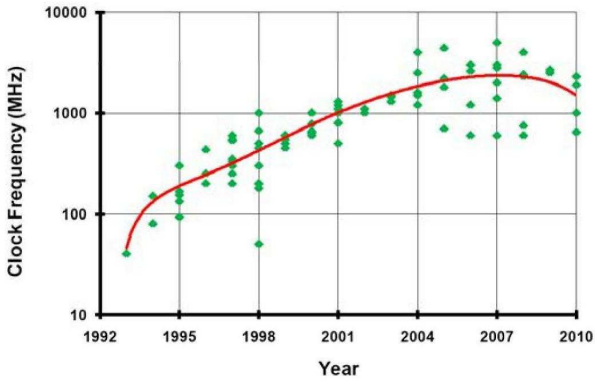

Figure 1.1: Operating frequency over time

Intel processor speed vs power have been obtained and plotted in Matlab as depicted in Figure 1.2 [3].

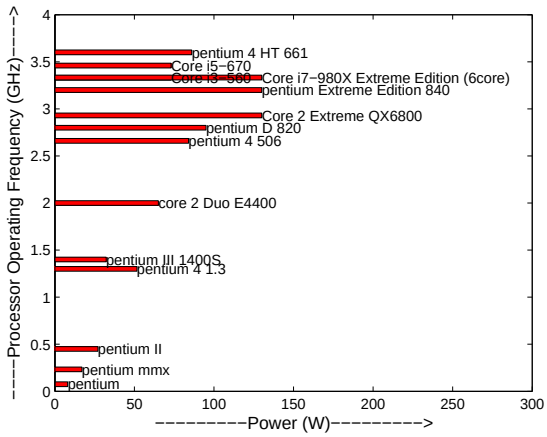

Figure 1.2: Operating Frequency vs. Power of Intel Processor

In Figure 1.2 it is observed that processor power consumption has been increased almost exponentially at operating frequencies beyond 2GHz. Also notable is that, core i-5 has higher operating frequency than core-i7 but the power consumption in core i-7 is almost double due to a higher number of cores. So reducing the number of cores will not only reduce power but can result in higher performance due to less data dependency.

Power consumption can also be reduced by moving to deep submicron technology. However, the main drawback to reducing feature size is it increases unit power density and chips cannot sustain that power, as shown in Figure 1.3 [4].

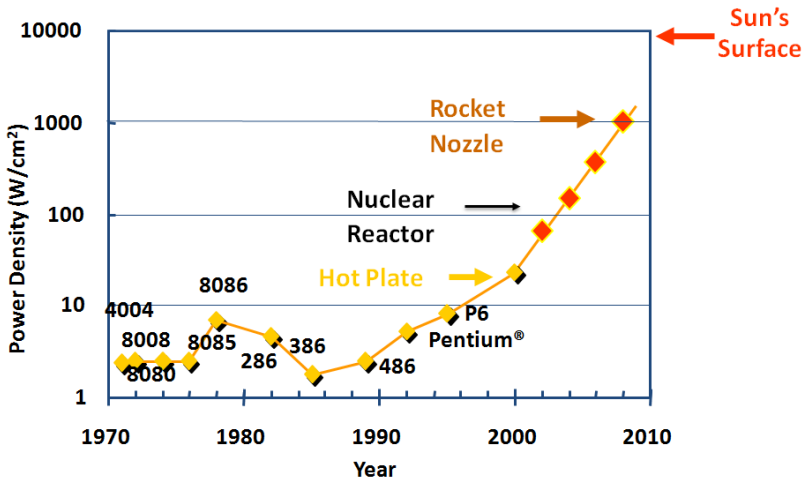

Figure 1.3: Power density per unit area

In CML, per unit power density is lower as it requires greater area than CMOS due to load resistances in CML logic. Therefore in CML, we can achieve higher operating frequency with less power density.

A previous mixed signal superscalar processor was developed in 1997, using 0.5μm BiCMOS process and required 3.6V and 2.1V power supplies. The reported operating frequency was 533MHz and the design used a PowerPC architecture that contained three pipelines and a large on-chip secondary cache to achieve a peak performance of 1600 MIPS. The 15mm x 10 mm die contained 2.7M transistors (2M CMOS and 0.7M bipolar) and dissipated less than 85W [5]. All logic circuits were implemented in three-level emitter coupled logic (ECL) and only RAM structures were implemented with CMOS circuits.

Although BJT has less switching noise and are faster than MOS transistors they are expensive and not frequently used in digital circuits [1]. This led to the idea to design a high speed processor datapath using MCML logic. Further CMOS SRAM cannot operate

at 3.33GHz using 0.12μm technology. Therefore only high speed datapath was developed in this thesis.

1.3 Contribution

CML logic architecture has been discussed in the literatures but it is not guaranteed that it can realize digital functions unless optimization has been performed [6]-[8]. Optimized CML logic can steer full bias current at different input combinations, resulting in full voltage swing that differentiate logic states. Also, technology files provide only transistors, unlike digital technology files that provides optimized CMOS logic gates. Therefore, due to lack of analog synthesizer tools handcrafting of CML logic architecture is necessary and then optimization is required for CML logic to realize digital function for a targeted frequency. All the basic components have been derived first, with maximum operating frequency 12GHz using 130nm CMOS technology. Bias current was chosen to be 70% of peak f_t current that gives us highest possible operating frequency without burning transistors when operate. Further, biasing CML gates with 70% of peak f_t or less may incur 10% propagation delay but can save more than 40% power [1] and [7]. Logic swing was determined to be 600mV, assuming it will be advantageous than CMOS logic at this frequency that has fixed swing 1V. These basic CML logic designs were later used in realizing 16-bit 3.33GHz datapath components. All the processor components - mux, register file, ALU were deliberately handcrafted in Cadence Virtuoso. Reported static power consumption of the multi-cycle CML processor is 41.264W and power density per unit area is $5.44\mu\text{W}/\mu m^2 = 544\text{W}/cm^2$, below traditional CMOS processor power density per unit area, as indicated in Figure 1.3. Estimated performance of the multi cycle CML processor datapath is 892 MIPS and expected chip area is 2.2mm x 3.45mm.

This work has been accepted at IEEE International Symposium On Circuits and Systems (ISCAS) 2012 Conference [9].

1.4 Organization

The thesis has been organized as follows: Chapter 2 provides background and describes high speed CML logic realization. Chapter 3 introduces datapath design. Chapter 4 describes CML datapath component realization and verification. Chapter 5 represents processor verification, performance and comparison. Chapter 6 draws conclusions and discusses future work.

Chapter 2

Background and High Speed CML Logic Realization

Recently, interest in high speed digital circuit baseds on MCML/BJT-CML is increasing due to low power consumption [10]. Noise coupling between digital circuitry and sensitive analog blocks has always been a major obstacle in complete system on chip design (SoC) [11]. MOS current mode logic (MCML) is a promising alternative to conventional MOS in mixed signal applications. Many efforts were exhausted to realize the potential of MCML [12]-[17]. Even though MCML has been shown to dissipate less power than CMOS at operation frequencies of more than 300MHz, designers have been reluctant to exchange MCML for CMOS [14]. The high complexity of MCML and the lack of automation tools made it impossible to produce robust and power-efficient designs while maintaining low cost and reasonable time to market.

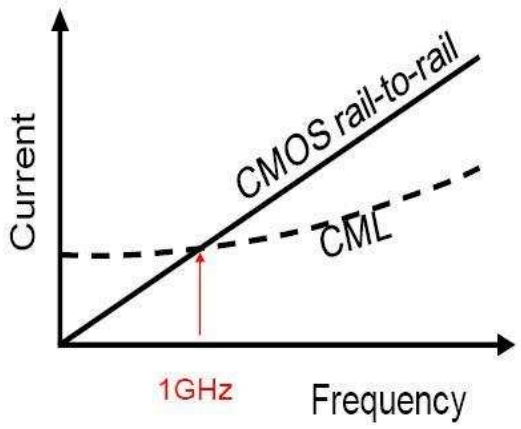

Figure 2.1: Current consumptions for CMOS vs. CML logic

Figure 2.1 shows typical current consumption for CMOS and CML logics [18]. As indicated, typically at slower frequency CMOS is beneficial whereas CML takes less power

at higher operating frequency. Figure 2.1 is based on assumption, and not according to circuit measurements, and is not shown in scale. It is supposed that typically beyond 1GHz CML is beneficial than CMOS.

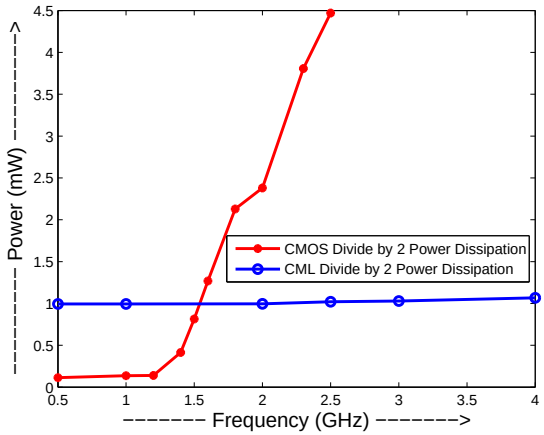

Figure 2.2: CMOS vs. CML power consumption

A divide by two circuits was realized in CMOS (W = 160nm, 160nm, 200nm, 600nm, 1m, 1.5m, 2.5m, 3m, 7m and 10m for 0.5, 1, 1.2, 1.4, 1.5, 1.6, 1.8, 2, 2.3 and 2.5GHz respectively) and CML (W = 160nm, 160nm, 160nm, 200nm, 250nm, 300nm each having W_{tail} = 120nm for 0.5, 1, 2, 2.5, 3 and 4GHz respectively) architecture. Channel length, L = 120nm was fixed for both cases and power has been plotted in Figure 2.2. At 2.5GHz CMOS D filp-flop propagation delay reaches 50% of input clock cycle whereas CML D flip-flop was able to generate correct output till 6GHz. It is obvious that CML dominates over CMOS beyond 1.5GHz. Benefits of MCML circuit topology over CMOS are largely independent of technology [6].

Many successful attempts have been made to expose the relationships between the MCML gate delay and the various design parameters [19]-[21]. These efforts have provided insight into the design considerations and have been described briefly for CML inverters and universal CML logic. There is no straight forward method to optimize CML logic and modeling accurate propagation delay for CML logic with pen and paper is very hard to

derive due to higher order effects. Therefore, we will rely on some approximation in the later subsections and compare with the simulation results in designing high speed CML gates.

2.1 CML Inverter

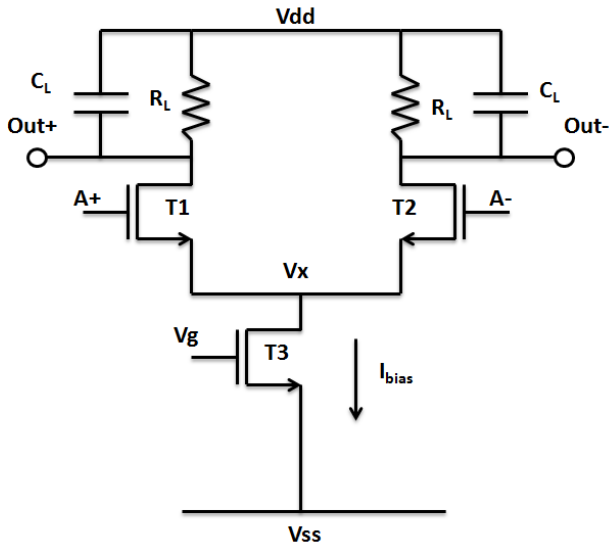

Figure 2.3: CML Inverter

Figure 2.3 shows a CML Inverter which is typically a differential pair. There exists a particular biasing voltage V_{in} (quiescent point), for which $I_{d1} = I_{d2} = I_{bias}/2$. For differential input, which is our logic swing ΔV, it is necessary to tune the circuit such that when logic-1 (V_{in}+ ΔV/2) is applied to A+ and logic-0 (V_{in}-ΔV/2) is applied to A-, T1 turns on and T2 turns off resulting in I_{d1}=I_{bias}, I_{d2}=0; steering all the current through T1. A Summary of CML inverter operation has been given in Table 2.1

A+	A-	T on	T off	Out+	Out-
V_{in}-ΔV/2	V_{in}+ΔV/2	2, 3	1	V_{in}+ΔV/2	V_{in}-ΔV/2
V_{in}+ΔV/2	V_{in}-ΔV/2	1, 3	2	V_{in}-ΔV/2	V_{in}+ΔV/2

Table 2.1: Inverter Operation

2.1.1 CML Inverter Optimization

If our supply voltage is V_{dd} as indicated in Figure 2.3, and the output of the CML gate drives another gate, then we must have to maintain:

$$\text{Logic-1} = V_{dd} = V_{in} + \Delta\text{V}/2$$
$$\text{and Logic-0} = V_{dd}\text{-}\ \Delta\text{V} = V_{in}\text{-}\ \Delta\text{V}/2$$

Let us assume logic-1 (V_{in}+ΔV/2) is applied to A+ and logic-0 (V_{in}-ΔV/2) is applied to A-. Therefore, to keep T1 in its forward-active region and T2 in its cut-off region, necessary conditions are:

$$V_{gs1} > V_{th1};\ V_{gd1} < V_{th1};\ V_{ds1} > V_{gs1} - V_{th1}$$
$$V_{gs2} < V_{th2};\ V_{gd2} < V_{th2}$$

If we assume that the magnitude of the voltage swing is just large enough to steer all the current from one side to the other then $\Delta\text{V} = \Delta V_{min}$, which is the minimum swing required (just large enough to turn off T2),

$$\begin{aligned}
V_{gs2} &= V_{th2} \\
V_{g2} - V_x &= V_{th2} \\
V_{in} - \Delta V_{min}/2 - V_x &= V_{th2} \\
V_{dd} - \Delta V_{min}/2 - \Delta V_{min}/2 - V_x &= V_{th2} \\
V_x &= V_{dd} - \Delta V_{min} - V_{th2} \qquad (2.1) \\
V_{gs1} &= V_{g1} - V_x \\
V_{gs1} &= V_{in} + \Delta V_{min}/2 - V_x \\
V_{gs1} &= V_{dd} - V_x \qquad (2.2) \\
I_{bias} &= (\mu_n C_{ox}/2)(W/L)(V_{gs1} - V_{th1})^{\alpha} \qquad (2.3)
\end{aligned}$$

where μ_n is electron mobility in nMOS device and C_{ox} is capacitance per unit area of gate oxide, C_{ox}=ϵ_{ox}/t_{ox}. Permittivity of SiO_2, $\epsilon_{ox} = 3.9\epsilon_0$; where permittivity of free space $\epsilon_0 = 8.85{\cdot}10^{-14}$F/cm and t_{ox} is thickness of gate oxide. For a short channel device, like 0.12μm feature size $\alpha \simeq 1.25$. For easiness of our pen and paper calcuation, sometimes we will assume either α is 2.

For matching purposes, T1 and T2 have to be same feature size such that an equal amount of current flows on both sides at the quiescent point. Therefore equation 2.1 yields

$$\begin{aligned} V_{th1} &= V_{dd} - \Delta V_{min} - V_x \\ \text{and, } V_{gs1} - V_{th1} &= V_{dd} - V_x - V_{dd} + \Delta V_{min} + V_x \\ &= \Delta V_{min} \qquad (2.4) \\ I_{bias} &= (\mu_n C_{ox}/2)(W/L)(\Delta V_{min})^{\alpha} \qquad (2.5) \end{aligned}$$

Therefore, I_{bias} can be realized as a function of minimum logic swing ΔV_{min}, with I_{bias} $\propto \Delta V_{min}$. It means, the less logic swing we define, the less bias current we need to realize a CML inverter. The minimum amount of current that will be required to realize this logic swing occurs when W is minimum. In other words, we can realize this full swing at high bias current which is true but we will burn more power. The minimum amount of W_n that can provide just large enough (smallest) bias current to realize full swing is:

$$I_L = (\mu_n C_{ox}/2)(W_n/L)(\Delta V_{min})^{\alpha} \qquad (2.6)$$

Again, let us consider a CML inverter circuit at its quiescent point, as indicated in Figure 2.3. In determining our minimum logic swing ΔV_{min} we need to consider how we are going to forward bias the transistors, meaning what our overdrive should be. Typically, for overdrive votage, $V_{ov} \geq$ 300mV, the transistor reaches soft saturation and when $V_{ov} \geq$ 500mV, the transistor reaches hard saturation. Overdrive voltage is referred to as $V_{ov} = V_{gs}$

- V_t. It is necessary to determine overdrive voltage because, if we apply logic-1 at A+ and logic-0 at A- and try to turn off T2 but if T2 at quiescent point reached velocity saturation then (ΔV)/2 is small enough to completely turn off T2. The consequence will be we cannot realize full swing. Making our logic swing large can turn T2 off but we will end up in greater RC constant and high propagation delay. Therefore, we need overdrive voltage such that it can push the transistor slightly in to the forward active region. A careful consideration was made such that $V_{ov} = V_{gs}$ - V_t = 263.9mV. Therefore it is necessary to expose the relation between logic swing and overdrive voltage. For an input voltage Vg1 at gate terminal of T1,

$$\begin{aligned} V_{gs1} &= Vg1 - V_x \\ V_{g1} &= V_{gs1} + V_x \\ \text{and } V_g2 &= V_{gs2} + V_x \end{aligned}$$

For differential input voltage, which is our logic swing,

$$\begin{aligned} \Delta V &= V_{g1} - V_{g2} \\ \Delta V &= V_{gs1} - V_{gs2} \\ I_{d1} &= (1/2)\mu C_{ox}(W/L)(V_{gs1} - V_t)^2 \ \alpha\text{=2 for simplicity} \\ I_{d2} &= (1/2)\mu C_{ox}(W/L)(V_{gs2} - V_t)^2 \\ \sqrt{Id1} - \sqrt{Id2} &= \Delta V\sqrt{(1/2)(\mu C_{ox}(W/L))} \\ I_{d1} + I_{d2} - 2\sqrt{Id1.Id2} &= (1/2)(\mu C_{ox}(W/L))\Delta V^2 \\ I_{bias} - 2\sqrt{Id1.Id2} &= (1/2)(\mu C_{ox}(W/L))\Delta V^2 \\ 2\sqrt{Id1.Id2} &= I_{bias} - (1/2)(\mu C_{ox}(W/L))\Delta V^2 \\ 2\sqrt{Id1.(Ibias - Id1)} &= I_{bias} - (1/2)(\mu C_{ox}(W/L))\Delta V^2 \\ -4I_{d1}^2 + 4I_{d1}.I_{bias} &= I_{bias}^2 - I_{bias}(\mu C_{ox}(W/L))\Delta V^2 + (1/4)(\mu C_{ox}(W/L))^2\Delta V^4 \end{aligned} \tag{2.7}$$

$$-4I_{d1}^2 + 4I_{d1} \cdot I_{bias} \;=\; I_{bias}^2 - I_{bias} \cdot K\Delta V^2 + (1/4)K^2\Delta V^4 \text{ K=}\mu\text{Cox(W/L)}$$

$$4I_{d1}^2 - 4Id1 \cdot I_{bias} + I_{bias}^2 - I_{bias}K\Delta V^2 + (1/4)K^2\Delta V^4 \;=\; 0$$

$$\begin{aligned}
I_{d1} &= [4I_{bias} \pm \sqrt{(16I_{bias}^2 - 16I_{bias}^2 + 16I_{bias}K\Delta V^2 - 4K^2\Delta V^4)}]/8 \\
I_{d1} &= [2I_{bias} \pm \Delta V\sqrt{(4I_{bias}K - K^2\Delta V^2)}]/4 \\
I_{d1} &= I_{bias}/2 \pm (1/4)\Delta V 2\sqrt{(I_{bias}.K)}\sqrt{[1 - K\Delta V^2/4I_{bias}]} \\
I_{d1} &= I_{bias}/2 \pm (\Delta V/2)\sqrt{(I_{bias}.K)}\sqrt{[1 - (\Delta V/2)^2/(I_{bias}/K)]}
\end{aligned}$$

As Logic 1 was applied to A+ and Logic-0 at A-, then differential input Vid > 0 resulting Id1 $>$ Id2.

$$I_{d1} = I_{bias}/2 + (\Delta V/2)\sqrt{(I_{bias}.K)}\sqrt{[1 - (\Delta V/2)^2/(I_{bias}/K)]} \tag{2.8}$$

$$I_{d2} = I_{bias}/2 - (\Delta V/2)\sqrt{(I_{bias}.K)}\sqrt{[1 - (\Delta V/2)^2/(I_{bias}/K)]} \tag{2.9}$$

At the biasing (quiescent) point, $V_{id} = 0$, resulting $I_{d1} = I_{d2} = I_{bias}/2$. Therefore equation 2.7 yields

$$\begin{aligned}
I_{bias}/2 &= (1/2)\mu C_{ox}(W/L)(V_{gs1} - V_t)^2 \\
I_{bias}/2 &= (K/2)V_{ov}^2 \\
K &= I_{bias}/V_{ov}^2
\end{aligned}$$

Plugging in the value of K in equation 2.8 we get

$$I_{d1} = I_{bias}/2 + (\Delta V/2)(I_{bias}/V_{ov})\sqrt{[1 - ((\Delta V/2)/V_{ov})^2]} \tag{2.10}$$

$$I_{d2} = I_{bias}/2 - (\Delta V/2)(I_{bias}/V_{ov})\sqrt{[1 - ((\Delta V/2)/V_{ov})^2]} \tag{2.11}$$

Now if we want to steer full current through T1, resulting $I_{d1} = I_{bias}$ and $I_{d2} = 0$, then equation 2.10 yields

$$\begin{aligned}
I_{bias} &= I_{bias}/2 + (\Delta V/2)(I_{bias}/V_{ov})\sqrt{[1 - ((\Delta V/2)/V_{ov})^2]} \\
I_{bias}/2 &= (\Delta V/2)(I_{bias}/V_{ov})\sqrt{[1 - ((\Delta V/2)/V_{ov})^2]} \\
1 &= \frac{\Delta V}{V_{ov}} \cdot \sqrt{1 - (\frac{\Delta V}{4V_{ov}})^2} \\
&= (\frac{\Delta V}{V_{ov}})^2 - \frac{1}{4} \cdot (\frac{\Delta V}{V_{ov}})^4 \\
&= \left(\frac{\Delta V}{V_{ov}}\right)^2 \left(1 - \frac{\Delta V^2}{4V_{ov}^2}\right) \\
\left(\frac{V_{ov}}{\Delta V}\right)^2 &= \frac{4V_{ov}^2 - \Delta V^2}{4V_{ov}^2} \\
4V_{ov}^4 &= 4V_{ov}^2 \cdot \Delta V^2 - \Delta V^4 \\
\left(2V_{ov}^2 - \Delta V^2\right)^2 &= 0 \\
\Delta V &= \pm\sqrt{2}V_{ov} \qquad (2.12)
\end{aligned}$$

It means for $\Delta V = +\sqrt{2}V_{ov}$ swing either one of the transistor will turn on pushing other one to be off and at $\Delta V = -\sqrt{2}V_{ov}$ it will be reversed, as indicated in Figure 2.4.

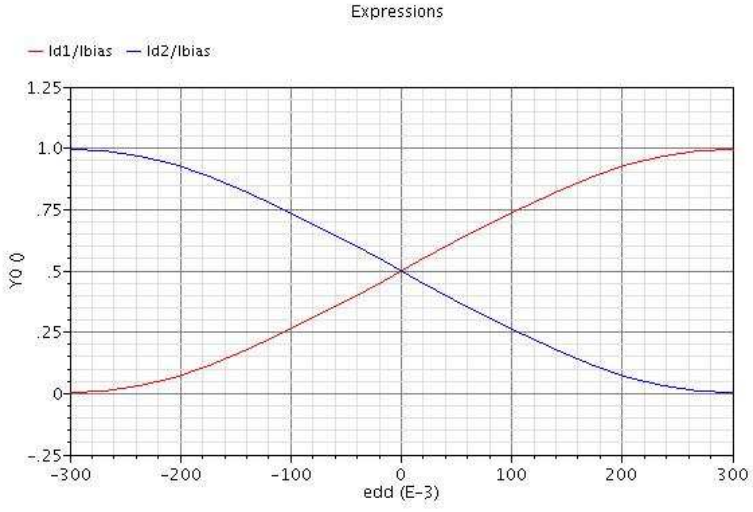

Figure 2.4: Normalized current for CML inverter

At the biasing point shown earlier, V_{ov} = 263.9mV, therefore our minimum logic swing from the biasing point will be $\Delta V/2=\sqrt{2}\cdot 263.9mV = \pm 373.21mV$. In Figure 2.4 it is shown, at +300mV swing from the biasing point, T1 passes all the bias current and at -300mV from the biasing point, T2 completely turns on resulting in a total logic swing ΔV=600mV. 73mV discrepancy occurs because it is a short channel device and α is not 2, but rather close to 1.25. Equation 2.12 can further be extended by plugging V_{ov} in terms of bias current and W.

$$\begin{aligned} \Delta V &= \pm\sqrt{2}V_{ov} \\ \Delta V &= \pm\sqrt{2}\sqrt{\frac{I_{bias}}{K}} \\ \Delta V &= \pm\sqrt{\frac{2I_{bias}}{\mu C_{ox}\frac{W}{L}}} \end{aligned} \tag{2.13}$$

It means the more W/L ratio we have, the faster we can move in switching region. But making W larger will increase parasitic capacitance at higher operating frequency and the RC constant will be dominating.

In order to achieve faster full swing, $\Delta V = I_{bias} * R_L$, we can increase bias current to reduce R_L. Bias current was tuned to 1.65mA for W=7μm and L=120nm, which is 70% of peak f_t current, and load resistance R_L was tuned to 348Ω.

Input gate capacitance of each CML basic component is, C_{gg} = 8fF. It is stated that wire capacitance is 0.10ps/μm [22]. Assuming 100μm is required to connect next stage then 10ps delay should be added and typically 1fF is necessary to mimic 1ps delay. Therefore 10+8 = 18fF was added as output load capacitance (C_L).

Propagation delay, P_d = 0.69RC, where C is the capacitance looking in to Drain of T1/T2.

$$\begin{aligned} P_d &= 0.69R_L(C_{intr} + C_L) \\ P_d &= 0.69R_L(C_{gd1} + C_{db1} + C_L) \end{aligned}$$

$$= \quad 0.69\frac{\Delta V}{I_{bias}}(C_{gd1} + C_{db1} + C_L) \tag{2.14}$$

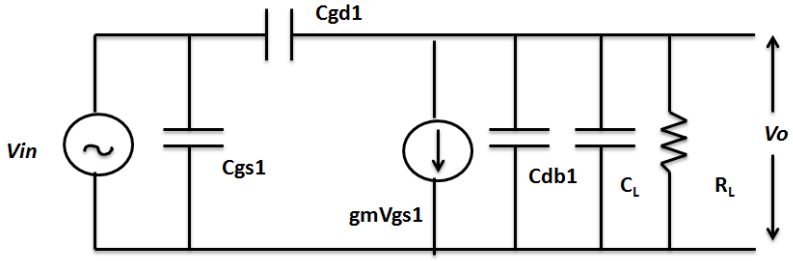

Figure 2.5: CML inverter half-circuit small-signal model

Where C_{intr} is intrinsic capacitance which equals the gate to drain (C_{gd1}) plus drain to bulk (C_{db1}) capacitance as shown in Figure 2.5. Equation 2.14 means, the higher the bias current, the lower the propagation delay. But, bias can be increased by increasing W, and at higher frequency further increasing bias current will not reduce propagation delay and parasitic capacitance will dominate. Hence, optimized width obtained for the upper level transistors are 7μm and 4μm width for the current source, all having channel length = 120nm with bias current, I_{bias}=1.65mA.

All these intuitive design considerations were included in designing CML logic circuits that can achieve maximum operating frequency of 12GHz with 0.12μm feature size. Supply voltage used was V_{dd}=2.8V, logic-1 = 2.8V, logic-0 = 2.2V, constant biasing voltage, V_g = 1V to bias the current source and logic swing ΔV = 600mV.

Figure 2.6 shows post vs. pre layout simulation of a CML inverter operating at 12GHz (input changes at 83ps) with 18fF input/output load capacitance. Rising delay of inverter in circuit level simulation is 17.47ps and extracted layout delay is 24.79ps. The power consumption of the CML inverter is P = V_{dd}*I_{bias} = 2.8V * 1.65mA = 4.62mW, which is constant. Operating this optimized CML inverter at 1MHz and 12GHz will dissipate the same amount of power. Therefore power consumption is independent of operating frequency. Figure 2.6 also shows layout simulation varies with circuit level simulation by an amount of

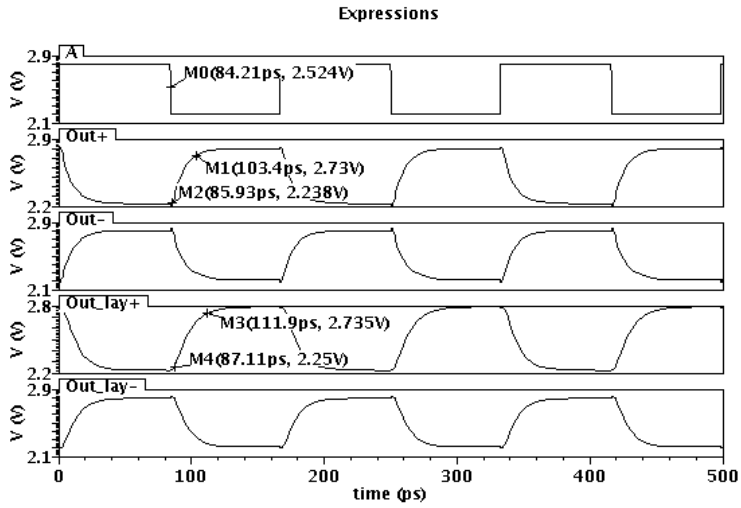

Figure 2.6: Post vs. pre layout simulation of CML inverter with 18fF input/output load capacitance (input changes at 83ps)

7.32ps. Layout of the CML inverter was also performed and the reported area is 15.960μm x 24.450μm, as indicated in Figure 2.7.

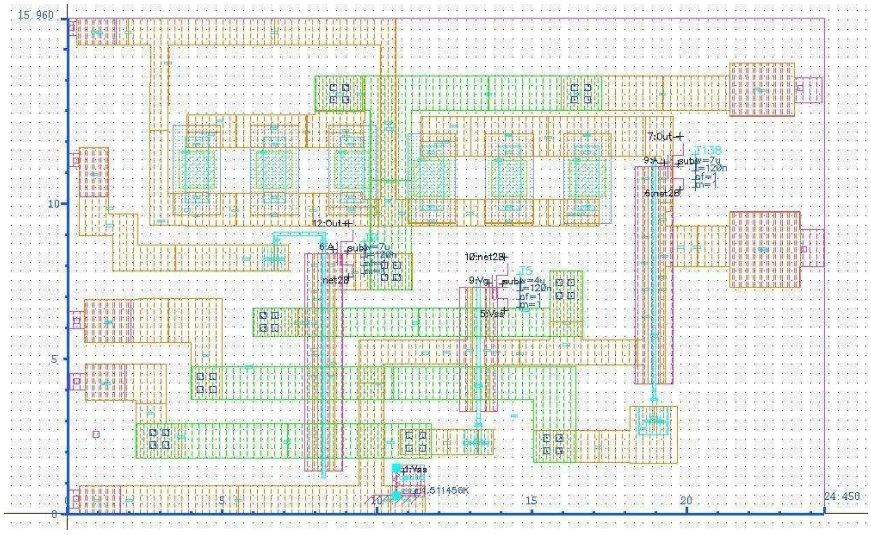

Figure 2.7: CML inverter layout

2.2 CML Universal Gate

A universal CML gate is represented in Figure 2.8 that can realize AND, NAND, OR and NOR functions. According to input logic combination as indicated in Figure 2.8, the

universal gate can realize an AND function. Reversing the output will realize a NAND function. Reversing all the inputs and outputs will realize an OR function and thus a NOR can be realized as well.

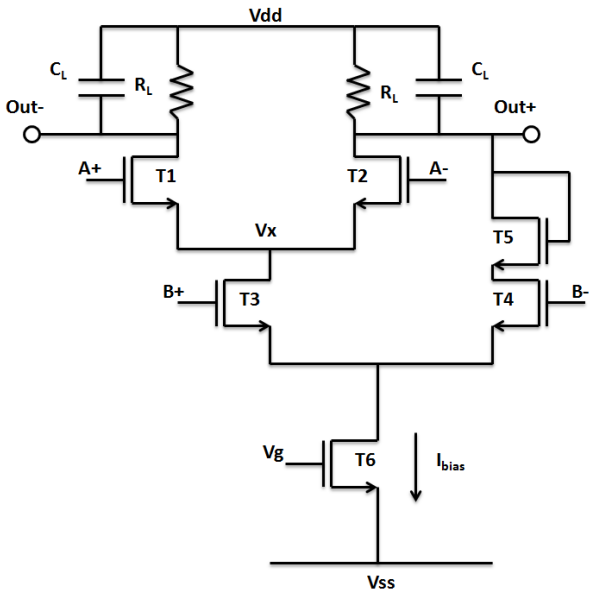

Figure 2.8: Universal CML gate

Careful consideration was made to choose V_{in}=2.5V and ΔV=600mV, leading Logic-1=2.8V and Logic-0=2.5V as shown earlier in CML inverter section. The biasing conditions are same as before. If we assume Logic-1 has applied in A+ and Logic-0 has been applied in A- then,

$$V_{gs1} > V_{th1};\ V_{gd1} < V_{th1};\ V_{ds1} > V_{gs1} - V_{th1}$$
$$V_{gs2} < V_{th2};\ V_{gd2} < V_{th2}$$

For input B, levels have to be at least V_{DS} amount down such that the condition $V_{GD3} < V_{th}$ and $V_{GD4} < V_{th}$ is met when T3 and T4 operate (are in the forward active region). Therefore logic-1 for input B is V_{in} + ΔV -V_{DS} and Logic-0 is V_{in} - ΔV - V_{DS}.

Due to the discrepancy of logic levels for the second input, a level shifter is necessary and has been embedded with in each gate as indicated in Figure 2.9. The previous power

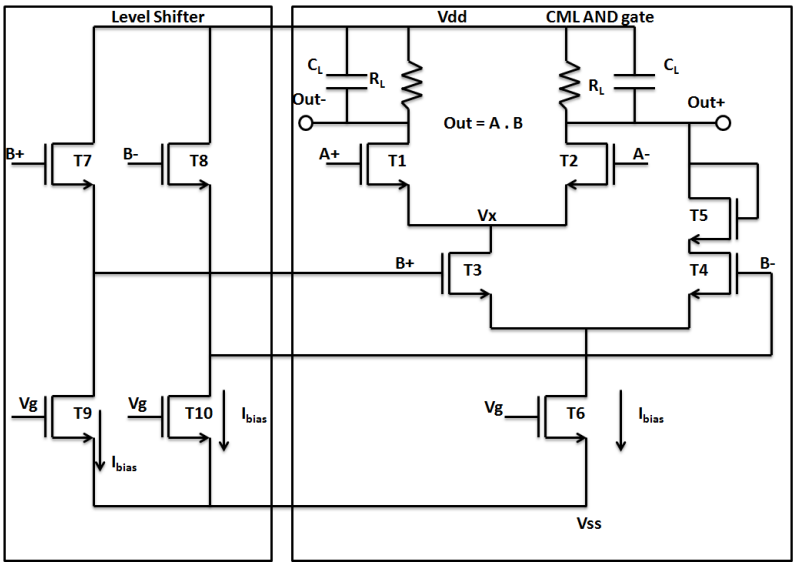

Figure 2.9: Universal CML gate with embedded level shifter

of the CML gate was V_{dd}*I_{bias} and now with the level shifter it is V_{dd}*$3I_{bias}$; total power increased by a factor of 3.

T7 and T8 have to be the same size as T1 and T2 for matching purposes, such that T7/T8 can mimic the voltage drop V_{DS} by T1/T2. To balance the differential architecture, T5 is necessary with T4; because T3 sees either T1 or T2 as a load and is also used for preventing breakdown. T6, T9 and T10 are of same size and constant biasing voltage (Vg) is applied to act as a current source.

There are three paths in the universal CML gate and it is necessary to make sure total bias current (I_{bias}) flows in any of these path from V_{dd} to V_{ss} depending on logic combination from. Splitting of I_{bias} is not acceptable because full swing realization will not be possible.

Table 2.2 shows the path that will be turned on, depending on input logic combination and realization of the CML AND function. It is notable that the entry at B+/B- is un-shifted logic. It is applied to input B at the level shifter and eventually it gets shifted down by an amount V_{DS} and propagates to input B of CML AND. It is indicated in Table 2.2 that lowest level (2^{nd} level, input B) dominates in path selection.

A+	A-	B+	B-	Out+	Out-	T off	T on	Path
V_{dd}-ΔV	V_{dd}	V_{dd}-ΔV	V_{dd}	V_{dd}-ΔV	V_{dd}	1, 2, 3	5, 4, 6	T5, T4, T6
V_{dd}-ΔV	V_{dd}	V_{dd}	V_{dd}-ΔV	V_{dd}-ΔV	V_{dd}	1, 4, 5	2, 3, 6	T2, T3, T6
V_{dd}	V_{dd}-ΔV	V_{dd}-ΔV	V_{dd}	V_{dd}-ΔV	V_{dd}	1, 2, 3	5, 4, 6	T5, T4, T6
V_{dd}	V_{dd}-ΔV	V_{dd}	V_{dd}-ΔV	V_{dd}	V_{dd}-ΔV	2, 4, 5	1, 3, 6	T1, T3, T6

Table 2.2: CML AND Operation

2.2.1 Universal CML Gate Optimization

When a path turns on, it is necessary to steer all the bias current (I_{bias}) to that path, as stated before, to realize full logic swing, $\Delta V = R_L * I_{bias}$. The more bias current we have, the faster the swing realized. Intuitive techniques to optimize the CML gate have been shown in the CML inverter optimization subsection. It was found, using 130nm CMOS technology, the highest operating frequency we can obtain for CML inverter is 12GHz with upper transistor W=7μm, L=120nm for bias current of 1.65mA for which current source size is W=4μm, L=120nm. As we cannot exceed this operating frequency for particular logic swing ΔV=600mV, supply voltage V_{dd}=2.8V, logic-1=2.8V, logic-0=2.2V parameters, the same size has been used in CML universal gate for upper transistors. But in this case, it has second level of input and here it contributes more parasitic capacitance, bias current has been increased very little from 1.65mA to 1.753mA (for which current source size is W=7μm, L=120nm) to operate at 12GHz.

For upper level (and 2^{nd} level) transistors, around V_{ov}=200mV has been provided such that a 300mV swing from the biasing point can turn off and on the transistors. It should be noted, at quiescent point $I_{r1} \neq I_{r2}$, as indicated in Figure 2.10. The reason is, Out+ has 2 paths at Q point but Out- has 1 path to V_{ss}. Therefore, typically $I_{r1} = \frac{1}{3} I_{r2}$, as it is observed in Figure 2.10. It is shown that I_{r1} = 517.5μA and I_{r2} = 1.238mA out of total bias current 1.753mA. Therefore, initially, Out+ tends to be close to logic-0 (2.2V), which is 2.371V and Out- tends to be close to logic-1 (2.8V), which 2.62V at Q point. As it can be seen in normalized current plot of universal CML gate in Figure 2.10 that $I_{d1} \neq I_{d2}$ at

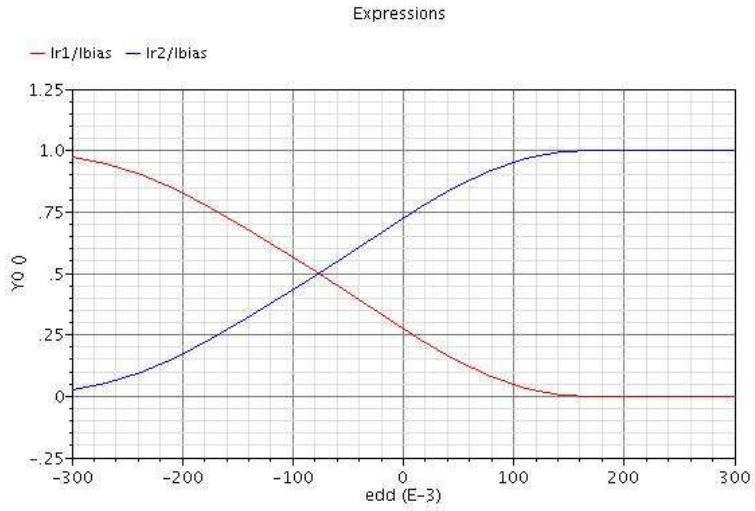

Figure 2.10: Normalized current for CML universal gate

biasing point. But full current steering was made possible at -300mV to +300mV swing, which is our objective.

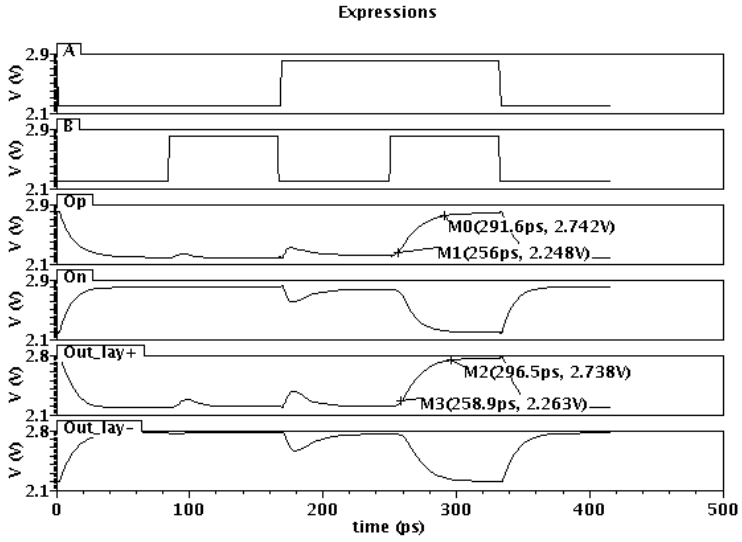

Figure 2.11: Post vs. pre layout simulation of CML AND with 18fF input/output load capacitance (input changes at 83ps)

Figure 2.11 shows post vs. pre layout simulation of the CML AND gate at 12GHz (input changes at 83ps). Load capacitance, (C_L) =18fF was added to mimic next stage input capacitance, C_{gg}=8fF plus 10fF to mimic 100μm wire for connecting next stage similar to the CML inverter [22]. The circuit level simulation shows rising delay is 35.6ps whereas

extracted layout simulation has rising delay of 37.6ps. Bias current I_{bias}=1.759mA and 2 other current source supply 2.019mA each, for a total power consumption of 2.8V * (1.753 + 2.019 + 2.019)mA = 16.2148mW.

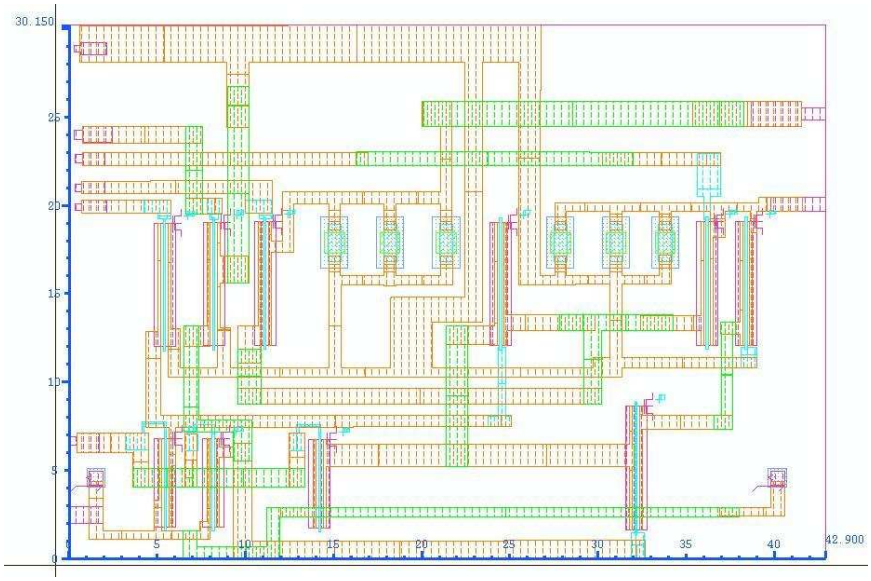

Figure 2.12: CML AND layout

The reported area of CML AND gate is 30.150μm x 42.900μm, as indicated in Figure 2.12 and this area should be the same for CML NAND/OR/NOR. Layouts of all these gates have been performed and have been simulated but not included because their architectures are not different from CML AND, just the reverse of inputs or outputs.

2.3 CML XOR/XNOR Gate

A CML XOR gate is shown in Figure 2.13 with an embedded level shifter. Table 2.3 briefly describes CML XOR operation and paths.

Logic-1 represented by V_{in}+ΔV/2 = 2.8V and logic-0 is V_{in}-ΔV/2 = 2.2V, where V_{in} = 2.5V is the biasing (quiescent) voltage and ΔV=600mV is the logic swing. Bias current I_{bias}=1.757mA and two other current sources supply 2.019mA each, having total power consumption of 2.8V * (1.757+2.019+2.019)mA = 16.226mW. Power consumption of the

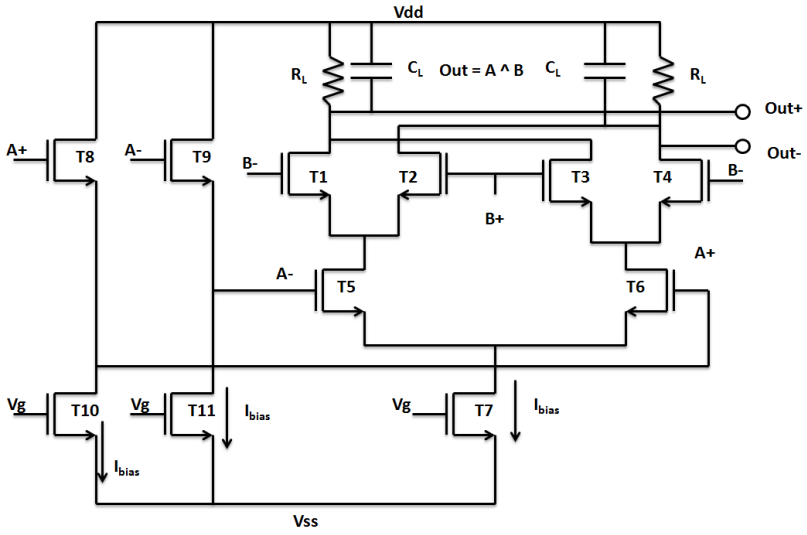

Figure 2.13: CML XOR gate

CML XOR is even less than the CML AND gate, which is impractical in CMOS realization but possible in CML architecture.

A+	A-	B+	B-	Out+	Out-	T on	T off
0	1	0	1	0	1	7, 5, 1	2, 3, 4, 6
0	1	1	0	1	0	7, 5, 2	1, 3, 4, 6
1	0	0	1	1	0	7, 6, 4	1, 2, 3, 4
1	0	1	0	0	1	7, 6, 3	1, 2, 4, 5

Table 2.3: CML XOR Operation

For the 00 combination, T1, T5 and T7 will turn on and I_{bias} will flow down to Vss, resulting in all other paths being turned off. For the 01 combination, either T2 or T3 will turn on; depending on A. As A+ is 0 and A- is 1, T5 will be on and T6 will be off, and the path will be T2, T5 and T7.

As the lowest level dominates the path, for the 10 combination T6 will be on, and as B+ is 0 and B-=1, T4 will turn on and the path will be T4, T6 and T7. For 11 combinations, T3, T6 and T7 will be turned on.

Figure 2.14 shows post vs. pre layout simulation of CML XOR gate with 18fF input/output load capacitance at 12GHz (input changes at 83ps). It is observed that circuit

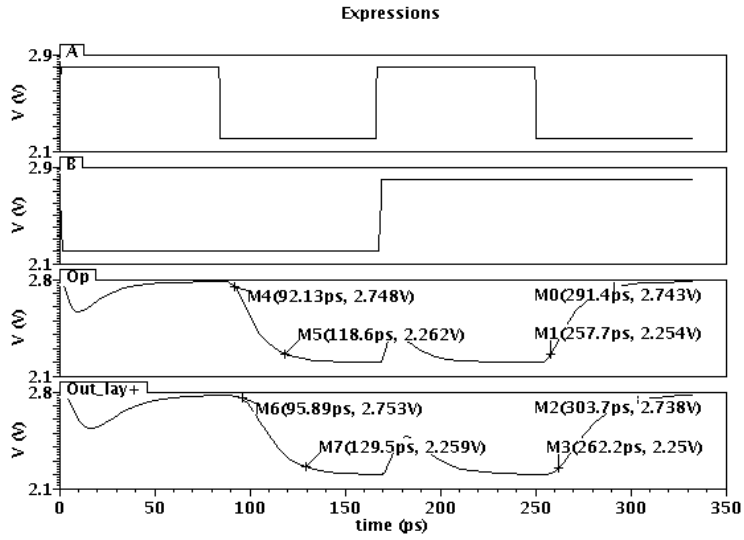

Figure 2.14: Post vs. pre layout simulation of CML XOR with 18fF input/output load capacitance (input changes at 83ps)

level rise delay is 33.7ps whereas extracted layout delay is 41.5ps. Therefore, 7.8ps discrepancy exists in between circuit level and device level simulation.

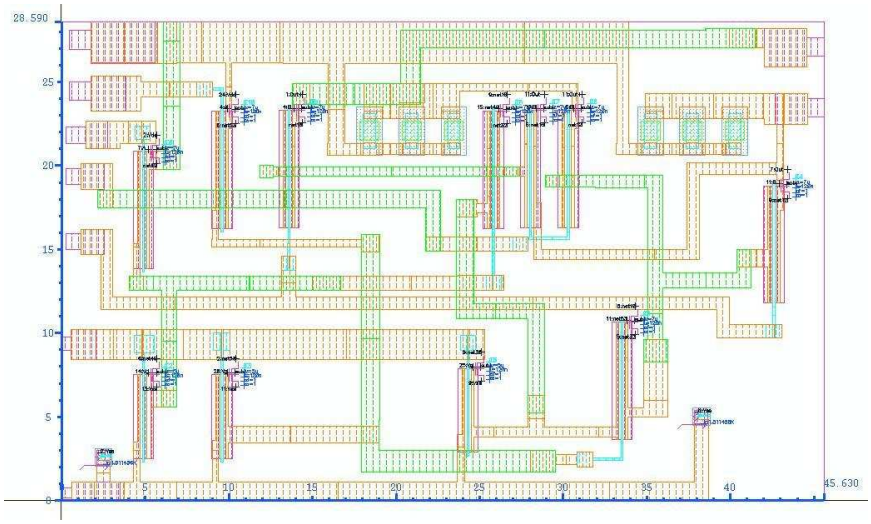

Figure 2.15: CML XOR layout

Reported area of CML XOR is 28.590μm x 45.630μm as indicated in Figure 2.15.

2.4 CML Mux Realization

A CML mux has been realized at the transistor level as indicated in Figure 2.16. The architecture is very similar to the XOR and the power dissipation is less than the CML AND/XOR gate. Realizing a CML mux with transistors reduces power at least $\frac{1}{3}$ over realizing the mux by cascading CML AND gates. Realizing mux with transistor has two fold benefits; one is less power and the other one is delay reduced by $\frac{1}{2}$. Table 2.4 describes CML mux operation.

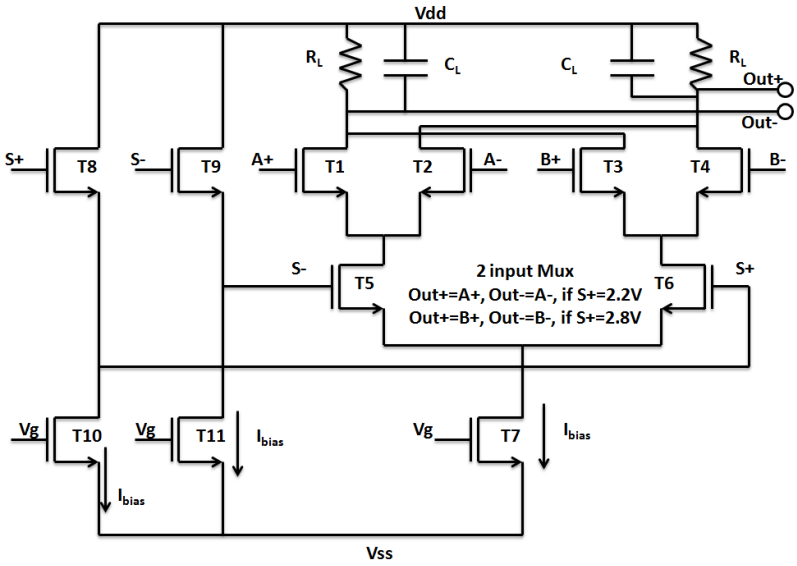

Figure 2.16: CML Mux realization

B+ (V)	A+(V)	S+(V)	Out+(V)	T on	T off
2.2	2.2	2.2	2.2	5, 2	6, 1, 3, 4
2.2	2.8	2.2	2.8	5, 1	6, 2, 3, 4
2.8	2.2	2.2	2.2	5, 2	6, 1, 3, 4
2.8	2.8	2.2	2.8	5, 1	6, 2, 3, 4
2.2	2.2	2.8	2.2	6, 4	5, 1, 2, 3
2.2	2.8	2.8	2.2	6, 4	5, 2, 2, 3
2.8	2.2	2.8	2.8	6, 5	4, 1, 2, 3
2.8	2.8	2.8	2.8	6, 5	4, 2, 2, 3

Table 2.4: CML Mux Operation

Figure 2.17 shows post vs. pre layout simulation of CML mux with 18fF input/output load capacitance at 12GHz (input changes at 83ps). It is observed that circuit level rise delay is 28.01ps whereas extracted layout delay is 33.99ps. Therefore, 5.98ps discrepancy exists in between circuit level and device level simulation.

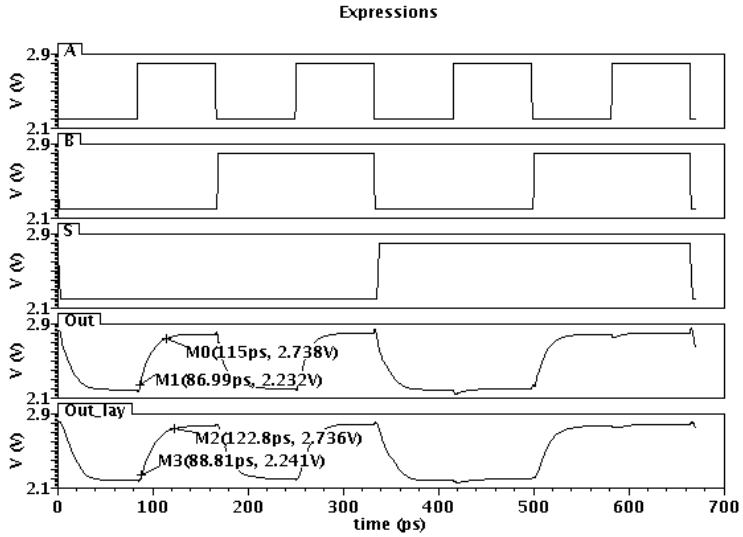

Figure 2.17: Post vs. pre layout simulation of CML Mux with 18fF input/output load capacitance (input changes at 83ps)

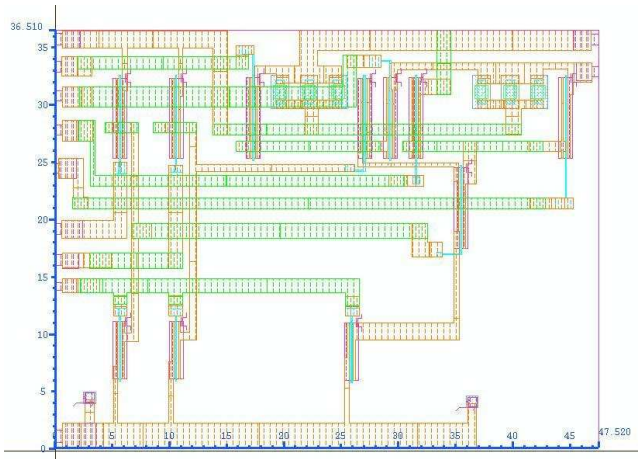

Figure 2.18: CML Mux layout

The reported area for the CML Mux is 36.510μm x 47.520μm, as indicated in Figure 2.18. Total power consumption of the CML Mux is 2.8V * (2.019 + 2.019 + 1.757)mA = 16.226mW.

2.5 CML D-latch Realization

A CML D-latch has been realized in transistors, as indicated in Figure 2.20, with four levels of hierarchy with a reset input, not by cascading CML AND gates, in contrast to CMOS architecture. This architecture not only reduces power consumption but also reduces

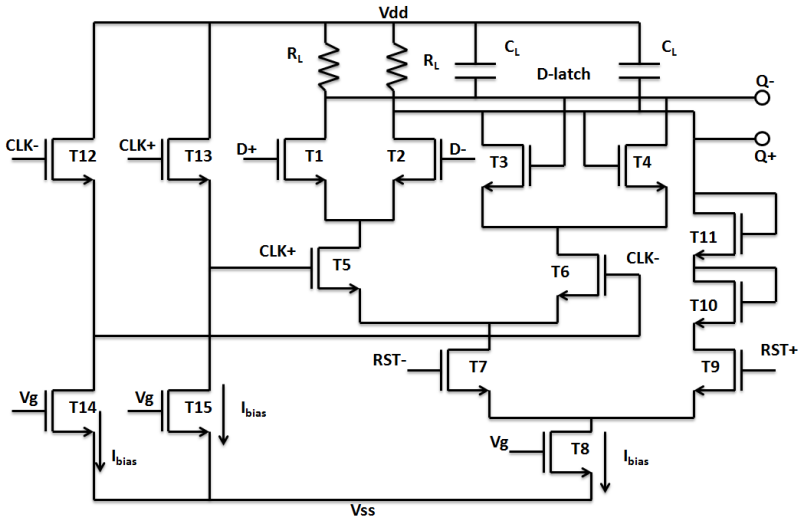

Figure 2.19: CML D-latch

delay. Very surprisingly power consumption is less than the CML AND gate, which cannot be realized in CMOS architecture. It is notable that, the lowest level dominates in selecting upper level paths. RST+=1.3V (RST-=0.7V) turns on T9 (T7 off) and ties Q+ to Vss, irrespective of any input combination.

CLK+ (V)	D+(V)	RST+(V)	Q_{t+1}+(V)	T on	T off
2.2	2.2	2.2	Q_t+	8, 7, 6, (3 / 4)	9, 5, (4 / 3), 1, 2, 10, 11
2.2	2.8	2.2	Q_t+	8, 7, 6, (3 / 4)	9, 5, (4 / 3), 1, 2, 10, 11
2.8	2.2	2.2	2.2	8, 7, 5, 2	9, 6, 1, 3, 4, 10, 11
2.8	2.8	2.2	2.8	8, 7, 5, 1	9, 6, 2, 3, 4, 10, 11
x	x	2.8	2.2	8, 9, 10, 11	1, 2, 3, 4, 5, 6, 7, 8

Table 2.5: CML D-latch Operation

Table 2.5 briefly describes the operation of the CML D-latch. Initially Q+ tends to be logic-0 (the reason explained for the CML AND gate). If the clock is high, then whichever

data comes at D+ is passes to the output Q+. Also, the present state pushes lower level transistors to retain their logic states when clock is not high. Therefore at beginning, reset is not necessary in CML architecture.

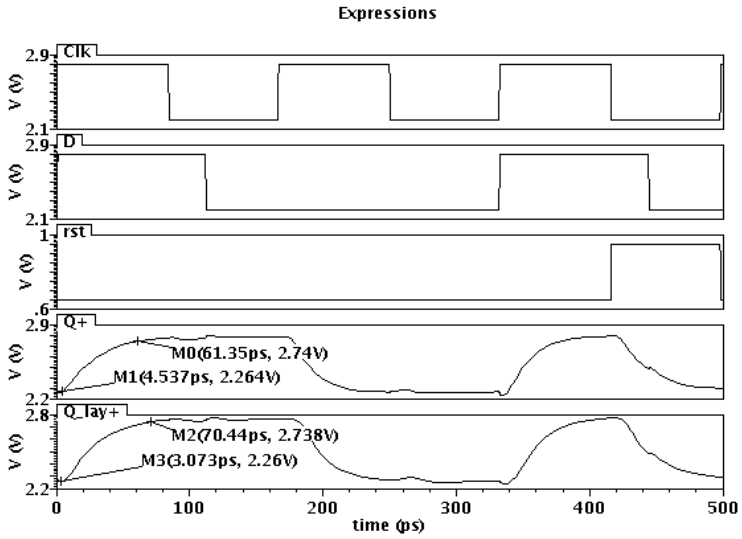

Figure 2.20: Post vs. pre layout simulation of CML D-latch with 18fF input/output load capacitance at 6GHz (166ps)

Figure 2.20 shows post vs. pre layout simulation of CML D-latch with 18fF input/output load capacitance at 6GHz. It is observed that circuit level rise delay is 56.813ps whereas extracted layout delay is 67.367ps. Therefore, 10.554ps discrepancy exists in between circuit level and device level simulation.

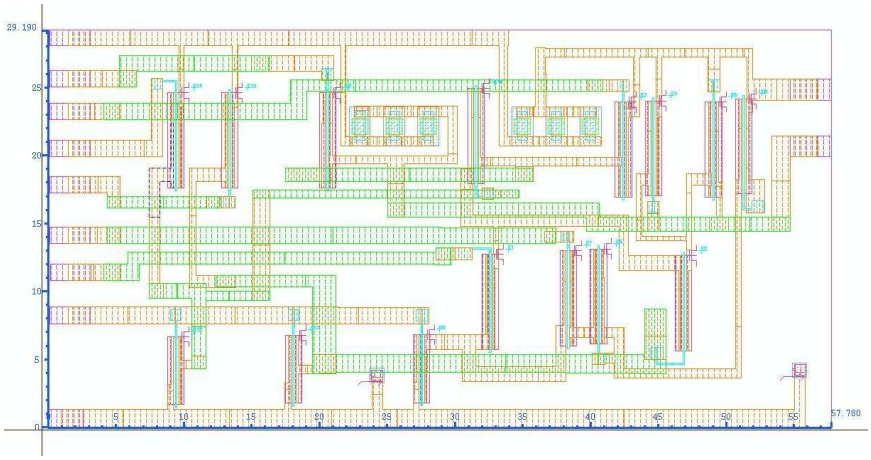

Figure 2.21: CML D-latch layout

Power consumption of the CML D-latch is 2.8V * (2.019 + 2.019 + 1.431)mA = 15.3132mW. Reported area for CML D-latch is 29.190μm x 57.780μm, as indicated in Figure 2.21.

2.6 Speed, Power, Area and Delay of Basic CML Components

Table 2.6 summarizes post simulation power, area and delay with 18fF load (10fF for wire 8fF for next stage C_{gg}) capacitance of each CML basic components. In later chapters, higher level circuit realization has been performed using these basic components to develop a CML processor datapath.. Therefore, in measuring theoretical worst case delay it has assumed that delay is additive, meaning if the output of a CML AND gate drives the input of a CML Mux, the total delay is the sum of the two. Total delay of any complex component in simulation is found to match the number of basic components in the critical path and assigning delay to each of them.

Component	Power (mW)	Area (μmxμm)	Delay (ps)
Inverter	4.62	15.96x24.45	24.79
AND/NAND	16.2148	30.15x42.90	37.6
OR/NOR	16.2148	30.15x42.90	46.09
XOR/XNOR	16.226	28.59x45.63	41.5
CML 2-to-1 Mux	16.226	36.51x47.52	33.99
CML D-latch	15.3132	29.19x57.78	67.367

Table 2.6: Power, Area and Delay of Basic Components (post layout simulation with 18fF load capacitance)

Layout of the processor datapath has not been performed but the circuit level simulation. It is observed that post vs. pre layout simulation differs maximum 10ps and wire delay is 10ps (for 100μm wire connecting next stage and assuming wire delay is 0.10ps/μm) [22]. Therefore 20fF load capacitance was added inside every CML logic in datapath simulation to mimic post layout + wire delay (in intermediate stages C_{gg} = 8fF has already been considered by simulation tool). Area was predicted by counting the number of gates * area of each gate + 50% area for wiring.

Chapter 3

Datapath

A multi-cycle 16-bit processor datapath has been designed using a RISC architecture that can execute 15 different operations and a 4-bit opcode has been used. Three different types of instructions can be performed and instruction set architecture is given in Table 3.1.

Address bus (16-bit)			
15-12	11-8	7-4	3-0
Opcode R-type	Rd	Rs	Rt
Opcode I-type	Rd	Rs	Immediate operand
Opcode J-type	Address		

Table 3.1: Instruction Set Architecture

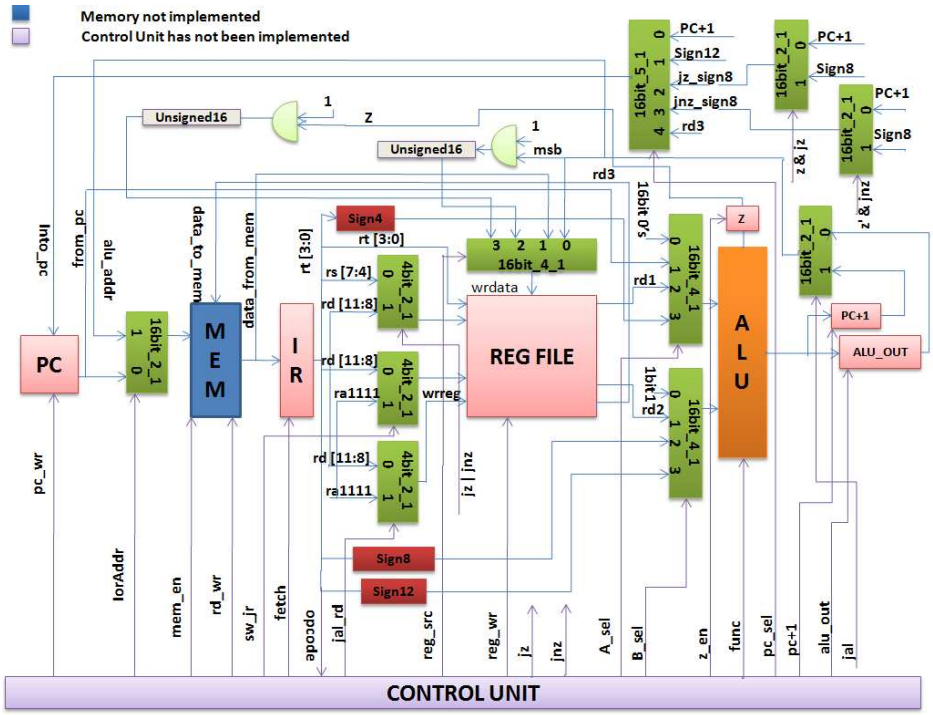

Figure 3.1: Processor datapath

The processor datapath is shown in Figure 3.1, where all the components have been realized in CML logic except the cache memory and control unit. Correct operation of the datapath has been verified by providing external control stimulus in every clock cycle and observing/providing data to/from memory. Fifteen different operations that can be performed are listed in Table 3.2. The Processor datapath does not contain any novel approach. However this datapath can be re-structured such that power consumption will be less and operating frequency can be increased.

Type	Opcode	Description	Cycle
R-type	0000 ADD	Rd=Rs+Rt	4
	0001 SUB	Rd=Rs-Rt	4
	0010 AND	Rd=Rs•Rt	4
	0011 XOR	Rd=Rs⊕Rt	4
	0100 SLT	Rd=1 if Rs<Rt else Rd=0	4
	0101 SEQ	Rd=1 if Rs=Rt else Rd=0	4
I-type	0110 LW	Rd=[Rs]+n; n=4bit	5
	0111 SW	[Rs+n]=Rd; n=4bit	4
	1000 ADDI	Rd=Rs+n; n=4bit	4
	1001 MOVI	Rd=n; n=8bit	4
J-type	1010 J LABEL	PC = LABEL; n=12bit	2
	1011 JZ	PC=LABEL if Rd=0; n=8bit	4
	1100 JNZ	PC=LABEL if Rd≠0; n=8bit	4
	1101 JAL	$RA=PC+1 &PC=LABEL; n=12bit	3
	1110 JR	PC=$RA ; least 12-bit unused	2

Table 3.2: Opcode and 15 Different Operations

The datapath contains 16-bit program counter register, 16-bit instruction register, 16x16 register file, 16-bit ALU, 16-bit pc+1 register and 16-bit ALU_out register. PC, IR, ALU_out, PC+1 are 16-bit registers with enable input that controls the write operation. The 16-bit ALU contains 16-bit CLA, BLA, AND and XOR. In the 16x16 register file, 4-bit rr1, rr2 and rr3 can address any of 16 register (16-bit each) and propagates the content of addressed register in 16-bit rd1, rd2 and rd3 through 3 16-bit 16-to-1 mux. 4-bit wrreg is used for addressing the write register and 1-bit reg_wr is used to enable the write operation. It is assumed, for mem_en=1 and rd_wr=0, that memory can be read and for mem_en=1 and

rd_wr=1 memory can perform write operation. It takes 2-5 cycles to perform any of the 15 instructions as listed in Table 3.2 with brief description.

In Table 3.2 RA is register-15 implemented in hardware. Also, register-0, which contains 0 is implemented in hardware as well. The critical path delay is 5 clock cycles for the LW instruction and the path is PC ⇒ IR ⇒ Reg_file ⇒ ALU_out ⇒ MEM ⇒ Reg_file.

3.1 First Clock Cycle of Every Instruction

The first clock cycle is same for any instruction and the selected path is shown in Figure 3.2, with control logic. As indicated in Figure 3.2, in the first cycle men_en = 1 and fetch = 1 and data comes into IR register, assuming that memory is fast enough to be read in one clock cycle. Also, at the same time, A_sel = 01, B_sel=00 selects the content of PC and 1-bit value 1 (from first input of 16bit_4_to_1 mux) and the ALU performs PC+1.

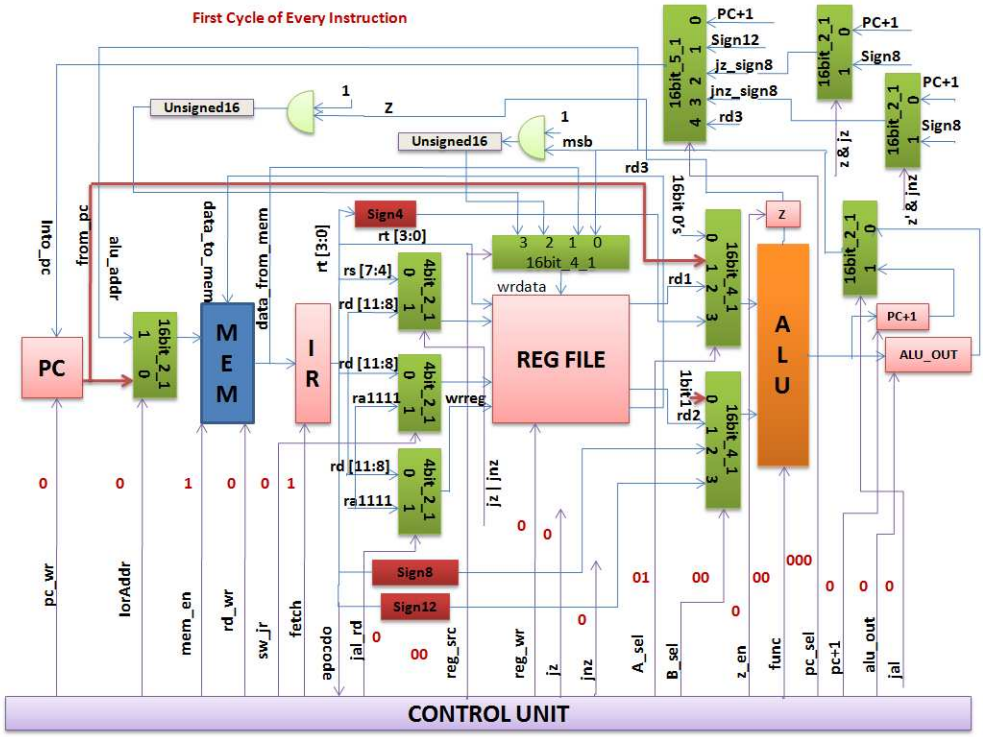

Figure 3.2: First clock cycle of any instruction

3.2 R-type Instruction

Six different type of instruction can be performed in R-type operation. They are ADD, SUB, AND, XOR, SLT and SEQ. ADD, SUB, AND, XOR instructions perform addition, subtraction, binary bitwise AND and binary bitwise XOR respectively. SLT stores $R_d = 1$ if $R_s < R_t$, else $R_d = 0$ (performs R_s - R_t and checks MSB). SEQ stores $R_d = 1$ if $R_s = R_t$, else $R_d = 0$ (performs R_s - R_t and checks Z flag). It takes four cycles to perform any R-type instruction.

3.2.1 R-type ADD/SUB/AND/XOR

In the second cycle of R-type ADD/SUB/AND/XOR instructions, A_sel = 10, B_sel = 01 selects rd1 and rd2 and pc+1 = 1 signal enables PC+1 to be saved in the PC+1 register. Control signal func = 00, 01, 10 and 11 at second cycle performs ADD, SUB, AND and

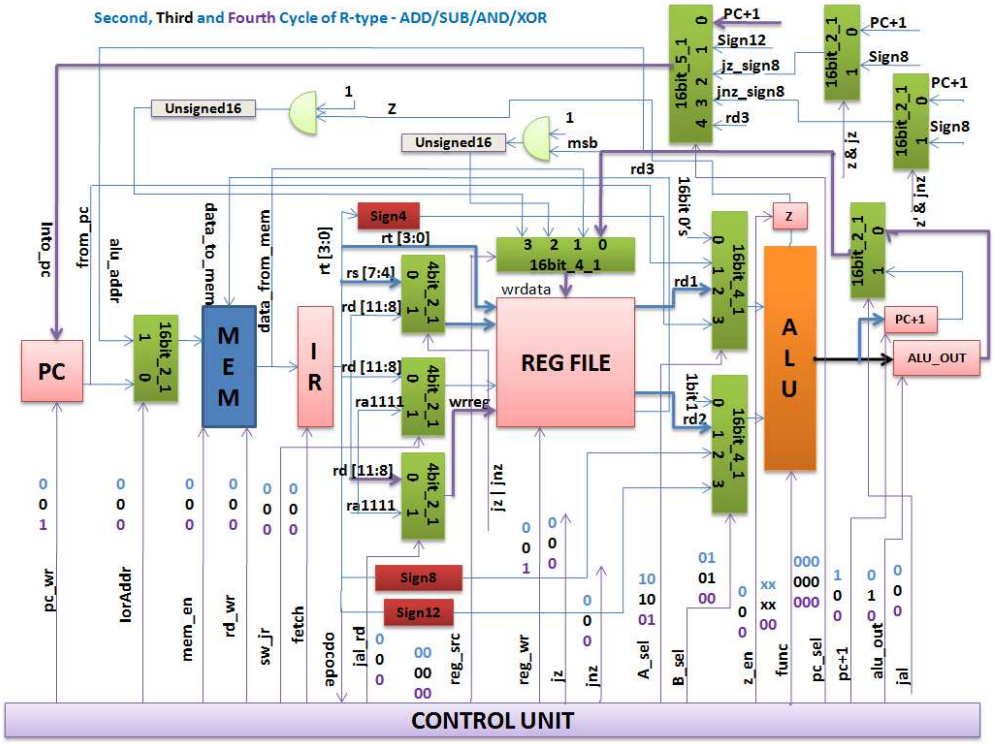

Figure 3.3: R-type ADD/SUB/AND/XOR

XOR respectively. The third clock cycle saves the ALU result in the ALU_OUT register and

control signal is alu_out = 1. Control signal jal = 1 passes the content of the ALU_OUT register through 16-bit 2-to-1 mux. At the fourth clock cycle reg_wr = 1 enables the register file to be written which was addressed by R_d[11:8] = wrreg[3:0] for control signal jal_rd = 0. At the same cycle pc_wr =1 enables PC register to be updated at the end of fourth clock cycle that can be used for next instruction. Second cycle to fourth cycle of R-type ADD/SUB/AND/XOR is shown in Figure 3.3 with control logic.

3.2.2 R-type SLT

At the second cycle PC+1 (computed in first cycle) gets updated in the PC+1 register and A_sel = 10, B_sel = 01 selects rd1 and rd2 as operands to the ALU, as indicated in Figure 3.4. Function of the ALU is func = 01, meaning it performs subtraction at the second cycle.

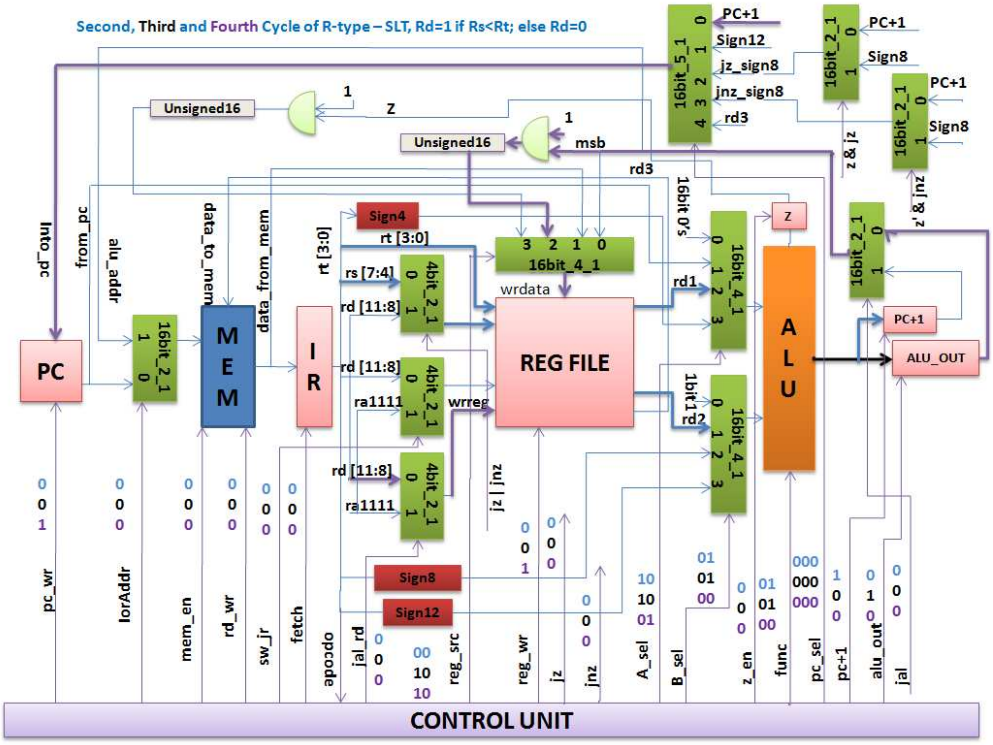

Figure 3.4: R-type SLT

At third clock cycle the result is stored in ALU_OUT register, holding control signal alu_out = 1. Control signal jal = 1 passes the content of the ALU_OUT register through the mux

and the MSB gets ANDed with 1. If $R_s < R_t$ then MSB = 1 and the AND result is 1 else MSB = 0. This result geta unsigned extension to 16-bit data. At fourth clock cycle, reg_wr = 1 enables the addressed register to be written by the data and PC is upadated as well as indicated in Figure 3.4.

3.2.3 R-type SEQ

At the second cycle PC+1 gets updated in the PC+1 register and rd1 and rd2 gets selected as operands to the ALU and the ALU performs the subtraction. At the third cycle

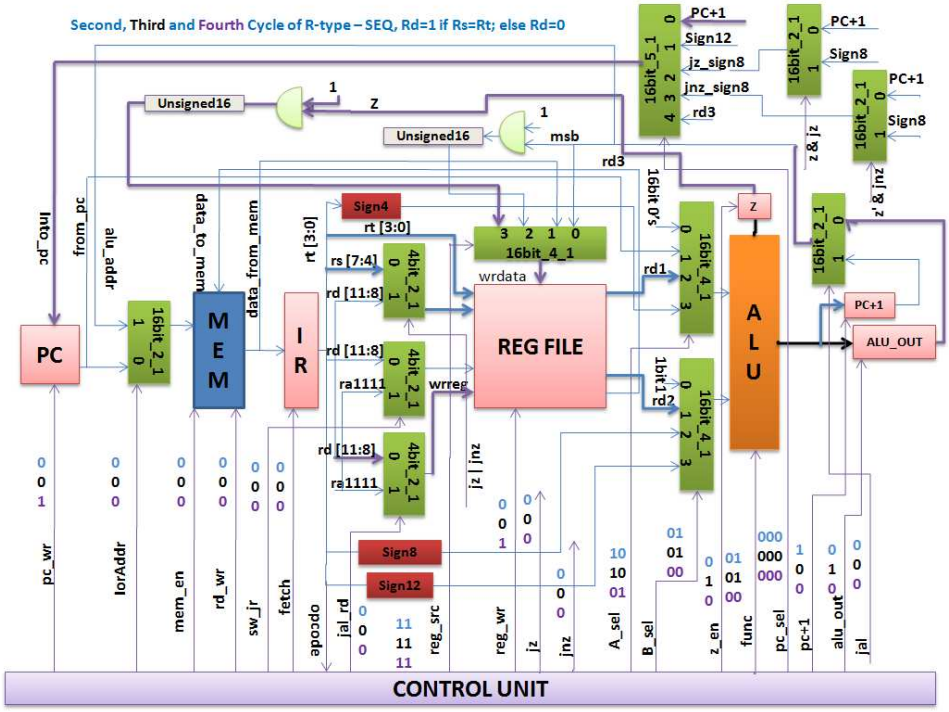

Figure 3.5: R-type SEQ

the ALU result gets updated in ALU_OUT and also control signal z = 1 enables the Z register to be updated. 1-bit zero flag register (Z) is used to hold the value, either 0 or 1 to indicate that the resultant bits are all zero or not, respectively. The content of the Z register (Z = 1 means rd1 = rd2 else rd1 $\neq$ rd2) is ANDed with 1-bit 1 at the third cycle. This result gets unsigned extension to 16-bit data and fed to fourth input of 16-bit 4-to-1 mux. At the fourth clock cycle reg_src = 11 selects that 4^{th} input to that mux and reg_wr = 1 enables

the write operation to the register file addressed by R_d[11:8] = wrreg[3:0], which is selected by control signal jal_rd = 0, as shown in Figure 3.5.

3.3 I-type Instruction

Four different types of instruction can be performed as immediate type operations. LW computes the contents of the R_s register and added to the sign extended 16-bit immediate data. The computed address is used to point to a location in memory and the data at this address is loaded into register, $R_d = [R_s] + n$. Likewise, SW computes the address but stores the content of the register addressed by 4-bit R_d into memory location, $[R_s+n] = R_d$. ADDI adds the content of register R_s to 4-bit immediate operand n (sign extended) and stores it to register R_d, $R_d = R_s+n$. MOVI converts 8-bit immediate operand to sign extended 16-bit data and moves it into register R_d, $R_d = n$.

3.3.1 I-type LW

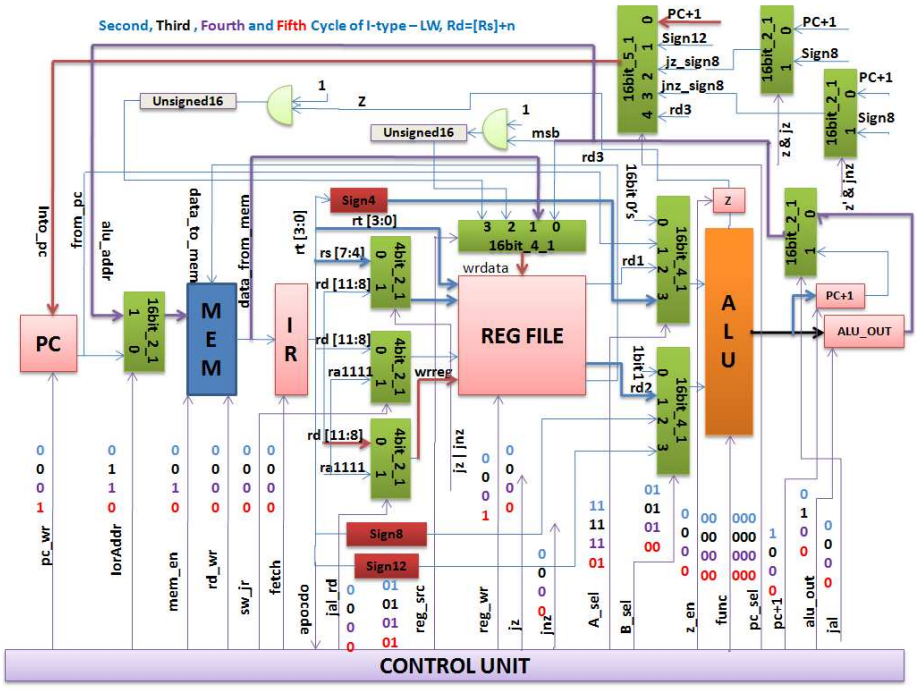

Figure 3.6: I-type LW

LW takes 5 clock cycles to complete and the second to fifth clock cycle is shown in Figure 3.6. At the second cycle, A_sel = 11 and B_sel = 01, selects sign extended rt[3:0] bits and content of register rs[3:0] through rd2. It also performs addition in second cycle. At the third clock cycle the result is stored in ALU_OUT register and at the fourth clock cycle the computed address (result) points to an address location in memory and memory is being when control signal IorAddr = 1 and mem_en = 1. Data from memory is written into register file at the fifth clock cycle and PC is updated while reg_wr = 1 and pc_wr = 1. The destination register is addressed by rd[11:8] and mux selection signal jal_rd = 0.

3.3.2 I-type SW

At the second cycle control signal A_sel = 11 and B_sel = 01 selects sign extended least significant 4-bits and the content of R_s register as operands to the ALU and performs addition. The computed address (result) is then updated in ALU_OUT in the third clock

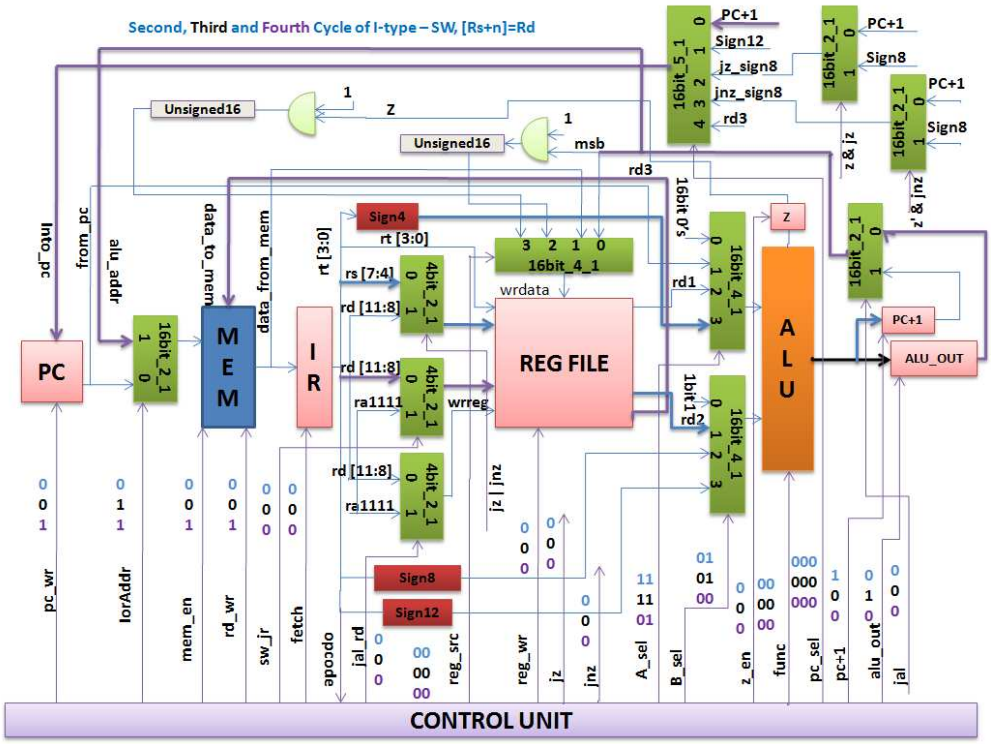

Figure 3.7: I-type SW

cycle, as indicated in Figure 3.7. At the fourth clock cycle, content of register file output rd3 = data_to_mem is written to the memory (MEM) addressed by computed result (R_s+n) when IorAddr = 1, mem_en = 1 and rd_wr = 1 and PC gets updated.

3.3.3 I-type ADDI

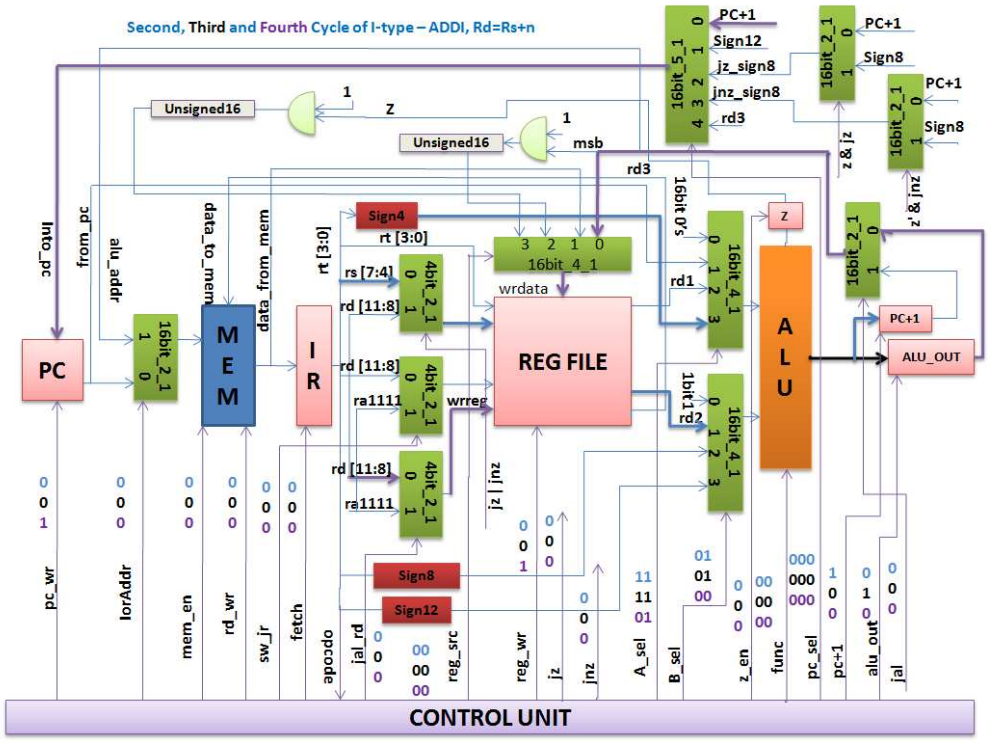

Figure 3.8: I-type ADDI

The second and third clock cycles for ADDI are the same as LW/SW, but this time the sum is considered as a result rather than an address and gets updated in ALU_OUT in the third clock cycle. The result is then saved to register R_d by selecting the path through reg_src = 00 and reg_wr =1 at the fourth clock cycle as indicated in Figure 3.8.

3.3.4 I-type MOVI

The MOVI operation moves least 8-bit data into register-R_d with sign extension as indicated in Figure 3.9. At the second clock cycle control signal A_sel = 00, B_sel = 10

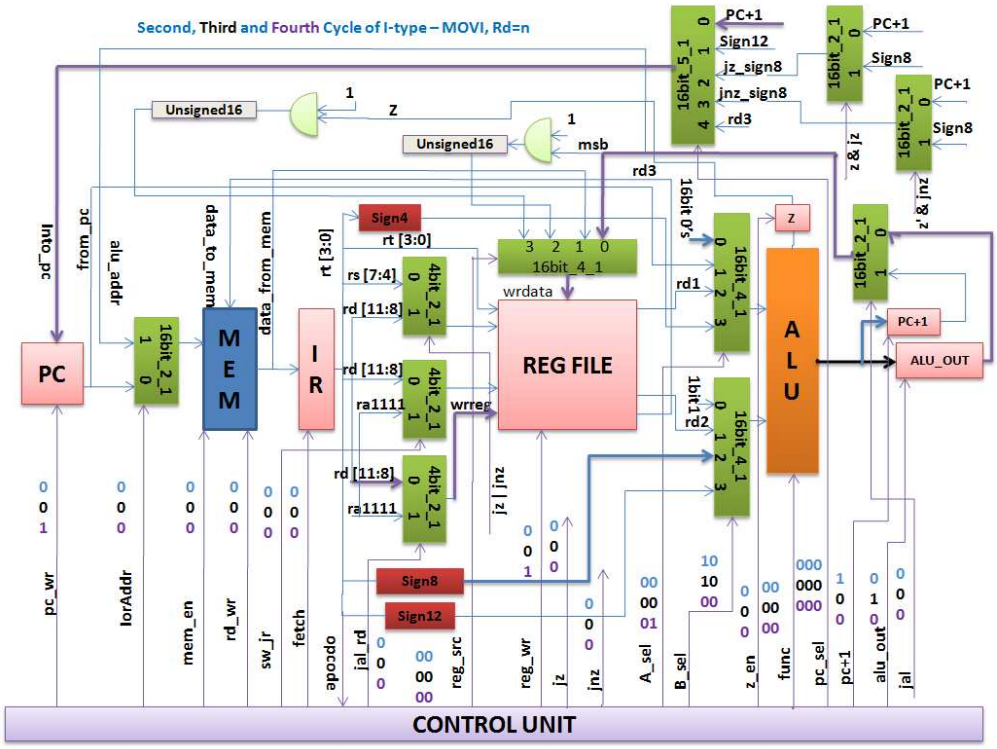

Figure 3.9: I-type MOVI

selects 16-bit 0 (from first input of 4-to-1 mux) and sign-extended 8-bit data as operands to ALU and performs addition, meaning 8-bit data is added with 0 and PC+1 is updated in the PC+1 register. At the third clock cycle the ALU result is saved into ALU_OUT register and control signal alu_out = 1. At the fourth clock cycle data is written into register file addressed by R_d[11:8] = wrreg[3:0] and reg_wr =1 and PC is updated.

3.4 J-type Instruction

J-type can perform five different types of jump instruction based on opcode. J LABEL performs jump unconditionally and does not save the program counter value. It takes 12-bit as an immediate operand and can jump in [-2048 - +2047] bit memory location. JZ and JNZ perform jump based on the contents of R_d. If $R_d = 0$ then JZ performs jump and if $R_d = 1$ then JNZ performs jump. If the condition is not satisfied then the program counter increments by one. Both JZ and JNZ take n = 8-bits as an immediate operand and can

jump in [-128 - +127] bit memory location. JAL instruction performs the jump operation unconditionally but it saves (jump and link) the incremented program counter into $RA = 1111 register in the register file. $RA is the fifteenth register in the register file that was implemented in hardware. The JR instruction returns from any location by loading the program counter from $RA that was saved earlier by some other instruction and can jump (return) in between [-32768 - +32767] bit memory location.

3.4.1 J-type J LABEL

As indicated in Figure 3.10, J LABEL can be executed in two clock cycles. The first

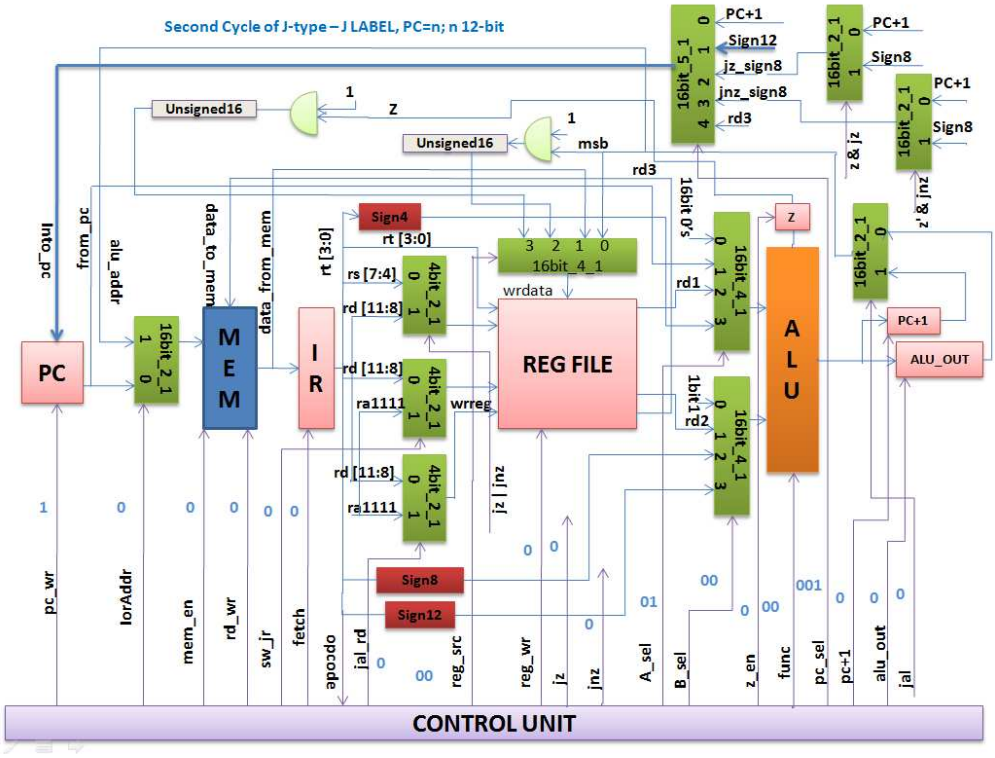

Figure 3.10: J-type J LABEL

cycle is the same as any instruction. At the second clock cycle, sign-extended 12-bit gets selected through 16-bit 5-to-1 mux by control signal pc_sel = 001. Also, at the second clock cycle control signal pc_wr =1 lets PC register be updated.

3.4.2 J-type JZ LABEL

JZ LABEL can be executed in four clock cycles as indicated in Figure 3.11. At the second clock cycle computed PC+1 gets updated in the PC+1 register and control signal jz =1 is activated and remains high until the fourth clock cycle. Also in the second clock

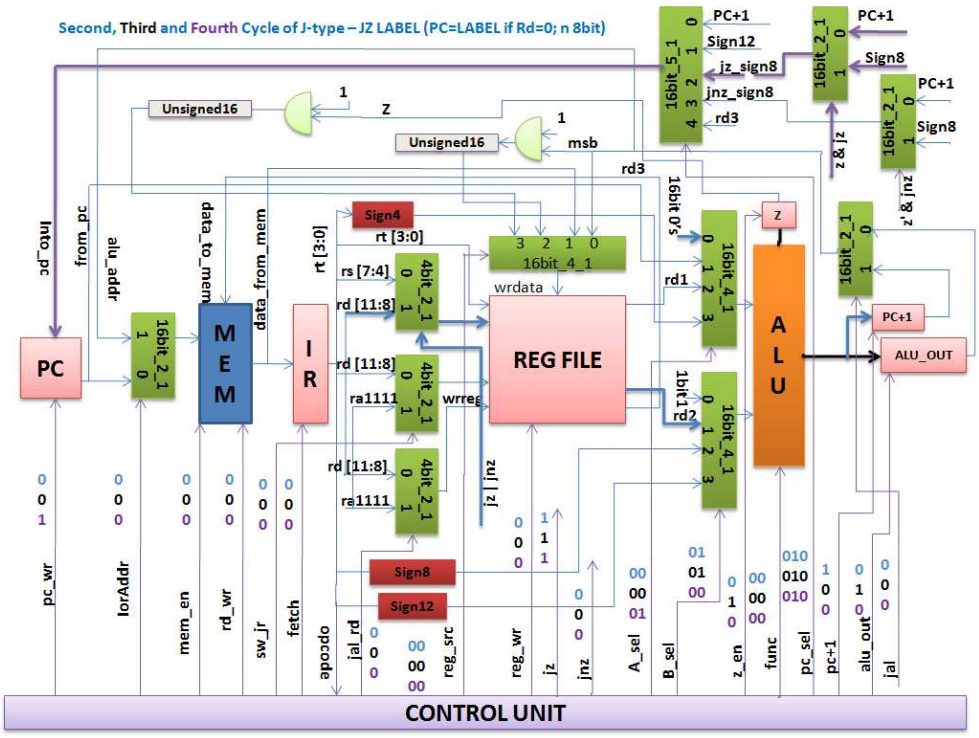

Figure 3.11: J-type JZ LABEL

cycle A_sel = 00 and B_sel = 01 selects 16-bit 0 (first input of 16-bit 4-to-1 mux) and the content of R_d register through rd2. Input to register file rr2 (which is R_d[11:8]) was selected through mux signal bit jz | jnz. The ALU performs addition and if the result is 0 then ALU output z will go high. At the third clock cycle z_en =1 activates z register to be updated. At the fourth clock cycle z & jz = 1 (if z = 1) selects 16-bit 2-to-1 mux control bit to pass Sign8 in the datapath and pc_sel = 010 lets it pass through 16-bit 5-to-1 mux and LABEL is updated in PC register.

3.4.3 J-type JNZ LABEL

JNZ can perform jump operation to 8-bit LABEL if $R_d \neq 0$, as indicated in Figure 3.12. At the second cycle control signal jnz = 1 enables R_d to be connected as input rr2 of register

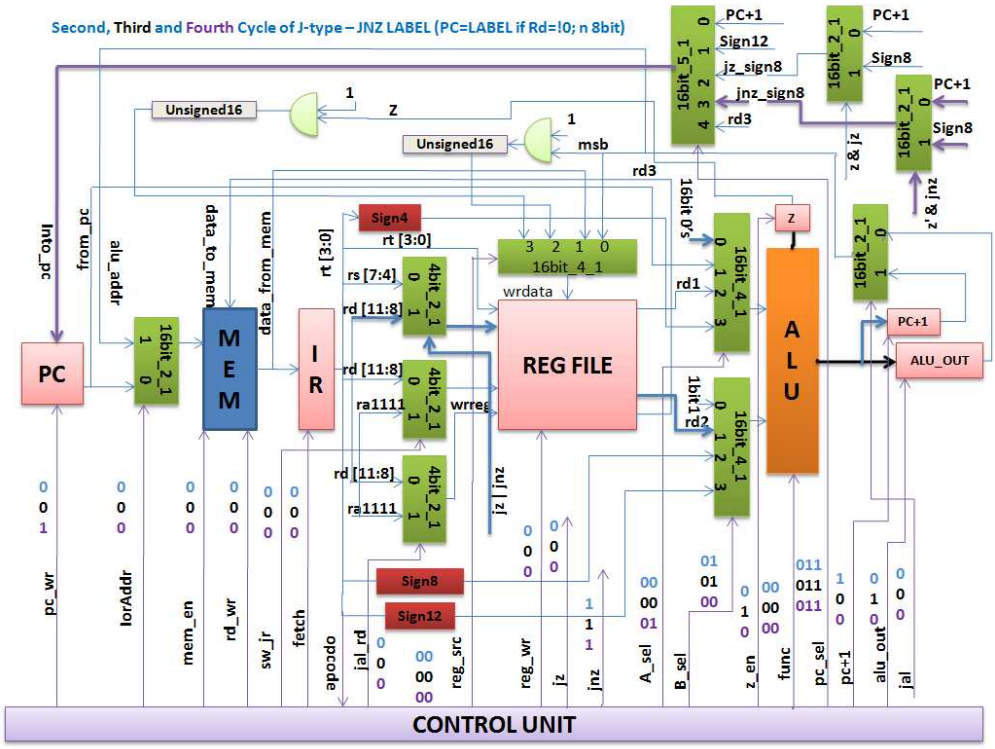

Figure 3.12: J-type JNZ LABEL

file through 4-bit 2-to-1 mux. Input rr2 of reg file generates content of this register at rr2 signal. A_sel = 00, B_sel = 01 selects 16-bit 0 and rr2 and func = 00 performs addition in this cycle. At the third cycle z_en = 1 let z register to be updated. At fourth clock cycle, z & jnz determines whether PC+1 or Sign8 will be passed through 16-bit 2-to-1 mux and pc_sel = 011 lets it pass. Also in this cycle pc_wr =1 lets PC register be updated.

3.4.4 J-type JAL

JAL can be performed in three clock cycles as indicated in Figure 3.13. At the first cycle the instruction is fetched to IR register and PC is incremented by 1 as stated before. At the second clock cycle control signal jal_rd =1 selects wrreg[3:0] = R_d[11:8], addressing

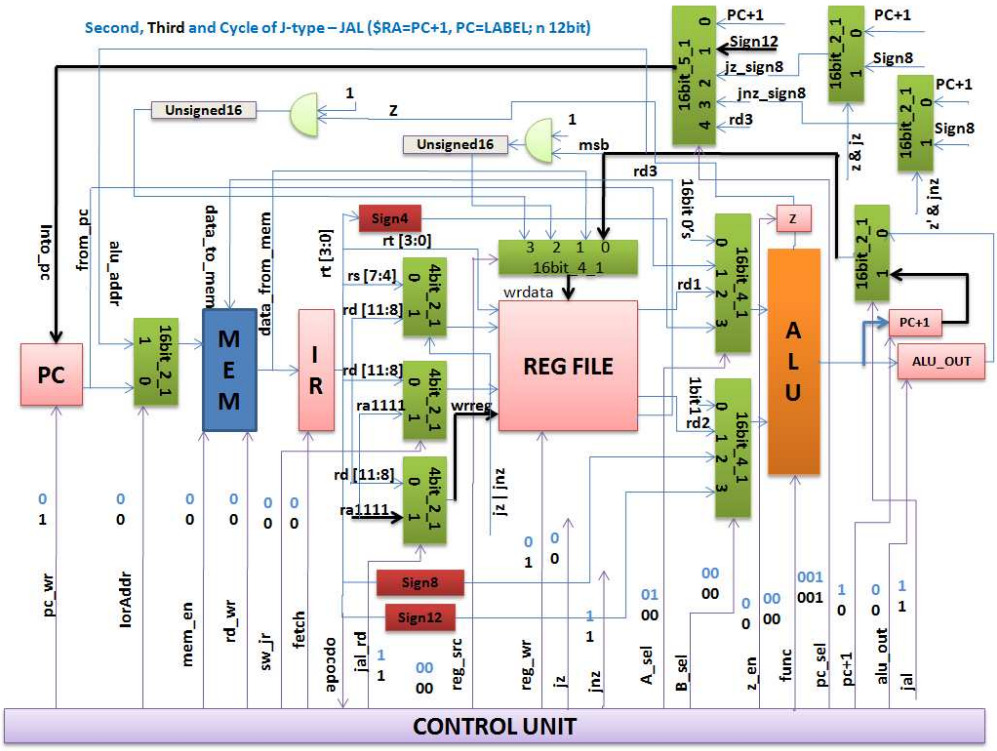

Figure 3.13: J-type JAL

which register needs to be written. Also at this cycle PC+1 register gets updated because of pc+1 = 1. At third clock cycle control signal jal = 1 selects PC+1 to pass through the 16-bit 2-to-1 mux and reg_wr = 1 lets it be written to the addressed register. Also Sign12 (12-bit sign extended LABEL) gets selected in the 16-bit 5-to-1 mux as control signal pc_sel = 001 and pc_wr = 1 enables updating of PC register.

3.4.5 J-type JR

JR can be executed in two clock cycles as indicated in Figure 3.14. At the second clock cycle, input to Ref file rr3[3:0] = ra[1111] gets selected as control signal sw_jr = 1. The corresponding content of rr3 becomes available at the rd3 16-bit output and gets passed through the 16-bit 5-to-1 mux as control signal is pc_sel = 100. At the same time result (return address coming through rd3) gets updated in PC register and control signal is pc_wr = 1.

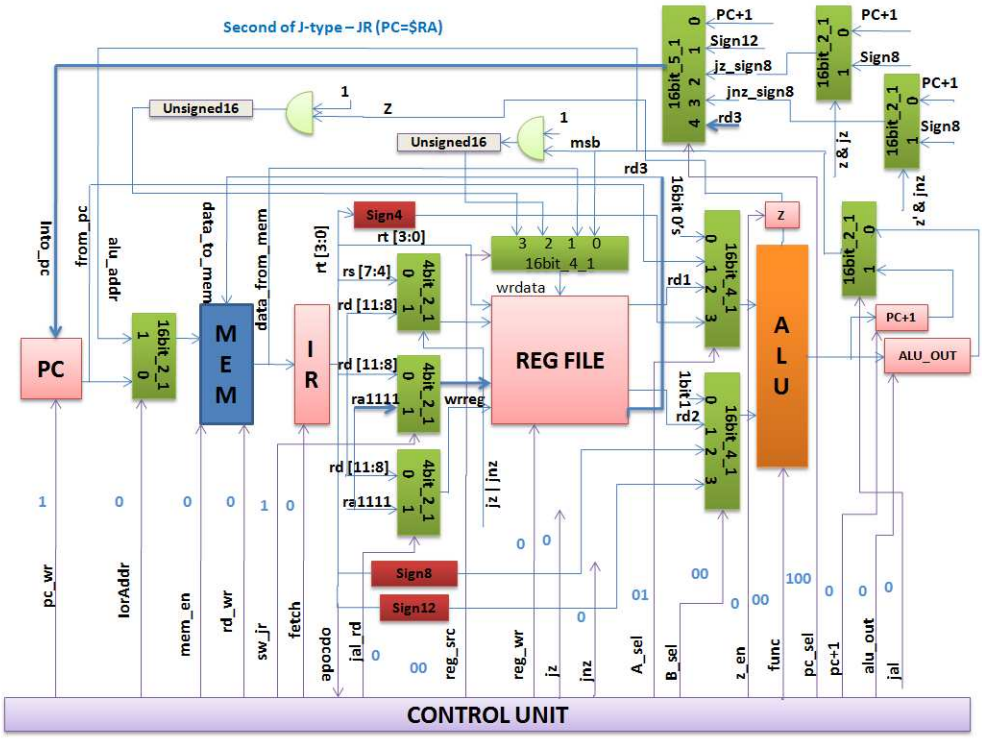

Figure 3.14: J-type JR

3.5 Control Signals

Control signals have been summarized for all fifteen instructions and listed in Table 3.3 and Table 3.4 for every clock cycle.

Cycle	pc_wr	IorAddr	mem_en	rd_wr	fetch	sw_jr	jal_rd	reg_src	reg_wr	A_sel	B_sel	func	z_en	pc+1	alu_out	jal	pc_sel	Jz	Jnz
All 1	0	0	1	0	1	0	0	00	0	01	00	00	0	0	0	0	000	0	0
Add 2	0	0	0	0	0	0	0	00	0	10	01	xx	0	1	0	0	000	0	0
Add 3	0	0	0	0	0	0	0	00	0	10	01	xx	0	0	1	0	000	0	0
Add 4	1	0	0	0	0	0	0	00	1	01	00	00	0	0	0	0	000	0	0
Slt 2	0	0	0	0	0	0	0	00	0	10	01	01	0	1	0	0	000	0	0
Slt 3	0	0	0	0	0	0	0	10	0	10	01	01	0	0	1	0	000	0	0
Slt 4	1	0	0	0	0	0	0	10	1	01	00	00	0	0	0	0	000	0	0
Seq 2	0	0	0	0	0	0	0	11	0	10	01	01	0	1	0	0	000	0	0
Seq 3	0	0	0	0	0	0	0	11	0	10	01	01	1	0	1	0	000	0	0
Seq 4	1	0	0	0	0	0	0	11	1	01	00	00	0	0	0	0	000	0	0
Lw 2	0	0	0	0	0	0	0	01	0	11	01	00	0	1	0	0	000	0	0
Lw 3	0	1	0	0	0	0	0	01	0	11	01	00	0	0	1	0	000	0	0
Lw 4	0	1	1	0	0	0	0	01	0	11	01	00	0	0	0	0	000	0	0
Lw 5	1	0	0	0	0	0	0	01	1	01	00	00	0	0	0	0	000	0	0
Sw 2	0	0	0	0	0	0	0	00	0	11	01	00	0	1	0	0	000	0	0
Sw 3	0	1	0	0	0	0	0	00	0	11	01	00	0	0	1	0	000	0	0
Sw 4	1	1	1	1	0	0	0	00	0	01	00	00	0	0	0	0	000	0	0

Table 3.3: Control Signal Table Part-1

Cycle	pc_wr	IorAddr	mem_en	rd_wr	fetch	sw_jr	jal_rd	reg_src	reg_wr	A_sel	B_sel	func	z_en	pc+1	alu_out	jal	pc_sel	Jz	Jnz
All 1	0	0	1	0	1	0	0	00	0	01	00	00	0	0	0	0	000	0	0
Addi 2	0	0	0	0	0	0	0	00	0	11	01	00	0	1	0	0	000	0	0
Addi 3	0	0	0	0	0	0	0	00	0	11	01	00	0	0	1	0	000	0	0
Addi 4	1	0	0	0	0	0	0	00	1	01	00	00	0	0	0	0	000	0	0
Movi 2	0	0	0	0	0	0	0	00	0	00	10	00	0	1	0	0	000	0	0
movi 3	0	0	0	0	0	0	0	00	0	00	10	00	0	0	1	0	000	0	0
Movi 4	1	0	0	0	0	0	0	00	1	01	00	00	0	0	0	0	000	0	0
J 2	1	0	0	0	0	0	0	00	0	01	00	00	0	0	0	0	001	0	0
JZ 2	0	0	0	0	0	0	0	00	0	00	01	00	0	1	0	0	010	1	0
JZ 3	0	0	0	0	0	0	0	00	0	00	01	00	1	0	1	0	010	1	0
Jz 4	1	0	0	0	0	0	0	00	0	01	00	00	0	0	0	0	010	1	0
Jnz 2	0	0	0	0	0	0	0	00	0	00	01	00	0	1	0	0	011	0	1
Jnz 3	0	0	0	0	0	0	0	00	0	00	01	00	1	0	1	0	011	0	1
Jnz 4	1	0	0	0	0	0	0	00	0	01	00	00	0	0	0	0	011	0	1
JAL 2	0	0	0	0	0	0	1	00	0	01	00	00	0	1	0	1	001	0	0
JAL 3	1	0	0	0	0	0	1	00	1	01	00	00	0	0	0	1	001	0	0
JR 2	1	0	0	0	0	1	0	00	0	00	00	00	0	0	0	0	100	0	0

Table 3.4: Control Signal Table Part-2

Chapter 4

Component Realization

The processor datapath is shown again in Figure 4.1 to indicate the components that are necessary to realize in CML. This chapter will demonstrate how each of these datapath components was realized using basic CML components developed in Chapter 2. Each basic component included 20fF load capacitance in designing datapath components to mimic post layout simulation and wire delay. Proper handcrafting in Cadence Virtuoso Composer Schematic has been done and Spectre simulation (equivalent to SPICE) was performed to verify correct operation of each block. The tool used is Analog Artist for Spectre simulation and the technology used is 130nm CMOS. The processor datapath consists of:

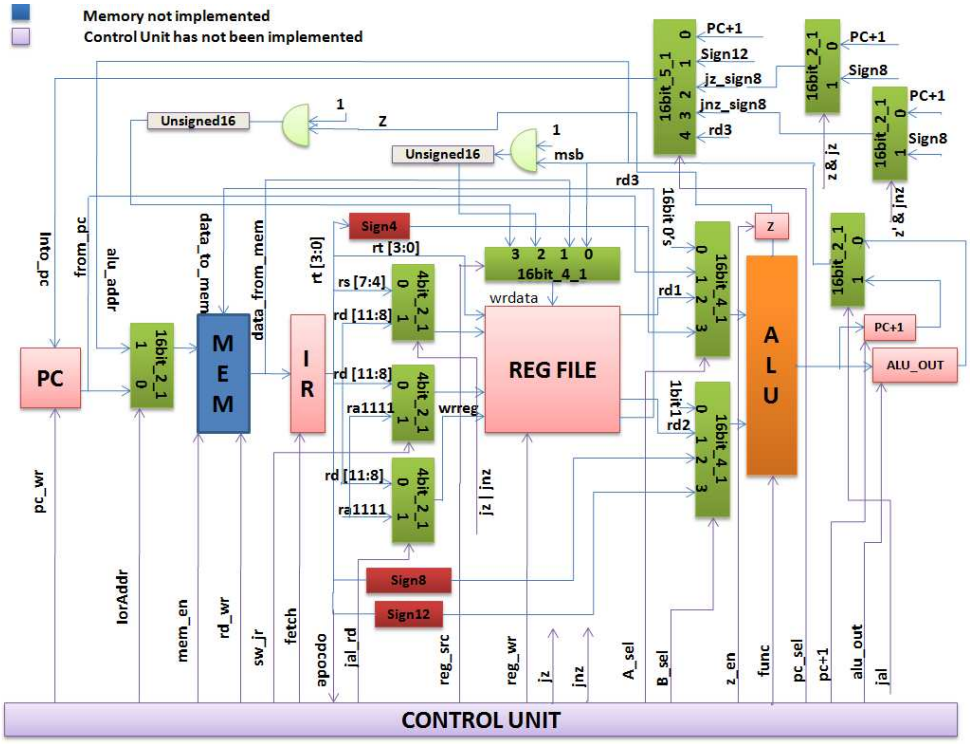

Figure 4.1: Datapath Components

1. 4 16-bit register with enable input (PC, IR, ALU_OUT, PC+1)
2. Z register (1-bit register with enable input)
3. 4 16-bit 2-to-1 mux
4. 3 4-bit 2-to-1 mux
5. 3 16-bit 4-to-1 mux
6. 16-bit 5-to-1 mux
7. 16-bit ALU
8. 16x16 register file (REG FILE)
9. Sign 4 to 16 extension (sign 4)
10. Sign 8 to 16 extension (sign 8)
11. Sign 12 to16 extension (sign 12)
12. 2 unsigned 1 to 16 extension (unsigned 16)
13. 4 1-bit AND gate
14. 1 1-bit OR gate

4.1 16-bit Register With Enable Input

A 1-bit register was developed by cascading CML d-latches. As CML comes in complementary form, there are two signals for any input/output. The negative clock pulse was provided to the first d-latch and the positive pulse was given to the second d-latch in order to realize a positive edge triggered D flip-flop. A two input mux was used in front of the master-slave DFF to hold the state when enable is 0 and pass the data to input D of

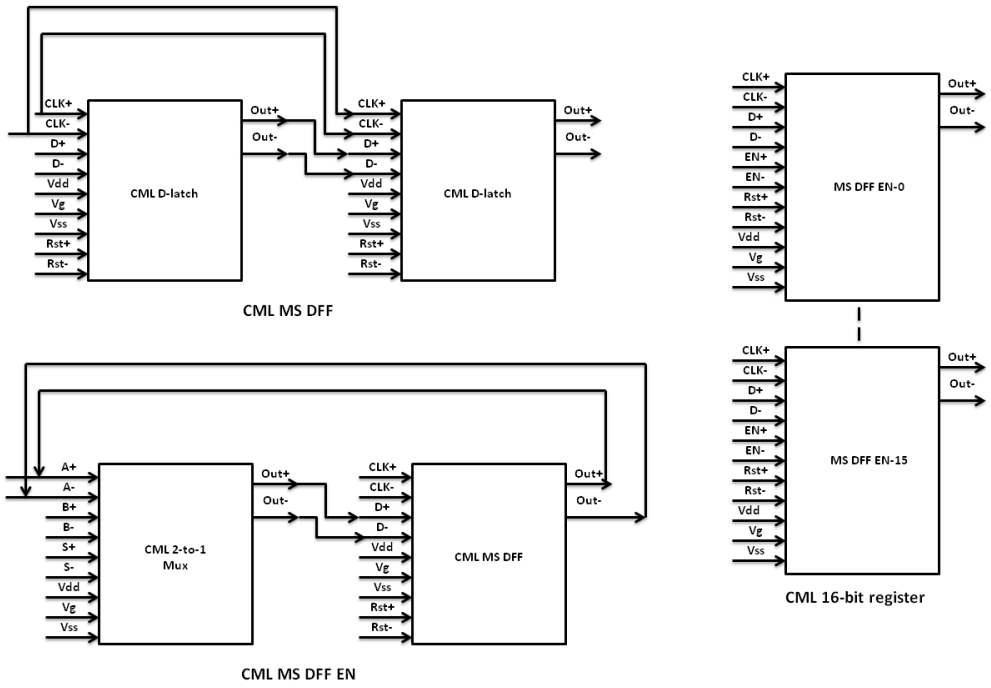

Figure 4.2: Block diagram of MS DFF, MS DFF-EN, 16-bit register

DFF when enable is 1. The circuit diagrams of the master-slave D flip-flop, master-slave D Flip-flop with an enable input and 16-bit register with enable input are shown in Figure 4.2.

MS DFF burns 2 * 15.3132mW = 30.6264mW and MS DFF-EN takes 16.226mW + 30.6264mW = 46.8524mW. For simplicity, only 1-bit MS DFF-EN output is shown in Figure 4.3. The 16-bit register output will be the same and delay is identical as well because they are parallel. In Figure 4.3, 101.41ps rise delay has been observed with 20fF load capacitance. In simulation it is observed that the 16-bit MS DFF-EN (16-bit register) takes 16 * 46.8524mW = 749.6384mW. Minimum setup time was found to be 40ps to operate at 6GHz.

Estimated area of the 16-bit register is 16 * (2*D latch area + 2-to1 mux area) = 16 * (2*(29.19 x 57.78)μm + 36.51 x 47.52μm) = 16*(94.89μm x 57.78μm) = 1518.24μm x 57.78μm. 50% additive area for wiring will be added later to the total area of the processor to get power density per unit area.

Therfore, in datapath 4 16-bit registers should dissipate 4 * 749.6384mW = 2998.55mW with an area of 4 * (1518.24μm x 57.78μm) = 6072.96μm x 57.78μm.

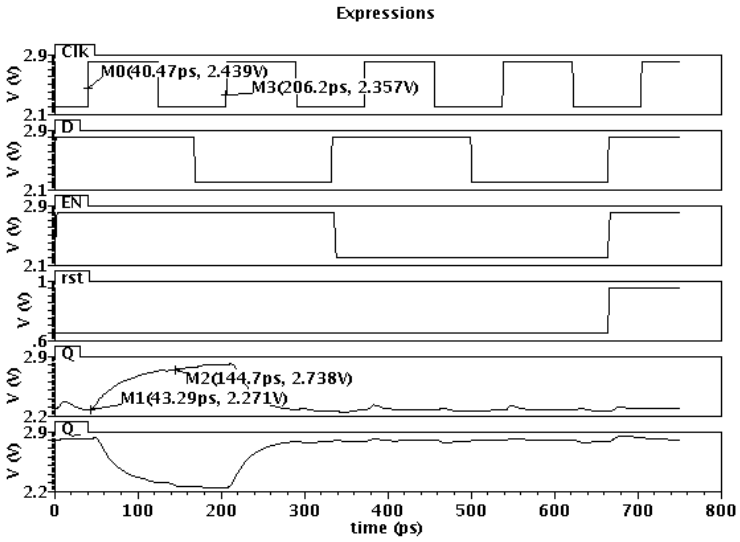

Figure 4.3: 1-bit register output at 6GHz with 20fF load capacitance (clock period 166ps)

4.2 Z Register (1-bit Register With Enable Input)

As indicated in section 4.1, the 1-bit register with an enable input has power consumption of 46.8524mW and an area of 94.89μm x 57.78μm and reported rise delay is 101.41ps.

4.3 4 16-bit 2-to-1 Mux

A 1-bit 2-to-1 mux has been shown in Figure 2.17, with a power consumption of 16.226mW and an area of 36.51μm x 47.52μm, with rise delay of 33.99ps. A 16-bit 2-to-1 mux is 16 instances of 1-bit 2-to-1 mux with 16 times power consumption of 259.616mW and 16 times area 584.16μm x 47.52μm. Delay is same because they are parallel in architecutre.

Therefore, 4 16-bit 2-to-1 mux power consumption is 1038.464mW, area = 2336.64μm x 47.52μm and rise delay = 33.99ps. Due to static power consumption, computing power for 16-bit 2-to-1 mux theoretically as 16 * 1-bit 2-to-1 mux produces the same value as simulating 16-bit 2-to-1 mux and getting the power from the tool. However, in some cases where larger number of basic CML components has been used, we simulate the component directly to get power consumption from the tool.

4.4 3 4-bit 2-to-1 Mux

As indicated in section 4.3, a 4-bit 2-to-1 mux power consumption is 64.904mW and area is 146.04μm x 47.52μm. Therefore 3 4-bit 2-to-1 mux power consumption is 194.712mW, with an area of 438.12μm x 47.52μm and rise delay is 33.99ps.

4.5 3 16-bit 4-to-1 Mux

A 1-bit 4-to-1 mux was developed using 3 1-bit 2-to-1 mux, providing S0 as the control bit for first stage and S1 as the control bit for the second stage as indicated in Figure 4.4. Expected power consumption is 3 * 1-bit 2-to-1 mux power = 3 * 16.226mW = 48.678mW and simulation power obtained is 2.8V * 17.38mA = 48.664mW, which is almost identical.

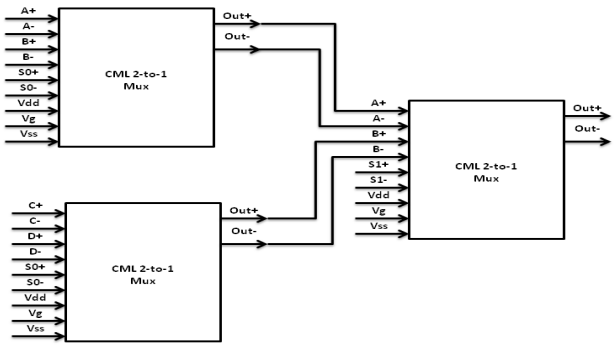

Figure 4.4: 1-bit 4-to-1 mux

The expected area of a 1-bit 4-to-1 mux would be 3 * (36.510μm x 47.52μm) = 109.53μm x 47.52μm. Expected rise delay is 2 * 33.99ps = 67.98ps, due to two stages of CML 2-to-1 mux, and simulated rise delay is 52.51ps, as indicated in Figure 4.5.

Therefore, the 3 16-bit 4-to-1 mux power consumption is 3 * 16 * 48.664mW = 3 * 778.624mW = 2335.872mW, area is 3 * 16 * (109.53μm x 47.52μm) = 3 * (1752.48μm x 47.52μm) = 5257.44μm x 47.52μm and simulated rise delay is 52.51ps.

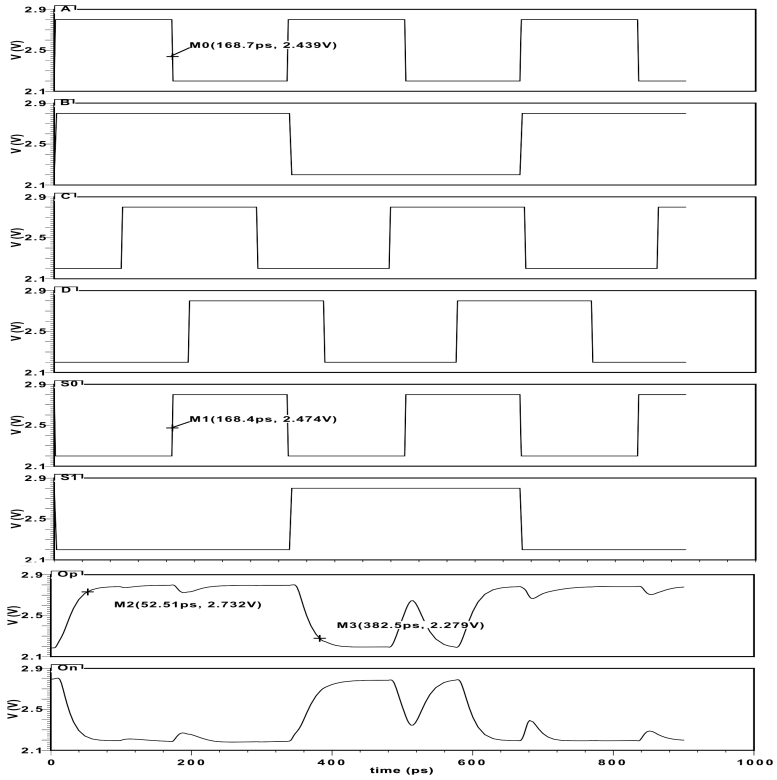

Figure 4.5: 1-bit 4-to-1 mux output with 20fF load capacitance (input changes at 83ps)

4.6 16-bit 5-to-1 mux

A 1-bit 5-to-1 mux was developed using a 1-bit 4-to-1 mux and a 1-bit 2-to-1 mux as shown in Figure 4.6. Expected power consumption is the sum of two = 48.664mW + 16.226mW = 64.89mW and simulated power is 2.8V * 23.18mA = 64.904mW.

The 16-bit 5-to-1 mux power consumption is 16 * 64.904mW = 1038.464mW and expected area is 16 * (1-bit 4-to-1 + 1-bit 2-to-1) = 16 * (109.53μm x 47.52μm + 36.51μm x 47.52μm) = 16 * (146.04μm + 47.52μm) = 2336.64μm x 47.52μm.

A 1-bit 5-to-1 mux output at where input changes at 166ps is shown in Figure 4.7. Simulated rise delay is 59.61ps, as indicated.

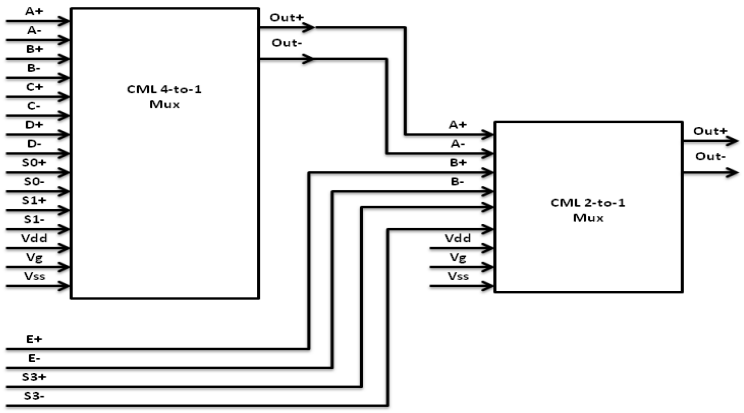

Figure 4.6: 1-bit 5-to-1 mux

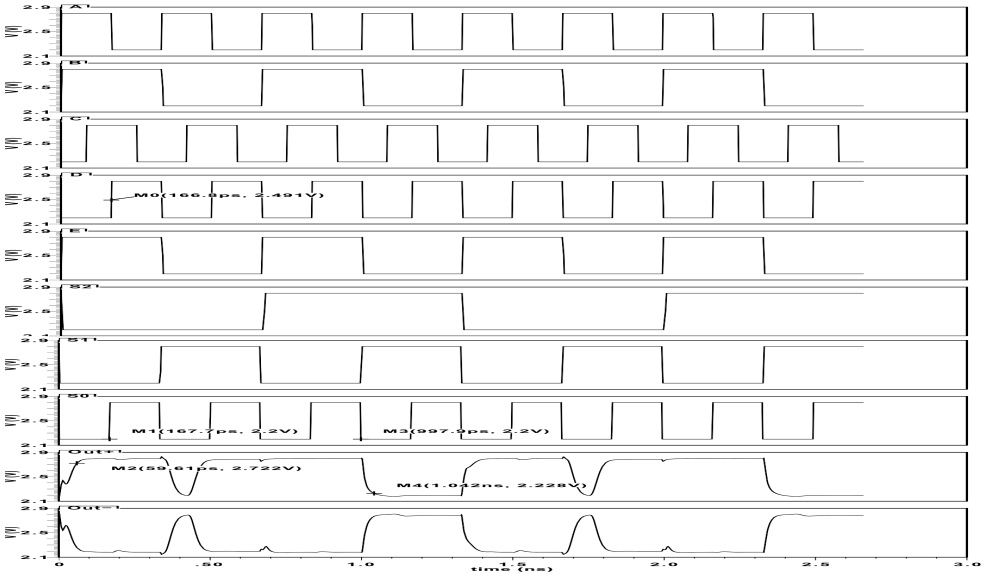

Figure 4.7: 1-bit 5-to-1 Mux output with 20fF load capacitance (input changes at 166ps)

4.7 16-bit ALU

A 16-bit ALU consists of a 16-bit CLA, 16-bit BLA, 16-bit AND, 16-bit XOR and 16-bit 4-to-1 mux. For 16-bit inputs at A and B - addition, subtraction, AND and XOR are performed irrespective to 2-bit function. Based on the function input, the 16-bit 4-to-1 mux passes the selected result to the output. The output bits are also OR-ed and passed to Z signal, which is connected to zero flag register.

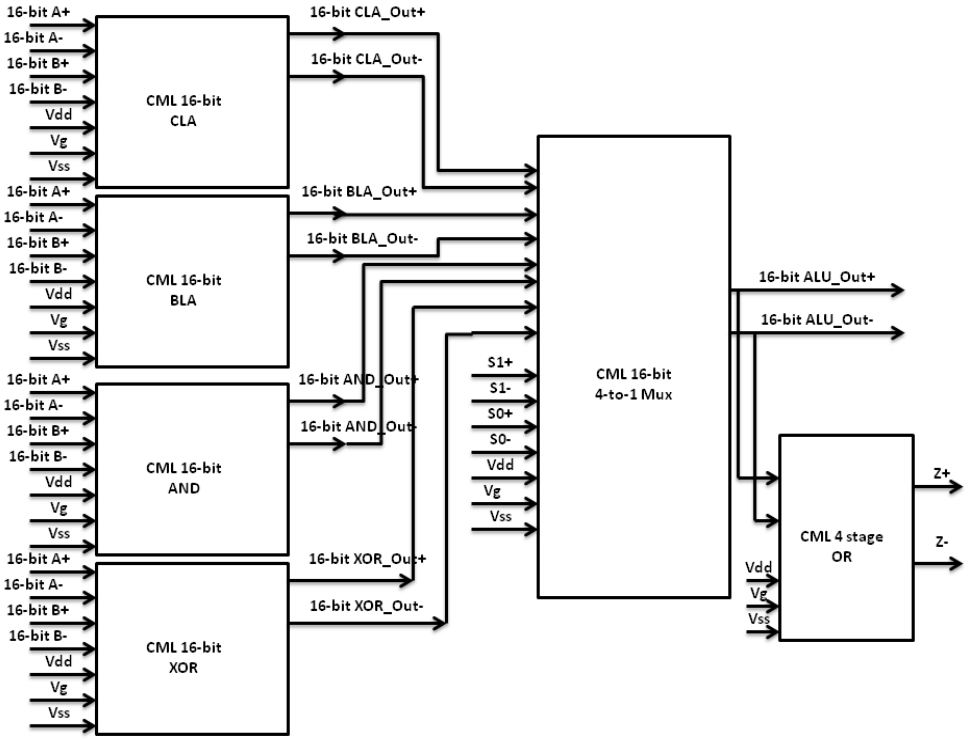

Figure 4.8: Block diagram of 16-bit ALU

A 16-bit carry look ahead adder was designed using the schematic shown in Figure 4.9, where carry generation is g_i, carry propagation is p_i and summation is s_i for each 1-bit CLA.

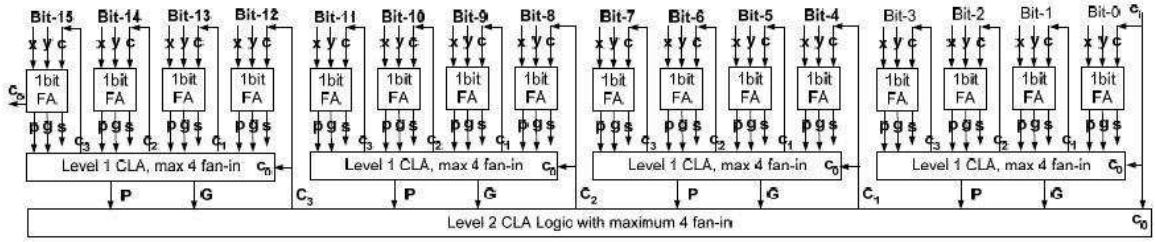

Figure 4.9: 16-bit CLA block diagram

$$g_i{=}x_i{\cdot}y_i;\ p_i{=}x_i{\oplus}y_i$$

$$s_i{=}x_i{\oplus}y_i{\oplus}c_{i-1}$$

Level-1 CLA output can be described as:

$$\mathrm{P0{=}p3{\cdot}p2{\cdot}p1{\cdot}p0}$$

P1=p7·p6·p5·p4

P2=p11·p10·p9·p8

P3=p15·p14·p13·p12

G0=g3+(p3·g2)+(p3·p2·g1)+(p3·p2·p1·g0)

G1=g7+(p7·g6)+(p7·p6·g5)+(p7·p6·p5·g4)

G2=g11+(p11·g10)+(p11·p10·g9)+(p11·p10·p9·g8)

G3=g15+(p15·g14)+(p15·p14·g13)+(p15·p14·p13·g12)

c1=g0+p0·c0

c2=g1+p1·g0+p1·p0·c0

c3=g2+p2·g1+p2·p1·g0+p2·p1·p0·c0

Level-2 CLA output can be described as:

C1=G0+P0·c0

C2=G1+P1·G0+P1·P0·c0

C3=G2+P2·G1+P2·P1·G0+P2·P1·P0·c0

Power consumption for the 16-bit CLA obtained in circuit simulation is 2.8V * 1.198A = 3.3544W. Critical path sensitization in 16-bit CLA was triggered by providing all 0's in 16-bit A and B and then All 1's in input A and 1-bit 1 in C_{in}. Simulation result shows rise delay of 16-bit CLA is 224.7ps, as indicated in Figure 4.10.

The 16-bit CLA expected area is 16 * (1-bit CLA) + 4 * Level-1 CLA + Level-2 CLA = 16 * (1 AND + 3 XOR) + 4 * (19 AND + 9 OR) + 10 AND + 6 OR = 16 * (30.15μm x 42.9μm + 3 * (28.59μm x 45.63μm)) + 4 * 28 * (30.15μm x 42.9μm) + 16 * (30.15μm x 42.9μm) = 1854.72μm x 45.63μm + 3376.8μm x 42.9μm + 482.4μm x 42.9μm = 5713.92μm x 42.9μm.

A 16-bit BLA can be realized using the same architecture of Level-1 and Level-2, but with a little difference in the 1-bit BLA.

$g_i=\bar{x_i}\cdot y_i$; $p_i=\bar{x_i}+y_i$

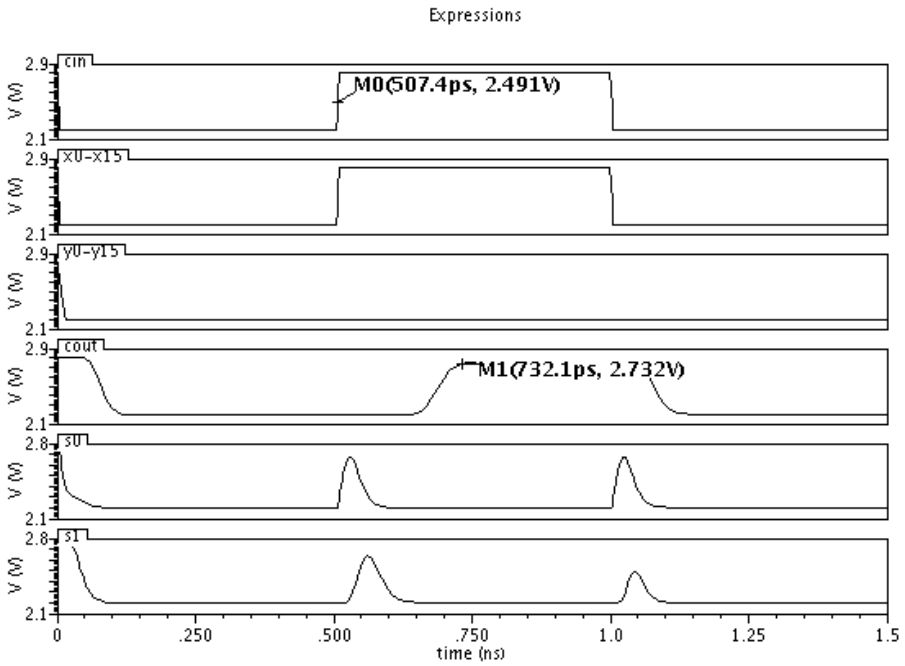

Figure 4.10: Critical path delay in 16-bit CLA is 224.7ps (input changes at 500ps)

$$s_i = x_i \oplus y_i \oplus c_{i-1}$$

Power consumption of the 16-bit BLA found in simulation is 2.8V * 1.198A = 3.3544W, which is the same as the 16-bit CLA and critical path delay should be same as well. Estimated area of 16-bit BLA would be 16 * 1-bit BLA + 4 * Levlel-1 CLA + Level-2 CLA = 16 * (1AND + 1OR + 2XOR) + 3376.8μm x 42.9μm + 482.4μm x 42.9μm = 16 * 2 * (AND + XOR) + 3859.2μm x 42.9μm = 32 * (58.74μm x 45.63μm) + 3859.2μm x 42.9μm = 5738.88μm x 45.63μm.

16-bit AND power consumption would be 16 * 16.2148mW = 259.4368mW, with an expected area of 16 * (30.15μm x 42.9μm) = 482.4μm x 42.9μm. Delay will be identical to the 1-bit AND gate which is 37.6ps.

16-bit XOR power consumption would be 16 * 16.226mW = 259.616mW, with an expected area of 16 * (28.59μm x 45.63μm) = 457.44μm x 45.63μm. Rise delay of the 16-bit XOR is 41.5ps.

16-bit 4-to-1 mux power consumption is 16 * 48.664mW = 778.624mW , area is 16 * (109.53μm x 47.52μm) = 1752.48μm x 47.52μm and simulated rise delay is 52.51ps.

Total expected area estimated is, 5713.92μm x 42.9μm + 5738.88μm x 45.63μm + 482.4μm x 42.9μm + 457.44μm x 45.63μm + 1752.48μm x 47.52μm = 14145.12μm x 47.52μm.

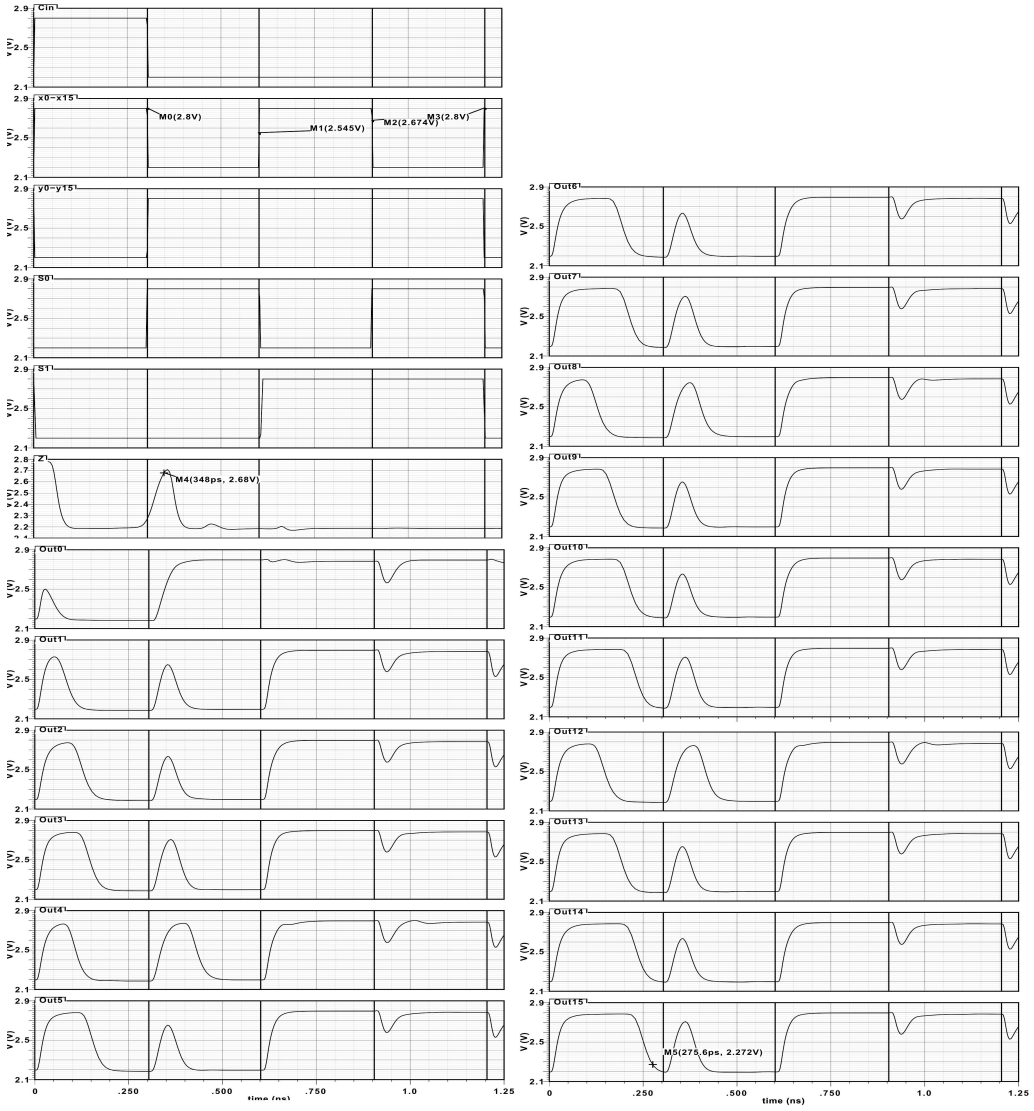

Figure 4.11: 16-bit ALU Output (input changes at 300ps)

Estimated path-delay from input to output of the ALU is max(16-bit CLA, 16-bit BLA, 16-bit AND, 16-bit XOR) + 16-bit 4-to-1 mux = 224.7ps + 52.51ps = 277.21ps. But critical path delay exists between the ALU input to Z flag where 4 stage of OR have been used.

The Z flag is used only in 3 instructions out of 15 instructions. In simulating 16-bit ALU at 300ps using path sensitization as indicated in Figure 4.10, generated correct output of Out signals at 275.6ps and Z signal took 348ps. Therefore 12 instructions can be executed at 300ps (3.33GHz) and 3 instruction (where z flag is necessary) can be executed in 350ps (2.85GHz).

In Figure 4.11 the ALU performs addition, ADD FFFF 0000 (with C_{in}=1) and result is all 0000 and Z = 1 when mux selection bit S0S1 = 00. Then for S0S1 = 01 it performs subtraction, SUB 0000 FFFF and the result is 0001. For S0S1=10 it performs 16-bit AND, AND FFFF FFFF and the result is FFFF. For S0S1=11 the ALU performs 16-bit XOR, XOR 0000 FFFF and the result is FFFF.

Estimated power consumption of 16-bit ALU is 3.3544W + 3.3544W + 259.4368mW + 259.616mW + 778.624mW + 15 * 16.2148mW = 8.2496988W and simulation result shows 16-bit ALU takes 2.8V * 2.944A = 8.2432W. Estimated area of ALU is 14145.12μm x 47.52μm. Time it takes to generate correct operation sensitizing critical path is 275.6ps for 12 instructions and 348ps for 3 instructions.

4.8 16x16 Register File

16x16 register file consists of 4-to-16 decoder, 16-bit AND, 16 16-bit registers and 3 16-bit 16-to-1 mux, as shown in Figure 4.12.

The register file has 3 4-bit inputs, rr1, rr2 and rr3, to address any of the 16 registers and corresponding 16-bit data will arrive at rd1, rd2 and rd3, respectively. 4-bit wrreg[3:0] is used to address any of the 16 register (16-bits each) and data will be written when control signal reg_wr is 1. Therefore, wrreg[3:0] input is fed to 4-to-16 decoder input. For wrreg[3:0] = 0000 first output bit of decoder will be high and rest of them will be low. For a particular combination of wrreg[3:0], the equivalent BCD value output line of the decoder will be high and rest of them will be low, as indicated in Figure 4.13.

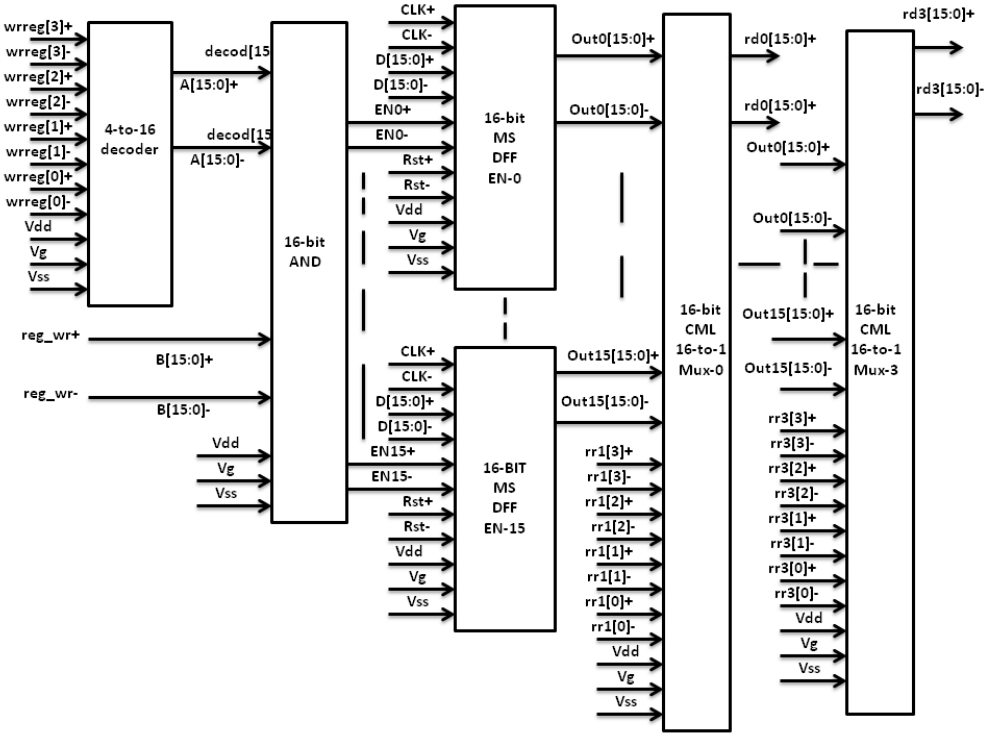

Figure 4.12: 16x16 Register File Schematic

4-to-16 decoder delay is dominated by the 2 stage AND delay. Theoretical estimation of rise delay = 75.2ps and simulated path delay observed is, rise delay = 72.01ps. Simulated power obtained for 4-to-16 decoder is 2.8V * 278.2mA = 778.96mW and expected area is 48 * AND gate area = 48 * (30.15μm x 42.9μm) = 1447.2μm x 42.9μm.

The output of the decoder is connected to a 16-bit AND and the second input of the 16-bit AND is reg_wr. Therefore, one of sixteen output lines of the 16-bit AND will be high if and only if the corresponding decoder line is high and reg_wr is high, to make sure that the selected register can be written only when reg_wr is high. The output of 16-bit AND was connected to the EN input of 16 registers. 16-bit AND power consumption is 16 * 16.2148mW = 259.4368mW, rise delay = 37.6ps, expected area = 16 * (30.15μm x 42.9μm) = 482.4μm x 42.9μm.

As stated in section 4.1, 16-bit register area = 1518.24μm x 57.78μm, with power conusmption of 923.44mW with rise delay = 101.41ps. Therefore, 16 16-bit register area

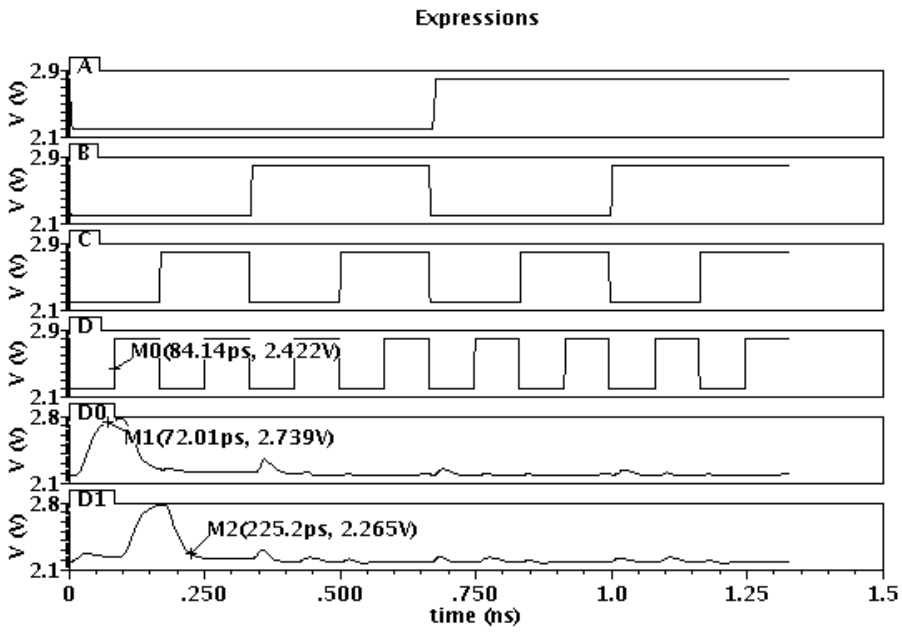

Figure 4.13: 4-to-16 Deocoder (input changes at 83ps)

= 16 * (1518.24μm x 57.78μm) = 24291.84μm x 57.78μm and power consumption is 16 * 749.6384mW = 11.994W.

A 1-bit 16-to-1 mux was developed using 1-bit 4-to-1 mux as indicated in Figure 4.14. The two least significant control bits, S0 and S1, were connected to the first stage of the 1-bit 4-to-1 mux and S2, S3 were connected to the second stage.

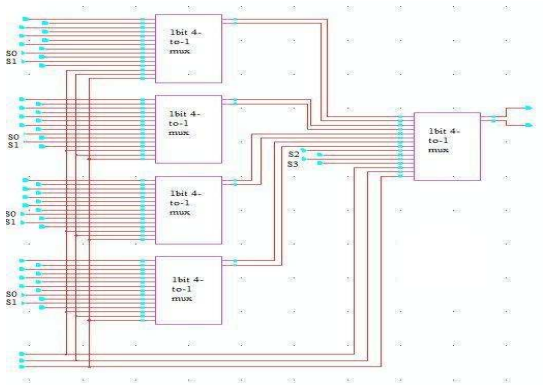

Figure 4.14: 1-bit 16-to-1 Mux Schematic

Simulated power consumption obtained for the 1-bit 16-to-1 mux is 2.8V * 86.92mA = 243.376mW. Expected area, from section 4.5, is 5 * (109.53μm x 47.52μm) = 547.65μm x

47.52μm. Simulated rise delay of 1-bit 4-to-1 mux is 52.51ps. Therefore expected rise delay of 1-bit 16-to-1 mux is 105.02ps, whereas simulated rise delay is 85.39ps, as indicated in Figure 4.15.

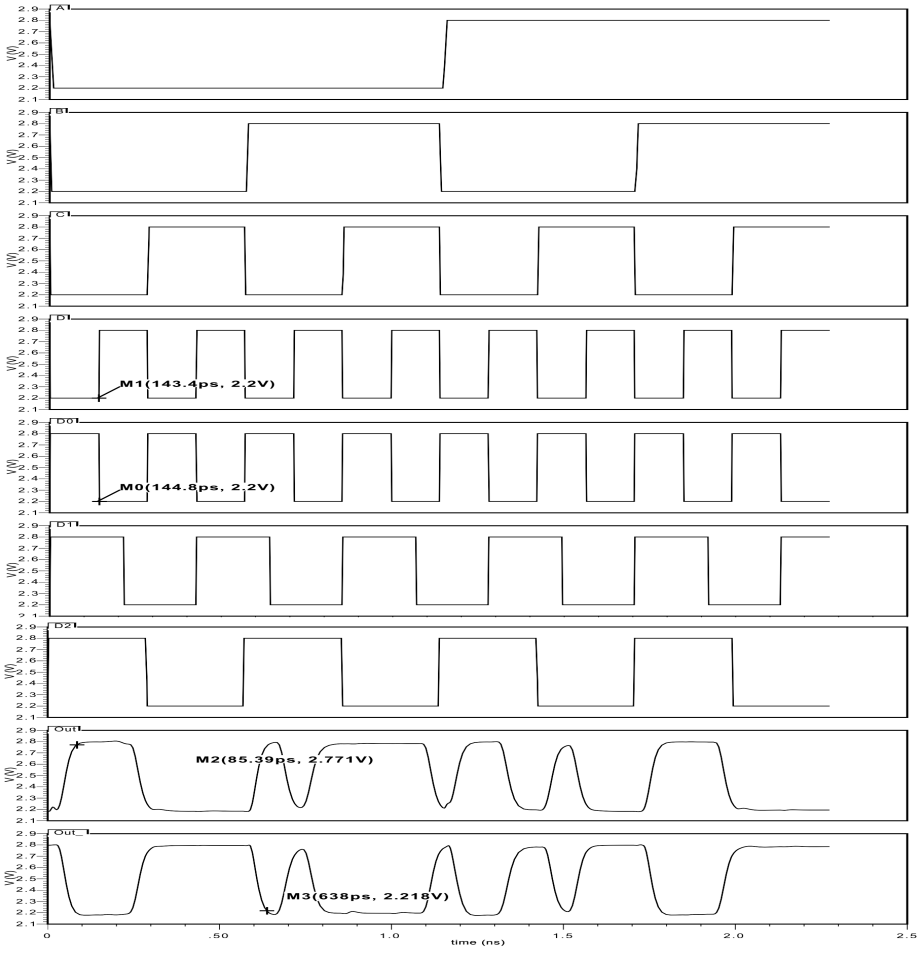

Figure 4.15: 1-bit 16-to-1 Mux Output (input changes at 144ps)

For simplicity, 3 inputs D0, D1, D2 have been used instead of 16 inputs. D0, D1 and D2 were connected to the first three inputs and fourth input was connected to D0, fifth input to D1, sixth input to D2, seventh input to D0 and so on and correct operation was verified at where intput changes at 144ps.

16-bit 16-to-1 mux power consumption is 16 * 243.376mW = 3.894016W and expected area is 16 * (547.65μm x 47.52μm) = 8762.4μm x 47.52μm with rise delay = 85.39ps.

Therefore, 3 16-bit 16-to-1 mux power consumption is 11.6872W with expected area of 26287.2μm x 47.52μm.

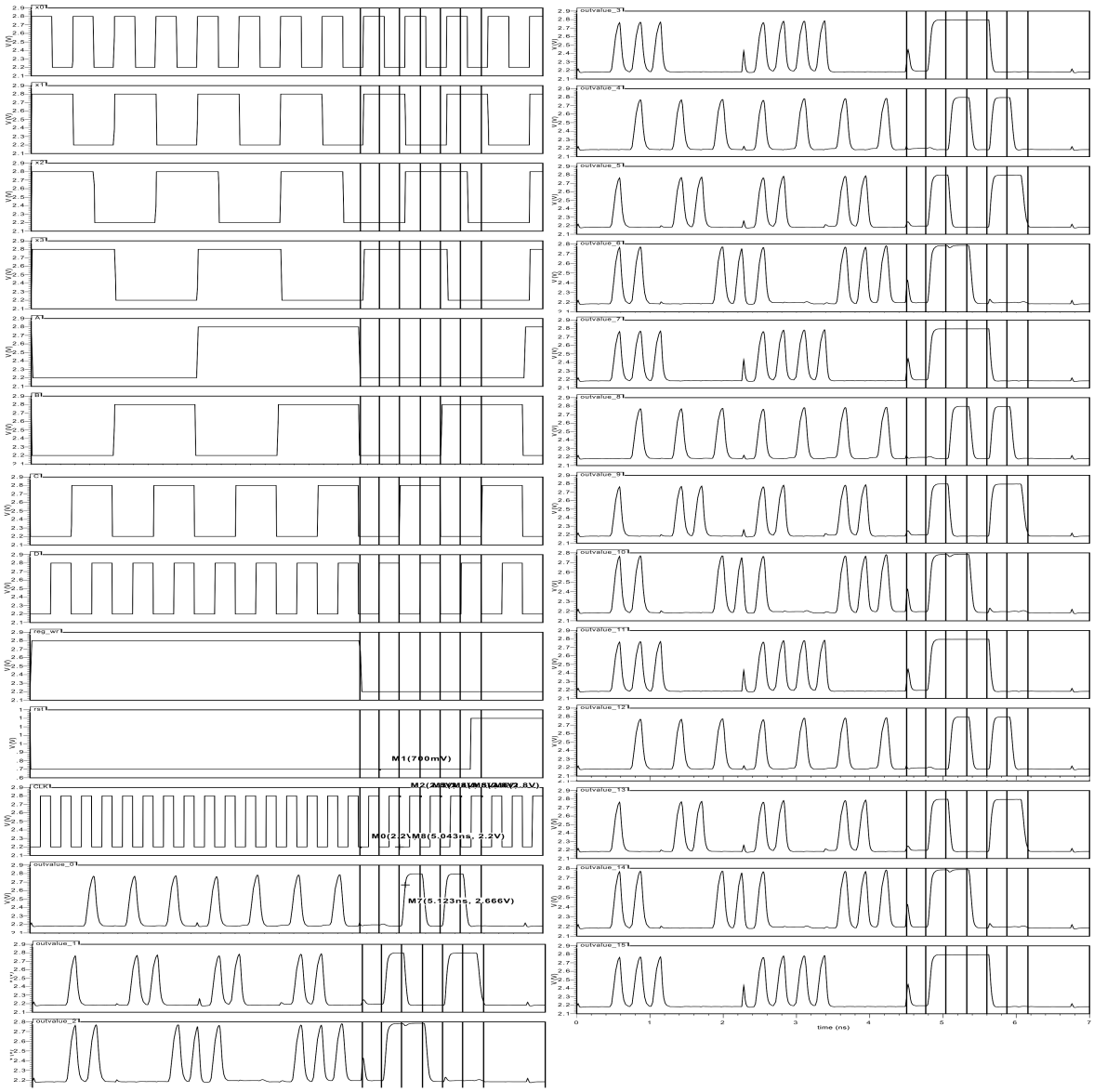

Figure 4.16: 16x16 Register File Output at 3.33GHz

16x16 register file path delay would be 4-to-16 decoder delay + 16-bit AND delay + 16-bit MS DFF-EN delay + 16-bit 16-to-1 mux delay = 72.01ps + 37.6ps + 101.41ps + 85.39ps = 296.41ps. Estimated power consumption of register file is 4-to-16 decoder + 16-bit AND + 16 16-bit register + 3 16-bit 16-to-1 mux = 778.96mW + 259.4368mW +

11.994W + 11.6872W = 24.719W. Estimated area of the 16x16 register file is 4-to-16 decoder area 1447.2μm x 42.9μm + 16-bit AND area 482.4μm x 42.9μm + 16 16-bit register area 24291.84μm x 57.78μm + 3 16-bit 16-to-1 mux area 26287.2μm x 47.52μm = 52508.64μm x 57.78μm.

Figure 4.16 shows verification of the 16x16 register file at 300ps clock period. An analog test bench was set up such that at the first clock cycle it writes FFFF in register-0 but as register-0 is implemented in hardware it will have 0 at the end. In consecutive cycles, data written on registers 1, 2, 3, 4, 5 up to 15 were EEEE, DDDD, 8888, 3333, 2222, 5555, 4444, FFFF, AAAA, 9999, 8888, 7777, 6666, 5555, 0000 and then control signal reg_wr goes low meaning write cannot be performed any more. Then rr1 = rr2 = rr3 = 0, 1, 2, 3, 4 and 5 was provided for simplicity and only 16-bit rd1 was probed to see the result at register-0, 1, .. 5 which were 0000, EEEE, DDDD, 8888, 3333 and 2222. Then in the following clock cycle reset = 1 sets all the outputs to be logic-0.

4.9 Sign 4-to-16 extension (Sign 4)

Sign 4 to 16 extension was developed using a single stage of sixteen inverters. First, 4-bits were passed to the inverter and the output was flipped to work as a buffer, due to the complementary form of CML logic. The fourth-bit was connected as input to rest of the inverters and those outputs were flipped as well.

Figure 4.17 shows the outputs of the sign 4 to 16 extension at where input changes at 72ps. The fourth-bit (D3) was copied to to input D4-D15 and fragment of output is shown Out0-Out7. Simulated power consumption of sign 4 to 16 extension is 2.8V * 26.4mA = 73.92mW and delay is same as 1-bit inverter but flipped, and simulation rise delay = 25.5ps. Expected area is 16 * (15.96μm x 24.45μm) = 255.36μm x 24.45μm.

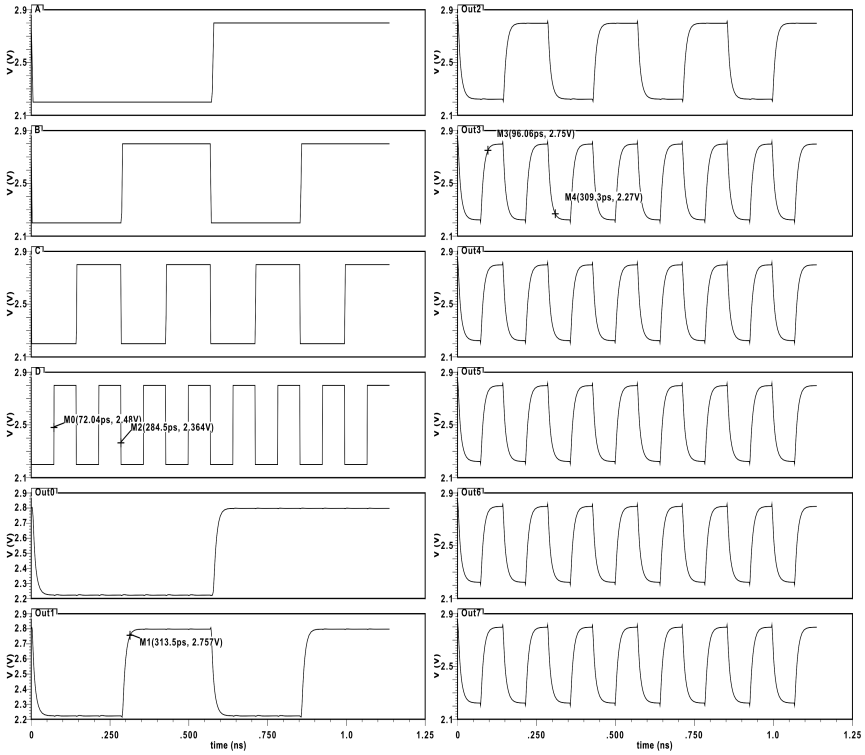

Figure 4.17: Sign 4 to 16 Extension Output (input changes at 72ps)

4.10 Sign 8-to-16 extension (Sign 8)

Sign 8 to 16 extension is similar to sign 4 to 16 extension, with the only difference being that the eighth bit (D7) is copied to D8-D15 bits. Area, power consumption and delay are the same as the sign 4-to-16 extension.

4.11 Sign 12-to-16 extension (Sign 12)

Sign 12 to 16 extension is similar to sign 4 to 16 extension with the only difference being that the twelfth bit (D11) is copied to D12 - D15 bits. Area, power consumption and delay are same as sign 4 to 16 extension.

4.12 2 unsigned 1-to-16 extension (Unsigned 16)

Unsigned 1-to-16 extension was developed using 16 inverters, similar to sign 4 to 16 extension. The only difference is the 1-bit input was connected to D0 and D1 - D15 were tied down to logic 0 (2.V). The output was flipped to work as a buffer, and area, power and delay are the same as sign 4 to 16 extension. Therefore 2 unsigned 1-to-16 extension power consumption is 2 * 73.92mW = 147.84mW and expected area is 2 * (255.36μm x 24.45μm) = 510.72μm x 24.45μm.

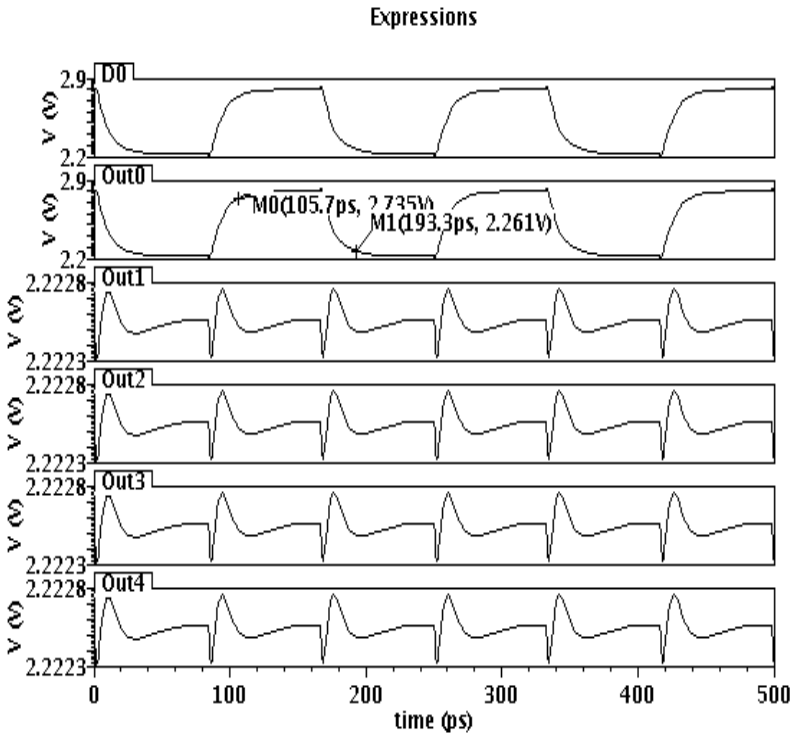

Figure 4.18: Unsigned 1 to 16 Extension Output (input changes at 83ps)

Figure 4.18 shows unsigned 1-to-16 extension output where output Q0 passes 1-bit input D0 and rest of the inputs are tied to Logic-0 which is 2.2V. The output glitch that is observed fluctuates only between 2.2228V - 2.2223V, 0.5mV only. The simulator tool usually zooms in when plotting output, meaning glitch is not as significant as it looks like. Only the first four bits of output are shown in Figure 4.17.

4.13 4 1-bit AND gate

4 1-bit AND gate power consumption is 4 * 16.2148mW = 64.8592mW and area is 4 * (30.15μm x 42.90μm) = 120.6μm x 42.9μm with rise delay = 37.6ps.

4.14 1 1-bit OR gate

One 1-bit OR gate power consumption is 16.2148mW having an area of 30.15μm x 42.90μm with rise delay = 46.09ps as stated in Table 2.6.

Table 4.1 summarizes power, expected area and time to generate correct operation for the complete datapath components. Simulation results presented in this table included 20fF load capacitance inside every basic components to mimic post layout simulation and wire capacitance.

Component	Power (mW)	Expected Area (μm x μm)	Delay (ps)
PC, PC+1, ALU_OUT, IR	2998.55	6072.96 x 57.78	101.41
Z	46.8524	94.89 x 57.78	101.41
16-bit 2-to-1 mux (4 used)	1038.464	2336.64 x 47.52	33.99
4-bit 2 to 1 mux (3 used)	194.712	438.12 x 47.52	33.99
16-bit 4 to 1 mux (3 used)	2335.872	5257.44 x 47.52	31.39
16-bit 5 to 1 mux	1038.464	2336.64 x 47.52	59.61
16-bit ALU	8243.2	14145.12 x 47.52	275.6
16x16 Reg file	24719	52508.64 x 57.78	296.41
Sign 4	73.92	255.36 x 24.45	25.5
Sign 8	73.92	255.36 x 24.45	25.5
Sign 12	73.92	255.36 x 24.45	25.5
Unsigned 16 (2 used)	147.84	510.72 x 24.45	25.5
1-bit AND gate (4 used)	64.8592	120.6 x 42.9	37.6
1 OR gate	16.2148	30.15 x 42.90	46.09
Total	41.065W	84618 x 57.78 = 2115.45 x 2311.2	296.41ps
Simulation Power & Frequency	41.264W	2.2mm x 2.3mm + 50% area for wiring = 2.2mm x 3.45mm	300ps

Table 4.1: Component Power Dissipation, Expected Area and Delay

As indicated in Table 4.1, estimated power obtained by summing up component power dissipation is 41.065W whereas simulation power obtained is 41.264W which is almost identical. Most of the power is dissipated in the 16x16 register file, which is 24.719W out of 41.264W. The 16x16 register file is also a time dominating component and critical path delay is 296.41ps. Register file was simulated at 300ps and has been verified as indicated in Figure 4.16. However as stated earlier, for 12 instruction ALU critical path delay is 275.6ps and for remaining 3 instruction where Z flag is used, critical path delay is 348ps.

Table 4.1 should be identical to post layout simulation although layout of the processor datapath has not been performed but in circuit level (transistor level). However to mimic post layout simulation and wire capacitance, 20fF load capacitance was added inside every building block. Typically post layout simulation varied with 10ps with circuit level as indicated in Chapter 2 and 1fF can mimic 1ps delay. Also, assuming 100μm is necessary to connect next stage then additional 10fF (0.1ps/μm) is necessary to mimic wire capacitance [22].

An attempt has been made to estimate total area, counting the number of basic component used + 50% area for wiring. Expected total area, including wire is, 2.2mm x 3.45mm (2200μm x 3450μm) leading power dissipation per unit area to be 41.264W / (2200 * 3450)μm^2 = 5.44μW/μm^2.

Chapter 5

Processor Verification and Performance

5.1 Processor Verification

A handcrafted 16-bit microprocessor datapath has been designed in CML logic as indicated in Figure 5.1. An analog test bench was set up to provide 16-bit data from memory using voltage sources (vpulse) and appropriate external control signals were provided to perform verification. Three instructions have been verified, providing consecutive data. At the first cycle reset was triggered and simulation was performed at 300ps (3.33GHz) clock period.

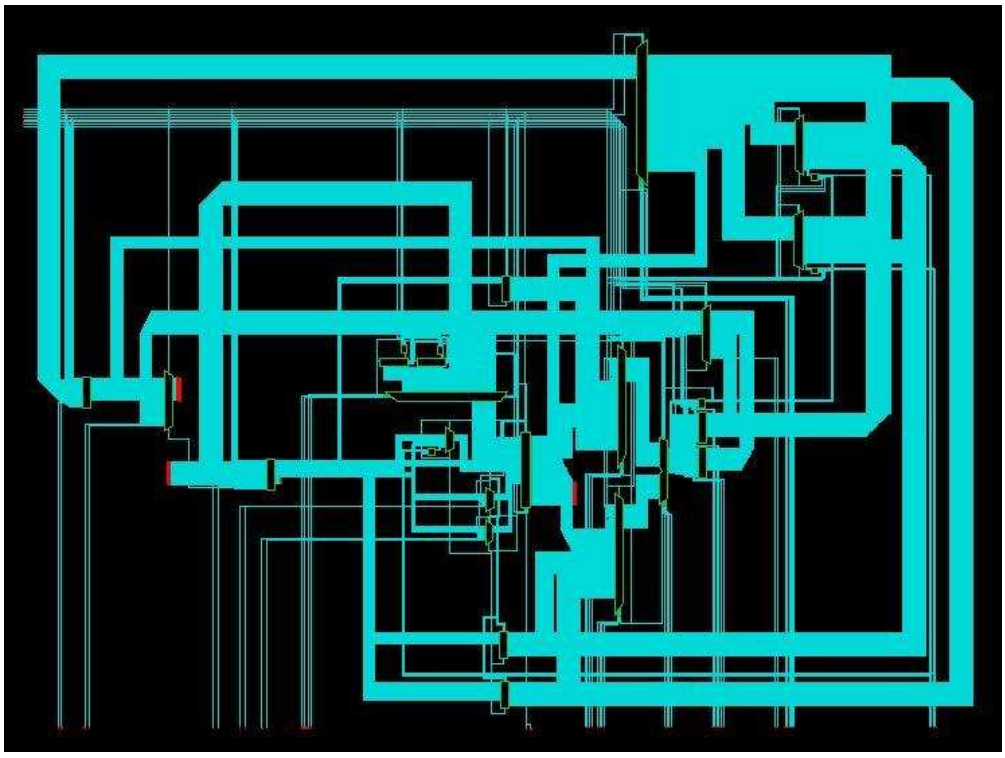

Figure 5.1: Handcrafted Processor Schematic

Each instruction took 4 clock cycles to execute and the result is provided in a sequence of Figure 5.2 - 5.4.

Instruction executed in cycle order:

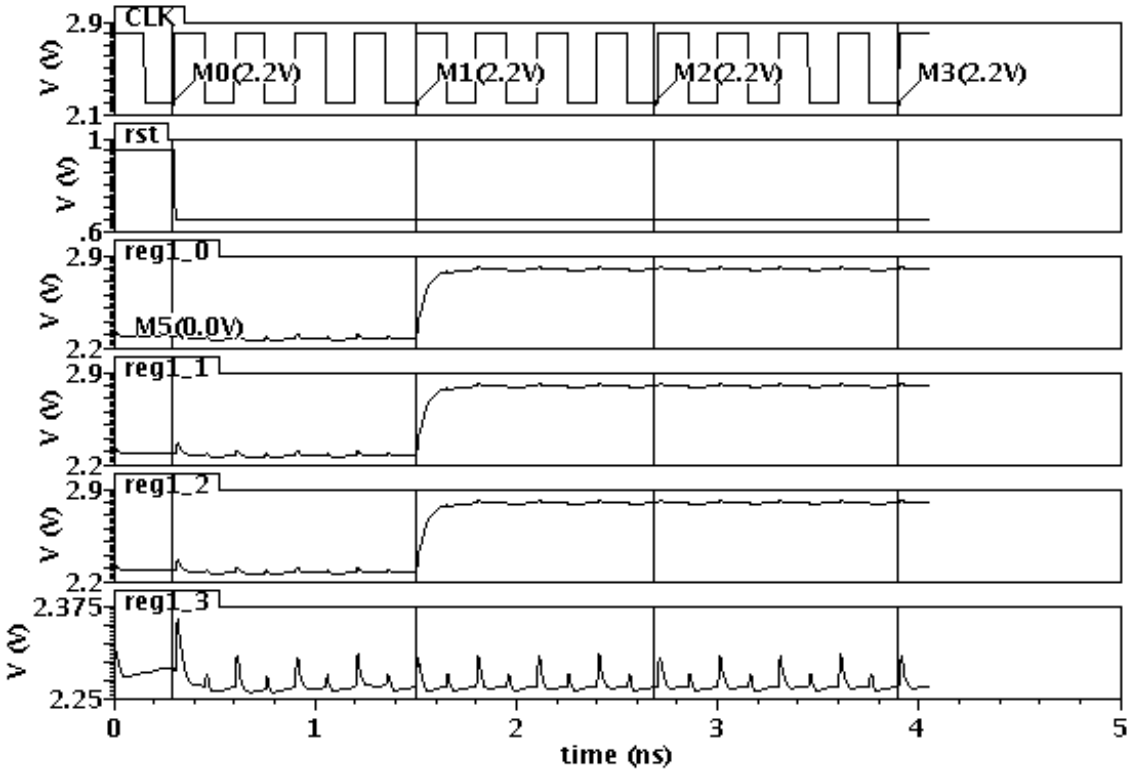

Figure 5.2: MOVI instruction

Reset (first cycle)

MOVI R_d = n; n 8-bit (I-type)

1001 0001 0001 0111 (applied at second cycle)

Explanation: R_d = Reg-1 = 23; move 23 in register-1. Appropriate external control signal has been applied at cycle = 2, 3, 4 and 5. Only selected signals of register-1 has been probed and expected result obtained at the beginning of sixth clock cycle as shown in Figure 5.2 (fifth bit of reg1_4 of register-1 is shown in Figure 5.3).

ADDI $R_d = R_s$ + n; n 4-bit (I-type)

1000 0111 0001 0111 (applied at sixth cycle)

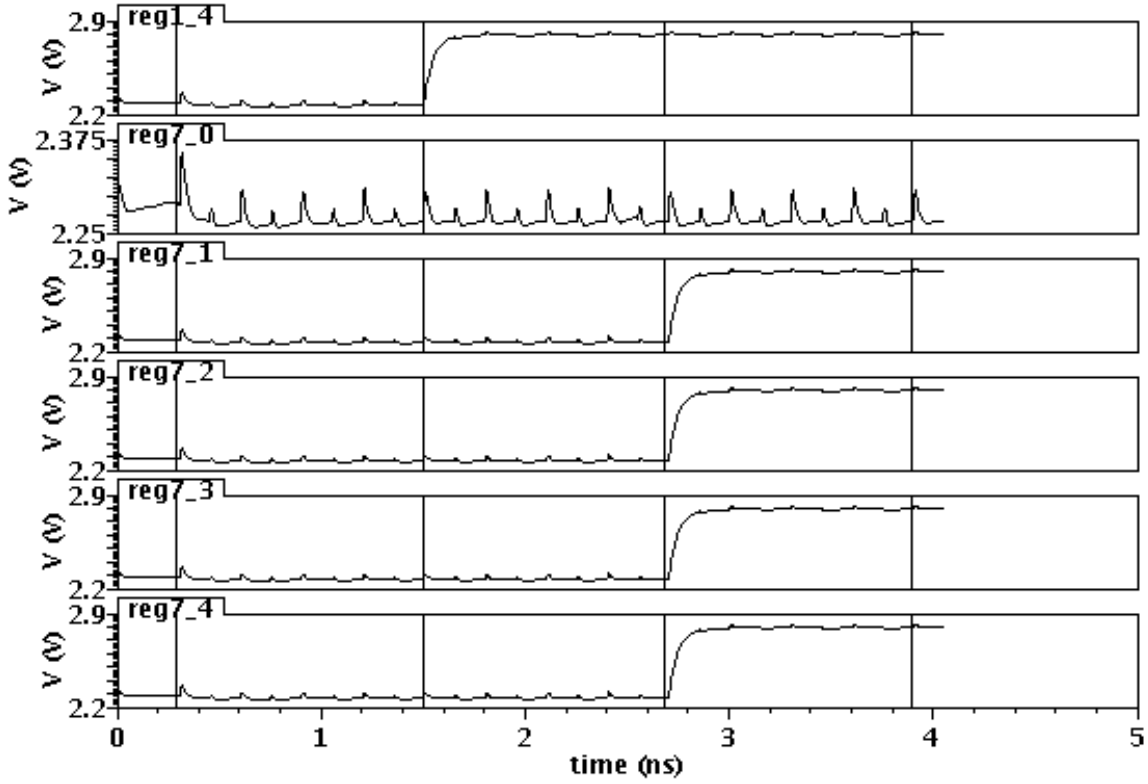

Figure 5.3: ADDI instruction

Explanation: $R_d = R_s$ + n; add content of register $R_s = 0001$ (register-1 contains 23 performed in previous instruction) to n = 0111 and save the result in $R_d = 0111$. Therefore register-7 will contain 23 + 7 = 30. Appropriate control signals were applied at cycle = 6, 7, 8 and 9. $R_d = 30 = 11110$ was observed at the beginning of tenth clock cycle, as indicated in Figure 5.3.

ADD $R_d = R_s + R_t$ (R-type)

0000 1101 0001 0111 (applied at tenth cycle)

Explanation: $R_d = R_s + R_t$; add content of register $R_s = 0001$ (register-1) with $R_t = 0111$ (register-7) and save the result in $R_d = 1101$ (register-13). Appropriate control signals were applied at cycle = 10, 11, 12 and 13 and content of $R_d = 13$ (register-13) was observed, 23 + 30 = 53 (110101) at the beginning of 14^{th} clock cycle, as indicated in Figure 5.4.

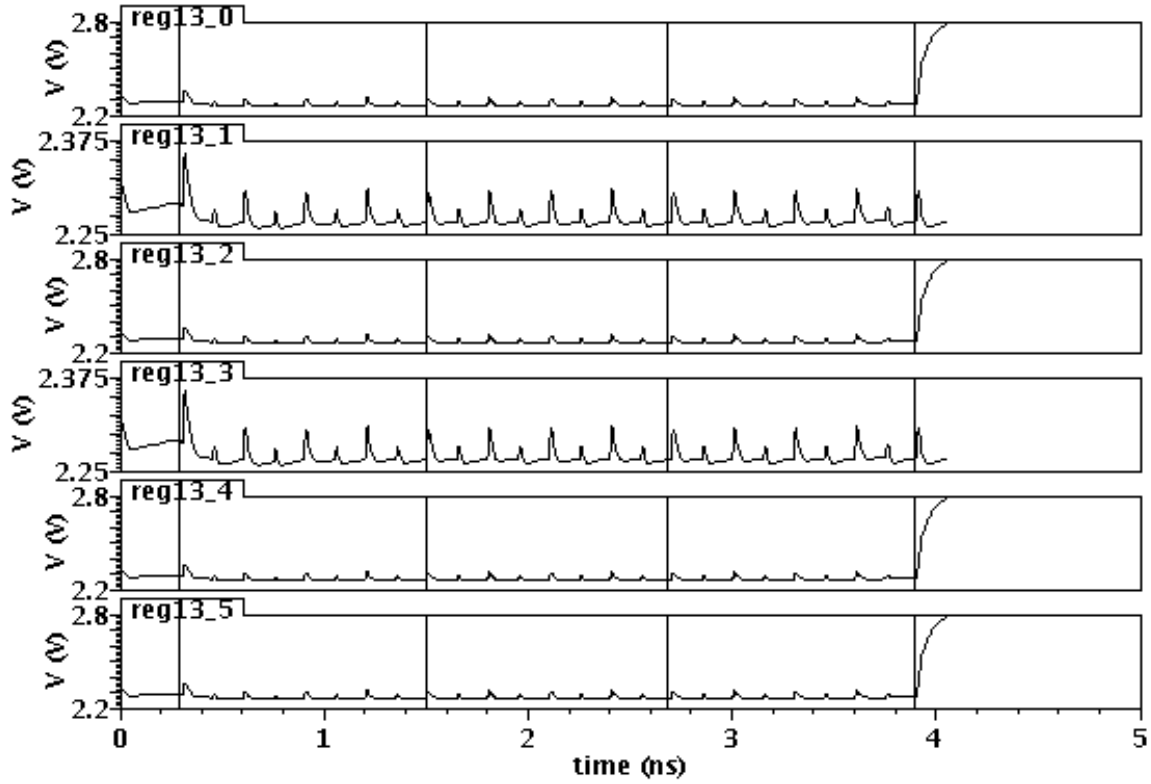

Figure 5.4: ADD instruction

Figure 5.5 shows static power consumption of CML processor datapath over 13 clock cycles (300ps clock period). Processor datapath peak power 41.51W reached at 306.5ps right after reset signal and average power dissipated is 41.264W, as indicated in Figure 5.4.

5.2 Performance

Performance metric MIPS was used to determine processor performance. 11 instructions take 4 clock cycles to execute, so the probability of executing any of these 11 instructions is 11/15. Similarly, 1 instruction (LW) takes 5 clock cycles to execute and thus probability is 1/15. Likewise, 1 instruction (JAL) takes 3 clock cycles to execute so probability is 1/15. Two instructions (J LABEL, JR) take 2 clock cycles to execute and probability is 2/15.

It has been stated earlier that 3 instructions (SEQ, JZ, and JNZ) require processor speed to be slowed down to operate at 2.87GHz (348ps) due to worst path delay. However

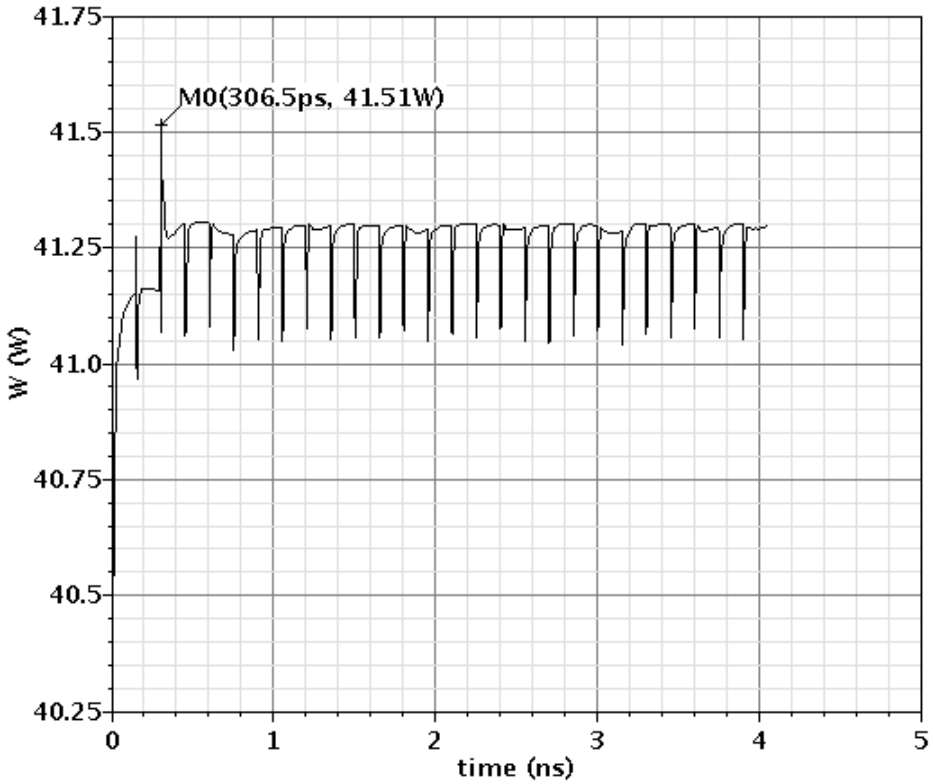

Figure 5.5: Static power consumption of CML processor datapath over 13 clock cycles

this problem can be solved by re-structuring the datapath. Assuming each instruction can be performed at 3.33GHz (300ps), estimated clock per instruction (CPI) is:

$$\begin{aligned} CPI &= 4 * \frac{11}{15} + 5 * \frac{1}{15} + 3 * \frac{1}{15} + 2 * \frac{2}{15} \\ &= 3.733 \end{aligned}$$

Assuming 1 million instructions have been executed,

$$\begin{aligned} Execution\ time &= No.\ of\ instruction\ count * clock\ period * CPI \\ &= 10^6 * 300ps * 3.733\ clock/instruction \\ &= 1.1199 * 10^{-3} sec \end{aligned}$$

$$
\begin{aligned}
MIPS &= \frac{Instruction\ count}{Execution\ time * 10^6} \\
&= \frac{10^6}{1.1199 * 10^{-3} sec * 10^6} \\
&= 892.93
\end{aligned}
$$

5.3 Comparison

There is no common ground to compare the CML processor with a CMOS processor. The same feature size CMOS can never operate as fast CML logic due small swing and differential architecture. The smaller swing we define, the faster we can operate CML logic. In CML there is no dynamic power, but only static power dissipation, and it is constant and does not increase with increasing frequency as stated earlier in Chapter 2. The highest operating frequency for a particular technology (minimum feature size) can be achieved through design intuition such that differential pair can just turn-off one side and steer full current in the other part based on input combinations that benefit CML over CMOS at higher frequency.

These days, Intel's latest processors in 90nm, 65nm, 32nm process technology can achieve 2-4GHz operating frequency with typical power consumption 150W - 200W and MIPS varies from 6000 - 10000 on an average [3] and [23]. The CML processor that was developed in this thesis work has been implemented at the circuit level with additional load capacitance to mimic post layout simulation and wire delay. Hardware/software co-ordination has not been performed, nor has a compiler been designed to implement program code (assembly language) to measure execution time. However, theoretical estimation shows the developed CML processor has MIPS = 892 with power consumption 41.264W. Today's processors are mostly pipelined and superscalar in architecture. Re-structuring the multi-cycle datapath of the CML processor into a pipeline can gain at least 4 times the MIPS

(3568), and making it superscalar can exceed 10000 MIPS while burning 50-70W in 120nm feature size.

Comparing just a single CMOS gate to CML gate is misleading. It is because of

1. Swing is not same. In CML we can define 100mV or 600mV swing, whatever we want, based on our requirement. Making the swing 100mV will burn less power than a 600mV swing. But in CMOS, swing is fixed, 1V.

2. At higher frequency, as long as our swing is less than 1V, CML is advantageous over CMOS.

3. At lower frequency (below 1GHz) it depends. Typically due to constant power dissipation CML is a worse choice than CMOS. Reducing swing to 100mV or less may benefit CML over CMOS at lower frequency.

4. Same feature size CMOS cannot operate as fast as CML. Therefore, in order to compare the two at a particular operating frequency we need smaller feature size for CMOS and larger feature size for CML, and thus breaking the common ground to compare.

5. One possible way to compare CMOS with CML at the same feature size is not to optimize CML at its highest operating frequency but at some lower frequency, where CMOS is capable to operate at, meaning frequency and feature size are the same. But in this case we are deliberately letting CML lose its benefits. As stated in Figure 2.1, CML power consumption does not increase exponentially, but rather slowly, similar to a horizontal line. Optimizing for lower frequency (where CMOS is capable to operate) may reduce some power but not too much. CML's higher constant power dissipation at lower frequency will let CMOS gate to be favorable because CMOS static power is almost zero and dynamic power will reduce due to lower operating frequency.

Chapter 6

Conclusions

The first ever, 3.33GHz MCML microprocessor datapath has been developed using 130nm CMOS technology. No prior work exists in literature in designing processor datapaths using MCML logic. It is first in its kind and power consumption is very low compared to traditional CMOS processors. However, a prior attempt was made to develop a BiCMOS superscalar RISC microprocessor where three level ECL logic gates have been used along with CMOS gates in same integrated chip [5].

Reported power consumption of the developed MCML processor is 41.264W with an estimated area of 2.2mm x 3.45m. Expected power density per unit area is $5.44\mu W/\mu m^2$. A RISC architecture was adopted in developing the 16-bit processor datapath. Out of fifteen instructions, twelve instructions can be performed at 3.33GHz and three instructions can be performed at 2.87GHz, with an estimated MIPS of 892.

This thesis work indicates that it is possible to realize superfast processors beyond 20GHz with minimum power consumption using today's technology. Either a voltage conversion is necessary that can be done by amplifying small signal swings to 1V and then shifted down or the total system could be implemented in CML to work with CML processor [24].

6.1 Future Work

It is possible to come up with a CML synthesizer tool such that the entire design can be automated. The idea behind the statement is, CML gates which are analog can be optimized for multiple operating frequencies and multi-input CML logics by proper handcrafting, similar to digital technology files. Implementing multi-input logic has two fold benefits; delay will be reduced but power consumption will be the same. However to provide enough biasing

for each level of transistors, we have to increase our supply voltage. Once CML logics for multi-input and multiple frequencies have been optimized by proper handcrafting, the rest of the process can be automated as is the case for digital circuits. A CML synthesizer tool should be able to pick different optimized CML logic such that it meets the area and timing constraints. It could be a time oriented interested topic for any PHD student in mixed signal design.

Bibliography

[1] Vasanth Kakani, Foster F. Dai and R.C Jaeger, "Delay analysis and optimal biasing for high speed low power current mode logic circuits," IEEE International Symposium on Circuits and Systems, vol. 2, pp. II-869-872, May 23-26, 2004.

[2] Dr. Fa Foster Dai's, "ELEC 6190 Introduction to Digital and Analog IC Design," Slide 10, Page 22.

[3] Wikipedia, "`http://en.wikipedia.org/wiki/List_of_CPU_power_dissipation`".

[4] Dr. Vishwani D. Agrawal's, "ELEC 6270 Low Power Design of Electronic Circuit," Slide no 1, page 6.

[5] Cliff A. Maier, James A. Markevitch, Tim Sippel, Earl T. Cohen, Jim Blomgren, James G. Ballard, Jay Pattin, Viki Moldenhauer, Jeffrey A. Thomas and George Taylor, "A 533-MHz BiCMOS superscalar RISC microprocessor," IEEE Journal of Solid-State Circuits, vol. 32, pp. 16251634, Nov. 1997.

[6] Armin Tajalli, Eric Vittoz and Yusuf Leblebici, "Ultra low power subthreshold MOS current mode logic circuits using a novel load device concept," European Solid State Circuits Conference, pp. 304-307, Sept. 11-13, 2007.

[7] M. Alioto and G. Palumbo, "Modeling and optimized design of current mode MUX / XOR and D-flip flop," IEEE Transactions of Circuits and Systems-II, vol. 47, no. 5, pp. 452-461, May 2000.

[8] Dr. Fa Foster Dai's, "ELEC 6190 Introduction to Digital and Analog IC Design," Slide 7, Page 4, 10, 11, 16, 17, 22.

[9] Abdullah Al Owahid and Foster F. Dai, "A 41.264W, 3.33GHz Processor Datapath Using Current Mode Logics In 130nm CMOS Technology, "IEEE International Symposium on Circuits and Systems, May 20-23, 2012.

[10] H. Rein, "Design consideration for very-high-speed Si-bipolar ICs operating up to 50 Gb/s," IEEE Journal of Solid-State Circuits, vol. 31, pp. 10761090, Aug. 1996.

[11] "International Technology RoadMap for Semiconductors," ITRS, Radio Frequency and Analog/Mixed-Signal Technologies for Wireless Communications Tech. Rep., 2005.

[12] S. Khabiri and M. Shams, "A mathematical programming approach to designing MOS current-mode logic circuits," in Proc. IEEE ISCAS, May 2005, vol. 3, pp. 24252428.

[13] H. Hassan, M. Anis, and M. Elmasry, "MOS current mode circuits: Analysis, design, and variability," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 13, no. 8, pp. 885898, Aug. 2005.

[14] J. Musicer and J. Rabaey, "MOS current mode logic for low power, low noise CORDIC computation in mixed-signal environments," in Proc. ISLPED, 2000, pp. 102107.

[15] A. Tanabe, M. Umetani, I. Fujiwara, T. Ogura, K. Kataoka, M. Okihara, H. Sakuraba, T. Endoh, and F. Masuoka, "A 10 Gb/s demultiplexer IC in 0.18 um CMOS using current mode logic with tolerance to the threshold voltage fluctuation," IEEE J. Solid-State Circuits, vol. 36, no. 6, pp. 988996, Jun. 2001.

[16] M. Allam and M. Elmasry, "Dynamic current mode logic (DyCML): A new low-power high-performance logic style," IEEE J. Solid-State Circuits, vol. 36, no. 3, pp. 550558, Mar. 2001.

[17] M. Yamashima and H. Yamada, "A MOS current mode logic (MCML) circuit for low-power sub-GHz processors," IECE Trans. Electron., vol. E75-C, no. 10, pp. 11811187, Oct. 1992.

[18] Dr. Fa Foster Dai's "ELEC 6190 Introduction to Digital and Analog IC Design," Slide 7, Page 2.

[19] M. Alioto and G. Palumbo, "Design strategies for source coupled logic gates," IEEE Trans. Circuits Syst. I, Fundam. Theory Appl., vol. 50, no. 5, pp. 640654, May 2003.

[20] A. Ismail and M. Elmasry, "A low power design approach for MOS current mode logic," in Proc. IEEE Int. SOC Conf., Sep. 2003, pp. 143146.

[21] Osman Musa and Maitham Shams, "An efficient delay model for MOS current mode logic automated design and optimization," IEEE Transactions on Circuits and Systems-I: Regular Papers, vol. 57, no. 8, August 2010.

[22] Xueyang Geng, Fa Foster Dai, J. David Irwin and Richard C. Jaeger, "24-Bit 5.0 GHz Direct Digital Synthesizer RFIC With Direct Digital Modulations in 0.13μm SiGe BiCMOS Technology," IEEE Journal of Solid-State Circuits, vol. 45, no. 5, pp. 944-954, May 2010.

[23] Wikipedia, "`http://en.wikipedia.org/wiki/List_of_Intel_microprocessors`".

[24] Xuelian Liu, Hadrian O. Aquino, Alexey Gutin and John. Mcdonald, "A 125-ps Access, 4GHz, 16KB BICMOS SRAM," IEEE International Midwest Symposium on Circuits and Systems, pp. 1222-1225, Aug 2010.