

ESE 545 | PROJECT

MICRO-ARCHITECTURE OF SYNERGISTIC PROCESSING UNIT OF SONY CELL

Aswin Natesh Venkatesh &
Salome Devkule

SBU ID: 111582677
SBU ID: 111678929

DESIGN IMPLEMENTATION

ABSTRACT:

This report details the implementation of a CELL SPU, a Dual-Issue Multimedia Processor Architecture. The design consists of the Register file, two pipes (odd & even) with multiple pipelined processing units, local memory and data forwarding circuits modelled using System Verilog.

An instruction sub-set consisting of 73 Instructions from Cell SPU (Synergetic Processing Unit) has been defined and tested for correct functionality. Additional features such as forwarding for data hazard and structural hazard elimination are also added. We have also implemented branch prediction to improve the performance. We discuss the performance of the processor for a standard program such as the matrix multiplication

SPU PROCESSOR CORE ARCHITECTURE:

The SPU processor core consists of two processing pipes, register file, and data forwarding circuits which is represented the Architecture block diagram. Each of the core units are explained below: -

Register File: The register file consists of 6 Read Ports and 2 Write Ports and has 128 entries with 128 bits each. The Register file has a latency of 1 clock cycle. Data from the register file is directly sent to even/odd pipes for execution and the result from the pipes are written back to the destination address after completion.

Fwd Circuit: The Instructions (Opcode & Immediate Values) are forwarded to the Register File for forwarding data between pipes after their specific latencies.

Even Pipe : The even pipe handles the following operational units as listed below:-

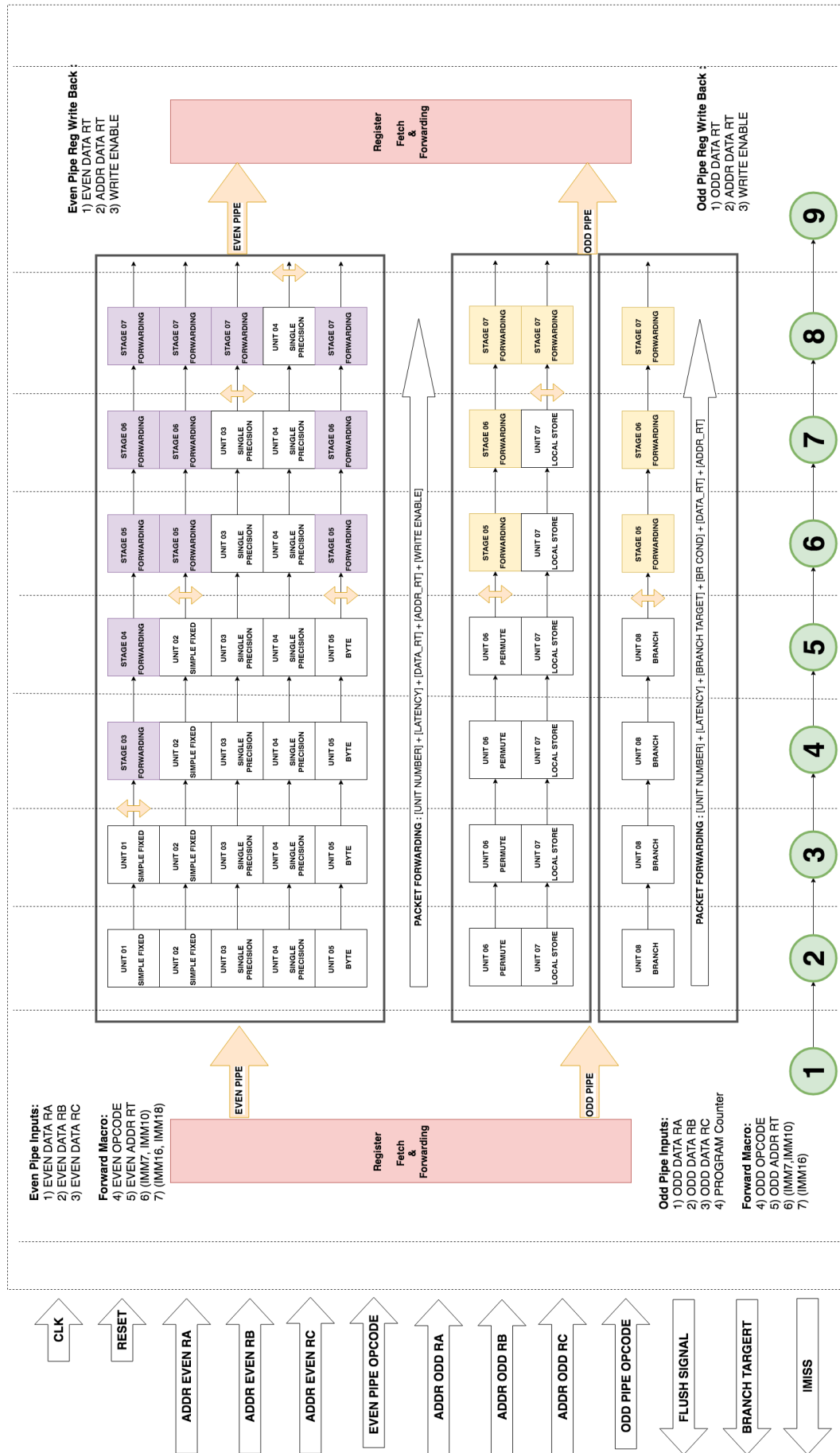
1. Simple Fixed -1 [Unit # - 01] [Latency - 2]
2. Simple Fixed -2 [Unit # - 02] [Latency - 4]
3. Single Precision [Unit # - 03] [Latency-6,7]
4. Byte [Unit # - 04] [Latency - 4]

Odd Pipe : The odd pipe handles the following operational units as listed below:-

1. Branch [Unit # - 07] [Latency - 4]
2. Permute [Unit # - 05] [Latency - 4]
3. Local Store [Unit # - 06] [Latency - 6]

At every clock pulse two instructions can be executed simultaneously by the odd and even pipe. The instructions are categorised based on their latencies, and have been numbered. All of these instructions passes through a 7 stage pipeline and are available for usage once they cross their respective latency period. This is illustrated by vertical arrow marks in the architecture block diagram.

The SPU Core Processing Unit Data path and Control Path is Shown in the next page.



PIPELINED PROCESSOR DESIGN (TOTAL 12 CYCLES):

The second part of the project involves the development of a dual-issue pipelined version of the multi-stage pipelined multimedia processor. The processor consists of 32kB of Local Store Memory, and 4kB of Cache (Direct Mapped). The Processor is 12 stage pipelined and each individual stages are mentioned below.

Stage 1: Fetch Stage

Stage 2: Decode Stage (Handles Structural Hazard and WAW Data Hazard)

Stage 3: Dependency Stage (Handles WAR Data Hazards)

Stage 4: Fetch from Register File

Stage 5: Even Pipe & Odd Pipe (Pipeline Stage 1)

Stage 6: Even Pipe & Odd Pipe (Pipeline Stage 2)

Stage 7: Even Pipe & Odd Pipe (Pipeline Stage 3)

Stage 8: Even Pipe & Odd Pipe (Pipeline Stage 4)

Stage 9: Even Pipe & Odd Pipe (Pipeline Stage 5)

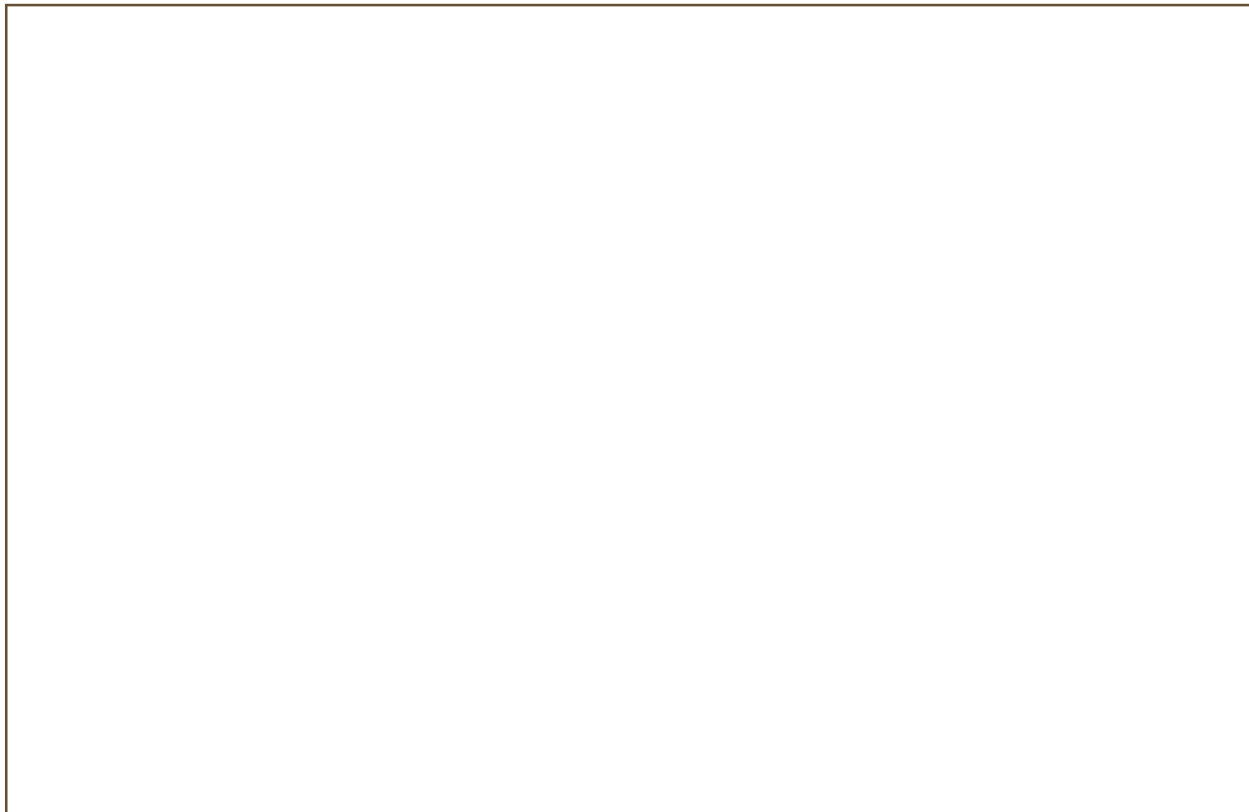
Stage 10: Even Pipe & Odd Pipe (Pipeline Stage 6)

Stage 11: Even Pipe & Odd Pipe (Pipeline Stage 7)

Stage 12: Register File Write Back

Thus, each stage of the pipeline, now called a pipe will be performing some action on some instruction.

The Control Path of the Pipelined basic pipelined model looks as follows:



INSTRUCTION SET:

Listed below are 73 instructions have been implemented and tested for functionality.

Simple Fixed (1) Instructions:

S.No	Mnemonic	Description	Instruction Operands	Opcode	Pipe	Unit	#	Latency
1	a	Add Word	rt, ra, rb	0001_1000_000	Even	Simple Fixed - 1	1	2
2	addx	Add Extended	rt, ra, rb	1101_0000_00	Even	Simple Fixed - 1	1	2
3	ai	Add Word Immediate	rt,ra,value	0001_1100	Even	Simple Fixed - 1	1	2
4	and	And	rt, ra, rb	0001_1000_001	Even	Simple Fixed - 1	1	2
5	andbi	And Byte Immediate	rt,ra,value	0001_0110	Even	Simple Fixed - 1	1	2
6	andc	And with Complement	rt, ra, rb	0101_1000_001	Even	Simple Fixed - 1	1	2
7	ceq	Compare Equal Word	rt, ra, rb	0111_1000_000	Even	Simple Fixed - 1	1	2
8	ceqb	Compare Equal Byte	rt, ra, rb	0111_1010_000	Even	Simple Fixed - 1	1	2
9	ceqbi	Compare Equal Byte Immediate	rt,ra,value	0111_1110	Even	Simple Fixed - 1	1	2
10	ceqi	Compare Equal Word Immediate	rt,ra,value	0111_1100	Even	Simple Fixed - 1	1	2
11	cgtb	Compare Greater Than Byte	rt, ra, rb	0100_1010_000	Even	Simple Fixed - 1	1	2
12	cgtbi	Compare Greater Than Byte Immediate	rt,ra,value	0100_1110	Even	Simple Fixed - 1	1	2
13	cgti	Compare Greater Than Word Immediate	rt,ra,value	0100_1100	Even	Simple Fixed - 1	1	2
14	clgtb	Compare Logical Greater Than Byte	rt, ra, rb	0101_1010_000	Even	Simple Fixed - 1	1	2
15	clgtbi	Compare Logical Greater Than Byte Immediate	rt,ra,value	0101_1110	Even	Simple Fixed - 1	1	2
16	il	Immediate Load Word	rt, symbol	0100_0000_1	Even	Simple Fixed - 1	1	2
17	ila	Immediate Load Address	rt, symbol	0100_001	Even	Simple Fixed - 1	1	2
18	ilh	Immediate Load Halfword	rt, symbol	0100_0001_1	Even	Simple Fixed - 1	1	2
19	nand	Nand	rt, ra, rb	0001_1001_001	Even	Simple Fixed - 1	1	2
20	nor	Nor	rt, ra, rb	0000_1001_001	Even	Simple Fixed - 1	1	2
21	or	Or	rt, ra, rb	0000_1000_001	Even	Simple Fixed - 1	1	2
22	orc	Or with Complement	rt, ra, rb	0101_1001_001	Even	Simple Fixed - 1	1	2
23	ori	Or Word Immediate	rt,ra,value	0000_0100	Even	Simple Fixed - 1	1	2
24	sf	Subtract from Word	rt, ra, rb	0000_1000_000	Even	Simple Fixed - 1	1	2
25	sfi	Subtract from Word Immediate	rt,ra,value	0000_1100	Even	Simple Fixed - 1	1	2
26	sfx	Subtract from Extended	rt, ra, rb	0110_1000_001	Even	Simple Fixed - 1	1	2
27	xor	Exclusive Or	rt, ra, rb	0100_1000_001	Even	Simple Fixed - 1	1	2
28	xorbi	Exclusive Or Byte Immediate	rt,ra,value	0100_0110	Even	Simple Fixed - 1	1	2
29	xori	Exclusive Or Word Immediate	rt,ra,value	0100_0100	Even	Simple Fixed - 1	1	2
30	xsbh	Extend Sign Byte to Halfword	rt,ra	0101_0110_110	Even	Simple Fixed - 1	1	2
31	xshw	Extend Sign Halfword to Word	rt,ra	0101_0101_110	Even	Simple Fixed - 1	1	2
32	xswd	Extend Sign Word to Doubleword	rt,ra	0101_0100_110	Even	Simple Fixed - 1	1	2

Byte Instructions:

S.No	Mnemonic	Description	Instruction Operands	Opcode	Pipe	Unit	#	Latency
33	absdb	Absolute Differences of Bytes	rt, ra, rb	0000_1010_011	Even	Byte	5	4
34	avgb	Average Bytes	rt, ra, rb	0001_1010_011	Even	Byte	5	4
35	cntb	Count Ones in Bytes	rt, ra	0101_0110_100	Even	Byte	5	4
36	sumb	Sum Bytes into Halfwords	rt, ra, rb	0100_1010_011	Even	Byte	5	4

Simple Fixed (2) Instructions:

S.No	Mnemonic	Description	Instruction Operands	Opcode	Pipe	Unit	#	Latency
37	rot	Rotate Word	rt, ra, rb	0000_1011_000	Even	Simple Fixed-2	2	4
38	rotm	Rotate and Mask Word	rt, ra, rb	0000_1011_001	Even	Simple Fixed-2	2	4
39	shl	Shift Left Word	rt, ra, rb	0000_1011_011	Even	Simple Fixed-2	2	4
40	shli	Shift Left Word Immediate	rt,ra,value	0000_1111_011	Even	Simple Fixed-2	2	4

Branch Instructions:

S.No	Mnemonic	Description	Instruction Operands	Opcode	Pipe	Unit	#	Latency
41	bi	Branch Indirect	ra	0011_0101_000	Odd	Branches	8	4
42	bisl	Branch Indirect and Set Link	rt, ra	0011_0101_001	Odd	Branches	8	4
43	br	Branch Relative	symbol	0011_0010_0	Odd	Branches	8	4
44	bra	Branch Absolute	symbol	0011_0000_0	Odd	Branches	8	4
45	brasl	Branch Absolute and Set Link	rt, symbol	0011_0001_0	Odd	Branches	8	4
46	brsl	Branch Relative and Set Link	rt, symbol	0011_0011_0	Odd	Branches	8	4
68	brnz	Branch if Not Zero Word	rt, symbol	0010_0001_0	Odd	Branches	8	4
69	brz	Branch if Zero Word	rt, symbol	0010_0000_0	Odd	Branches	8	4

Permute Instructions:

S.No	Mnemonic	Description	Instruction Operands	Opcode	Pipe	Unit	#	Latency
47	rotqby	Rotate Quadword by Bytes	rt, ra, rb	0011_1011_100	Odd	Permute	6	4
48	rotqbyi	Rotate Quadword by Bytes Immediate	rt, ra, value	0011_1111_100	Odd	Permute	6	4
49	rotqmby	Rotate and Mask Quadword by Bytes	rt, ra, rb	0011_1011_101	Odd	Permute	6	4
50	rotqmbyi	Rotate and Mask Quadword by Bytes Immediate	rt, ra, value	0011_1111_101	Odd	Permute	6	4
51	shlqby	Shift Left Quadword by Bytes	rt, ra, rb	0011_1011_111	Odd	Permute	6	4
52	shlqbyi	Shift Left Quadword by Bytes Immediate	rt, ra, value	0011_1111_111	Odd	Permute	6	4

Local Store Instructions:

S.No	Mnemonic	Description	Instruction Operands	Opcode	Pipe	Unit	#	Latency
58	lqa	Load Quadword (a-form)	rt, symbol	0011_0000_1	Odd	Local Store	7	6
59	lqx	Load Quadword (x-form)	rt, ra, rb	0011_1000_100	Odd	Local Store	7	6
60	stqa	Store Quadword (a-form)	rt, symbol	0010_0000_1	Odd	Local Store	7	6
61	stqx	Store Quadword (x-form)	rt, ra, rb	0010_1000_100	Odd	Local Store	7	6

Single Precision Instructions:

S.No	Mnemonic	Description	Instruction Operands	Opcode	Pipe	Unit	#	Latency
53	fa	Floating Add	rt, ra, rb	0101_1000_100	Even	Single Precision	3	6
54	fm	Floating Multiply	rt, ra, rb	0101_1000_110	Even	Single Precision	3	6
55	fma	Floating Multiply and Add	rt, ra, rb,rc	1110_	Even	Single Precision	3	6
56	fms	Floating Multiply and Subtract	rt, ra, rb,rc	1111_	Even	Single Precision	3	6
57	fs	Floating Subtract	rt, ra, rb	0101_1000_101	Even	Single Precision	3	6
62	mpy	Multiply	rt, ra, rb	0111_1000_100	Even	Single Precision	4	7
63	mpya	Multiply and Add	rt, ra, rb,rc	1100_	Even	Single Precision	4	7
64	mpyi	Multiply Immediate	rt,ra,value	0111_0100	Even	Single Precision	4	7
65	mpys	Multiply and Shift Right	rt, ra, rb	0111_1000_111	Even	Single Precision	4	7
66	mpyu	Multiply Unsigned	rt, ra, rb	0111_1001_100	Even	Single Precision	4	7
67	mpyui	Multiply Unsigned Immediate	rt,ra,value	0111_0101	Even	Single Precision	4	7

Control Instructions:

S.No	Mnemonic	Description	Instruction Operands	Opcode	Pipe	Unit	#	Latency
70	lnop	No Operation (Execute)	rt	0100_0000_001	Odd	No-Op	-	1
71	nop	No Operation (Execute)	rt	0100_0000_001	Even	No-Op	-	1
72		Stop & Signal	-	-	-	-	-	-
73	lmiss	IMISS	-	-	-	-	-	-

TESTBENCH & VERIFICATION METHODOLOGY (SPU LITE MODEL CODE)

The test-bench passes all required inputs for the system to execute two instructions simultaneously by both the pipes, and monitors the output produced after the register file write back stage. The test-bench is scripted in System Verilog and follows simple code coverage method to test all the 73 instructions and their functionalities including control statements. The test-bench also saves all the inputs passed and outputs received to a file with the time stamp which is used to observe and cross check the results produced. This file provides detailed report about the results produced and is attached with this report for reference. Sample Inputs/outputs are tabulated below for all Instruction Units.

Simple Fixed Unit 1:

Instruction	Inputs	Outputs
Add Word	EVEN INSTRUCTION Add Word >> Input at : 15ns >> Output at : 95ns -> Opcode_Even = 00011000000 -> Addr_Even_Ra = 20 -> Addr_Even_Rb = 25 -> Addr_Even_Rc = 0 -> Imm7_Even = 0 -> Imm10_Even = 0 -> Imm16_Even = 0 -> Imm18_Even = 0	# CLOCK IN NS: 95ns # EVEN PIPE RESULT >>> # Addr_Rt2= 50 # Data_Rt= 45 # Data_Rt(hex)= 00000000000000000000000000002d # Wr_en= 1
Compare Greater Than Byte Immediate	EVEN INSTRUCTION Compare Greater Than Byte Immediate >> Input at : 125ns >> Output at : 205ns -> Opcode_Even = 00001001110 -> Addr_Even_Ra = 20 -> Addr_Even_Rb = 0 -> Addr_Even_Rc = 0 -> Imm7_Even = 0 -> Imm10_Even = 25 -> Imm16_Even = 0 -> Imm18_Even = 0	# CLOCK IN NS: 205ns # EVEN PIPE RESULT >>> # Addr_Rt2= 50 # Data_Rt= 0 # Data_Rt(hex)= 000000000000000000000000000000 # Wr_en= 1

Simple Fixed Unit 2:

Instruction	Inputs	Outputs
Rotate Word	EVEN INSTRUCTION Rotate word >> Input at : 365ns >> Output at : 445ns -> Opcode_Even = 00001011000 -> Addr_Even_Ra = 126 -> Addr_Even_Rb = 126 -> Addr_Even_Rc = 0 -> Imm7_Even = 0 -> Imm10_Even = 0 -> Imm16_Even = 0 -> Imm18_Even = 0	# CLOCK IN NS: 445ns # EVEN PIPE RESULT >>> # Addr_Rt2= 50 # Data_Rt=10684611793521601475482153351632683136 # Data_Rt(hex)= 0809c808808080808080808080808080 # Wr_en= 1
Shift left word immediate	EVEN INSTRUCTION Shift left word immediate >> Input at : 395ns >> Output at : 475ns -> Opcode_Even = 00001111011 -> Addr_Even_Ra = 126 -> Addr_Even_Rb = 0 -> Addr_Even_Rc = 0 -> Imm7_Even = 7 -> Imm10_Even = 0 -> Imm16_Even = 0 -> Imm18_Even = 0	# CLOCK IN NS: 475ns # EVEN PIPE RESULT >>> # Addr_Rt2= 50 # Data_Rt=85404480777014643781468766618232307712 # Data_Rt(hex)=40404e00404040004040400040404000 # Wr_en= 1

Single Precision Unit:

Instruction	Inputs	Outputs
Floating Add	EVEN INSTRUCTION Floating Add	# CLOCK IN NS: 555ns
	>> Input at : 475ns	# EVEN PIPE RESULT >>>
	>> Output at : 555ns	# Addr_Rt2= 75
	-> Opcode_Even = 01011000100	# Data_Rt= 1091357901
	-> Addr_Even_Ra = 124	# Data_Rt(hex)= 0000000000000000000000000410ccccd,
	-> Addr_Even_Rb = 124	# Wr_en= 1
	-> Addr_Even_Rc = 0	
	-> Imm7_Even = 0	
	-> Imm10_Even = 0	
	-> Imm16_Even = 0	
-> Imm18_Even = 0		
Multiply Immediate	EVEN INSTRUCTION Multiply Immediate	# CLOCK IN NS: 515ns
	>> Input at : 435ns	# EVEN PIPE RESULT >>>
	>> Output at : 515ns	# Addr_Rt2= 50
	-> Opcode_Even = 00001110100	# Data_Rt= 500
	-> Addr_Even_Ra = 20	# Data_Rt(hex)= 000000000000000000000000000001f4
	-> Addr_Even_Rb = 0	# Wr_en= 1
	-> Addr_Even_Rc = 0	
	-> Imm7_Even = 0	
	-> Imm10_Even = 25	
	-> Imm16_Even = 0	
-> Imm18_Even = 0		

Byte Unit:

Instruction	Inputs	Outputs
Count Ones in Bytes	EVEN INSTRUCTION Count Ones in Bytes	# CLOCK IN NS: 435ns
	>> Input at : 355ns	# EVEN PIPE RESULT >>>
	>> Output at : 435ns	# Addr_Rt2= 50
	-> Opcode_Even = 01010110100	# Data_Rt= 2
	-> Addr_Even_Ra = 20	# Data_Rt(hex)= 00000000000000000000000000000002,
	-> Addr_Even_Rb = 0	# Wr_en= 1
	-> Addr_Even_Rc = 0	
	-> Imm7_Even = 0	
	-> Imm10_Even = 0	
	-> Imm16_Even = 0	
-> Imm18_Even = 0		
Absolute Differences of Bytes	EVEN INSTRUCTION Absolute Differences of Bytes	# CLOCK IN NS: 415ns
	>> Input at : 335ns	# EVEN PIPE RESULT >>>
	>> Output at : 415ns	# Addr_Rt2= 50
	-> Opcode_Even = 00001010011	# Data_Rt= 5
	-> Addr_Even_Ra = 20	# Data_Rt(hex)= 00000000000000000000000000000005,
	-> Addr_Even_Rb = 25	# Wr_en= 1
	-> Addr_Even_Rc = 0	
	-> Imm7_Even = 0	
	-> Imm10_Even = 0	
	-> Imm16_Even = 0	
-> Imm18_Even = 0		

Branch Unit:

Instruction	Inputs	Outputs
Branch if Zero Word	ODD INSTRUCTION Branch if Zero Word	# CLOCK IN NS: 265ns
	>> Input at : 185ns	# ODD PIPE RESULT >>>
	>> Output at : 265ns	# Addr_Rt2= 0
	-> Opcode_Odd = 00001000000	# Data_Rt= 0
	-> Addr_Odd_Ra = 0	# Data_Rt(hex)= 00000000000000000000000000000000
	-> Addr_Odd_Rb = 0	# Wr_en= 1
	-> Addr_Odd_Rc = 0	# Br_Pc_out= 00000000000001010000010011001100
	-> Imm7_Odd = 0	
	-> Imm10_Odd = 0	
	-> Imm16_Odd = 36	
	-> Imm18_Odd = 0	
	-> PC = 328766	

Branch Indirect	ODD INSTRUCTION Branch Indirect >> Input at : 55ns >> Output at : 135ns -> Opcode_Odd = 00110101000 -> Addr_Odd_Ra = 127 -> Addr_Odd_Rb = 0 -> Addr_Odd_Rc = 0 -> Imm7_Odd = 0 -> Imm10_Odd = 0 -> Imm16_Odd = 0 -> Imm18_Even = 0	# CLOCK IN NS: 135ns # ODD PIPE RESULT >>> # Addr_Rt2= 55, # Data_Rt= 122, # Data_Rt(hex)= 00000000000000000000000000007a, # Wr_en= 1, # Br_Pc_out= 10000000100000001000000010000000,

Permute Unit:

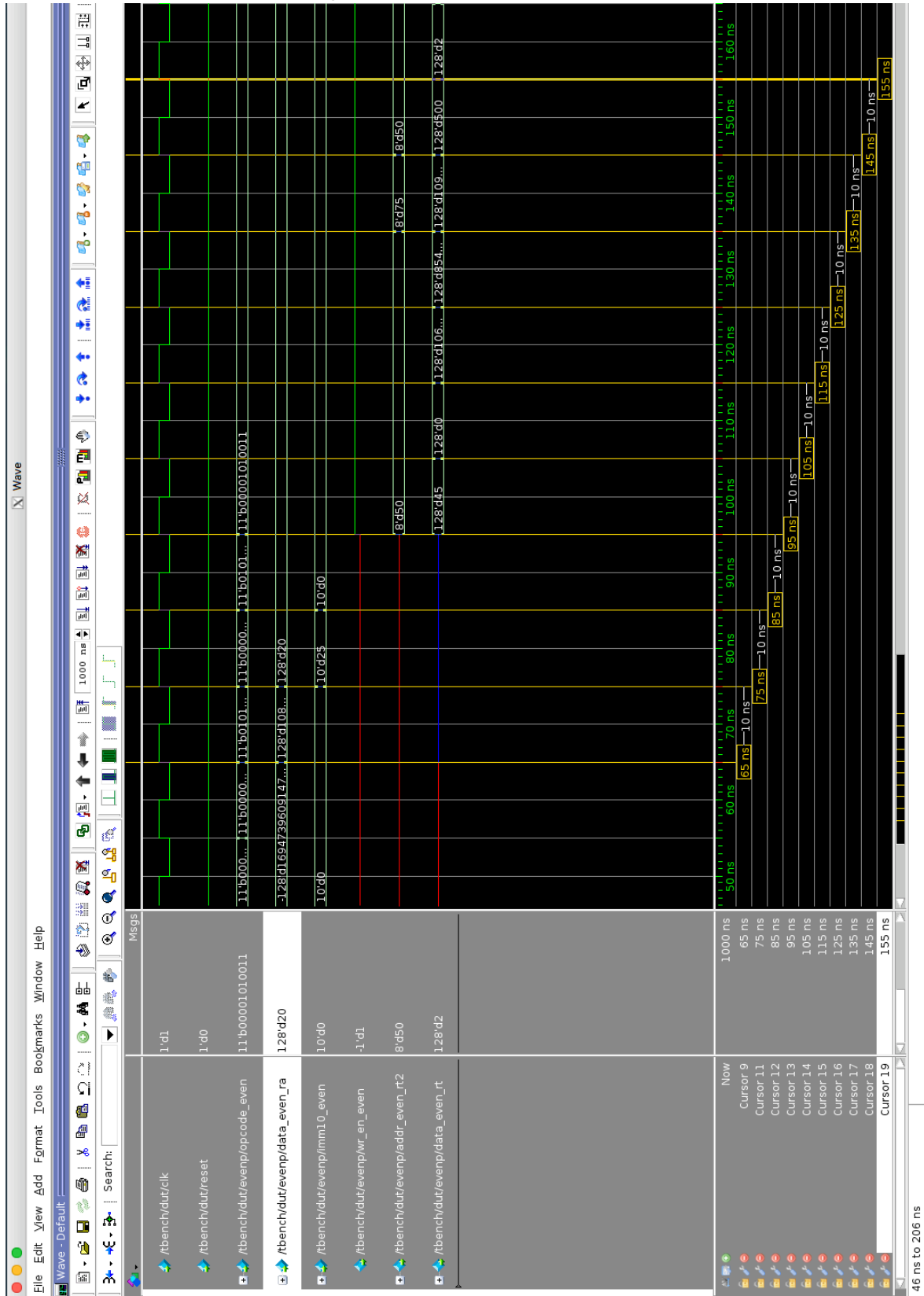
Instruction	Inputs	Outputs
Rotate Quadword by Bytes	ODD INSTRUCTION Rotate Quadword by Bytes >> Input at : 115ns >> Output at : 195ns -> Opcode_Odd = 00111011100 -> Addr_Odd_Ra = 126 -> Addr_Odd_Rb = 126 -> Addr_Odd_Rc = 0 -> Imm7_Odd = 0 -> Imm10_Odd = 0 -> Imm16_Odd = 0 -> Imm18_Even = 0	# CLOCK IN NS: 195ns # ODD PIPE RESULT >>> # Addr_Rt2= 75 # Data_Rt=170808403787765190019692950735500116096 # Data_Rt(hex)=808080808080809c8080808080808080 # Wr_en= 1 # Br_Pc_out= 00000000000010000000001000000100
Shift left quadword by bytes	ODD INSTRUCTION Shift left quadword by bytes >> Input at : 155ns >> Output at : 235ns -> Opcode_Odd = 00111011111 -> Addr_Odd_Ra = 126 -> Addr_Odd_Rb = 126 -> Addr_Odd_Rc = 0 -> Imm10_Even = 25	# CLOCK IN NS: 235ns # ODD PIPE RESULT >>> # Addr_Rt2= 75 # Data_Rt=170808406006153739902585569290863280256 # Data_Rt(hex)= 8080809c808080808080808080808080 # Wr_en= 1 # Br_Pc_out= 000000000000100000000001000000100

Local Store Unit:

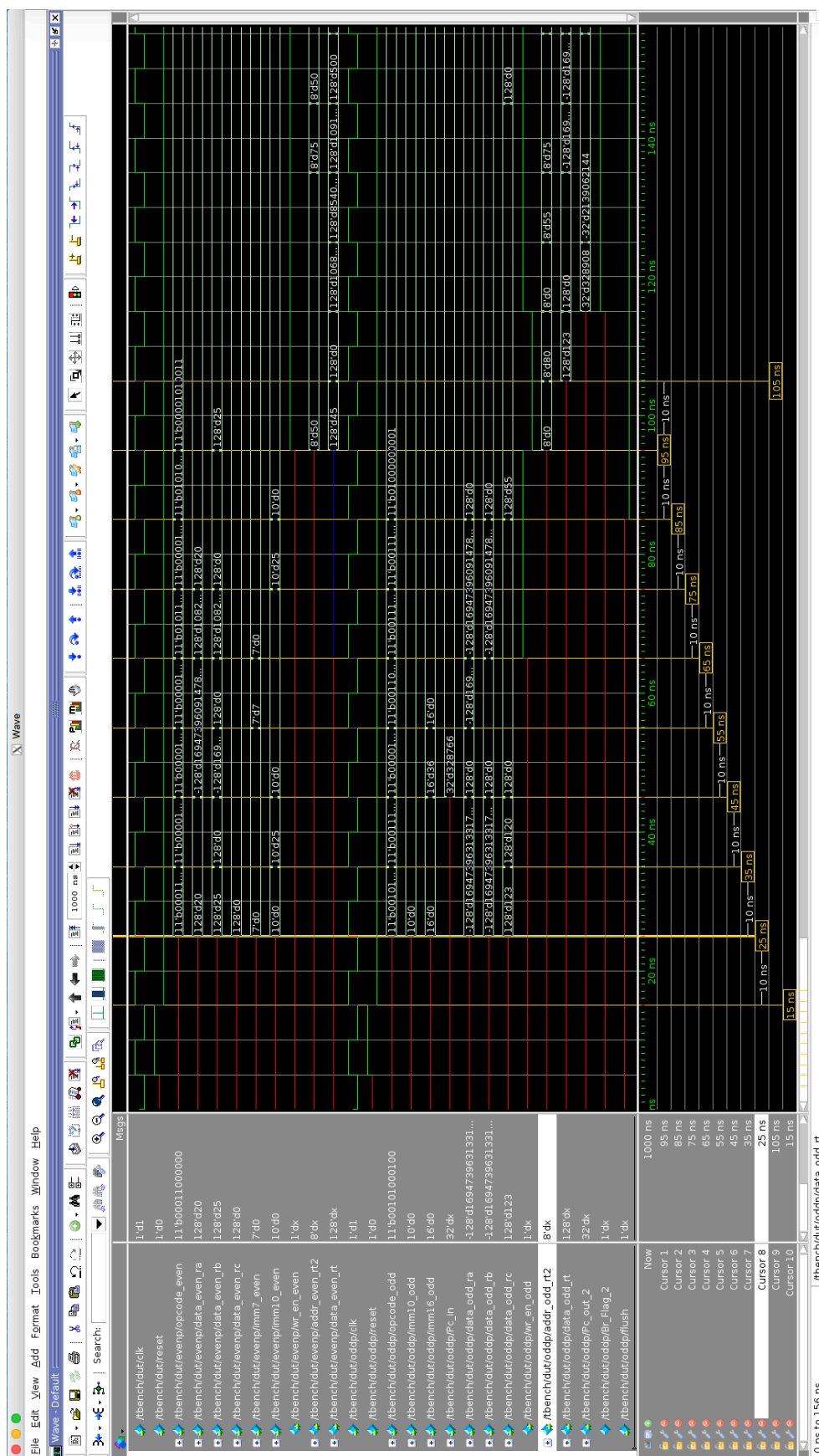
Instruction	Inputs	Outputs
Store Quadword (x-form)	ODD INSTRUCTION Store Quadword (x-form) >> Input at : 15ns >> Output at : 95ns -> Opcode_Odd = 00101000100 -> Addr_Odd_Ra = 127 -> Addr_Odd_Rb = 127 -> Addr_Odd_Rc = 123 -> Imm7_Odd = 0 -> Imm10_Odd = 0 -> Imm16_Odd = 0 -> Imm18_Even = 0	# CLOCK IN NS: 95ns # ODD PIPE RESULT >>> # Addr_Rt2= 0 # Data_Rt= x(Data stores in LS at same cycle) # Data_Rt(hex)= xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx # Wr_en= 0 # Br_Pc_out= xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Load Quadword (x-form)	ODD INSTRUCTION Load Quadword (x-form) >> Input at : 25ns >> Output at : 105ns -> Opcode_Odd = 00111000100 -> Addr_Odd_Ra = 127 -> Addr_Odd_Rb = 127 -> Addr_Odd_Rc = 120 -> Imm7_Odd = 0 -> Imm10_Odd = 0 -> Imm16_Odd = 0 -> Imm18_Even = 0	# CLOCK IN NS: 105ns # ODD PIPE RESULT >>> # Addr_Rt2= 80 # Data_Rt= 123(Data retrieved from LS) # Data_Rt(hex)= 000000000000000000000000000007b # Wr_en= 0

Static Timing Waveforms

Output Waveform 1 | Even Pipe performing Multiply Immediate Operation:



Output Waveform 2 | Both pipes in operation:



TESTBENCH & VERIFICATION METHODOLOGY (PARSER)

Inputs	Outputs
a 30 40 50 60	00011000000011001001010000011110
addx 30 40 50 60	11010000000011001001010000011110
ai 30 40 50 60	00011100000011001001010000011110
and 30 40 50 60	00011000001011001001010000011110
andbi 30 40 50 60	00010110000011001001010000011110
andc 30 40 50 60	01011000001011001001010000011110
ceq 30 40 50 60	01111000000011001001010000011110
ceqb 30 40 50 60	01111010000011001001010000011110
ceqbi 30 40 50 60	01111110000011001001010000011110
ceqi 30 40 50 60	01111100000011001001010000011110
cgtb 30 40 50 60	01001010000011001001010000011110
cgtbi 30 40 50 60	01001110000011001001010000011110
cgti 30 40 50 60	01001100000011001001010000011110
clgtb 30 40 50 60	01011010000011001001010000011110
clgtbi 30 40 50 60	01011110000011001001010000011110
il 30 40 50 60	01000000100000000001010000011110
ila 30 40 50 60	01000010000000000001010000011110
ilh 30 40 50 60	01000001100000000001010000011110
nand 30 40 50 60	00011001001011001001010000011110
nor 30 40 50 60	00001001001011001001010000011110
or 30 40 50 60	00001000001011001001010000011110
orc 30 40 50 60	01011001001011001001010000011110
ori 30 40 50 60	00000100000011001001010000011110
sf 30 40 50 60	00001000000011001001010000011110
sfi 30 40 50 60	00001100000011001001010000011110
sfx 30 40 50 60	01101000001011001001010000011110
xor 30 40 50 60	01001000001011001001010000011110
xorbi 30 40 50 60	01000110000011001001010000011110
xori 30 40 50 60	01000100000011001001010000011110
xsbh 30 40 50 60	01010110110000000001010000011110
xshw 30 40 50 60	01010101110000000001010000011110
xswd 30 40 50 60	01010100110000000001010000011110
absdb 30 40 50 60	00001010011011001001010000011110
avgb 30 40 50 60	00011010011011001001010000011110
cntb 30 40 50 60	01010110100000000001010000011110
sumb 30 40 50 60	01001010011011001001010000011110
rot 30 40 50 60	00001011000011001001010000011110
rotm 30 40 50 60	00001011001011001001010000011110
shl 30 40 50 60	00001011011011001001010000011110
shli 30 40 50 60	00001111011011001001010000011110
bi 30 40 50 60	00110101000000000001111000000000
bisl 30 40 50 60	00110101001000000001010000011110
br 30 40 50 60	00110010000000000001010000000000
bra 30 40 50 60	00110000000000000001010000000000
brasl 30 40 50 60	00110001000000000001010000011110

brsl 30 40 50 60	00110011000000000001010000011110
rotqby 30 40 50 60	00111011100011001001010000011110
rotqbyi 30 40 50 60	00111111100011001001010000011110
rotqmby 30 40 50 60	00111011101011001001010000011110
rotqmbyi 30 40 50 60	00111111101011001001010000011110
shlqby 30 40 50 60	00111011111011001001010000011110
shlqbyi 30 40 50 60	00111111111011001001010000011110
fa 30 40 50 60	01011000100011001001010000011110
fm 30 40 50 60	01011000110011001001010000011110
fma 30 40 50 60	11100011110011001001010000111100
fms 30 40 50 60	11110011110011001001010000111100
fs 30 40 50 60	01011000101011001001010000011110
lqa 30 40 50 60	00110000100000000001010000011110
lqx 30 40 50 60	00111000100011001001010000011110
stqa 30 40 50 60	00100000100000000001010000011110
stqx 30 40 50 60	00101000100011001001010000011110
mpy 30 40 50 60	01111000100011001001010000011110
mpya 30 40 50 60	11000011110011001001010000111100
mpyi 30 40 50 60	01110100000011001001010000011110
mpys 30 40 50 60	01111000111011001001010000011110
mpyu 30 40 50 60	01111001100011001001010000011110
mpyui 30 40 50 60	01110101000011001001010000011110
brnz 30 40 50 60	00100001000000000001010000011110
brz 30 40 50 60	00100000000000000001010000011110
onop 30 40 50 60	01000000010000000000000000000000
enop 30 40 50 60	01000000001000000000000000000000

TESTBENCH & VERIFICATION METHODOLOGY (PARSER)

Output Window

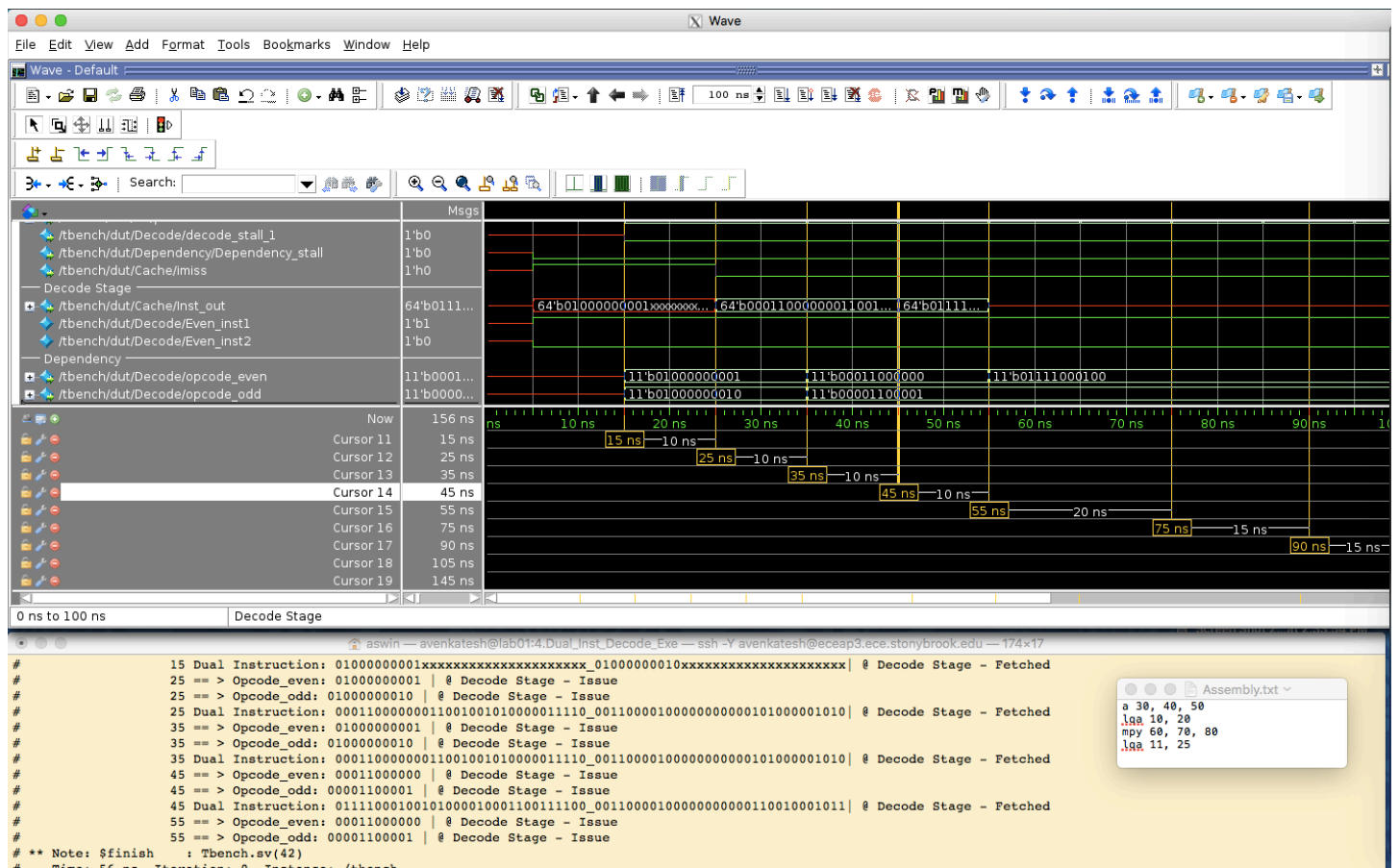
```
# run -all
#      15 Local Store Memory Instruction 1:00011000_00001100_10010100_00011110
#      15 Local Store Memory Instruction 2:00011000_00001100_10010100_00010000
#      15 Local Store Memory Instruction 3:00011000_00001100_10010100_00011111
#      15 Local Store Memory Instruction 4:00011000_00001100_10010100_00010001
#      15 Local Store Memory Instruction 5:10011000_00001100_10010100_00011110
#      15 Local Store Memory Instruction 6:10011000_00001100_10010100_00010000
#      15 Local Store Memory Instruction 7:xxxxxxxx_xxxxxxxxx_xxxxxxxxx_xxxxxxxxx
# Cache_valid_B1:1
# Cache_tag_B1:00000000
# Cache Miss Loading Block 1
#      16 Cache Block 1 Instruction 1:00011000_00001100_10010100_00011110
#      16 Cache Block 1 Instruction 2:00011000_00001100_10010100_00010000
#      16 Cache Block 1 Instruction 3:00011000_00001100_10010100_00011111
#      16 Cache Block 1 Instruction 4:00011000_00001100_10010100_00010001
#      16 Cache Block 1 Instruction 5:10011000_00001100_10010100_00011110
#      16 Cache Block 1 Instruction 6:10011000_00001100_10010100_00010000
#      16 Cache Block 1 Instruction 7:xxxxxxxx_xxxxxxxxx_xxxxxxxxx_xxxxxxxxx
# Hit 01
# ** Note: $finish : cache.sv(244)
# Time: 105 ns Iteration: 0 Instance: /tbench
# End time: 11:45:58 on Apr 30,2018, Elapsed time: 0:00:01
```

TESTBENCH & VERIFICATION METHODOLOGY (DUAL INSTRUCTION FETCH + DECODE + ISSUE)

Output Window

```
# run -all
#
# 15 Dual Instruction: 01000000001xxxxxxxxxxxxxxxxxxxxx_01000000010xxxxxxxxxxxxxxxxxxxxx | @ Decode Stage - Fetched
#
# 25 == > Opcode_even: 01000000001 | @ Decode Stage - Issue
#
# 25 == > Opcode_odd: 01000000010 | @ Decode Stage - Issue
#
# 25 Dual Instruction: 00011000000011001001010000011110_00110000100000000000101000001010 | @ Decode Stage - Fetched
#
# 35 == > Opcode_even: 01000000001 | @ Decode Stage - Issue
#
# 35 == > Opcode_odd: 01000000010 | @ Decode Stage - Issue
#
# 35 Dual Instruction: 00011000000011001001010000011110_00110000100000000000101000001010 | @ Decode Stage - Fetched
#
# 45 == > Opcode_even: 00011000000 | @ Decode Stage - Issue
#
# 45 == > Opcode_odd: 00001100001 | @ Decode Stage - Issue
#
# 45 Dual Instruction: 01111000100101000010001100111100_00110000100000000000110010001011 | @ Decode Stage - Fetched
#
# 55 == > Opcode_even: 00011000000 | @ Decode Stage - Issue
#
# 55 == > Opcode_odd: 00001100001 | @ Decode Stage - Issue

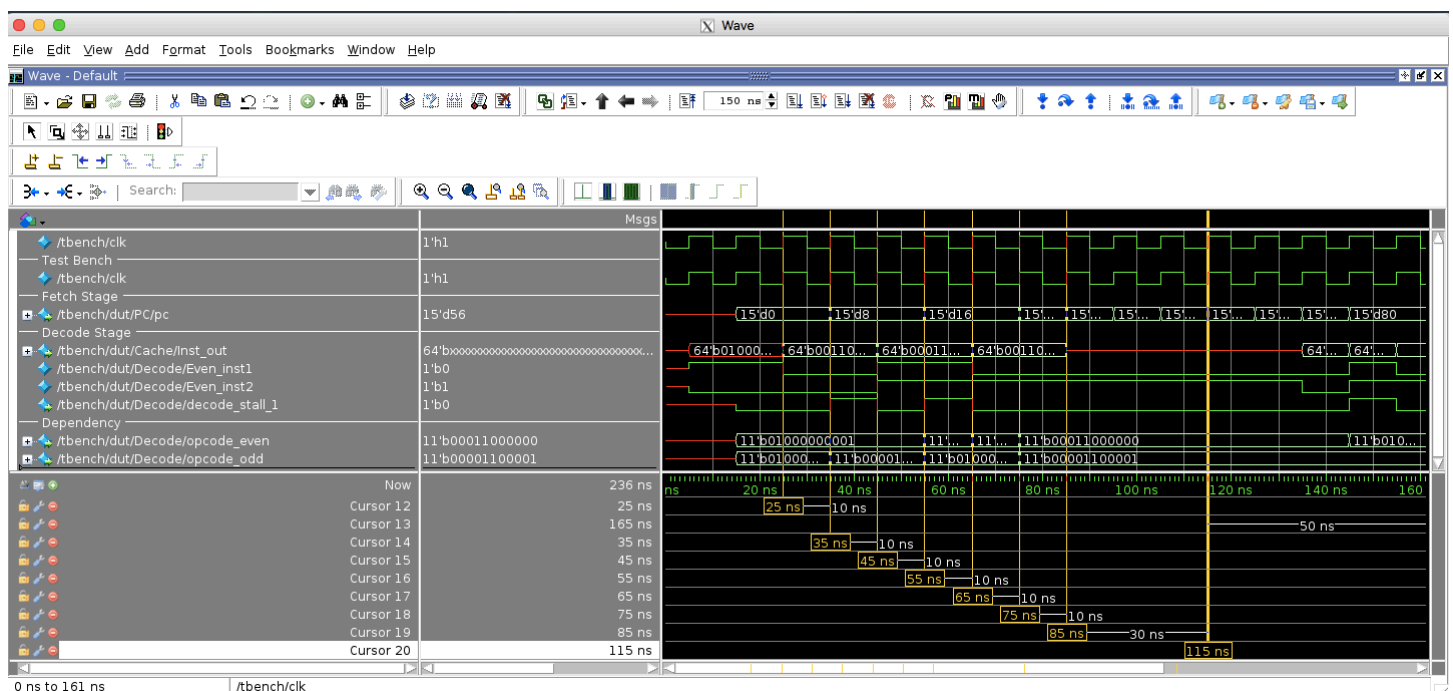
# ** Note: $finish : Tbench.sv(42)
# Time: 56 ns Iteration: 0 Instance: /tbench
# End time: 12:12:05 on Apr 30,2018, Elapsed time: 0:00:00
# Errors: 0, Warnings: 0
[avenkatesh@lab01 4.Dual_Inst_Decode_Exe]$
```



TESTBENCH & VERIFICATION METHODOLOGY (DECODE – STRUCTURAL HAZARD RESOLUTION)

Output Window

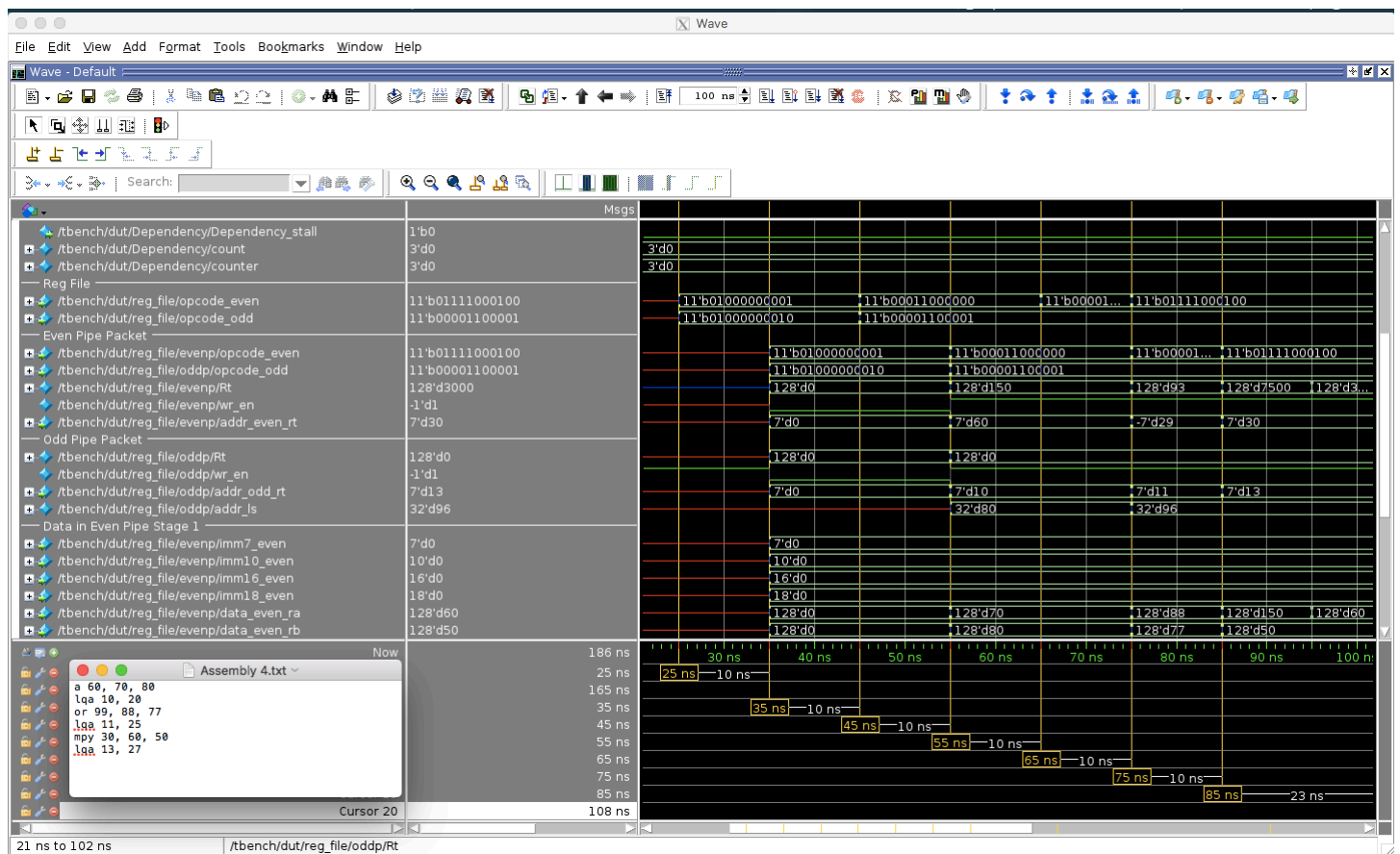
```
# run -all
#      5 No_Structural_Hazard
#     15 No_Structural_Hazard
#    25 == > Opcode_even: 01000000001 | @ Decode Stage - Issue
#    25 == > Opcode_odd: 01000000010 | @ Decode Stage - Issue
#    15 Dual Instruction: 0100000001xxxxxxxxxxxxxxxx_0100000010xxxxxxxxxxxxxxxxxxxxxx | @ Decode Stage - Fetched
#    25 No_Structural_Hazard
#    25 Dual Instruction: 00110000100000000000101000001010_00110000100000000000110010001011 | @ Decode Stage - Fetched
#    35 Odd_Structural_Hazard Resolution in Progress
#    35 == > Opcode_even: 01000000001 | @ Decode Stage - Issue
#    35 == > Opcode_odd: 01000000010 | @ Decode Stage - Issue
#    45 == > Opcode_even: 01000000001 | @ Decode Stage - Issue
#    45 == > Opcode_odd: 00001100001 | @ Decode Stage - Issue
#    35 Dual Instruction: 00110000100000000000101000001010_00110000100000000000110010001011 | @ Decode Stage - Fetched
#    45 Odd_Structural_Hazard Resolution in Progress
#    45 Dual Instruction: 00011000000011001001010000011110_01111000100101000010001100111100 | @ Decode Stage - Fetched
#    55 Even_Structural_Hazard Resolution in Progress
#    55 == > Opcode_even: 01000000001 | @ Decode Stage - Issue
#    55 == > Opcode_odd: 00001100001 | @ Decode Stage - Issue
#    65 == > Opcode_even: 00011000000 | @ Decode Stage - Issue
#    65 == > Opcode_odd: 01000000001 | @ Decode Stage - Issue
#    55 Dual Instruction: 00011000000011001001010000011110_01111000100101000010001100111100 | @ Decode Stage - Fetched
#    65 Even_Structural_Hazard Resolution in Progress
#    65 Dual Instruction: 001100001000000000001000110010001_00011000000101101010010111000010 | @ Decode Stage - Fetched
#    75 No_Structural_Hazard
#    75 == > Opcode_even: 01111000100 | @ Decode Stage - Issue
#    75 == > Opcode_odd: 01000000001 | @ Decode Stage - Issue
#    85 == > Opcode_even: 00011000000 | @ Decode Stage - Issue
#    85 == > Opcode_odd: 00001100001 | @ Decode Stage - Issue
#    75 Dual Instruction: 001100001000000000001000110010001_00011000000101101010010111000010 | @ Decode Stage - Fetched
#    85 No_Structural_Hazard
# ** Note: $finish : Tbench.sv(42)
# Time: 86 ns Iteration: 0 Instance: /tbench
# End time: 12:47:30 on Apr 30,2018, Elapsed time: 0:00:00
```



TESTBENCH & VERIFICATION METHODOLOGY (DATA HAZARD - FORWARDING)

Output Window

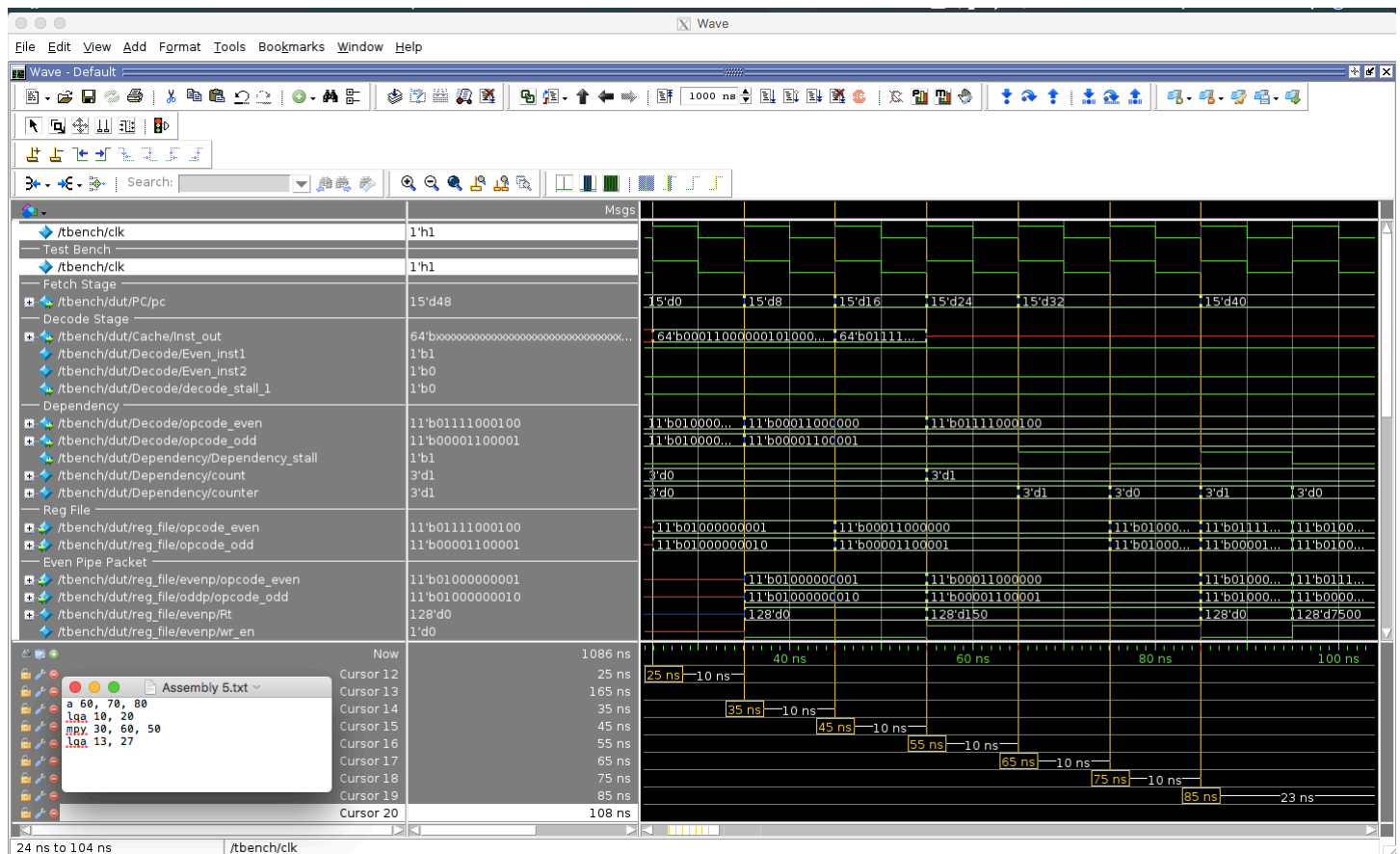
```
# run -all
# Opcode:xxxxxxxx, data_even_ra:      x, data_even_rb:      x, data_even_rc:      x
# Opcode:xxxxxxxx, data_even_ra:      x, data_even_rb:      x, data_even_rc:      x
# Opcode:xxxxxxxx, data_even_ra:      x, data_even_rb:      x, data_even_rc:      x
# Opcode:01000000001, data_even_ra:    x, data_even_rb:      x, data_even_rc:      x
# Opcode:01000000001, data_even_ra:    0, data_even_rb:      0, data_even_rc:      0
# Opcode:00011000000, data_even_ra:    0, data_even_rb:      0, data_even_rc:      0
# Opcode:00011000000, data_even_ra:    70, data_even_rb:     80, data_even_rc:      0
# Opcode:00001000001, data_even_ra:    70, data_even_rb:     80, data_even_rc:      0
# Opcode:01111000100, data_even_ra:    88, data_even_rb:     77, data_even_rc:      0
# Opcode:01111000100, data_even_ra:    150, data_even_rb:     50, data_even_rc:      0
# Opcode:01111000100, data_even_ra:    60, data_even_rb:     50, data_even_rc:      0
# ** Note: $finish : Tbench.sv(42)
# Time: 166 ns Iteration: 0 Instance: /tbench
# End time: 13:58:21 on Apr 30,2018, Elapsed time: 0:00:01
# Errors: 0, Warnings: 0
[avenkatesh@lab01 6.Forwarding_Data_Hazard]$
```



TESTBENCH & VERIFICATION METHODOLOGY (DATA HAZARD - FORWARDING)

Output Window

```
# run -all
# Opcode:xxxxxxxx, data_even_ra:          x, data_even_rb:          x, data_even_rc:          x
# Opcode:xxxxxxxx, data_even_ra:          x, data_even_rb:          x, data_even_rc:          x
# Opcode:xxxxxxxx, data_even_ra:          x, data_even_rb:          x, data_even_rc:          x
# Opcode:01000000001, data_even_ra:        x, data_even_rb:          x, data_even_rc:          x
# Opcode:01000000001, data_even_ra:        0, data_even_rb:          0, data_even_rc:          0
# Opcode:00011000000, data_even_ra:        0, data_even_rb:          0, data_even_rc:          0
# Opcode:00011000000, data_even_ra:        70, data_even_rb:        80, data_even_rc:          0
# Opcode:00001000001, data_even_ra:        70, data_even_rb:        80, data_even_rc:          0
# Opcode:01111000100, data_even_ra:        88, data_even_rb:        77, data_even_rc:          0
# 1st EVEN If - Data Hazard
# 1st EVEN If - Data Hazard
# 1st EVEN If - Data Hazard
# Opcode:01111000100, data_even_ra:        150, data_even_rb:        50, data_even_rc:          0
# Opcode:01111000100, data_even_ra:        60, data_even_rb:        50, data_even_rc:          0
# ** Note: $finish : Tbench.sv(42)
# Time: 166 ns Iteration: 0 Instance: /tbench
# End time: 13:40:23 on Apr 30,2018, Elapsed time: 0:00:01
# Errors: 0, Warnings: 0
[avenkatesh@lab01 6.Forwarding_Data_Hazard]$
```



----- END OF REPORT -----

Script: Parser

```
#include <iostream>
#include <cstring>
#include <fstream>
#include <string>
#include <sstream>

using namespace std;

char *DecToBin(string nums, unsigned bit);

int main()
{
    ifstream inFile("Assembly.txt");
    ofstream outFile;
    outFile.open("Instruction.txt");
    string line, str_null= "000";
    string str[10];

    while(getline(inFile, line))
    {
        char *token = std::strtok(&line[0], " \t , ");
        int i=0;

        while (token != NULL)
        {
            str[i] = token;
            token = std::strtok(NULL, " \t , ");
            i++;
        }
        if(i!=0){ // Not to enter first time
            //if(str[0] == "MUL") {outFile << "Text"<< DecToBin(str[1], 10)<< endl;}
            if(str[0] == "a") {outFile << "00011000000"<<DecToBin(str[3], 7)<<DecToBin(str[2], 7)<<DecToBin(str[1],
7)<<endl;} // 1 - a - Add Word
            else if(str[0] == "addx") {outFile << "11010000000"<<DecToBin(str[3], 7)<<DecToBin(str[2], 7)<<DecToBin(str[1],
7)<<endl;} // 2 - addx - Add Extended
            else if(str[0] == "ai") {outFile << "00011100"<<DecToBin(str[3], 10)<<DecToBin(str[2], 7)<<DecToBin(str[1],
7)<<endl;} // 3 - ai - Add Word Immediate
            else if(str[0] == "and") {outFile << "00011000001"<<DecToBin(str[3], 7)<<DecToBin(str[2], 7)<<DecToBin(str[1],
7)<<endl;} // 4 - and - And
            else if(str[0] == "andbi") {outFile << "00010110"<<DecToBin(str[3], 10)<<DecToBin(str[2], 7)<<DecToBin(str[1],
7)<<endl;} // 5 - andbi - And Byte Immediate
            else if(str[0] == "andc") {outFile << "01011000001"<<DecToBin(str[3], 7)<<DecToBin(str[2], 7)<<DecToBin(str[1],
7)<<endl;} // 6 - andc - And with Complement
            else if(str[0] == "ceq") {outFile << "01111000000"<<DecToBin(str[3], 7)<<DecToBin(str[2], 7)<<DecToBin(str[1],
7)<<endl;} // 7 - ceq - Compare Equal Word
            else if(str[0] == "ceqb") {outFile << "01111010000"<<DecToBin(str[3], 7)<<DecToBin(str[2], 7)<<DecToBin(str[1],
7)<<endl;} // 8 - ceqb - Compare Equal Byte
            else if(str[0] == "ceqbi") {outFile << "01111110"<<DecToBin(str[3], 10)<<DecToBin(str[2], 7)<<DecToBin(str[1],
7)<<endl;} // 9 - ceqbi - Compare Equal Byte Immediate
            else if(str[0] == "ceqi") {outFile << "01111100"<<DecToBin(str[3], 10)<<DecToBin(str[2], 7)<<DecToBin(str[1],
7)<<endl;} // 10 - ceqi - Compare Equal Word Immediate
            else if(str[0] == "cgtb") {outFile << "01001010000"<<DecToBin(str[3], 7)<<DecToBin(str[2], 7)<<DecToBin(str[1],
7)<<endl;} // 11 - cgtb - Compare Greater Than Byte
            else if(str[0] == "cgtbi") {outFile << "01001110"<<DecToBin(str[3], 10)<<DecToBin(str[2], 7)<<DecToBin(str[1],
7)<<endl;} // 12 - cgtbi - Compare Greater Than Byte Immediate
            else if(str[0] == "cgti") {outFile << "01001100"<<DecToBin(str[3], 10)<<DecToBin(str[2], 7)<<DecToBin(str[1],
7)<<endl;} // 13 - cgti - Compare Greater Than Word Immediate
            else if(str[0] == "clgtb") {outFile << "01011010000"<<DecToBin(str[3], 7)<<DecToBin(str[2], 7)<<DecToBin(str[1],
7)<<endl;} // 14 - clgtb - Compare Logical Greater Than Byte
```

```

        else if(str[0] == "clgtbi") {outFile << "01011110"<<DecToBin(str[3], 10)<<DecToBin(str[2], 7)<<DecToBin(str[1],
7)<<endl;} // 15 - clgtbi - Compare Logical Greater Than Byte Immediate
        else if(str[0] == "il") {outFile << "010000001"<<DecToBin(str[2], 16)<<DecToBin(str[1], 7)<<endl;} // 16 - il -
Immediate Load Word
        else if(str[0] == "ila") {outFile << "0100001"<<DecToBin(str[2], 18)<<DecToBin(str[1], 7)<<endl;} // 17 - ila -
Immediate Load Address
        else if(str[0] == "ilh") {outFile << "010000011"<<DecToBin(str[2], 16)<<DecToBin(str[1], 7)<<endl;} // 18 - ilh -
Immediate Load Halfword
        else if(str[0] == "nand") {outFile << "00011001001"<<DecToBin(str[3], 7)<<DecToBin(str[2], 7)<<DecToBin(str[1],
7)<<endl;} // 19 - nand - Nand
        else if(str[0] == "nor") {outFile << "00001001001"<<DecToBin(str[3], 7)<<DecToBin(str[2], 7)<<DecToBin(str[1],
7)<<endl;} // 20 - nor - Nor
        else if(str[0] == "or") {outFile << "00001000001"<<DecToBin(str[3], 7)<<DecToBin(str[2], 7)<<DecToBin(str[1],
7)<<endl;} // 21 - or - Or
        else if(str[0] == "orc") {outFile << "01011001001"<<DecToBin(str[3], 7)<<DecToBin(str[2], 7)<<DecToBin(str[1],
7)<<endl;} // 22 - orc - Or with Complement
        else if(str[0] == "ori") {outFile << "00000100"<<DecToBin(str[3], 10)<<DecToBin(str[2], 7)<<DecToBin(str[1],
7)<<endl;} // 23 - ori - Or Word Immediate
        else if(str[0] == "sf") {outFile << "00001000000"<<DecToBin(str[3], 7)<<DecToBin(str[2], 7)<<DecToBin(str[1],
7)<<endl;} // 24 - sf - Subtract from Word
        else if(str[0] == "sfi") {outFile << "00001100"<<DecToBin(str[3], 10)<<DecToBin(str[2], 7)<<DecToBin(str[1],
7)<<endl;} // 25 - sfi - Subtract from Word Immediate
        else if(str[0] == "sfx") {outFile << "01101000001"<<DecToBin(str[3], 7)<<DecToBin(str[2], 7)<<DecToBin(str[1],
7)<<endl;} // 26 - sfx - Subtract from Extended
        else if(str[0] == "xor") {outFile << "01001000001"<<DecToBin(str[3], 7)<<DecToBin(str[2], 7)<<DecToBin(str[1],
7)<<endl;} // 27 - xor - Exclusive Or
        else if(str[0] == "xorbi") {outFile << "01000110"<<DecToBin(str[3], 10)<<DecToBin(str[2], 7)<<DecToBin(str[1],
7)<<endl;} // 28 - xorbi - Exclusive Or Byte Immediate
        else if(str[0] == "xori") {outFile << "01000100"<<DecToBin(str[3], 10)<<DecToBin(str[2], 7)<<DecToBin(str[1],
7)<<endl;} // 29 - xori - Exclusive Or Word Immediate
        else if(str[0] == "xsbh") {outFile << "01010110110"<<DecToBin(str_null, 7)<<DecToBin(str[2], 7)<<DecToBin(str[1],
7)<<endl;} // 30 - xsbh - Extend Sign Byte to Halfword
        else if(str[0] == "xshw") {outFile << "01010101110"<<DecToBin(str_null, 7)<<DecToBin(str[2], 7)<<DecToBin(str[1],
7)<<endl;} // 31 - xshw - Extend Sign Halfword to Word
        else if(str[0] == "xswd") {outFile << "01010100110"<<DecToBin(str_null, 7)<<DecToBin(str[2], 7)<<DecToBin(str[1],
7)<<endl;} // 32 - xswd - Extend Sign Word to Doubleword
        else if(str[0] == "absdb") {outFile << "00001010011"<<DecToBin(str[3], 7)<<DecToBin(str[2], 7)<<DecToBin(str[1],
7)<<endl;} // 33 - absdb - Absolute Differences of Bytes
        else if(str[0] == "avgb") {outFile << "00011010011"<<DecToBin(str[3], 7)<<DecToBin(str[2], 7)<<DecToBin(str[1],
7)<<endl;} // 34 - avgb - Average Bytes
        else if(str[0] == "cntb") {outFile << "01010110100"<<DecToBin(str_null, 7)<<DecToBin(str[2], 7)<<DecToBin(str[1],
7)<<endl;} // 35 - cntb - Count Ones in Bytes
        else if(str[0] == "sumb") {outFile << "01001010011"<<DecToBin(str[3], 7)<<DecToBin(str[2], 7)<<DecToBin(str[1],
7)<<endl;} // 36 - sumb - Sum Bytes into Halfwords
        else if(str[0] == "rot") {outFile << "00001011000"<<DecToBin(str[3], 7)<<DecToBin(str[2], 7)<<DecToBin(str[1],
7)<<endl;} // 37 - rot - Rotate Word
        else if(str[0] == "rotm") {outFile << "00001011001"<<DecToBin(str[3], 7)<<DecToBin(str[2], 7)<<DecToBin(str[1],
7)<<endl;} // 38 - rotm - Rotate and Mask Word
        else if(str[0] == "shl") {outFile << "00001011011"<<DecToBin(str[3], 7)<<DecToBin(str[2], 7)<<DecToBin(str[1],
7)<<endl;} // 39 - shl - Shift Left Word
        else if(str[0] == "shli") {outFile << "00001111011"<<DecToBin(str[3], 7)<<DecToBin(str[2], 7)<<DecToBin(str[1],
7)<<endl;} // 40 - shli - Shift Left Word Immediate
        else if(str[0] == "bi") {outFile << "00110101000"<<DecToBin(str_null, 7)<<DecToBin(str[1], 7)<<DecToBin(str_null,
7)<<endl;} // 41 - bi - Branch Indirect
        else if(str[0] == "bisl") {outFile << "00110101001"<<DecToBin(str_null, 7)<<DecToBin(str[2], 7)<<DecToBin(str[1],
7)<<endl;} // 42 - bisl - Branch Indirect and Set Link
        else if(str[0] == "br") {outFile << "001100100"<<DecToBin(str[2], 16)<<DecToBin(str_null, 7)<<endl;} // 43 - br -
Branch Relative
        else if(str[0] == "bra") {outFile << "001100000"<<DecToBin(str[2], 16)<<DecToBin(str_null, 7)<<endl;} // 44 - bra
- Branch Absolute
        else if(str[0] == "brasil") {outFile << "001100010"<<DecToBin(str[2], 16)<<DecToBin(str[1], 7)<<endl;} // 45 - brasil
- Branch Absolute and Set Link

```

```

        else if(str[0] == "brsl") {outFile << "001100110"<<DecToBin(str[2], 16)<<DecToBin(str[1], 7)<<endl;} // 46 - brsl -
Branch Relative and Set Link
        else if(str[0] == "rotqby") {outFile << "00111011100"<<DecToBin(str[3], 7)<<DecToBin(str[2], 7)<<DecToBin(str[1],
7)<<endl;} // 47 - rotqby - Rotate Quadword by Bytes
        else if(str[0] == "rotqbyi") {outFile << "00111111100"<<DecToBin(str[3], 7)<<DecToBin(str[2], 7)<<DecToBin(str[1],
7)<<endl;} // 48 - rotqbyi - Rotate Quadword by Bytes Immediate
        else if(str[0] == "rotqmby") {outFile << "00111011101"<<DecToBin(str[3], 7)<<DecToBin(str[2],
7)<<DecToBin(str[1], 7)<<endl;} // 49 - rotqmby - Rotate and Mask Quadword by Bytes
        else if(str[0] == "rotqmbyi") {outFile << "00111111101"<<DecToBin(str[3], 7)<<DecToBin(str[2],
7)<<DecToBin(str[1], 7)<<endl;} // 50 - rotqmbyi - Rotate and Mask Quadword by Bytes Immediate
        else if(str[0] == "shlqby") {outFile << "00111011111"<<DecToBin(str[3], 7)<<DecToBin(str[2], 7)<<DecToBin(str[1],
7)<<endl;} // 51 - shlqby - Shift Left Quadword by Bytes
        else if(str[0] == "shlqbyi") {outFile << "00111111111"<<DecToBin(str[3], 7)<<DecToBin(str[2], 7)<<DecToBin(str[1],
7)<<endl;} // 52 - shlqbyi - Shift Left Quadword by Bytes Immediate
        else if(str[0] == "fa") {outFile << "01011000100"<<DecToBin(str[3], 7)<<DecToBin(str[2], 7)<<DecToBin(str[1],
7)<<endl;} // 53 - fa - Floating Add
        else if(str[0] == "fm") {outFile << "01011000110"<<DecToBin(str[3], 7)<<DecToBin(str[2], 7)<<DecToBin(str[1],
7)<<endl;} // 54 - fm - Floating Multiply
        else if(str[0] == "fma") {outFile << "1110"<<DecToBin(str[1], 7)<<DecToBin(str[3], 7)<<DecToBin(str[2],
7)<<DecToBin(str[4], 7)<<endl;} // 55 - fma - Floating Multiply and Add
        else if(str[0] == "fms") {outFile << "1111"<<DecToBin(str[1], 7)<<DecToBin(str[3], 7)<<DecToBin(str[2],
7)<<DecToBin(str[4], 7)<<endl;} // 56 - fms - Floating Multiply and Subtract
        else if(str[0] == "fs") {outFile << "01011000101"<<DecToBin(str[3], 7)<<DecToBin(str[2], 7)<<DecToBin(str[1],
7)<<endl;} // 57 - fs - Floating Subtract
        else if(str[0] == "lqa") {outFile << "001100001"<<DecToBin(str[2], 16)<<DecToBin(str[1], 7)<<endl;} // 58 - lqa -
Load Quadword (a-form)
        else if(str[0] == "lqx") {outFile << "00111000100"<<DecToBin(str[3], 7)<<DecToBin(str[2], 7)<<DecToBin(str[1],
7)<<endl;} // 59 - lqx - Load Quadword (x-form)
        else if(str[0] == "stqa") {outFile << "001000001"<<DecToBin(str[2], 16)<<DecToBin(str[1], 7)<<endl;} // 60 - stqa
- Store Quadword (a-form)
        else if(str[0] == "stqx") {outFile << "00101000100"<<DecToBin(str[3], 7)<<DecToBin(str[2], 7)<<DecToBin(str[1],
7)<<endl;} // 61 - stqx - Store Quadword (x-form)
        else if(str[0] == "mpy") {outFile << "01111000100"<<DecToBin(str[3], 7)<<DecToBin(str[2], 7)<<DecToBin(str[1],
7)<<endl;} // 62 - mpy - Multiply
        else if(str[0] == "mpya") {outFile << "1100"<<DecToBin(str[1], 7)<<DecToBin(str[3], 7)<<DecToBin(str[2],
7)<<DecToBin(str[4], 7)<<endl;} // 63 - mpya - Multiply and Add
        else if(str[0] == "mpyi") {outFile << "01110100"<<DecToBin(str[3], 10)<<DecToBin(str[2], 7)<<DecToBin(str[1],
7)<<endl;} // 64 - mpyi - Multiply Immediate
        else if(str[0] == "mpys") {outFile << "01111000111"<<DecToBin(str[3], 7)<<DecToBin(str[2], 7)<<DecToBin(str[1],
7)<<endl;} // 65 - mpys - Multiply and Shift Right
        else if(str[0] == "mpyu") {outFile << "01111001100"<<DecToBin(str[3], 7)<<DecToBin(str[2], 7)<<DecToBin(str[1],
7)<<endl;} // 66 - mpyu - Multiply Unsigned
        else if(str[0] == "mpyui") {outFile << "01110101"<<DecToBin(str[3], 10)<<DecToBin(str[2], 7)<<DecToBin(str[1],
7)<<endl;} // 67 - mpyui - Multiply Unsigned Immediate
        else if(str[0] == "brnz") {outFile << "001000010"<<DecToBin(str[2], 16)<<DecToBin(str[1], 7)<<endl;} // 68 - brnz
- Branch if Not Zero Word
        else if(str[0] == "brz") {outFile << "001000000"<<DecToBin(str[2], 16)<<DecToBin(str[1], 7)<<endl;} // 69 - brz -
Branch if Zero Word
        else if(str[0] == "onop") {outFile << "01000000010"<<DecToBin(str_null, 7)<<DecToBin(str_null,
7)<<DecToBin(str_null, 7)<<endl;} // 70 - Inop - No Operation (Execute)
        else if(str[0] == "enop") {outFile << "01000000001"<<DecToBin(str_null, 7)<<DecToBin(str_null,
7)<<DecToBin(str_null, 7)<<endl;} // 71 - nop - No Operation (Execute)
        else {cout<<str[0]<<" Instruction not Found"<<endl;}
    }
    outFile.close();
}

//Decimal to Binary
char* DecToBin(string nums, unsigned bit)
{
    int num;
    istringstream (nums) >> num;

```

```

char *binStr = new char (bit + 1);
int len = bit;

binStr[bit] = '\0';
while (bit-->0) binStr[bit] = '0';

if (num == 0)
    return binStr;

int r;
while (num && len)
{
    r = num % 2;
    binStr[--len] = r + '0';
    num /= 2;
}

return binStr;
}

```

Script: SPU CELL Top Module

```

//===== Register File
=====

module reg_file (clk, reset, Pc_in, addr_even_ra, addr_even_rb, addr_even_rc, opcode_even, imm10_even, imm16_even,
imm18_even, imm7_even, addr_even_rt,
addr_odd_ra, addr_odd_rb, addr_odd_rc, opcode_odd, imm10_odd, imm16_odd, imm18_odd, imm7_odd, addr_odd_rt,
flush,Rt_Temp_even_Rout, Rt_Temp1_even_Rout, Rt_Temp2_even_Rout, Rt_Temp3_even_Rout, Rt_Temp4_even_Rout,
Rt_Temp5_even_Rout, Rt_Temp6_even_Rout,Rt_Temp_odd_Rout, Rt_Temp1_odd_Rout, Rt_Temp2_odd_Rout,
Rt_Temp3_odd_Rout, Rt_Temp4_odd_Rout, Rt_Temp5_odd_Rout, Rt_Temp6_odd_Rout, Even_Test_Packet, Odd_Test_Packet);

    // Input to REG File for Processing
    input clk, reset;
    input [0:6] addr_even_ra, addr_even_rb, addr_even_rc;
    input [0:6] addr_odd_ra, addr_odd_rb, addr_odd_rc;
    output logic [0:136] Even_Test_Packet; //Packet Data Sent to Testbench for Verication Purpose
    output logic [0:168] Odd_Test_Packet; //Packet Data Sent to Testbench for Verication Purpose

    // Even Pipe Forwarding and Data Stuff
    input [0:10] opcode_even; // Input from TB
    input signed [0:6] imm7_even; // logic from TB
    input signed [0:9] imm10_even; // logic from TB
    input signed [0:15] imm16_even; // logic from TB
    input signed [0:17] imm18_even; // logic from TB
    input [0:6] addr_even_rt; // Input from TB
    logic [0:6] addr_even_rt2; // Receiving from Even Pipe
    logic signed [0:127] data_even_rt; // Receiving from Even Pipe
    logic wr_en_even; // Receiving from Even Pipe
    logic [0:142]Rt_Temp_even, Rt_Temp1_even, Rt_Temp2_even, Rt_Temp3_even, Rt_Temp4_even, Rt_Temp5_even,
Rt_Temp6_even;
    output logic [0:142]Rt_Temp_even_Rout, Rt_Temp1_even_Rout, Rt_Temp2_even_Rout, Rt_Temp3_even_Rout,
Rt_Temp4_even_Rout, Rt_Temp5_even_Rout, Rt_Temp6_even_Rout;

    logic [0:142] Rt_Temp_even_Test;

    logic signed [0:127] data_even_ra, data_even_rb, data_even_rc; // Data to Even Pipe
    logic signed [0:127] data_even_ra1,
data_even_ra2,data_even_ra3,data_even_ra4,data_even_ra5,data_even_ra6,data_even_ra_new;
    logic signed [0:127] data_even_rb1,
data_even_rb2,data_even_rb3,data_even_rb4,data_even_rb5,data_even_rb6,data_even_rb_new;

```

```

logic signed [0:127] data_even_rc1,
data_even_rc2,data_even_rc3,data_even_rc4,data_even_rc5,data_even_rc6,data_even_rc_new;
logic [0:10] opcode_even_fwd;      // Forward Reg for 1 Cycle Delay
logic signed [0:6] imm7_even_fwd;  // Forward Reg for 1 Cycle Delay
logic signed [0:9] imm10_even_fwd;  // Forward Reg for 1 Cycle Delay
logic signed [0:15] imm16_even_fwd; // Forward Reg for 1 Cycle Delay
logic signed [0:17] imm18_even_fwd; // Forward Reg for 1 Cycle Delay
logic [0:6] addr_even_rt_fwd;      // Forward Reg for 1 Cycle Delay

// Odd Pipe Forwarding and Data Stuff
input flush;
input [0:10] opcode_odd;           // Input from TB
input [0:6] imm7_odd;              // logic from TB
input [0:9] imm10_odd;             // logic from TB
input [0:15] imm16_odd;           // logic from TB
input [0:17] imm18_odd;           // logic from TB
input [0:6] addr_odd_rt;           // Input from TB
input [0:14] Pc_in;               // Input from TB
logic [0:6] addr_odd_rt2;          // Receiving from odd Pipe
logic [0:127] data_odd_rt;         // Receiving from odd Pipe
logic [0:14] Pc_out_2;
logic wr_en_odd, flush_fwd;        // Receiving from odd Pipe

output logic [0:175] Rt_Temp_odd_Rout, Rt_Temp1_odd_Rout, Rt_Temp2_odd_Rout, Rt_Temp3_odd_Rout,
Rt_Temp4_odd_Rout, Rt_Temp5_odd_Rout, Rt_Temp6_odd_Rout;
logic [0:175] Rt_Temp_odd, Rt_Temp1_odd, Rt_Temp2_odd, Rt_Temp3_odd, Rt_Temp4_odd, Rt_Temp5_odd,
Rt_Temp6_odd;

logic [0:127] data_odd_ra, data_odd_rb, data_odd_rc; // Data to odd Pipe
logic [0:10] opcode_odd_fwd;      // Forward Reg for 1 Cycle Delay
logic [0:6] imm7_odd_fwd;         // Forward Reg for 1 Cycle Delay
logic [0:9] imm10_odd_fwd;        // Forward Reg for 1 Cycle Delay
logic [0:15] imm16_odd_fwd;       // Forward Reg for 1 Cycle Delay
logic [0:17] imm18_odd_fwd;       // Forward Reg for 1 Cycle Delay
logic [0:6] addr_odd_rt_fwd;      // Forward Reg for 1 Cycle Delay
logic [0:14] Pc_in_fwd;           // Forward Reg for 1 Cycle Delay

// Odd and Even Pipe Connections

evenpipe evenp ( .clk(clk),                // Input
.reset(reset),                            // Input
.opcode_even(opcode_even_fwd),             // Input
.imm7_even(imm7_even_fwd),                 // Input
.imm10_even(imm10_even_fwd),               // Input
.imm16_even(imm16_even_fwd),               // Input
.imm18_even(imm18_even_fwd),               // Input
.data_even_ra(data_even_ra),               // Input
.data_even_rb(data_even_rb),               // Input
.data_even_rc(data_even_rc),               // Input
.addr_even_rt(addr_even_rt_fwd),            // Input
.addr_even_rt2(addr_even_rt2),              // Output Receiving
.data_even_rt(data_even_rt),               // Output Receiving
.wr_en_even(wr_en_even),                   // Output Receiving
.Rt_Temp(Rt_Temp_even), // Rt_Temp_even

.Rt_Temp1(Rt_Temp1_even),
.Rt_Temp2(Rt_Temp2_even),
.Rt_Temp3(Rt_Temp3_even),
.Rt_Temp4(Rt_Temp4_even),
.Rt_Temp5(Rt_Temp5_even),
.Rt_Temp6(Rt_Temp6_even));

```

```

oddpipeline oddp (      .clk(clk),           // Input
                        .reset(reset),       // Input
                        .opcode_odd(opcode_odd_fwd), // Input
                        .imm7_odd(imm7_odd_fwd), // Input
                        .imm10_odd(imm10_odd_fwd), // Input
                        .imm16_odd(imm16_odd_fwd), // Input
                        .imm18_odd(imm18_odd_fwd), // Input
                        .data_odd_ra(data_odd_ra), // Input
                        .data_odd_rb(data_odd_rb), // Input
                        .data_odd_rc(data_odd_rc), // Input
                        .addr_odd_rt(addr_odd_rt_fwd), // Input
                        .addr_odd_rt2(addr_odd_rt2), //Output Receiving
                        .data_odd_rt(data_odd_rt), //Output Receiving
                        .wr_en_odd(wr_en_odd), //Output Receiving
                        .Pc_in(Pc_in_fwd), //Output Receiving
                        .Pc_out_2(Pc_out_2), //Output Receiving
                        .flush(flush_fwd), // Flush Signal from TB
                        .Rt_Temp0_fwd(Rt_Temp_odd),
                        .Rt_Temp1(Rt_Temp1_odd),
                        .Rt_Temp2(Rt_Temp2_odd),
                        .Rt_Temp3(Rt_Temp3_odd),
                        .Rt_Temp4(Rt_Temp4_odd),
                        .Rt_Temp5(Rt_Temp5_odd),
                        .Rt_Temp6(Rt_Temp6_odd));

```

```

always_comb begin

```

```

    Rt_Temp_odd_Rout = Rt_Temp_odd;
    Rt_Temp1_odd_Rout = Rt_Temp1_odd;
    Rt_Temp2_odd_Rout = Rt_Temp2_odd;
    Rt_Temp3_odd_Rout = Rt_Temp3_odd;
    Rt_Temp4_odd_Rout = Rt_Temp4_odd;
    Rt_Temp5_odd_Rout = Rt_Temp5_odd;
    Rt_Temp6_odd_Rout = Rt_Temp6_odd;

```

```

    Rt_Temp_even_Rout = Rt_Temp_even;
    Rt_Temp1_even_Rout = Rt_Temp1_even;
    Rt_Temp2_even_Rout = Rt_Temp2_even;
    Rt_Temp3_even_Rout = Rt_Temp3_even;
    Rt_Temp4_even_Rout = Rt_Temp4_even;
    Rt_Temp5_even_Rout = Rt_Temp5_even;
    Rt_Temp6_even_Rout = Rt_Temp6_even;

```

```

end

```

```

//always_ff @(posedge clk) begin $display("@REG FILE: %d", Rt_Temp_even_Rout[0:2]); $display("@REG 2 FILE: %d",
Rt_Temp_even[0:2]); end

```

```

logic [0:127][0:127] mem;

```

```

assign Even_Test_Packet = {data_even_rt, addr_even_rt2, wr_en_even};
assign Odd_Test_Packet = {data_odd_rt, addr_odd_rt2, wr_en_odd, Pc_out_2};

```

```

always_ff @(posedge clk) begin

```

```

    if(reset) begin
        integer i;
        for (i=0; i<124; i=i+1)
            mem[i] = i;
        mem[124]=32'h408ccccd;
        mem[125]=128'd79228162532711081671548469249;
        mem[126]=128'd170808406006153739902585569290863280256;
        mem[127]=128'd170808403787765189503184116671632670848;
    end

```

```

        mem[12] =
128'b00111111000000000000000000000000_01000000000000000000000000000000_0100000001000000000000000000
00000_01000000100000000000000000000000; // A Reg Matrix 1 (1,2,3,4)
        //mem[112] = 128'h3F800000400000004040000040800000; // A Reg Matrix 1 (1,2,3,4)
        mem[13] =
128'b01000000101000000000000000000000_01000000111000000000000000000000_0100000010100000000000000000
00000_01000000111000000000000000000000; // B1 Reg Matrix (5 7 5 7)
        mem[14] =
128'b01000000110000000000000000000000_01000001000000000000000000000000_0100000011000000000000000000
00000_01000001000000000000000000000000; // B2 Reg Matrix (6 8 6 8)

    end
end

always_ff @(posedge clk) begin          // Write data to Even Pipe
    if(wr_en_even)
        mem[addr_even_rt2] <= data_even_rt;
    end

    always_ff @(posedge clk) begin          // Write data to Odd Pipe
        if(wr_en_odd)
            mem[addr_odd_rt2] <= data_odd_rt;
        end
    end

/*always_ff @(posedge clk) begin

    data_even_ra1 <= data_even_ra;
    data_even_ra2 <= data_even_ra1;
    data_even_ra3 <= data_even_ra2;
    data_even_ra4 <= data_even_ra3;
    data_even_ra5 <= data_even_ra4;
    data_even_ra6 <= data_even_ra5;
    end
*/

always_ff @(posedge clk) begin

    if (Rt_Temp1_even [7:14] == addr_even_ra && Rt_Temp1_even[3:5] == 2) begin // Even vs Even

        data_even_ra <= Rt_Temp1_even [15:142];
        data_even_rb <= mem[addr_even_rb];
        data_even_rc <= mem[addr_even_rc];

        data_odd_ra <= mem[addr_odd_ra];
        data_odd_rb <= mem[addr_odd_rb];
        data_odd_rc <= mem[addr_odd_rc];end

    else if(Rt_Temp1_even [7:14] == addr_even_rb && Rt_Temp1_even[3:5] == 2) begin // Even vs Even
        data_even_ra <= mem[addr_even_ra];
        data_even_rb <= Rt_Temp1_even [15:142];
        data_even_rc <= mem[addr_even_rc];

        data_odd_ra <=mem[addr_odd_ra];
        data_odd_rb <= mem[addr_odd_rb];
        data_odd_rc <= mem[addr_odd_rc]; end

    else if (Rt_Temp1_even [7:14] == addr_even_rc && Rt_Temp1_even[3:5] == 2) begin // Even vs Even
        data_even_ra <=mem[addr_even_ra];
        data_even_rb <= mem[addr_even_rb];
        data_even_rc <= Rt_Temp1_even [15:142];

        data_odd_ra <=mem[addr_odd_ra];
        data_odd_rb <= mem[addr_odd_rb];

```

```

data_odd_rc <= mem[addr_odd_rc]; end

else if (Rt_Temp1_even [7:14] == addr_odd_ra && Rt_Temp1_even[3:5] == 2) begin // Even vs Odd
data_odd_ra <= Rt_Temp1_even [15:142];
data_odd_rb <= mem[addr_odd_rb];
data_odd_rc <= mem[addr_odd_rc];

data_even_ra <=mem[addr_even_ra];
data_even_rb <= mem[addr_even_rb];
data_even_rc <= mem[addr_even_rc]; end

else if(Rt_Temp1_even [7:14] == addr_odd_rb && Rt_Temp1_even[3:5] == 2) begin // Even vs Odd
data_odd_ra <= mem[addr_odd_ra];
data_odd_rb <= Rt_Temp1_even [15:142];
data_odd_rc <= mem[addr_odd_rc];

data_even_ra <=mem[addr_even_ra];
data_even_rb <= mem[addr_even_rb];
data_even_rc <= mem[addr_even_rc]; end

else if (Rt_Temp1_even [7:14] == addr_odd_rc && Rt_Temp1_even[3:5] == 2) begin // Even vs Odd
data_odd_ra <= mem[addr_odd_ra];
data_odd_rb <= mem[addr_odd_rb];
data_odd_rc <= Rt_Temp1_even [15:142];

data_even_ra <=mem[addr_even_ra];
data_even_rb <= mem[addr_even_rb];
data_even_rc <= mem[addr_even_rc];end

// -----

else if ( Rt_Temp2_even [7:14] == addr_even_ra && Rt_Temp2_even[3:5] == 3) begin // Even vs Even
data_even_ra <= Rt_Temp1_even [15:142];
data_even_rb <= mem[addr_even_rb];
data_even_rc <= mem[addr_even_rc];

data_odd_ra <=mem[addr_odd_ra];
data_odd_rb <= mem[addr_odd_rb];
data_odd_rc <= mem[addr_odd_rc];end

else if( Rt_Temp2_even [7:14] == addr_even_rb && Rt_Temp2_even[3:5] == 3) begin // Even vs Even
data_even_ra <= mem[addr_even_ra];
data_even_rb <= Rt_Temp1_even [15:142];
data_even_rc <= mem[addr_even_rc];

data_odd_ra <=mem[addr_odd_ra];
data_odd_rb <= mem[addr_odd_rb];
data_odd_rc <= mem[addr_odd_rc]; end

else if ( Rt_Temp2_even [7:14] == addr_even_rc && Rt_Temp2_even[3:5] == 3) begin // Even vs Even
data_even_ra <=mem[addr_even_ra];
data_even_rb <= mem[addr_even_rb];
data_even_rc <= Rt_Temp1_even [15:142];

data_odd_ra <=mem[addr_odd_ra];
data_odd_rb <= mem[addr_odd_rb];
data_odd_rc <= mem[addr_odd_rc]; end

else if (Rt_Temp2_even [7:14] == addr_odd_ra && Rt_Temp2_even[3:5] == 3) begin // Even vs Odd
data_odd_ra <= Rt_Temp1_even [15:142];
data_odd_rb <= mem[addr_odd_rb];
data_odd_rc <= mem[addr_odd_rc];

```



```

data_even_ra <=mem[addr_even_ra];
data_even_rb <= mem[addr_even_rb];
data_even_rc <= mem[addr_even_rc]; end

else if( Rt_Temp2_even [7:14] == addr_odd_rb && Rt_Temp2_even[3:5] == 3) begin // Even vs Odd
data_odd_ra <= mem[addr_odd_ra];
data_odd_rb <= Rt_Temp1_even [15:142];
data_odd_rc <= mem[addr_odd_rc];

data_even_ra <=mem[addr_even_ra];
data_even_rb <= mem[addr_even_rb];
data_even_rc <= mem[addr_even_rc]; end

else if ( Rt_Temp2_even [7:14] == addr_odd_rc && Rt_Temp2_even[3:5] == 3) begin // Even vs Odd
data_odd_ra <= mem[addr_odd_ra];
data_odd_rb <= mem[addr_odd_rb];
data_odd_rc <= Rt_Temp1_even [15:142];

data_even_ra <=mem[addr_even_ra];
data_even_rb <= mem[addr_even_rb];
data_even_rc <= mem[addr_even_rc];end

//latency=4,stage=4
else if (Rt_Temp3_even [7:14] == addr_even_ra && Rt_Temp3_even[3:5] == 4) begin // Even vs Even
data_even_ra <= Rt_Temp3_even [15:142];
data_even_rb <= mem[addr_even_rb];
data_even_rc <= mem[addr_even_rc];

data_odd_ra <=mem[addr_odd_ra];
data_odd_rb <= mem[addr_odd_rb];
data_odd_rc <= mem[addr_odd_rc];end

else if(Rt_Temp3_even [7:14] == addr_even_rb && Rt_Temp3_even[3:5] == 4) begin // Even vs Even
data_even_ra <= mem[addr_even_ra];
data_even_rb <= Rt_Temp3_even [15:142];
data_even_rc <= mem[addr_even_rc];

data_odd_ra <=mem[addr_odd_ra];
data_odd_rb <= mem[addr_odd_rb];
data_odd_rc <= mem[addr_odd_rc]; end

else if (Rt_Temp3_even [7:14] == addr_even_rc && Rt_Temp3_even[3:5] == 4) begin // Even vs Even
data_even_ra <=mem[addr_even_ra];
data_even_rb <= mem[addr_even_rb];
data_even_rc <= Rt_Temp3_even [15:142];

data_odd_ra <=mem[addr_odd_ra];
data_odd_rb <= mem[addr_odd_rb];
data_odd_rc <= mem[addr_odd_rc]; end

else if (Rt_Temp3_even [7:14] == addr_odd_ra && Rt_Temp3_even[3:5] == 4) begin // Even vs Odd
data_odd_ra <= Rt_Temp3_even [15:142];
data_odd_rb <= mem[addr_odd_rb];
data_odd_rc <= mem[addr_odd_rc];

data_even_ra <=mem[addr_even_ra];
data_even_rb <= mem[addr_even_rb];
data_even_rc <= mem[addr_even_rc]; end

else if(Rt_Temp3_even [7:14] == addr_odd_rb && Rt_Temp3_even[3:5] == 4) begin // Even vs Odd
data_odd_ra <= mem[addr_odd_ra];
data_odd_rb <= Rt_Temp3_even [15:142];
data_odd_rc <= mem[addr_odd_rc];

```

```

data_even_ra <= mem[addr_even_ra];
data_even_rb <= mem[addr_even_rb];
data_even_rc <= mem[addr_even_rc]; end

else if (Rt_Temp3_even [7:14] == addr_odd_rc && Rt_Temp3_even[3:5] == 4) begin // Even vs Odd
data_odd_ra <= mem[addr_odd_ra];
data_odd_rb <= mem[addr_odd_rb];
data_odd_rc <= Rt_Temp3_even [15:142];

data_even_ra <= mem[addr_even_ra];
data_even_rb <= mem[addr_even_rb];
data_even_rc <= mem[addr_even_rc]; end

else if (Rt_Temp4_even [7:14] == addr_even_ra && Rt_Temp4_even[3:5] == 5) begin // Even vs Even
data_even_ra <= Rt_Temp4_even [15:142];
data_even_rb <= mem[addr_even_rb];
data_even_rc <= mem[addr_even_rc];

data_odd_ra <= mem[addr_odd_ra];
data_odd_rb <= mem[addr_odd_rb];
data_odd_rc <= mem[addr_odd_rc]; end

else if (Rt_Temp4_even [7:14] == addr_even_rb && Rt_Temp4_even[3:5] == 5) begin // Even vs Even
data_even_ra <= mem[addr_even_ra];
data_even_rb <= Rt_Temp4_even [15:142];
data_even_rc <= mem[addr_even_rc];

data_odd_ra <= mem[addr_odd_ra];
data_odd_rb <= mem[addr_odd_rb];
data_odd_rc <= mem[addr_odd_rc]; end

else if (Rt_Temp4_even [7:14] == addr_even_rc && Rt_Temp4_even[3:5] == 5) begin // Even vs Even
data_even_ra <= mem[addr_even_ra];
data_even_rb <= mem[addr_even_rb];
data_even_rc <= Rt_Temp4_even [15:142];

data_odd_ra <= mem[addr_odd_ra];
data_odd_rb <= mem[addr_odd_rb];
data_odd_rc <= mem[addr_odd_rc]; end

else if (Rt_Temp4_even [7:14] == addr_odd_ra && Rt_Temp4_even[3:5] == 5) begin // Even vs Odd
data_odd_ra <= Rt_Temp4_even [15:142];
data_odd_rb <= mem[addr_odd_rb];
data_odd_rc <= mem[addr_odd_rc];

data_even_ra <= mem[addr_even_ra];
data_even_rb <= mem[addr_even_rb];
data_even_rc <= mem[addr_even_rc]; end

else if (Rt_Temp4_even [7:14] == addr_odd_rb && Rt_Temp4_even[3:5] == 5) begin // Even vs Odd
data_odd_ra <= mem[addr_odd_ra];
data_odd_rb <= Rt_Temp4_even [15:142];
data_odd_rc <= mem[addr_odd_rc];

data_even_ra <= mem[addr_even_ra];
data_even_rb <= mem[addr_even_rb];
data_even_rc <= mem[addr_even_rc]; end

else if (Rt_Temp4_even [7:14] == addr_odd_rc && Rt_Temp4_even[3:5] == 5) begin // Even vs Odd
data_odd_ra <= mem[addr_odd_ra];
data_odd_rb <= mem[addr_odd_rb];

```

```

data_odd_rc <= Rt_Temp4_even [15:142];

data_even_ra <= mem[addr_even_ra];
data_even_rb <= mem[addr_even_rb];
data_even_rc <= mem[addr_even_rc];end

else if (Rt_Temp5_even [7:14] == addr_even_ra && Rt_Temp5_even[3:5] == 6) begin    // Even vs Even
    data_even_ra <= Rt_Temp5_even [15:142];
    data_even_rb <= mem[addr_even_rb];
    data_even_rc <= mem[addr_even_rc];

    data_odd_ra <= mem[addr_odd_ra];
    data_odd_rb <= mem[addr_odd_rb];
    data_odd_rc <= mem[addr_odd_rc];end

else if(Rt_Temp5_even [7:14] == addr_even_rb && Rt_Temp5_even[3:5] == 6) begin // Even vs Even
    data_even_ra <= mem[addr_even_ra];
    data_even_rb <= Rt_Temp5_even [15:142];
    data_even_rc <= mem[addr_even_rc];

    data_odd_ra <= mem[addr_odd_ra];
    data_odd_rb <= mem[addr_odd_rb];
    data_odd_rc <= mem[addr_odd_rc]; end

else if (Rt_Temp5_even [7:14] == addr_even_rc && Rt_Temp5_even[3:5] == 6) begin  // Even vs Even
    data_even_ra <= mem[addr_even_ra];
    data_even_rb <= mem[addr_even_rb];
    data_even_rc <= Rt_Temp5_even [15:142];

    data_odd_ra <= mem[addr_odd_ra];
    data_odd_rb <= mem[addr_odd_rb];
    data_odd_rc <= mem[addr_odd_rc]; end

else if (Rt_Temp5_even [7:14] == addr_odd_ra && Rt_Temp5_even[3:5] == 6) begin // Even vs Odd
    data_odd_ra <= Rt_Temp5_even [15:142];
    data_odd_rb <= mem[addr_odd_rb];
    data_odd_rc <= mem[addr_odd_rc];

    data_even_ra <= mem[addr_even_ra];
    data_even_rb <= mem[addr_even_rb];
    data_even_rc <= mem[addr_even_rc]; end

else if(Rt_Temp5_even [7:14] == addr_odd_rb && Rt_Temp5_even[3:5] == 6) begin  // Even vs Odd
    data_odd_ra <= mem[addr_odd_ra];
    data_odd_rb <= Rt_Temp5_even [15:142];
    data_odd_rc <= mem[addr_odd_rc];

    data_even_ra <= mem[addr_even_ra];
    data_even_rb <= mem[addr_even_rb];
    data_even_rc <= mem[addr_even_rc]; end

else if (Rt_Temp5_even [7:14] == addr_odd_rc && Rt_Temp5_even[3:5] == 6) begin // Even vs Odd
    data_odd_ra <= mem[addr_odd_ra];
    data_odd_rb <= mem[addr_odd_rb];
    data_odd_rc <= Rt_Temp5_even [15:142];

    data_even_ra <= mem[addr_even_ra];
    data_even_rb <= mem[addr_even_rb];
    data_even_rc <= mem[addr_even_rc];end

else if (Rt_Temp6_even [7:14] == addr_even_ra && Rt_Temp6_even[3:5] == 7) begin    // Even vs Even

```

```

data_even_ra <= Rt_Temp6_even [15:142];
data_even_rb <= mem[addr_even_rb];
data_even_rc <= mem[addr_even_rc];

data_odd_ra <=mem[addr_odd_ra];
data_odd_rb <= mem[addr_odd_rb];
data_odd_rc <= mem[addr_odd_rc];end

else if(Rt_Temp6_even [7:14] == addr_even_rb && Rt_Temp6_even[3:5] == 7) begin // Even vs Even
data_even_ra <= mem[addr_even_ra];
data_even_rb <= Rt_Temp6_even [15:142];
data_even_rc <= mem[addr_even_rc];

data_odd_ra <=mem[addr_odd_ra];
data_odd_rb <= mem[addr_odd_rb];
data_odd_rc <= mem[addr_odd_rc]; end

else if (Rt_Temp6_even [7:14] == addr_even_rc && Rt_Temp6_even[3:5] == 7) begin // Even vs Even
data_even_ra <=mem[addr_even_ra];
data_even_rb <= mem[addr_even_rb];
data_even_rc <= Rt_Temp6_even [15:142];

data_odd_ra <=mem[addr_odd_ra];
data_odd_rb <= mem[addr_odd_rb];
data_odd_rc <= mem[addr_odd_rc]; end

else if (Rt_Temp6_even [7:14] == addr_odd_ra && Rt_Temp6_even[3:5] == 7) begin // Even vs Odd
data_odd_ra <= Rt_Temp6_even [15:142];
data_odd_rb <= mem[addr_odd_rb];
data_odd_rc <= mem[addr_odd_rc];

data_even_ra <=mem[addr_even_ra];
data_even_rb <= mem[addr_even_rb];
data_even_rc <= mem[addr_even_rc]; end

else if(Rt_Temp6_even [7:14] == addr_odd_rb && Rt_Temp6_even[3:5] == 7) begin // Even vs Odd
data_odd_ra <= mem[addr_odd_ra];
data_odd_rb <= Rt_Temp6_even [15:142];
data_odd_rc <= mem[addr_odd_rc];

data_even_ra <=mem[addr_even_ra];
data_even_rb <= mem[addr_even_rb];
data_even_rc <= mem[addr_even_rc]; end

else if (Rt_Temp6_even [7:14] == addr_odd_rc && Rt_Temp6_even[3:5] == 7) begin // Even vs Odd
data_odd_ra <= mem[addr_odd_ra];
data_odd_rb <= mem[addr_odd_rb];
data_odd_rc <= Rt_Temp6_even [15:142];

data_even_ra <=mem[addr_even_ra];
data_even_rb <= mem[addr_even_rb];
data_even_rc <= mem[addr_even_rc];end

else begin

data_even_ra <= mem[addr_even_ra];
data_even_rb <= mem[addr_even_rb];
data_even_rc <= mem[addr_even_rc];

data_odd_ra <= mem[addr_odd_ra];
data_odd_rb <= mem[addr_odd_rb];
data_odd_rc <= mem[addr_odd_rc];
end
end

```

```

end
always_ff @(posedge clk) $display("Opcode:%b, data_even_ra:%d, data_even_rb:%d, data_even_rc:%d",opcode_even,
data_even_ra, data_even_rb, data_even_rc);

//end
always_ff @(posedge clk) begin      // Instruction Forward to Pipes
    opcode_even_fwd <= opcode_even; addr_even_rt_fwd <= addr_even_rt;
    imm7_even_fwd <= imm7_even; imm10_even_fwd <= imm10_even;
    imm16_even_fwd <= imm16_even; imm18_even_fwd <= imm18_even;

    opcode_odd_fwd <= opcode_odd; addr_odd_rt_fwd <= addr_odd_rt;
    imm7_odd_fwd <= imm7_odd; imm10_odd_fwd <= imm10_odd;
    imm16_odd_fwd <= imm16_odd; imm18_odd_fwd <= imm18_odd;
    Pc_in_fwd <= Pc_in; flush_fwd <= flush;
end

//always_comb $display ("=====>>> unit_temp:%d , latency_temp:%d , wr_en:%d , addr_even_rt:%d",
Rt_Temp_even[0:2], Rt_Temp_even[3:5], Rt_Temp_even[6], Rt_Temp_even[7:14]);

endmodule

```

Script: Fetch (Cache + PC)

```

//===== LOCAL STORE
//===== (0 CYCLE LAT)

module Local_Store (clk, reset, data_in, data_out, addr, wr_en, pc, imiss);

    input clk, reset, wr_en, imiss;
    input [0:14] pc;
    input [0:31] data_in [0:127];
    input [0:14] addr;
    output logic [0:1023] data_out;

    integer i;

    logic [0:7] local_st [0:32768]; // 32 KB Local Store
    logic [0:31] data_in_tmp;

    always_comb begin
        if (wr_en) begin
            for(i=0;i<127;i=i+1) begin
                data_in_tmp = data_in[i];
                for(int k=0; k<4; k=k+1) begin
                    local_st[(4*i)+k] = data_in_tmp[(k*8)+:8]; end
            end
        end

        if (imiss == 1) begin
            for (int j=0; j<128;j=j+1) begin
                data_out[j*8+:8] = local_st[j+pc];
            end end
        // $display("Local_St Addr: %b, Data Out:%b",addr, data_out);
    end

endmodule

// $display("Cache 01:%b_%b_%b_%b",Cache[0],Cache[1],Cache[2],Cache[3]);
//===== PRG CTR
//===== (0 CYCLE LAT)

module PC(clk, reset, pc, imiss, decode_stall_1, Dependency_stall);

```

```
input clk, reset, imiss, decode_stall_1, Dependency_stall;
output logic [0:14] pc; logic [0:14] pc_old;
```

```
always_ff @(posedge clk) begin
    if (reset) pc = 0; end
```

```
always_ff @(posedge clk) begin
```

```
    if(imiss || decode_stall_1 || Dependency_stall) pc<=pc;
    //else if (Dependency_stall) pc = pc_old;
```

```
    else begin
        pc <= pc +8;
        pc_old <= pc; end
    end
```

```
// always_comb begin
//     if(reset)
//         pc = 0;
//     else if(imiss)
//         pc = pc;
//     else if (decode_stall_1) pc = pc;
//     else if(Dependency_stall) pc = pc;// - 8;
//     else
//         pc = pc + 8;
//         pc_NonBlock <= pc;
//     $display("Program Counter:%d", pc_NonBlock);
// end
```

```
endmodule
```

```
//===== CACHE ILB
//===== (1 CYCLE LAT)
```

```
module Cache (clk, reset, Cache_load_addr, Cache_load_data, pc, Inst_out, imiss);
```

```
integer i, j;
input clk, reset;
input [0:14] pc;
output logic [0:14] Cache_load_addr;
output logic [0:63] Inst_out;
input [0:1023] Cache_load_data;
output logic imiss;
```

```
logic [0:31] Inst_Fetch_1, Inst_Fetch_2;
logic [0:7] Cache [0:511];
logic Cache_valid_B1, Cache_valid_B2, Cache_valid_B3, Cache_valid_B4; // 4 Valid Bits
logic [0:7] Cache_tag_B1,Cache_tag_B2,Cache_tag_B3,Cache_tag_B4; // 4 Valid Bits
```

```
always_comb begin
    // $display("imiss:%b",imiss);
    if((imiss==1)&&(pc[0:14] >=0)) begin Cache_valid_B1 = 1'b1; end//$display("Cache_valid_B1:%b", Cache_valid_B1); end
    if((imiss==1)&&(pc[0:14] >=128))begin Cache_valid_B2 = 1'b1;end//$display("Cache_valid_B2:%b", Cache_valid_B2); end
    if((imiss==1)&&(pc[0:14] >=256))begin Cache_valid_B3 = 1'b1; end//$display("Cache_valid_B3:%b", Cache_valid_B3); end
    if((imiss==1)&&(pc[0:14] >=384))begin Cache_valid_B4 = 1'b1; end//$display("Cache_valid_B4:%b", Cache_valid_B4); end
    if((imiss==1)&&(pc[7]==0 & pc[8]==0))begin Cache_tag_B1=pc[0:7]; end//$display("Cache_tag_B1:%b", Cache_tag_B1); end
    if((imiss==1)&&(pc[7]==0 & pc[8]==1))begin Cache_tag_B2=pc[0:7]; end//$display("Cache_tag_B2:%b", Cache_tag_B2); end
    if((imiss==1)&&(pc[7]==1 & pc[8]==0))begin Cache_tag_B3=pc[0:7]; end//$display("Cache_tag_B3:%b", Cache_tag_B3); end
    if((imiss==1)&&(pc[7]==1 & pc[8]==1))begin Cache_tag_B4=pc[0:7]; end//$display("Cache_tag_B4:%b", Cache_tag_B4); end
end
```

```

assign Inst_Fetch_1 = Inst_out[0:10];
assign Inst_Fetch_2 = Inst_out[32:42];

always_ff @(posedge clk) begin // Cache Hit - Passing - Output
    if((Cache_valid_B1 == 1)&&(pc[0:7]==Cache_tag_B1)) begin //$display ("Hit 01");
        for(j=0;j<64;j=j+8) begin
            imiss = 0;
            Inst_out[j+:8] = Cache[j/8+pc[9:14]];
        end
        //$display("Program Counter_ Block Offset: %d",pc[8:14]);
        //$display("Cache Hit Instruction Block 1:%b, -- %b",Inst_out[0:31], Inst_out[32:63]);
    end

    else if ((Cache_valid_B2 == 1)&&(pc[0:7]==Cache_tag_B2)) begin //$display ("Hit 02");
        for(j=0;j<64;j=j+8) begin
            imiss = 0;
            Inst_out[j+:8] = Cache[j/8+pc[9:14]+128];
        end end

    else if ((Cache_valid_B3 == 1)&&(pc[0:7]==Cache_tag_B3)) begin //$display ("Hit 03");
        for(j=0;j<64;j=j+8) begin
            imiss = 0;
            Inst_out[j+:8] = Cache[j/8+pc[9:14]+256];
        end end

    else if ((Cache_valid_B4 == 1)&&(pc[0:7]==Cache_tag_B4)) begin //$display ("Hit 04");
        for(j=0;j<64;j=j+8) begin
            imiss = 0;
            Inst_out[j+:8] = Cache[j/8+pc[9:14]+384];
        end end

    else begin
        imiss=1; //$display ("-----Default of Else");
        Inst_out[0:10] <= 11'b0100_0000_001; Inst_out[32:42] <= 11'b0100_0000_010; end // Pass Imiss Later
    end

always_comb begin // Cache Miss - Loading - Input
    if(imiss == 1) begin
        if(pc[7]==0 & pc[8]==0)begin
            //$display("Cache Miss Loading Block 1");
            for (i=0;i<128;i=i+1) begin
                Cache[i]= Cache_load_data[i*8+:8];end
            //$display("Cache Block 1 Instruction 1:%b_%b_%b_%b",Cache[0], Cache[1], Cache[2], Cache[3]);
            //$display("Cache Block 1 Instruction 2:%b_%b_%b_%b",Cache[4], Cache[5], Cache[6], Cache[7]);
            //$display("Cache Block 1 Instruction 3:%b_%b_%b_%b",Cache[8], Cache[9], Cache[10], Cache[11]);
            //$display("Cache Block 1 Instruction 4:%b_%b_%b_%b",Cache[12], Cache[13], Cache[14], Cache[15]);
            //$display("Cache Block 1 Instruction 5:%b_%b_%b_%b",Cache[16], Cache[17], Cache[18], Cache[19]);
            //$display("Cache Block 1 Instruction 6:%b_%b_%b_%b",Cache[20], Cache[21], Cache[22], Cache[23]);
            //$display("Cache Block 1 Instruction 7:%b_%b_%b_%b",Cache[24], Cache[25], Cache[26], Cache[27]);
            //$display("Cache Block 1 Data Received:%b",Cache_load_data); end
            //$display("Instruction 1 in Cache:%b_%b_%b_%b",Cache[0],Cache[1],Cache[2], Cache[3]);
        end end
        else if(pc[7]==0 & pc[8]==1)begin
            //$display("Cache Miss Loading Block 2");
            for (i=0;i<128;i=i+1) begin
                Cache[i]= Cache_load_data[i*8+:8];end
        end
        else if(pc[7]==1 & pc[8]==0)begin
            //$display("Cache Miss Loading Block 3");
            for (i=0;i<128;i=i+1) begin
                Cache[i]= Cache_load_data[i*8+:8];end
        end
    end
end

```

```

else if(pc[7]==1 & pc[8]==1)begin
  //$display("Cache Miss Loading Block 4");
  for (i=0;i<128;i=i+1) begin
    Cache[i]= Cache_load_data[i*8+:8];end
  end
end
endmodule

```

Script: Decode

```

# //===== DECODE
===== (1 CYCLE LAT)

```

```

module Decode (clk, reset, Dependency_Stall, Inst_out, decode_stall_1, addr_even_ra, addr_even_rb, addr_even_rc, opcode_even,
imm10_even, imm16_even, imm18_even, imm7_even, addr_even_rt, addr_odd_ra, addr_odd_rb, addr_odd_rc, opcode_odd,
imm10_odd, imm16_odd, imm18_odd, imm7_odd, addr_odd_rt );

```

```

input clk, reset, Dependency_Stall;
input [0:63] Inst_out;

```

```

output logic [0:6] addr_even_ra, addr_even_rb, addr_even_rc;
output logic [0:6] addr_odd_ra, addr_odd_rb, addr_odd_rc;
output logic [0:10] opcode_even;
output logic signed [0:6] imm7_even;
output logic signed [0:9] imm10_even;
output logic signed [0:15] imm16_even;
output logic signed [0:17] imm18_even;
output logic [0:6] addr_even_rt;
output logic [0:10] opcode_odd;
output logic [0:6] imm7_odd;
output logic [0:9] imm10_odd;
output logic [0:15] imm16_odd;
output logic [0:17] imm18_odd;
output logic [0:6] addr_odd_rt;

```

```

output logic decode_stall_1;

```

```

logic Prev_stall;

```

```

// output [0:6] addr_even_ra, addr_even_rb, addr_even_rc;
// output [0:6] addr_odd_ra, addr_odd_rb, addr_odd_rc;
// output [0:10] opcode_even;
// output signed [0:6] imm7_even;
// output signed [0:9] imm10_even;
// output signed [0:15] imm16_even;
// output signed [0:17] imm18_even;
// output [0:6] addr_even_rt;
// output [0:10] opcode_odd;
// output [0:6] imm7_odd;
// output [0:9] imm10_odd;
// output [0:15] imm16_odd;
// output [0:17] imm18_odd;
// output [0:6] addr_odd_rt;

```

```

// output logic decode_stall_1;

```

```

logic Even_inst1, Even_inst2, Prev_Even_Hazard, Prev_Odd_Hazard, Even_Structural_Hazard, No_Structural_Hazard,
Odd_Structural_Hazard;
logic [0:6] addr_even_ra_inst1, addr_even_rb_inst1, addr_even_rc_inst1, addr_even_ra_inst2, addr_even_rb_inst2,
addr_even_rc_inst2;

```



```

logic [0:6] addr_odd_ra_inst1, addr_odd_rb_inst1, addr_odd_rc_inst1, addr_odd_ra_inst2, addr_odd_rb_inst2,
addr_odd_rc_inst2;
logic [0:10] opcode_even_inst1, opcode_even_inst2;    // logic from TB

logic signed [0:6] imm7_even_inst1, imm7_even_inst2; // logic from TB
logic signed [0:9] imm10_even_inst1, imm10_even_inst2; // logic from TB
logic signed [0:15] imm16_even_inst1, imm16_even_inst2; // logic from TB
logic signed [0:17] imm18_even_inst1, imm18_even_inst2; // logic from TB
logic [0:6] addr_even_rt_inst1, addr_even_rt_inst2;    // logic from TB

logic [0:10] opcode_odd_inst1, opcode_odd_inst2;    // logic from TB
logic [0:6] imm7_odd_inst1, imm7_odd_inst2;    // logic from TB
logic [0:9] imm10_odd_inst1, imm10_odd_inst2 ;    // logic from TB
logic [0:15] imm16_odd_inst1, imm16_odd_inst2;    // logic from TB
logic [0:17] imm18_odd_inst1, imm18_odd_inst2;    // logic from TB
logic [0:6] addr_odd_rt_inst1, addr_odd_rt_inst2;    // logic from TB

logic [0:6] addr_even_ra_tmp, addr_even_rb_tmp, addr_even_rc_tmp, addr_even_ra_bck, addr_even_rb_bck,
addr_even_rc_bck;
logic [0:6] addr_odd_ra_tmp, addr_odd_rb_tmp, addr_odd_rc_tmp, addr_odd_ra_bck, addr_odd_rb_bck, addr_odd_rc_bck;
logic [0:10] opcode_even_tmp, opcode_even_bck;    // logic from TB

logic signed [0:6] imm7_even_tmp, imm7_even_bck; // logic from TB
logic signed [0:9] imm10_even_tmp, imm10_even_bck; // logic from TB
logic signed [0:15] imm16_even_tmp, imm16_even_bck; // logic from TB
logic signed [0:17] imm18_even_tmp, imm18_even_bck; // logic from TB
logic [0:6] addr_even_rt_tmp, addr_even_rt_bck;    // logic from TB

logic [0:10] opcode_odd_tmp, opcode_odd_bck;    // logic from TB
logic [0:6] imm7_odd_tmp, imm7_odd_bck;    // logic from TB
logic [0:9] imm10_odd_tmp, imm10_odd_bck ;    // logic from TB
logic [0:15] imm16_odd_tmp, imm16_odd_bck;    // logic from TB
logic [0:17] imm18_odd_tmp, imm18_odd_bck;    // logic from TB
logic [0:6] addr_odd_rt_tmp, addr_odd_rt_bck;    // logic from TB

logic decode_stall_Remove, Even_inst_nop, Odd_inst_nop;

always_comb begin
    // -----01
    // Decode Inst_out[0:31] - Instruction 1 - Even Pipe
    if((Inst_out[0:3] == 4'b1110)
    || (Inst_out[0:3] == 4'b1111)
    || (Inst_out[0:3] == 4'b1100)) begin Even_inst1 = 1; Even_inst_nop = 0;
        opcode_even_inst1 = Inst_out[0:3];
        addr_even_ra_inst1 = Inst_out[18:24];
        addr_even_rb_inst1 = Inst_out[11:17];
        addr_even_rc_inst1 = Inst_out[25:31];
        imm7_even_inst1 = 0;
        imm10_even_inst1 = 0;
        imm16_even_inst1 = 0;
        imm18_even_inst1 = 0;
        addr_even_rt_inst1 = Inst_out[4:10];
    end

    else if((Inst_out[0:6] == 7'b0100_001)) begin Even_inst1 = 1; Even_inst_nop = 0;
        opcode_even_inst1 = Inst_out[0:6];
        addr_even_ra_inst1 = 0;
        addr_even_rb_inst1 = 0;
        addr_even_rc_inst1 = 0;
    end
end

```

```

imm7_even_inst1    = 0;
imm10_even_inst1   = 0;
imm16_even_inst1   = 0;
imm18_even_inst1   = Inst_out[7:24];
addr_even_rt_inst1 = Inst_out[25:31];
end

else if((Inst_out[0:7] == 8'b0001_1100)
|| (Inst_out[0:7] == 8'b0001_0110)
|| (Inst_out[0:7] == 8'b0111_1110)
|| (Inst_out[0:7] == 8'b0111_1100)
|| (Inst_out[0:7] == 8'b0100_1110)
|| (Inst_out[0:7] == 8'b0100_1100)
|| (Inst_out[0:7] == 8'b0101_1110)
|| (Inst_out[0:7] == 8'b0000_0100)
|| (Inst_out[0:7] == 8'b0000_1100)
|| (Inst_out[0:7] == 8'b0100_0110)
|| (Inst_out[0:7] == 8'b0100_0100)
|| (Inst_out[0:7] == 8'b0111_0100)
|| (Inst_out[0:7] == 8'b0111_0101)) begin Even_inst1 = 1; Even_inst_nop = 0;
opcode_even_inst1   = Inst_out[0:7];
addr_even_ra_inst1  = Inst_out[18:24];
addr_even_rb_inst1  = 0;
addr_even_rc_inst1  = 0;
imm7_even_inst1     = 0;
imm10_even_inst1    = Inst_out[8:17];
imm16_even_inst1    = 0;
imm18_even_inst1    = 0;
addr_even_rt_inst1  = Inst_out[25:31];
end

else if((Inst_out[0:8] == 9'b0100_0000_1)
|| (Inst_out[0:8] == 9'b0100_0001_1)) begin Even_inst1 = 1; Even_inst_nop = 0;
opcode_even_inst1   = Inst_out[0:8];
addr_even_ra_inst1  = 0;
addr_even_rb_inst1  = 0;
addr_even_rc_inst1  = 0;
imm7_even_inst1     = 0;
imm10_even_inst1    = 0;
imm16_even_inst1    = Inst_out[9:24];
imm18_even_inst1    = 0;
addr_even_rt_inst1  = Inst_out[25:31];
end

else if((Inst_out[0:10] == 11'b0001_1000_000)
|| (Inst_out[0:10] == 11'b1101_0000_000)
|| (Inst_out[0:10] == 11'b0001_1000_001)
|| (Inst_out[0:10] == 11'b0101_1000_001)
|| (Inst_out[0:10] == 11'b0111_1000_000)
|| (Inst_out[0:10] == 11'b0111_1010_000)
|| (Inst_out[0:10] == 11'b0100_1010_000)
|| (Inst_out[0:10] == 11'b0101_1010_000)
|| (Inst_out[0:10] == 11'b0001_1001_001)
|| (Inst_out[0:10] == 11'b0000_1001_001)
|| (Inst_out[0:10] == 11'b0000_1000_001)
|| (Inst_out[0:10] == 11'b0101_1001_001)
|| (Inst_out[0:10] == 11'b0000_1000_000)
|| (Inst_out[0:10] == 11'b0110_1000_001)
|| (Inst_out[0:10] == 11'b0100_1000_001)
|| (Inst_out[0:10] == 11'b0000_1010_011)
|| (Inst_out[0:10] == 11'b0001_1010_011)
|| (Inst_out[0:10] == 11'b0000_1011_000)
|| (Inst_out[0:10] == 11'b0000_1011_001)

```

```

    ll (Inst_out[0:10] == 11'b0000_1011_011)
    ll (Inst_out[0:10] == 11'b0101_1000_100)
    ll (Inst_out[0:10] == 11'b0101_1000_110)
    ll (Inst_out[0:10] == 11'b0101_1000_101)
    ll (Inst_out[0:10] == 11'b0100_1010_011)
    ll (Inst_out[0:10] == 11'b0111_1000_100)
    ll (Inst_out[0:10] == 11'b0111_1000_111)
    ll (Inst_out[0:10] == 11'b0111_1001_100)) begin Even_inst1 = 1; Even_inst_nop = 0;
opcode_even_inst1 = Inst_out[0:10];
addr_even_ra_inst1 = Inst_out[18:24];
addr_even_rb_inst1 = Inst_out[11:17];
addr_even_rc_inst1 = 0;
imm7_even_inst1 = 0;
imm10_even_inst1 = 0;
imm16_even_inst1 = 0;
imm18_even_inst1 = 0;
addr_even_rt_inst1 = Inst_out[25:31];
end

else if((Inst_out[0:10] == 11'b0101_0110_100)
ll (Inst_out[0:10] == 11'b0101_0110_110)
ll (Inst_out[0:10] == 11'b0101_0101_110)
ll (Inst_out[0:10] == 11'b0101_0100_110)) begin Even_inst1 = 1; Even_inst_nop = 0;
opcode_even_inst1 = Inst_out[0:10];
addr_even_ra_inst1 = Inst_out[18:24];
addr_even_rb_inst1 = 0;
addr_even_rc_inst1 = 0;
imm7_even_inst1 = 0;
imm10_even_inst1 = 0;
imm16_even_inst1 = 0;
imm18_even_inst1 = 0;
addr_even_rt_inst1 = Inst_out[25:31];
end

else if((Inst_out[0:10] == 11'b0000_1111_011)) begin Even_inst1 = 1; Even_inst_nop = 0;
opcode_even_inst1 = Inst_out[0:10];
addr_even_ra_inst1 = Inst_out[18:24];
addr_even_rb_inst1 = 0;
addr_even_rc_inst1 = 0;
imm7_even_inst1 = Inst_out[11:17];
imm10_even_inst1 = 0;
imm16_even_inst1 = 0;
imm18_even_inst1 = 0;
addr_even_rt_inst1 = Inst_out[25:31];
end

else if((Inst_out[0:10] == 11'b0100_0000_001)) begin Even_inst1 = 1;
//else begin
//Even_inst_nop = 1;
opcode_even_inst1 = 11'b0100_0000_001;
addr_even_ra_inst1 = 7'b0;
addr_even_rb_inst1 = 7'b0;
addr_even_rc_inst1 = 7'b0;
imm7_even_inst1 = 7'b0;
imm10_even_inst1 = 10'b0;
imm16_even_inst1 = 16'b0;
imm18_even_inst1 = 18'b0;
addr_even_rt_inst1 = 7'b0;
end

// else begin
// // $display ("===== >>>> Else Reached No-Op Passed -

```

371");

```

// Even_inst1 = 0;
// opcode_even_inst1 = 11'b0100_0000_001;
// addr_even_ra_inst1 = 0;
// addr_even_rb_inst1 = 0;
// addr_even_rc_inst1 = 0;
// imm7_even_inst1 = 0;
// imm10_even_inst1 = 0;
// imm16_even_inst1 = 0;
// imm18_even_inst1 = 0;
// addr_even_rt_inst1 = 0;
// end

// -----03
// Decode Inst_out[0:31] - Instruction 1 - Odd Pipe

if((Inst_out[0:8] == 9'b0010_0001_0)
|| (Inst_out[0:8] == 9'b0010_0000_0)
|| (Inst_out[0:8] == 9'b0011_0001_0)
|| (Inst_out[0:8] == 9'b0011_0011_0)
|| (Inst_out[0:8] == 9'b0011_0000_1)
|| (Inst_out[0:8] == 9'b0010_0000_1))begin Even_inst1 = 0; Odd_inst_nop = 0;
opcode_odd_inst1 =Inst_out[0:8];
addr_odd_ra_inst1 =0;
addr_odd_rb_inst1 =0;
addr_odd_rc_inst1 =0;
addr_odd_rt_inst1=Inst_out[25:31];
imm7_odd_inst1 =0;
imm10_odd_inst1 =0;
imm16_odd_inst1 =Inst_out[9:24];
imm18_odd_inst1 =0;
end

else if ((Inst_out[0:8] == 9'b0011_0010_0)
|| (Inst_out[0:8] == 9'b0011_0000_0))begin Even_inst1 = 0; Odd_inst_nop = 0;
opcode_odd_inst1 =Inst_out[0:8];
addr_odd_ra_inst1 =0;
addr_odd_rb_inst1 =0;
addr_odd_rc_inst1 =0;
addr_odd_rt_inst1=0;
imm7_odd_inst1 =0;
imm10_odd_inst1 =0;
imm16_odd_inst1 =Inst_out[9:24];
imm18_odd_inst1 =0;
end

else if ((Inst_out[0:10] == 11'b0011_0101_000)
|| (Inst_out[0:10] == 11'b0011_0101_001)) begin Even_inst1 = 0; Odd_inst_nop = 0;
opcode_odd_inst1 =Inst_out[0:10];
addr_odd_ra_inst1 =Inst_out[18:24];
addr_odd_rb_inst1 =0;
addr_odd_rc_inst1 =0;
addr_odd_rt_inst1=0;
imm7_odd_inst1 =0;
imm10_odd_inst1 =0;
imm16_odd_inst1 =0;
imm18_odd_inst1 =0;
end

else if ((Inst_out[0:10] == 11'b0011_1111_100)
|| (Inst_out[0:10] == 11'b0011_1111_101)
|| (Inst_out[0:10] == 11'b0011_1111_111)) begin Even_inst1 = 0; Odd_inst_nop = 0;
opcode_odd_inst1 =Inst_out[0:10];
addr_odd_ra_inst1 =Inst_out[18:24];

```

```

addr_odd_rb_inst1 =0;
addr_odd_rc_inst1 =0;
addr_odd_rt_inst1=Inst_out[25:31];
imm7_odd_inst1   =Inst_out[11:17];
imm10_odd_inst1  =0;
imm16_odd_inst1  =0;
imm18_odd_inst1  =0;
end

else if ((Inst_out[0:10] == 11'b0011_1011_100)
|| (Inst_out[0:10] == 11'b0011_1011_101)
|| (Inst_out[0:10] == 11'b0011_1011_111)
|| (Inst_out[0:10] == 11'b0011_1000_100)
|| (Inst_out[0:10] == 11'b0010_1000_100))begin Even_inst1 = 0; Odd_inst_nop = 0;
opcode_odd_inst1  =Inst_out[0:10];
addr_odd_ra_inst1 =Inst_out[18:24];
addr_odd_rb_inst1 =Inst_out[11:17];
addr_odd_rc_inst1 =0;
addr_odd_rt_inst1=Inst_out[25:31];
imm7_odd_inst1   =0;
imm10_odd_inst1  =0;
imm16_odd_inst1  =0;
imm18_odd_inst1  =0;
end

else if ((Inst_out[0:10] == 11'b0100_0000_010))begin Even_inst1 = 0;
//Odd_inst_nop = 1;
opcode_odd_inst1  =Inst_out[0:10];
addr_odd_ra_inst1 =7'b0;
addr_odd_rb_inst1 =7'b0;;
addr_odd_rc_inst1 =7'b0;;
addr_odd_rt_inst1=7'b0;;
imm7_odd_inst1   =7'b0;;
imm10_odd_inst1  =10'b0;;
imm16_odd_inst1  =16'b0;;
imm18_odd_inst1  =18'b0;;
end

// -----000-----02
// Decode Inst_out[0:31] - Instruction 1 - Even Pipe
if((Inst_out[32:35] == 4'b1110)
|| (Inst_out[32:35] == 4'b1111)
|| (Inst_out[32:35] == 4'b1100)) begin Even_inst2 = 1; Even_inst_nop = 0;
opcode_even_inst2 = Inst_out[32:35];
addr_even_ra_inst2 = Inst_out[50:56];
addr_even_rb_inst2 = Inst_out[43:49];
addr_even_rc_inst2 = Inst_out[57:63];
imm7_even_inst2   = 0;
imm10_even_inst2  = 0;
imm16_even_inst2  = 0;
imm18_even_inst2  = 0;
addr_even_rt_inst2 = Inst_out[36:42];
end

else if((Inst_out[32:38] == 7'b0100_001)) begin Even_inst2 = 1; Even_inst_nop = 0;
opcode_even_inst2 = Inst_out[32:38];
addr_even_ra_inst2 = 0;
addr_even_rb_inst2 = 0;
addr_even_rc_inst2 = 0;
imm7_even_inst2   = 0;
imm10_even_inst2  = 0;
imm16_even_inst2  = 0;
imm18_even_inst2  = Inst_out[39:56];

```

```

addr_even_rt_inst2 = Inst_out[57:63];
end

else if((Inst_out[32:39] == 8'b0001_1100)
|| (Inst_out[32:39] == 8'b0001_0110)
|| (Inst_out[32:39] == 8'b0111_1110)
|| (Inst_out[32:39] == 8'b0111_1100)
|| (Inst_out[32:39] == 8'b0100_1110)
|| (Inst_out[32:39] == 8'b0100_1100)
|| (Inst_out[32:39] == 8'b0101_1110)
|| (Inst_out[32:39] == 8'b0000_0100)
|| (Inst_out[32:39] == 8'b0000_1100)
|| (Inst_out[32:39] == 8'b0100_0110)
|| (Inst_out[32:39] == 8'b0100_0100)
|| (Inst_out[32:39] == 8'b0111_0100)
|| (Inst_out[32:39] == 8'b0111_0101)) begin Even_inst2 = 1; Even_inst_nop = 0;
opcode_even_inst2 = Inst_out[32:39];
addr_even_ra_inst2 = Inst_out[50:56];
addr_even_rb_inst2 = 0;
addr_even_rc_inst2 = 0;
imm7_even_inst2 = 0;
imm10_even_inst2 = Inst_out[40:49];
imm16_even_inst2 = 0;
imm18_even_inst2 = 0;
addr_even_rt_inst2 = Inst_out[57:63];
end

else if((Inst_out[32:40] == 9'b0100_0000_1)
|| (Inst_out[32:40] == 9'b0100_0001_1)) begin Even_inst2 = 1; Even_inst_nop = 0;
opcode_even_inst2 = Inst_out[32:40];
addr_even_ra_inst2 = 0;
addr_even_rb_inst2 = 0;
addr_even_rc_inst2 = 0;
imm7_even_inst2 = 0;
imm10_even_inst2 = 0;
imm16_even_inst2 = Inst_out[41:56];
imm18_even_inst2 = 0;
addr_even_rt_inst2 = Inst_out[57:63];
end

else if((Inst_out[32:42] == 11'b0001_1000_000)
|| (Inst_out[32:42] == 11'b1101_0000_000)
|| (Inst_out[32:42] == 11'b0001_1000_001)
|| (Inst_out[32:42] == 11'b0101_1000_001)
|| (Inst_out[32:42] == 11'b0111_1000_000)
|| (Inst_out[32:42] == 11'b0111_1010_000)
|| (Inst_out[32:42] == 11'b0100_1010_000)
|| (Inst_out[32:42] == 11'b0101_1010_000)
|| (Inst_out[32:42] == 11'b0001_1001_001)
|| (Inst_out[32:42] == 11'b0000_1001_001)
|| (Inst_out[32:42] == 11'b0000_1000_001)
|| (Inst_out[32:42] == 11'b0101_1001_001)
|| (Inst_out[32:42] == 11'b0000_1000_000)
|| (Inst_out[32:42] == 11'b0110_1000_001)
|| (Inst_out[32:42] == 11'b0100_1000_001)
|| (Inst_out[32:42] == 11'b0000_1010_011)
|| (Inst_out[32:42] == 11'b0001_1010_011)
|| (Inst_out[32:42] == 11'b0000_1011_000)
|| (Inst_out[32:42] == 11'b0000_1011_001)
|| (Inst_out[32:42] == 11'b0000_1011_011)
|| (Inst_out[32:42] == 11'b0101_1000_100)
|| (Inst_out[32:42] == 11'b0101_1000_110)
|| (Inst_out[32:42] == 11'b0101_1000_101)

```

```

    || (Inst_out[32:42] == 11'b0100_1010_011)
    || (Inst_out[32:42] == 11'b0111_1000_100)
    || (Inst_out[32:42] == 11'b0111_1000_111)
    || (Inst_out[32:42] == 11'b0111_1001_100)) begin Even_inst2 = 1; Even_inst_nop = 0;
    opcode_even_inst2 = Inst_out[32:42];
    addr_even_ra_inst2 = Inst_out[50:56];
    addr_even_rb_inst2 = Inst_out[43:49];
    addr_even_rc_inst2 = 0;
    imm7_even_inst2 = 0;
    imm10_even_inst2 = 0;
    imm16_even_inst2 = 0;
    imm18_even_inst2 = 0;
    addr_even_rt_inst2 = Inst_out[57:63];
    end

    else if((Inst_out[32:42] == 11'b0101_0110_100)
    || (Inst_out[32:42] == 11'b0101_0110_110)
    || (Inst_out[32:42] == 11'b0101_0101_110)
    || (Inst_out[32:42] == 11'b0101_0100_110)) begin Even_inst2 = 1; Even_inst_nop = 0;
    opcode_even_inst2 = Inst_out[32:42];
    addr_even_ra_inst2 = Inst_out[50:56];
    addr_even_rb_inst2 = 0;
    addr_even_rc_inst2 = 0;
    imm7_even_inst2 = 0;
    imm10_even_inst2 = 0;
    imm16_even_inst2 = 0;
    imm18_even_inst2 = 0;
    addr_even_rt_inst2 = Inst_out[57:63];
    end

    else if((Inst_out[32:42] == 11'b0000_1111_011)) begin Even_inst2 = 1; Even_inst_nop = 0;
    opcode_even_inst2 = Inst_out[32:42];
    addr_even_ra_inst2 = Inst_out[50:56];
    addr_even_rb_inst2 = 0;
    addr_even_rc_inst2 = 0;
    imm7_even_inst2 = Inst_out[43:49];
    imm10_even_inst2 = 0;
    imm16_even_inst2 = 0;
    imm18_even_inst2 = 0;
    addr_even_rt_inst2 = Inst_out[57:63];
    end

    else if((Inst_out[32:42] == 11'b0100_0000_001)) begin
        Even_inst2 = 1;
    //else begin
        //Even_inst_nop = 1;
    opcode_even_inst2 = 11'b0100_0000_001;
    addr_even_ra_inst2 = 7'b0;
    addr_even_rb_inst2 = 7'b0;
    addr_even_rc_inst2 = 7'b0;
    imm7_even_inst2 = 7'b0;
    imm10_even_inst2 = 10'bx;
    imm16_even_inst2 = 16'bx;
    imm18_even_inst2 = 18'bx;
    addr_even_rt_inst2 = 7'b0;
    end

    // else begin
    // $display ("=====>>>> Else Reached No-Op Passed -
528");

    // Even_inst2 = 0;
    // opcode_even_inst2 = 11'b0100_0000_001;
    // addr_even_ra_inst2 = 0;

```

```

// addr_even_rb_inst2 = 0;
// addr_even_rc_inst2 = 0;
// imm7_even_inst2 = 0;
// imm10_even_inst2 = 0;
// imm16_even_inst2 = 0;
// imm18_even_inst2 = 0;
// addr_even_rt_inst2 = 0;
// end

// -----04
// Decode Inst_out[0:31] - Instruction 1 - Odd Pipe

if((Inst_out[32:40] == 9'b0010_0001_0)
|| (Inst_out[32:40] == 9'b0010_0000_0)
|| (Inst_out[32:40] == 9'b0011_0001_0)
|| (Inst_out[32:40] == 9'b0011_0011_0)
|| (Inst_out[32:40] == 9'b0011_0000_1)
|| (Inst_out[32:40] == 9'b0010_0000_1))begin Even_inst2 = 0; Odd_inst_nop = 0;
opcode_odd_inst2 =Inst_out[32:40];
addr_odd_ra_inst2 =0;
addr_odd_rb_inst2 =0;
addr_odd_rc_inst2 =0;
addr_odd_rt_inst2 =Inst_out[57:63];
imm7_odd_inst2 =0;
imm10_odd_inst2 =0;
imm16_odd_inst2 =Inst_out[41:56];
imm18_odd_inst2 =0;
end

else if ((Inst_out[32:40] == 9'b0011_0010_0)
|| (Inst_out[32:40] == 9'b0011_0000_0))begin Even_inst2 = 0; Odd_inst_nop = 0;
opcode_odd_inst2 =Inst_out[32:40];
addr_odd_ra_inst2 =0;
addr_odd_rb_inst2 =0;
addr_odd_rc_inst2 =0;
addr_odd_rt_inst2 =0;
imm7_odd_inst2 =0;
imm10_odd_inst2 =0;
imm16_odd_inst2 =Inst_out[41:56];
imm18_odd_inst2 =0;
end

else if ((Inst_out[32:42] == 11'b0011_0101_000)
|| (Inst_out[32:42] == 11'b0011_0101_001)) begin Even_inst2 = 0; Odd_inst_nop = 0;
opcode_odd_inst2 =Inst_out[32:42];
addr_odd_ra_inst2 =Inst_out[50:56];
addr_odd_rb_inst2 =0;
addr_odd_rc_inst2 =0;
addr_odd_rt_inst2 =0;
imm7_odd_inst2 =0;
imm10_odd_inst2 =0;
imm16_odd_inst2 =0;
imm18_odd_inst2 =0;
end

else if ((Inst_out[32:42] == 11'b0011_1111_100)
|| (Inst_out[32:42] == 11'b0011_1111_101)
|| (Inst_out[32:42] == 11'b0011_1111_111))begin Even_inst2 = 0; Odd_inst_nop = 0;
opcode_odd_inst2 =Inst_out[32:42];
addr_odd_ra_inst2 =Inst_out[50:56];
addr_odd_rb_inst2 =0;
addr_odd_rc_inst2 =0;
addr_odd_rt_inst2 =Inst_out[57:63];

```



```

    imm7_odd_inst2    =Inst_out[43:49];
    imm10_odd_inst2   =0;
    imm16_odd_inst2   =0;
    imm18_odd_inst2   =0;
end

    else if ((Inst_out[32:42] == 11'b0011_1011_100)
    || (Inst_out[32:42] == 11'b0011_1011_101)
    || (Inst_out[32:42] == 11'b0011_1011_111)
    || (Inst_out[32:42] == 11'b0011_1000_100)
    || (Inst_out[32:42] == 11'b0010_1000_100))begin Even_inst2 = 0; Odd_inst_nop = 0;
    opcode_odd_inst2  =Inst_out[32:42];
    addr_odd_ra_inst2 =Inst_out[50:56];
    addr_odd_rb_inst2 =Inst_out[43:49];
    addr_odd_rc_inst2 =0;
    addr_odd_rt_inst2 =Inst_out[57:63];
    imm7_odd_inst2    =0;
    imm10_odd_inst2   =0;
    imm16_odd_inst2   =0;
    imm18_odd_inst2   =0;
end

    else if ((Inst_out[32:42] == 11'b0100_0000_010)) begin Even_inst2 = 0;
    //else begin
        //Odd_inst_nop = 1;
        opcode_odd_inst2  =Inst_out[32:42];
        addr_odd_ra_inst2 =7'b0;
        addr_odd_rb_inst2 =7'b0;
        addr_odd_rc_inst2 =7'b0;
        addr_odd_rt_inst2 =7'b0;
        imm7_odd_inst2    =7'b0;
        imm10_odd_inst2   =10'bx;
        imm16_odd_inst2   =16'bx;
        imm18_odd_inst2   =18'bx;
    end

end

//-----
always_comb begin
    if((Even_inst1 == 1)&&(Even_inst2 == 1) &&
        (Inst_out[32:42] != 11'b0100_0000_010) && (Inst_out[0:10] != 11'b0100_0000_010) &&
        (Inst_out[32:42] != 11'b0100_0000_001) && (Inst_out[0:10] != 11'b0100_0000_001))
        begin
            Even_Structural_Hazard = 1; No_Structural_Hazard = 0; Odd_Structural_Hazard = 0;

            //if(decode_stall_Remove) decode_stall_1 = 0;
            //$display("Inside Structural Hazard - Even Case");
        end

    else if((Even_inst1 == 0)&&(Even_inst2 == 0) &&
        (Inst_out[32:42] != 11'b0100_0000_010) && (Inst_out[0:10] != 11'b0100_0000_010) &&
        (Inst_out[32:42] != 11'b0100_0000_001) && (Inst_out[0:10] != 11'b0100_0000_001) )
        begin
            Odd_Structural_Hazard = 1; No_Structural_Hazard = 0; Even_Structural_Hazard = 0;
            // decode_stall_1 = 1;
            // if(decode_stall_Remove) decode_stall_1 = 0;
        end

    else begin No_Structural_Hazard = 1; Even_Structural_Hazard = 0; Odd_Structural_Hazard = 0; end
end

always_comb begin
    if(Even_Structural_Hazard == 1) begin

```

```

opcode_even_tmp  = opcode_even_inst1;
addr_even_ra_tmp = addr_even_ra_inst1;
addr_even_rb_tmp = addr_even_rb_inst1;
addr_even_rc_tmp = addr_even_rc_inst1;
imm7_even_tmp    = imm7_even_inst1;
imm10_even_tmp   = imm10_even_inst1;
imm16_even_tmp   = imm16_even_inst1;
imm18_even_tmp   = imm18_even_inst1;
addr_even_rt_tmp = addr_even_rt_inst1;
//-----
opcode_odd_tmp   = 11'b0100_0000_001;
addr_odd_ra_tmp  = 7'b0;
addr_odd_rb_tmp  = 7'b0;
addr_odd_rc_tmp  = 7'b0;
addr_odd_rt_tmp  = 7'b0;
imm7_odd_tmp     = 7'b0;
imm10_odd_tmp    = 10'b0;
imm16_odd_tmp    = 16'b0;
imm18_odd_tmp    = 18'b0;
end

```

```

if(Prev_Even_Hazard == 1) begin
opcode_even_tmp  = opcode_even_inst2;
addr_even_ra_tmp = addr_even_ra_inst2;
addr_even_rb_tmp = addr_even_rb_inst2;
addr_even_rc_tmp = addr_even_rc_inst2;
imm7_even_tmp    = imm7_even_inst2;
imm10_even_tmp   = imm10_even_inst2;
imm16_even_tmp   = imm16_even_inst2;
imm18_even_tmp   = imm18_even_inst2;
addr_even_rt_tmp = addr_even_rt_inst2;
//-----
opcode_odd_tmp   = 11'b0100_0000_001;
addr_odd_ra_tmp  = 7'b0;
addr_odd_rb_tmp  = 7'b0;
addr_odd_rc_tmp  = 7'b0;
addr_odd_rt_tmp  = 7'b0;
imm7_odd_tmp     = 7'b0;
imm10_odd_tmp    = 10'b0;
imm16_odd_tmp    = 16'b0;
imm18_odd_tmp    = 18'b0;
end

```

```

end

if(Odd_Structural_Hazard == 1) begin
opcode_odd_tmp  = opcode_odd_inst1;
addr_odd_ra_tmp = addr_odd_ra_inst1;
addr_odd_rb_tmp = addr_odd_rb_inst1;
addr_odd_rc_tmp = addr_odd_rc_inst1;
imm7_odd_tmp    = imm7_odd_inst1;
imm10_odd_tmp   = imm10_odd_inst1;
imm16_odd_tmp   = imm16_odd_inst1;
imm18_odd_tmp   = imm18_odd_inst1;
addr_odd_rt_tmp = addr_odd_rt_inst1;
//-----
opcode_even_tmp = 11'b0100_0000_001;
addr_even_ra_tmp = 7'b0;
addr_even_rb_tmp = 7'b0;
addr_even_rc_tmp = 7'b0;
addr_even_rt_tmp = 7'b0;
imm7_even_tmp    = 7'b0;
imm10_even_tmp   = 10'b0;
imm16_even_tmp   = 16'b0;

```

```

imm18_even_tmp  =18'b0;
end

if(Prev_Odd_Hazard == 1) begin

opcode_odd_tmp   = opcode_odd_inst2;
addr_odd_ra_tmp  = addr_odd_ra_inst2;
addr_odd_rb_tmp  = addr_odd_rb_inst2;
addr_odd_rc_tmp  = addr_odd_rc_inst2;
imm7_odd_tmp     = imm7_odd_inst2;
imm10_odd_tmp    = imm10_odd_inst2;
imm16_odd_tmp    = imm16_odd_inst2;
imm18_odd_tmp    = imm18_odd_inst2;
addr_odd_rt_tmp  = addr_odd_rt_inst2;
//-----
opcode_even_tmp  = 11'b0100_0000_001;
addr_even_ra_tmp =7'b0;
addr_even_rb_tmp =7'b0;
addr_even_rc_tmp =7'b0;
addr_even_rt_tmp =7'b0;
imm7_even_tmp    =7'b0;
imm10_even_tmp   =10'b0;
imm16_even_tmp   =16'b0;
imm18_even_tmp   =18'b0;

end

if(No_Structural_Hazard) begin
if((Even_inst1 == 1)&&(Even_inst2 == 0))begin
opcode_even_tmp  = opcode_even_inst1;
addr_even_ra_tmp = addr_even_ra_inst1;
addr_even_rb_tmp = addr_even_rb_inst1;
addr_even_rc_tmp = addr_even_rc_inst1;
imm7_even_tmp    = imm7_even_inst1;
imm10_even_tmp   = imm10_even_inst1;
imm16_even_tmp   = imm16_even_inst1;
imm18_even_tmp   = imm18_even_inst1;
addr_even_rt_tmp = addr_even_rt_inst1;
//-----
opcode_odd_tmp   = opcode_odd_inst2;
addr_odd_ra_tmp  = addr_odd_ra_inst2;
addr_odd_rb_tmp  = addr_odd_rb_inst2;
addr_odd_rc_tmp  = addr_odd_rc_inst2;
imm7_odd_tmp     = imm7_odd_inst2;
imm10_odd_tmp    = imm10_odd_inst2;
imm16_odd_tmp    = imm16_odd_inst2;
imm18_odd_tmp    = imm18_odd_inst2;
addr_odd_rt_tmp  = addr_odd_rt_inst2;
end

else if((Even_inst1 == 0)&&(Even_inst2 == 1))begin
opcode_even_tmp  = opcode_even_inst2;
addr_even_ra_tmp = addr_even_ra_inst2;
addr_even_rb_tmp = addr_even_rb_inst2;
addr_even_rc_tmp = addr_even_rc_inst2;
imm7_even_tmp    = imm7_even_inst2;
imm10_even_tmp   = imm10_even_inst2;
imm16_even_tmp   = imm16_even_inst2;
imm18_even_tmp   = imm18_even_inst2;
addr_even_rt_tmp = addr_even_rt_inst2;
//-----
opcode_odd_tmp   = opcode_odd_inst1;
addr_odd_ra_tmp  = addr_odd_ra_inst1;

```

```

addr_odd_rb_tmp = addr_odd_rb_inst1;
addr_odd_rc_tmp = addr_odd_rc_inst1;
imm7_odd_tmp    = imm7_odd_inst1;
imm10_odd_tmp   = imm10_odd_inst1;
imm16_odd_tmp   = imm16_odd_inst1;
imm18_odd_tmp   = imm18_odd_inst1;
addr_odd_rt_tmp = addr_odd_rt_inst1;
end
end end

always_ff @(posedge clk) begin
    if (reset==0) //$display ($time-10, " Dual Instruction: %b_%bl @ Decode Stage - Fetched" , Inst_out[0:31], Inst_out[32:63]);

    if(Even_Structural_Hazard == 1) begin
        decode_stall_1 = 1; //$display($time, " Even_Structural_Hazard Resolution in Progress");
        Prev_Even_Hazard <= 1; end

    if(Prev_Even_Hazard == 1) begin
        decode_stall_1 = 0;
        Prev_Even_Hazard <= 0; end

    if(Odd_Structural_Hazard == 1) begin
        Prev_Odd_Hazard <= 1; //$display($time, " Odd_Structural_Hazard Resolution in Progress");
        decode_stall_1 = 1; end

    if(Prev_Odd_Hazard == 1) begin
        Prev_Odd_Hazard <= 0;
        decode_stall_1 = 0; end

    if(No_Structural_Hazard) begin //$display($time, " No_Structural_Hazard");
        if((Even_inst1 == 1)&&(Even_inst2 == 0))begin
            decode_stall_1 = 0;end

        else if((Even_inst1 == 0)&&(Even_inst2 == 1))begin
            decode_stall_1 = 0; end end
    end

    // -----
    always_ff @(posedge clk) begin

        if(Dependency_Stall) begin
            //$display("=====Dependency Stall is High");

            opcode_even_bck <= opcode_even_tmp;
            addr_even_ra_bck <= addr_even_ra_tmp;
            addr_even_rb_bck <= addr_even_rb_tmp;
            addr_even_rc_bck <= addr_even_rc_tmp;
            imm7_even_bck   <= imm7_even_tmp;
            imm10_even_bck  <= imm10_even_tmp;
            imm16_even_bck  <= imm16_even_tmp;
            imm18_even_bck  <= imm18_even_tmp;
            addr_even_rt_bck <= addr_even_rt_tmp;
            //-----
            opcode_odd_bck  <= opcode_odd_tmp;
            addr_odd_ra_bck  <= addr_odd_ra_tmp;
            addr_odd_rb_bck  <= addr_odd_rb_tmp;
            addr_odd_rc_bck  <= addr_odd_rc_tmp;
            imm7_odd_bck     <= imm7_odd_tmp;
            imm10_odd_bck    <= imm10_odd_tmp;
            imm16_odd_bck    <= imm16_odd_tmp;
            imm18_odd_bck    <= imm18_odd_tmp;
            addr_odd_rt_bck  <= addr_odd_rt_tmp;// end
        end
    end
end

```

```

Prev_stall =1 ;

end

else if(Dependency_Stall==0 && Prev_stall==1)begin

    // decode_stall_1 = 1;
    opcode_even  <= opcode_even_bck;
    addr_even_ra <= addr_even_ra_bck;
    addr_even_rb <= addr_even_rb_bck;
    addr_even_rc <= addr_even_rc_bck;
    imm7_even    <= imm7_even_bck;
    imm10_even   <= imm10_even_bck;
    imm16_even   <= imm16_even_bck;
    imm18_even   <= imm18_even_bck;
    addr_even_rt <= addr_even_rt_bck;
    //-----
    opcode_odd   <= opcode_odd_bck;
    addr_odd_ra  <= addr_odd_ra_bck;
    addr_odd_rb  <= addr_odd_rb_bck;
    addr_odd_rc  <= addr_odd_rc_bck;
    imm7_odd     <= imm7_odd_bck;
    imm10_odd    <= imm10_odd_bck;
    imm16_odd    <= imm16_odd_bck;
    imm18_odd    <= imm18_odd_bck;
    addr_odd_rt  <= addr_odd_rt_bck;
    Prev_stall = 1;

end

else begin
    // decode_stall_1 = 0;
    opcode_even  <= opcode_even_tmp;
    addr_even_ra <= addr_even_ra_tmp;
    addr_even_rb <= addr_even_rb_tmp;
    addr_even_rc <= addr_even_rc_tmp;
    imm7_even    <= imm7_even_tmp;
    imm10_even   <= imm10_even_tmp;
    imm16_even   <= imm16_even_tmp;
    imm18_even   <= imm18_even_tmp;
    addr_even_rt <= addr_even_rt_tmp;
    //-----
    opcode_odd   <= opcode_odd_tmp;
    addr_odd_ra  <= addr_odd_ra_tmp;
    addr_odd_rb  <= addr_odd_rb_tmp;
    addr_odd_rc  <= addr_odd_rc_tmp;
    imm7_odd     <= imm7_odd_tmp;
    imm10_odd    <= imm10_odd_tmp;
    imm16_odd    <= imm16_odd_tmp;
    imm18_odd    <= imm18_odd_tmp;
    addr_odd_rt  <= addr_odd_rt_tmp; end

    // else if (Prev_stall == 1 && Dependency_Stall==0) begin

```

```

// $display ("opcode_even: %b, addr_even_ra: %b, addr_even_rb: %b, addr_even_rc: %b, imm7_even: %b, imm10_even: %b,
imm16_even: %b, imm18_even: %b, addr_even_rt: %b", opcode_even, addr_even_ra, addr_even_rb, addr_even_rc, imm7_even,
imm10_even, imm16_even, imm18_even, addr_even_rt);
// $display ("opcode_odd: %b, addr_odd_ra: %b, addr_odd_rb: %b, addr_odd_rc: %b, addr_odd_rt: %b, imm7_odd: %b,
imm10_odd: %b, imm16_odd: %b, imm18_odd: %b", opcode_odd, addr_odd_ra, addr_odd_rb, addr_odd_rc, addr_odd_rt,
imm7_odd, imm10_odd, imm16_odd, imm18_odd);

end

// always_ff @(posedge clk) begin
// if(reset==0) begin
// $display ($time, " == > Opcode_even: %b | @ Decode Stage - Issue" , opcode_even);
// $display ($time, " == > Opcode_odd: %b | @ Decode Stage - Issue", opcode_odd); end
// end
endmodule // Decode

```

Script: Dependency

```

//=====Dependency=====
=====

```

```

module Dependency (clk, reset,addr_even_ra, addr_even_rb, addr_even_rc, opcode_even, imm10_even, imm16_even,
imm18_even, imm7_even, addr_even_rt,
addr_odd_ra, addr_odd_rb, addr_odd_rc, opcode_odd, imm10_odd, imm16_odd, imm18_odd, imm7_odd, addr_odd_rt,
flush,opcode_even_depend, imm7_even_depend, imm10_even_depend, imm16_even_depend, imm18_even_depend,
addr_even_rt_depend, addr_even_ra_depend, addr_even_rb_depend, addr_even_rc_depend ,
opcode_odd_depend,imm7_odd_depend,imm10_odd_depend,imm16_odd_depend,imm18_odd_depend,addr_odd_rt_depend,addr_o
dd_ra_depend, addr_odd_rb_depend, addr_odd_rc_depend, Dependency_stall, Rt_Temp_even, Rt_Temp1_even, Rt_Temp2_even,
Rt_Temp3_even, Rt_Temp4_even, Rt_Temp5_even, Rt_Temp6_even, Rt_Temp_odd, Rt_Temp1_odd, Rt_Temp2_odd,
Rt_Temp3_odd, Rt_Temp4_odd, Rt_Temp5_odd, Rt_Temp6_odd);

```

```

input clk, reset, flush;
input [0:6] addr_even_ra, addr_even_rb, addr_even_rc;
input [0:6] addr_odd_ra, addr_odd_rb, addr_odd_rc;
input [0:10] opcode_even;
input signed [0:6] imm7_even;
input signed [0:9] imm10_even;
input signed [0:15] imm16_even;
input signed [0:17] imm18_even;
input [0:6] addr_even_rt;
input [0:142] Rt_Temp_even, Rt_Temp1_even, Rt_Temp2_even, Rt_Temp3_even, Rt_Temp4_even, Rt_Temp5_even,
Rt_Temp6_even;

```

```

output logic [0:10] opcode_even_depend;
output logic signed [0:6] imm7_even_depend;
output logic signed [0:9] imm10_even_depend;
output logic signed [0:15] imm16_even_depend;
output logic signed [0:17] imm18_even_depend;
output logic [0:6] addr_even_rt_depend;
output logic [0:6] addr_even_ra_depend, addr_even_rb_depend, addr_even_rc_depend;

```

```

input [0:10] opcode_odd;
input [0:6] imm7_odd;
input [0:9] imm10_odd;
input [0:15] imm16_odd;
input [0:17] imm18_odd;
input [0:6] addr_odd_rt;
input [0:175] Rt_Temp_odd, Rt_Temp1_odd, Rt_Temp2_odd, Rt_Temp3_odd, Rt_Temp4_odd, Rt_Temp5_odd, Rt_Temp6_odd;

```

```

output logic [0:10] opcode_odd_depend;
output logic signed [0:6] imm7_odd_depend;
output logic signed [0:9] imm10_odd_depend;
output logic signed [0:15] imm16_odd_depend;
output logic signed [0:17] imm18_odd_depend;
output logic signed [0:6] addr_odd_rt_depend;
output logic [0:6] addr_odd_ra_depend, addr_odd_rb_depend, addr_odd_rc_depend;
logic [0:6] addr_odd_rt_depend2;
logic signed [0:15] imm16_odd_depend2;

logic [0:10] opcode_odd_depend_bck;
logic signed [0:6] imm7_odd_depend_bck;
logic signed [0:9] imm10_odd_depend_bck;
logic signed [0:15] imm16_odd_depend_bck;
logic signed [0:17] imm18_odd_depend_bck;
logic signed [0:6] addr_odd_rt_depend_bck;
logic [0:6] addr_odd_ra_depend_bck, addr_odd_rb_depend_bck, addr_odd_rc_depend_bck;

logic [0:10] opcode_even_depend_bck;
logic signed [0:6] imm7_even_depend_bck;
logic signed [0:9] imm10_even_depend_bck;
logic signed [0:15] imm16_even_depend_bck;
logic signed [0:17] imm18_even_depend_bck;
logic [0:6] addr_even_rt_depend_bck;
logic [0:6] addr_even_ra_depend_bck, addr_even_rb_depend_bck, addr_even_rc_depend_bck;

logic [0:10] opcode_odd_depend2, opcode_even_depend2 ;

output logic Dependency_stall;

logic [0:10] opcode_even_depend_temp, opcode_odd_depend_temp;
logic [0:6] addr_even_rt_depend_temp, addr_odd_rt_depend_temp;
logic [0:2] opcode_even_depend_lat, opcode_odd_depend_lat;
logic [0:2] count, counter;
logic Prev_Stall;

assign Dependency_stall = (counter > 0);

//always_comb begin
//  always_ff @(posedge clk) begin
//    if (Dependency_stall) begin
//      opcode_even_depend = 11'b0100_0000_001;
//      opcode_odd_depend = 11'b0100_0000_010; end
//
//    else begin
//      opcode_even_depend = opcode_even_depend2;
//      opcode_odd_depend = opcode_odd_depend2;
//      imm16_odd_depend = imm16_odd_depend2;
//      addr_odd_rt_depend = addr_odd_rt_depend2;end
//  end

always_ff @(posedge clk) begin
  //$display("Counter:%d, Count:%d ", counter, count);

  Prev_Stall <= Dependency_stall;

  if(counter > 0) begin // count == 0 &&
    counter = counter - 1; $display("-----| | | | | | | | | |----->>> Data Hazard");
    //Dependency_stall = 1;
    // opcode_even_depend = 11'b0100_0000_001;
    // opcode_odd_depend = 11'b0100_0000_010;

```

```

end

else if(count == 1) counter <= 1;
else if (count == 2 ) counter <= 2;
else if (count == 3 ) counter <= 3;
else if (count == 4 ) counter <= 4;
else if (count == 5 ) counter <= 5;
else if (count == 6 ) counter <= 6;

else counter <= 0;

if (Dependency_stall) begin
opcode_even_depend = 11'b0100_0000_001;
opcode_odd_depend = 11'b0100_0000_010; end

else if (Prev_Stall == 1 && Dependency_stall == 0) begin
addr_even_ra_depend <= addr_even_ra_depend_bck;
addr_even_rb_depend <= addr_even_rb_depend_bck;
addr_even_rc_depend <= addr_even_rc_depend_bck;
opcode_even_depend <= opcode_even_depend_bck;
addr_even_rt_depend <= addr_even_rt_depend_bck;
imm7_even_depend <= imm7_even_depend_bck;
imm10_even_depend <= imm10_even_depend_bck;
imm16_even_depend <= imm16_even_depend_bck;
imm18_even_depend <= imm18_even_depend_bck;

addr_odd_ra_depend <= addr_odd_ra_depend_bck;
addr_odd_rb_depend <= addr_odd_rb_depend_bck;
addr_odd_rc_depend <= addr_odd_rc_depend_bck;
opcode_odd_depend <= opcode_odd_depend_bck;
addr_odd_rt_depend <= addr_odd_rt_depend_bck;
imm7_odd_depend <= imm7_odd_depend_bck;
imm10_odd_depend <= imm10_odd_depend_bck;
imm16_odd_depend <= imm16_odd_depend_bck;
imm18_odd_depend <= imm18_odd_depend_bck;

end

else if(count == 0 && counter == 0) begin
//$display("----->>> No Data Hazard, Counter:%d, Count:%d ", counter,
count);
//Dependency_stall = 0;
addr_even_ra_depend<=addr_even_ra;
addr_even_rb_depend<=addr_even_rb;
addr_even_rc_depend<=addr_even_rc;
opcode_even_depend <= opcode_even; addr_even_rt_depend <= addr_even_rt;
imm7_even_depend <= imm7_even; imm10_even_depend <= imm10_even;
imm16_even_depend <= imm16_even; imm18_even_depend <= imm18_even;

addr_odd_ra_depend<=addr_odd_ra;
addr_odd_rb_depend<=addr_odd_rb;
addr_odd_rc_depend<=addr_odd_rc;
opcode_odd_depend <= opcode_odd; addr_odd_rt_depend <= addr_odd_rt;
imm7_odd_depend <= imm7_odd; imm10_odd_depend <= imm10_odd;
imm16_odd_depend <= imm16_odd; imm18_odd_depend <= imm18_odd;

// ----- For Latency Calculation @ Reg File

opcode_even_depend_temp <= opcode_even;
addr_even_rt_depend_temp <= addr_even_rt;
opcode_odd_depend_temp <= opcode_odd;
addr_odd_rt_depend_temp <= addr_odd_rt;

```



```

end

// $display ("opcode_even: %b | @ Dependency Stge" , opcode_even_depend);
// $display ("opcode_odd: %b | @ Dependency Stage", opcode_odd_depend);
end

always_ff @(posedge clk) begin

if(Dependency_stall) begin
    addr_even_ra_depend_bck<=addr_even_ra;
    addr_even_rb_depend_bck<=addr_even_rb;
    addr_even_rc_depend_bck<=addr_even_rc;
    opcode_even_depend_bck <= opcode_even;
    addr_even_rt_depend_bck <= addr_even_rt;
    imm7_even_depend_bck <= imm7_even;
    imm10_even_depend_bck <= imm10_even;
    imm16_even_depend_bck <= imm16_even;
    imm18_even_depend_bck <= imm18_even;

    addr_odd_ra_depend_bck<=addr_odd_ra;
    addr_odd_rb_depend_bck<=addr_odd_rb;
    addr_odd_rc_depend_bck<=addr_odd_rc;
    opcode_odd_depend_bck <= opcode_odd;
    addr_odd_rt_depend_bck <= addr_odd_rt;
    imm7_odd_depend_bck <= imm7_odd;
    imm10_odd_depend_bck <= imm10_odd;
    imm16_odd_depend_bck <= imm16_odd;
    imm18_odd_depend_bck <= imm18_odd;
end end

always_comb begin
if (opcode_even_depend_temp == 7'b0100_001||opcode_even_depend_temp == 8'b0001_1100||opcode_even_depend_temp ==
8'b0001_0110||opcode_even_depend_temp == 8'b0111_1110||
opcode_even_depend_temp == 8'b0111_1100||opcode_even_depend_temp == 8'b0100_1110||opcode_even_depend_temp ==
8'b0100_1100||opcode_even_depend_temp == 8'b0101_1110||
opcode_even_depend_temp == 8'b0000_0100||opcode_even_depend_temp == 8'b0000_1100||opcode_even_depend_temp ==
8'b0100_0110||opcode_even_depend_temp == 8'b0100_0100||
opcode_even_depend_temp == 9'b0100_0000_1||opcode_even_depend_temp == 9'b0100_0001_1||opcode_even_depend_temp
== 11'b0001_1000_000||opcode_even_depend_temp == 11'b1101_0000_000||
opcode_even_depend_temp == 11'b0001_1000_001||opcode_even_depend_temp ==
11'b0101_1000_001||opcode_even_depend_temp == 11'b0111_1000_000||opcode_even_depend_temp ==
11'b0111_1010_000||
opcode_even_depend_temp == 11'b0100_1010_000||opcode_even_depend_temp ==
11'b0101_1010_000||opcode_even_depend_temp == 11'b0001_1001_001||opcode_even_depend_temp ==
11'b0000_1001_001||
opcode_even_depend_temp == 11'b0000_1000_001||opcode_even_depend_temp ==
11'b0101_1001_001||opcode_even_depend_temp == 11'b0000_1000_000||opcode_even_depend_temp ==
11'b0110_1000_001||
opcode_even_depend_temp == 11'b0100_1000_001||opcode_even_depend_temp ==
11'b0101_0110_110||opcode_even_depend_temp == 11'b0101_0101_110||opcode_even_depend_temp ==
11'b0101_0100_110)
opcode_even_depend_lat = 3'd2;
else if(opcode_even_depend_temp == 11'b0000_1111_011|| opcode_even_depend_temp == 11'b0000_1011_000||
opcode_even_depend_temp == 11'b0000_1011_001||opcode_even_depend_temp == 11'b0000_1011_011||
opcode_even_depend_temp == 11'b0101_0110_100||opcode_even_depend_temp ==
11'b0000_1010_011||opcode_even_depend_temp == 11'b0001_1010_011||opcode_even_depend_temp ==
11'b0100_1010_011)
opcode_even_depend_lat = 3'd4;

```

```

else if(opcode_even_depend_temp == 4'b1110_llopcod_opcode_even_depend_temp == 4'b1111_llopcod_opcode_even_depend_temp ==
11'b0101_1000_100llopcod_opcode_even_depend_temp == 11'b0101_1000_110llopcod_opcode_even_depend_temp ==
11'b0101_1000_101)
opcode_even_depend_lat = 3'd6;
else if (opcode_even_depend_temp == 4'b1100_ll_opcode_even_depend_temp == 8'b0111_0100ll_opcode_even_depend_temp
== 8'b0111_0101llopcod_opcode_even_depend_temp == 11'b0111_1000_100ll_opcode_even_depend_temp ==
11'b0111_1000_111llopcod_opcode_even_depend_temp == 11'b0111_1001_100)
opcode_even_depend_lat = 3'd7;
else opcode_even_depend_lat = 3'd0;

if(opcode_odd_depend_temp == 11'b0011_1011_101llopcod_opcode_odd_depend_temp ==
11'b0011_1011_111llopcod_opcode_odd_depend_temp == 9'b0011_0001_0llopcod_opcode_odd_depend_temp == 9'b0011_0011_0ll
opcode_odd_depend_temp == 9'b0011_0010_0llopcod_opcode_odd_depend_temp == 9'b0011_0000_0llopcod_opcode_odd_depend_temp
== 11'b0011_0101_000llopcod_opcode_odd_depend_temp == 11'b0011_0101_001ll
opcode_odd_depend_temp == 9'b0010_0001_0llopcod_opcode_odd_depend_temp == 9'b0010_0000_0)
opcode_odd_depend_lat = 3'd4;
else if (opcode_odd_depend_temp == 9'b0011_0000_1llopcod_opcode_odd_depend_temp ==
9'b0010_0000_1llopcod_opcode_odd_depend_temp == 11'b0011_1000_100llopcod_opcode_odd_depend_temp == 11'b0010_1000_100)
opcode_odd_depend_lat = 3'd6;
else opcode_odd_depend_lat = 3'd0;

end // always_comb

always_comb begin

//$display("===== > addr_even_rb: %d | Rt_Temp_odd:%d",addr_even_rb, Rt_Temp_odd [7:14]);

if (((addr_even_ra!=7'b0) && (addr_even_ra == addr_even_rt_depend_temp)) ||
((addr_even_rb!=7'b0) && (addr_even_rb == addr_even_rt_depend_temp)) ||
((addr_even_rc!=7'b0) && (addr_even_rc == addr_even_rt_depend_temp)) ||
((addr_odd_ra!=7'b0) && (addr_odd_ra == addr_even_rt_depend_temp)) ||
((addr_odd_rb!=7'b0) && (addr_odd_rb == addr_even_rt_depend_temp)) ||
((addr_odd_rc!=7'b0) && (addr_odd_rc == addr_even_rt_depend_temp))) begin
count=opcode_even_depend_lat-1; $display("1st EVEN If - Data Hazard");
end

else if (((addr_even_ra!=7'b0) && (addr_even_ra == Rt_Temp_even [7:14])) ||
((addr_even_rb!=7'b0) && (addr_even_rb == Rt_Temp_even [7:14])) ||
((addr_even_rc!=7'b0) && (addr_even_rc == Rt_Temp_even [7:14])) ||
((addr_odd_ra!=7'b0) && (addr_odd_ra == Rt_Temp_even [7:14])) ||
((addr_odd_rb!=7'b0) && (addr_odd_rb == Rt_Temp_even [7:14])) ||
((addr_odd_rc!=7'b0) && (addr_odd_rc == Rt_Temp_even [7:14]))) begin
count=Rt_Temp_even[3:5]-2; $display("2st EVEN If - Data Hazard");
//count=0; $display("2st EVEN If - Data Hazard");
end

else if (((addr_even_ra!=7'bx) && (addr_even_ra == Rt_Temp1_even [7:14])) ||
((addr_even_rb!=7'bx) && (addr_even_rb == Rt_Temp1_even [7:14])) ||
((addr_even_rc!=7'bx) && (addr_even_rc == Rt_Temp1_even [7:14])) ||
((addr_odd_ra!=7'bx) && (addr_odd_ra == Rt_Temp1_even [7:14])) ||
((addr_odd_rb!=7'bx) && (addr_odd_rb == Rt_Temp1_even [7:14])) ||
((addr_odd_rc!=7'bx) && (addr_odd_rc == Rt_Temp1_even [7:14]))) begin
count=Rt_Temp1_even[3:5]-3; $display("3nd EVEN If - Data Hazard"); end

else if (((addr_even_ra!=7'bx) && (addr_even_ra == Rt_Temp2_even [7:14])) ||
((addr_even_rb!=7'bx) && (addr_even_rb == Rt_Temp2_even [7:14])) ||
((addr_even_rc!=7'bx) && (addr_even_rc == Rt_Temp2_even [7:14])) ||
((addr_odd_ra!=7'bx) && (addr_odd_ra == Rt_Temp2_even [7:14])) ||
((addr_odd_rb!=7'bx) && (addr_odd_rb == Rt_Temp2_even [7:14])) ||
((addr_odd_rc!=7'bx) && (addr_odd_rc == Rt_Temp2_even [7:14]))) begin

```

```

count=Rt_Temp2_even[3:5]-4; $display("4rd EVEN If - Data Hazard"); end

else if (((addr_even_ra!=7'bx) && (addr_even_ra == Rt_Temp3_even [7:14])) ||
((addr_even_rb!=7'bx) && (addr_even_rb == Rt_Temp3_even [7:14])) ||
((addr_even_rc!=7'bx) && (addr_even_rc == Rt_Temp3_even [7:14])) ||
((addr_odd_ra!=7'bx) && (addr_odd_ra == Rt_Temp3_even [7:14])) ||
((addr_odd_rb!=7'bx) && (addr_odd_rb == Rt_Temp3_even [7:14])) ||
((addr_odd_rc!=7'bx) && (addr_odd_rc == Rt_Temp3_even [7:14]))) begin
count=Rt_Temp3_even[3:5]-5; $display("5th EVEN If - Data Hazard"); end

else if (((addr_even_ra!=7'bx) && (addr_even_ra == Rt_Temp4_even [7:14])) ||
((addr_even_rb!=7'bx) && (addr_even_rb == Rt_Temp4_even [7:14])) ||
((addr_even_rc!=7'bx) && (addr_even_rc == Rt_Temp4_even [7:14])) ||
((addr_odd_ra!=7'bx) && (addr_odd_ra == Rt_Temp4_even [7:14])) ||
((addr_odd_rb!=7'bx) && (addr_odd_rb == Rt_Temp4_even [7:14])) ||
((addr_odd_rc!=7'bx) && (addr_odd_rc == Rt_Temp4_even [7:14]))) begin
count=Rt_Temp4_even[3:5]-6; $display("6th EVEN If - Data Hazard"); end

else if (((addr_even_ra!=7'bx) && (addr_even_ra == Rt_Temp5_even [7:14])) ||
((addr_even_rb!=7'bx) && (addr_even_rb == Rt_Temp5_even [7:14])) ||
((addr_even_rc!=7'bx) && (addr_even_rc == Rt_Temp5_even [7:14])) ||
((addr_odd_ra!=7'bx) && (addr_odd_ra == Rt_Temp5_even [7:14])) ||
((addr_odd_rb!=7'bx) && (addr_odd_rb == Rt_Temp5_even [7:14])) ||
((addr_odd_rc!=7'bx) && (addr_odd_rc == Rt_Temp5_even [7:14]))) begin
count=Rt_Temp5_even[3:5]-7; $display("7th EVEN If - Data Hazard"); end

// else if (((addr_even_ra!=7'bx) && (addr_even_ra == Rt_Temp6_even [7:14])) ||
// ((addr_even_rb!=7'bx) && (addr_even_rb == Rt_Temp6_even [7:14])) ||
// ((addr_even_rc!=7'bx) && (addr_even_rc == Rt_Temp6_even [7:14])) ||
// ((addr_odd_ra!=7'bx) && (addr_odd_ra == Rt_Temp6_even [7:14])) ||
// ((addr_odd_rb!=7'bx) && (addr_odd_rb == Rt_Temp6_even [7:14])) ||
// ((addr_odd_rc!=7'bx) && (addr_odd_rc == Rt_Temp6_even [7:14]))) begin
// count=Rt_Temp6_even[3:5]-7; $display("8th EVEN If - Data Hazard"); end

// -----

else if (((addr_even_ra!=7'b0) && (addr_even_ra == addr_odd_rt_depend_temp)) ||
((addr_even_rb!=7'b0) && (addr_even_rb == addr_odd_rt_depend_temp)) ||
((addr_even_rc!=7'b0) && (addr_even_rc == addr_odd_rt_depend_temp)) ||
((addr_odd_ra!=7'b0) && (addr_odd_ra == addr_odd_rt_depend_temp)) ||
((addr_odd_rb!=7'b0) && (addr_odd_rb == addr_odd_rt_depend_temp)) ||
((addr_odd_rc!=7'b0) && (addr_odd_rc == addr_odd_rt_depend_temp))) begin
count=opcode_odd_depend_lat-1; $display("1st ODD If - Data Hazard" );
end

else if (((addr_even_ra!=7'bx) && (addr_even_ra == Rt_Temp_odd [7:14])) ||
((addr_even_rb!=7'bx) && (addr_even_rb == Rt_Temp_odd [7:14])) ||
((addr_even_rc!=7'bx) && (addr_even_rc == Rt_Temp_odd [7:14])) ||
((addr_odd_ra!=7'bx) && (addr_odd_ra == Rt_Temp_odd [7:14])) ||
((addr_odd_rb!=7'bx) && (addr_odd_rb == Rt_Temp_odd [7:14])) ||
((addr_odd_rc!=7'bx) && (addr_odd_rc == Rt_Temp_odd [7:14]))) begin
count=Rt_Temp_odd[3:5]-2; $display("2st ODD If - Data Hazard"); end

else if (((addr_even_ra!=7'bx) && (addr_even_ra == Rt_Temp1_odd [7:14])) ||
((addr_even_rb!=7'bx) && (addr_even_rb == Rt_Temp1_odd [7:14])) ||
((addr_even_rc!=7'bx) && (addr_even_rc == Rt_Temp1_odd [7:14])) ||
((addr_odd_ra!=7'bx) && (addr_odd_ra == Rt_Temp1_odd [7:14])) ||
((addr_odd_rb!=7'bx) && (addr_odd_rb == Rt_Temp1_odd [7:14])) ||
((addr_odd_rc!=7'bx) && (addr_odd_rc == Rt_Temp1_odd [7:14]))) begin
count=Rt_Temp1_odd[3:5]-3; $display("3nd ODD If - Data Hazard"); end

else if (((addr_even_ra!=7'bx) && (addr_even_ra == Rt_Temp2_odd [7:14])) ||
((addr_even_rb!=7'bx) && (addr_even_rb == Rt_Temp2_odd [7:14])) ||

```

```

((addr_even_rc!=7'bx) && (addr_even_rc == Rt_Temp2_odd [7:14])) ||
((addr_odd_ra!=7'bx) && (addr_odd_ra == Rt_Temp2_odd [7:14])) ||
((addr_odd_rb!=7'bx) && (addr_odd_rb == Rt_Temp2_odd [7:14])) ||
((addr_odd_rc!=7'bx) && (addr_odd_rc == Rt_Temp2_odd [7:14])))) begin
count=Rt_Temp2_odd[3:5]-4; $display("4rd ODD If - Data Hazard"); end

else if (((addr_even_ra!=7'bx) && (addr_even_ra == Rt_Temp3_odd [7:14])) ||
((addr_even_rb!=7'bx) && (addr_even_rb == Rt_Temp3_odd [7:14])) ||
((addr_even_rc!=7'bx) && (addr_even_rc == Rt_Temp3_odd [7:14])) ||
((addr_odd_ra!=7'bx) && (addr_odd_ra == Rt_Temp3_odd [7:14])) ||
((addr_odd_rb!=7'bx) && (addr_odd_rb == Rt_Temp3_odd [7:14])) ||
((addr_odd_rc!=7'bx) && (addr_odd_rc == Rt_Temp3_odd [7:14])))) begin
count=Rt_Temp3_odd[3:5]-5; $display("5th ODD If - Data Hazard"); end

else if (((addr_even_ra!=7'bx) && (addr_even_ra == Rt_Temp4_odd [7:14])) ||
((addr_even_rb!=7'bx) && (addr_even_rb == Rt_Temp4_odd [7:14])) ||
((addr_even_rc!=7'bx) && (addr_even_rc == Rt_Temp4_odd [7:14])) ||
((addr_odd_ra!=7'bx) && (addr_odd_ra == Rt_Temp4_odd [7:14])) ||
((addr_odd_rb!=7'bx) && (addr_odd_rb == Rt_Temp4_odd [7:14])) ||
((addr_odd_rc!=7'bx) && (addr_odd_rc == Rt_Temp4_odd [7:14])))) begin
count=Rt_Temp4_odd[3:5]-6; $display("6th ODD If - Data Hazard"); end

else if (((addr_even_ra!=7'bx) && (addr_even_ra == Rt_Temp5_odd [7:14])) ||
((addr_even_rb!=7'bx) && (addr_even_rb == Rt_Temp5_odd [7:14])) ||
((addr_even_rc!=7'bx) && (addr_even_rc == Rt_Temp5_odd [7:14])) ||
((addr_odd_ra!=7'bx) && (addr_odd_ra == Rt_Temp5_odd [7:14])) ||
((addr_odd_rb!=7'bx) && (addr_odd_rb == Rt_Temp5_odd [7:14])) ||
((addr_odd_rc!=7'bx) && (addr_odd_rc == Rt_Temp5_odd [7:14])))) begin
count=Rt_Temp5_odd[3:5]-7; $display("7th ODD If - Data Hazard"); end

// else if (((addr_even_ra!=7'bx) && (addr_even_ra == Rt_Temp6_odd [7:14])) ||
// ((addr_even_rb!=7'bx) && (addr_even_rb == Rt_Temp6_odd [7:14])) ||
// ((addr_even_rc!=7'bx) && (addr_even_rc == Rt_Temp6_odd [7:14])) ||
// ((addr_odd_ra!=7'bx) && (addr_odd_ra == Rt_Temp6_odd [7:14])) ||
// ((addr_odd_rb!=7'bx) && (addr_odd_rb == Rt_Temp6_odd [7:14])) ||
// ((addr_odd_rc!=7'bx) && (addr_odd_rc == Rt_Temp6_odd [7:14])))) begin
// count=Rt_Temp6_odd[3:5]-7; $display("7th ODD If - Data Hazard"); end

else
count=0;
end

//always_ff@(posedge clk) begin $display("@ DEPENDENCY Rt_Temp_even: %d",Rt_Temp_even[0:2]); end

endmodule

// ----- 1st Backup

// always_ff @(posedge clk) begin
// $display("Counter:%d, Count:%d ", counter, count);
// if(count == 1) counter <= 1;
// else if (count == 2 ) counter <= 2;
// else if (count == 3 ) counter <= 3;
// else if (count == 4 ) counter <= 4;
// else if (count == 5 ) counter <= 5;
// else if (count == 6 ) counter <= 6;
// else counter <= 0;

// if(count == 0 && counter > 0) begin
// counter <= counter - 1; //$display("----->>> Data Hazard");

```

```

//      Dependency_stall = 1;
//      opcode_even_depend = 11'b0100_0000_001;
//      opcode_odd_depend = 11'b0100_0000_010;
//      end

//      else if(count == 0 && counter == 0) begin
//          //$display("----->>> No Data Hazard, Counter:%d, Count:%d ", counter,
count);
//          $display ("opcode_even: %b | @ Dependency Stge" , opcode_even_depend);
//          $display ("opcode_odd: %b | @ Dependency Stage", opcode_odd_depend);

//          Dependency_stall = 0;
//          addr_even_ra_depend<=addr_even_ra;
//          addr_even_rb_depend<=addr_even_rb;
//          addr_even_rc_depend<=addr_even_rc;
//          opcode_even_depend <= opcode_even; addr_even_rt_depend <= addr_even_rt;
//          imm7_even_depend <= imm7_even; imm10_even_depend <= imm10_even;
//          imm16_even_depend <= imm16_even; imm18_even_depend <= imm18_even;

//          addr_odd_ra_depend<=addr_odd_ra;
//          addr_odd_rb_depend<=addr_odd_rb;
//          addr_odd_rc_depend<=addr_odd_rc;
//          opcode_odd_depend <= opcode_odd; addr_odd_rt_depend <= addr_odd_rt;
//          imm7_odd_depend <= imm7_odd; imm10_odd_depend <= imm10_odd;
//          imm16_odd_depend <= imm16_odd; imm18_odd_depend <= imm18_odd;

// // ----- For Latency Calculation @ Reg File

//      opcode_even_depend_temp <= opcode_even;
//      addr_even_rt_depend_temp <= addr_even_rt;
//      opcode_odd_depend_temp <= opcode_odd;
//      addr_odd_rt_depend_temp <= addr_odd_rt;

//      end
//      end

// ----- 2nd Backup

//=====Dependency=====
=====

// module Dependency (clk, reset,addr_even_ra, addr_even_rb, addr_even_rc, opcode_even, imm10_even, imm16_even,
imm18_even, imm7_even, addr_even_rt,
// addr_odd_ra, addr_odd_rb, addr_odd_rc, opcode_odd, imm10_odd, imm16_odd, imm18_odd, imm7_odd, addr_odd_rt,
flush,opcode_even_depend, imm7_even_depend, imm10_even_depend, imm16_even_depend, imm18_even_depend,
addr_even_rt_depend, addr_even_ra_depend, addr_even_rb_depend, addr_even_rc_depend ,
opcode_odd_depend,imm7_odd_depend,imm10_odd_depend,imm16_odd_depend,imm18_odd_depend,addr_odd_rt_depend,addr_o
dd_ra_depend, addr_odd_rb_depend, addr_odd_rc_depend, Dependency_stall, Rt_Temp_even, Rt_Temp1_even, Rt_Temp2_even,
Rt_Temp3_even, Rt_Temp4_even, Rt_Temp5_even, Rt_Temp6_even, Rt_Temp_odd, Rt_Temp1_odd, Rt_Temp2_odd,
Rt_Temp3_odd, Rt_Temp4_odd, Rt_Temp5_odd, Rt_Temp6_odd);

// input clk, reset, flush;
// input [0:6] addr_even_ra, addr_even_rb, addr_even_rc;
// input [0:6] addr_odd_ra, addr_odd_rb, addr_odd_rc;
// input [0:10] opcode_even;
// input signed [0:6] imm7_even;
// input signed [0:9] imm10_even;
// input signed [0:15] imm16_even;
// input signed [0:17] imm18_even;
// input [0:6] addr_even_rt;

```

```

// input [0:142] Rt_Temp_even, Rt_Temp1_even, Rt_Temp2_even, Rt_Temp3_even, Rt_Temp4_even, Rt_Temp5_even,
Rt_Temp6_even;

// output logic [0:10] opcode_even_depend;
// output logic signed [0:6] imm7_even_depend;
// output logic signed [0:9] imm10_even_depend;
// output logic signed [0:15] imm16_even_depend;
// output logic signed [0:17] imm18_even_depend;
// output logic [0:6] addr_even_rt_depend;
// output logic [0:6] addr_even_ra_depend, addr_even_rb_depend, addr_even_rc_depend;

// input [0:10] opcode_odd;
// input [0:6] imm7_odd;
// input [0:9] imm10_odd;
// input [0:15] imm16_odd;
// input [0:17] imm18_odd;
// input [0:6] addr_odd_rt;
// input [0:175] Rt_Temp_odd, Rt_Temp1_odd, Rt_Temp2_odd, Rt_Temp3_odd, Rt_Temp4_odd, Rt_Temp5_odd,
Rt_Temp6_odd;

// output logic [0:10] opcode_odd_depend;
// output logic signed [0:6] imm7_odd_depend;
// output logic signed [0:9] imm10_odd_depend;
// output logic signed [0:15] imm16_odd_depend;
// output logic signed [0:17] imm18_odd_depend;
// output logic signed [0:6] addr_odd_rt_depend;
// output logic [0:6] addr_odd_ra_depend, addr_odd_rb_depend, addr_odd_rc_depend;

// logic [0:10] opcode_odd_depend2, opcode_even_depend2 ;

// output logic Dependency_stall;

// logic [0:10] opcode_even_depend_temp, opcode_odd_depend_temp;
// logic [0:6] addr_even_rt_depend_temp, addr_odd_rt_depend_temp;
// logic [0:2] opcode_even_depend_lat, opcode_odd_depend_lat;
// logic [0:2] count, counter;

// assign Dependency_stall = (counter > 0);

// always_comb begin
//   if (Dependency_stall) begin
//     opcode_even_depend = 11'b0100_0000_001;
//     opcode_odd_depend = 11'b0100_0000_010; end

//   else begin
//     opcode_even_depend = opcode_even_depend2;
//     opcode_odd_depend = opcode_odd_depend2;end
// end

// always_ff @(posedge clk) begin
//   $display("Counter:%d, Count:%d ", counter, count);

//   if(counter > 0) begin // count == 0 &&
//     counter = counter - 1; $display("-----|||||||----->>> Data Hazard");
//     //Dependency_stall = 1;
//     // opcode_even_depend = 11'b0100_0000_001;
//     // opcode_odd_depend = 11'b0100_0000_010;
//     end

//   else if(count == 1) counter <= 0;

```

```

// else if (count == 2 ) counter <= 1;
// else if (count == 3 ) counter <= 2;
// else if (count == 4 ) counter <= 3;
// else if (count == 5 ) counter <= 4;
// else if (count == 6 ) counter <= 5;

// else if(count == 0 && counter == 0) begin
//   //$display("----->>> No Data Hazard, Counter:%d, Count:%d ", counter,
count);
//   //Dependency_stall = 0;
//   addr_even_ra_depend<=addr_even_ra;
//   addr_even_rb_depend<=addr_even_rb;
//   addr_even_rc_depend<=addr_even_rc;
//   opcode_even_depend2 <= opcode_even; addr_even_rt_depend <= addr_even_rt;
//   imm7_even_depend <= imm7_even; imm10_even_depend <= imm10_even;
//   imm16_even_depend <= imm16_even; imm18_even_depend <= imm18_even;

//   addr_odd_ra_depend<=addr_odd_ra;
//   addr_odd_rb_depend<=addr_odd_rb;
//   addr_odd_rc_depend<=addr_odd_rc;
//   opcode_odd_depend2 <= opcode_odd; addr_odd_rt_depend <= addr_odd_rt;
//   imm7_odd_depend <= imm7_odd; imm10_odd_depend <= imm10_odd;
//   imm16_odd_depend <= imm16_odd; imm18_odd_depend <= imm18_odd;

// // ----- For Latency Calculation @ Reg File

//   opcode_even_depend_temp <= opcode_even;
//   addr_even_rt_depend_temp <= addr_even_rt;
//   opcode_odd_depend_temp <= opcode_odd;
//   addr_odd_rt_depend_temp <= addr_odd_rt;

// end

// else counter <= 0;

//   $display ("opcode_even: %b | @ Dependency Stge" , opcode_even_depend);
//   $display ("opcode_odd: %b | @ Dependency Stage", opcode_odd_depend);
// end

// always_comb begin
// if (opcode_even_depend_temp == 7'b0100_001|opcode_even_depend_temp == 8'b0001_1100|opcode_even_depend_temp
== 8'b0001_0110|opcode_even_depend_temp == 8'b0111_1110||
// opcode_even_depend_temp == 8'b0111_1100|opcode_even_depend_temp == 8'b0100_1110|opcode_even_depend_temp
== 8'b0100_1100|opcode_even_depend_temp == 8'b0101_1110||
// opcode_even_depend_temp == 8'b0000_0100|opcode_even_depend_temp == 8'b0000_1100|opcode_even_depend_temp
== 8'b0100_0110|opcode_even_depend_temp == 8'b0100_0100||
// opcode_even_depend_temp == 9'b0100_0000_1|opcode_even_depend_temp ==
9'b0100_0001_1|opcode_even_depend_temp == 11'b0001_1000_000|opcode_even_depend_temp == 11'b1101_0000_000||
// opcode_even_depend_temp == 11'b0001_1000_001|opcode_even_depend_temp ==
11'b0101_1000_001|opcode_even_depend_temp == 11'b0111_1000_000|opcode_even_depend_temp ==
11'b0111_1010_000||
// opcode_even_depend_temp == 11'b0100_1010_000|opcode_even_depend_temp ==
11'b0101_1010_000|opcode_even_depend_temp == 11'b0001_1001_001|opcode_even_depend_temp ==
11'b0000_1001_001||
// opcode_even_depend_temp == 11'b0000_1000_001|opcode_even_depend_temp ==
11'b0101_1001_001|opcode_even_depend_temp == 11'b0000_1000_000|opcode_even_depend_temp ==
11'b0110_1000_001||
// opcode_even_depend_temp == 11'b0100_1000_001|opcode_even_depend_temp ==
11'b0101_0110_110|opcode_even_depend_temp == 11'b0101_0101_110|opcode_even_depend_temp ==
11'b0101_0100_110)

```

```

// opcode_even_depend_lat = 3'd2;
// else if(opcode_even_depend_temp == 11'b0000_1111_0111) opcode_even_depend_temp == 11'b0000_1011_0001
opcode_even_depend_temp == 11'b0000_1011_0011)opcode_even_depend_temp == 11'b0000_1011_0111
// opcode_even_depend_temp == 11'b0101_0110_1001)opcode_even_depend_temp ==
11'b0000_1010_0111)opcode_even_depend_temp == 11'b0001_1010_0111)opcode_even_depend_temp ==
11'b0100_1010_0111)
// opcode_even_depend_lat = 3'd4;
// else if(opcode_even_depend_temp == 4'b1110_1111)opcode_even_depend_temp == 4'b1111_1111)opcode_even_depend_temp ==
11'b0101_1000_1001)opcode_even_depend_temp == 11'b0101_1000_1101)opcode_even_depend_temp ==
11'b0101_1000_1011)
// opcode_even_depend_lat = 3'd6;
// else if (opcode_even_depend_temp == 4'b1100_1111) opcode_even_depend_temp == 8'b0111_0100)
opcode_even_depend_temp == 8'b0111_0101)opcode_even_depend_temp == 11'b0111_1000_1001
opcode_even_depend_temp == 11'b0111_1000_1111)opcode_even_depend_temp == 11'b0111_1001_1001)
// opcode_even_depend_lat = 3'd7;
// else opcode_even_depend_lat = 3'd0;

// if(opcode_odd_depend_temp == 11'b0011_1011_1011)opcode_odd_depend_temp ==
11'b0011_1011_1111)opcode_odd_depend_temp == 9'b0011_0001_0111)opcode_odd_depend_temp == 9'b0011_0011_0111
// opcode_odd_depend_temp == 9'b0011_0010_0111)opcode_odd_depend_temp == 9'b0011_0000_0111)opcode_odd_depend_temp
== 11'b0011_0101_0001)opcode_odd_depend_temp == 11'b0011_0101_0011
// opcode_odd_depend_temp == 9'b0010_0001_0111)opcode_odd_depend_temp == 9'b0010_0000_0111
// opcode_odd_depend_lat = 3'd4;
// else if (opcode_odd_depend_temp == 9'b0011_0000_1111)opcode_odd_depend_temp ==
9'b0010_0000_1111)opcode_odd_depend_temp == 11'b0011_1000_1001)opcode_odd_depend_temp == 11'b0010_1000_1001)
// opcode_odd_depend_lat = 3'd6;
// else opcode_odd_depend_lat = 3'd0;

// end // always_comb

// always_comb begin

// $display("===== > addr_even_rb: %d | Rt_Temp_odd:%d",addr_even_rb, Rt_Temp_odd [7:14]);

// if (((addr_even_ra!=7'b0) && (addr_even_ra == addr_even_rt_depend_temp)) ||
//      ((addr_even_rb!=7'b0) && (addr_even_rb == addr_even_rt_depend_temp)) ||
//      ((addr_even_rc!=7'b0) && (addr_even_rc == addr_even_rt_depend_temp)) ||
//      ((addr_odd_ra!=7'b0) && (addr_odd_ra == addr_even_rt_depend_temp)) ||
//      ((addr_odd_rb!=7'b0) && (addr_odd_rb == addr_even_rt_depend_temp)) ||
//      ((addr_odd_rc!=7'b0) && (addr_odd_rc == addr_even_rt_depend_temp)))) begin
//      count=opcode_even_depend_lat-1; $display("1st EVEN If - Data Hazard");
//      end

// else if (((addr_even_ra!=7'b0) && (addr_even_ra == Rt_Temp_even [7:14])) ||
//          ((addr_even_rb!=7'b0) && (addr_even_rb == Rt_Temp_even [7:14])) ||
//          ((addr_even_rc!=7'b0) && (addr_even_rc == Rt_Temp_even [7:14])) ||
//          ((addr_odd_ra!=7'b0) && (addr_odd_ra == Rt_Temp_even [7:14])) ||
//          ((addr_odd_rb!=7'b0) && (addr_odd_rb == Rt_Temp_even [7:14])) ||
//          ((addr_odd_rc!=7'b0) && (addr_odd_rc == Rt_Temp_even [7:14])))) begin
//          count=Rt_Temp_even[3:5]-2; $display("2st EVEN If - Data Hazard");
//          end

// else if (((addr_even_ra!=7'bx) && (addr_even_ra == Rt_Temp1_even [7:14])) ||
//          ((addr_even_rb!=7'bx) && (addr_even_rb == Rt_Temp1_even [7:14])) ||
//          ((addr_even_rc!=7'bx) && (addr_even_rc == Rt_Temp1_even [7:14])) ||
//          ((addr_odd_ra!=7'bx) && (addr_odd_ra == Rt_Temp1_even [7:14])) ||
//          ((addr_odd_rb!=7'bx) && (addr_odd_rb == Rt_Temp1_even [7:14])) ||
//          ((addr_odd_rc!=7'bx) && (addr_odd_rc == Rt_Temp1_even [7:14])))) begin
//          count=Rt_Temp1_even[3:5]-3; $display("3rd EVEN If - Data Hazard"); end

```



```

// else if (((addr_even_ra!=7'bx) && (addr_even_ra == Rt_Temp2_even [7:14])) ||
//          ((addr_even_rb!=7'bx) && (addr_even_rb == Rt_Temp2_even [7:14])) ||
//          ((addr_even_rc!=7'bx) && (addr_even_rc == Rt_Temp2_even [7:14])) ||
//          ((addr_odd_ra!=7'bx) && (addr_odd_ra == Rt_Temp2_even [7:14])) ||
//          ((addr_odd_rb!=7'bx) && (addr_odd_rb == Rt_Temp2_even [7:14])) ||
//          ((addr_odd_rc!=7'bx) && (addr_odd_rc == Rt_Temp2_even [7:14]))) begin
//     count=Rt_Temp2_even[3:5]-4; $display("4rd EVEN If - Data Hazard"); end

// else if (((addr_even_ra!=7'bx) && (addr_even_ra == Rt_Temp3_even [7:14])) ||
//          ((addr_even_rb!=7'bx) && (addr_even_rb == Rt_Temp3_even [7:14])) ||
//          ((addr_even_rc!=7'bx) && (addr_even_rc == Rt_Temp3_even [7:14])) ||
//          ((addr_odd_ra!=7'bx) && (addr_odd_ra == Rt_Temp3_even [7:14])) ||
//          ((addr_odd_rb!=7'bx) && (addr_odd_rb == Rt_Temp3_even [7:14])) ||
//          ((addr_odd_rc!=7'bx) && (addr_odd_rc == Rt_Temp3_even [7:14]))) begin
//     count=Rt_Temp3_even[3:5]-5; $display("5th EVEN If - Data Hazard"); end

// else if (((addr_even_ra!=7'bx) && (addr_even_ra == Rt_Temp4_even [7:14])) ||
//          ((addr_even_rb!=7'bx) && (addr_even_rb == Rt_Temp4_even [7:14])) ||
//          ((addr_even_rc!=7'bx) && (addr_even_rc == Rt_Temp4_even [7:14])) ||
//          ((addr_odd_ra!=7'bx) && (addr_odd_ra == Rt_Temp4_even [7:14])) ||
//          ((addr_odd_rb!=7'bx) && (addr_odd_rb == Rt_Temp4_even [7:14])) ||
//          ((addr_odd_rc!=7'bx) && (addr_odd_rc == Rt_Temp4_even [7:14]))) begin
//     count=Rt_Temp4_even[3:5]-6; $display("6th EVEN If - Data Hazard"); end

// else if (((addr_even_ra!=7'bx) && (addr_even_ra == Rt_Temp5_even [7:14])) ||
//          ((addr_even_rb!=7'bx) && (addr_even_rb == Rt_Temp5_even [7:14])) ||
//          ((addr_even_rc!=7'bx) && (addr_even_rc == Rt_Temp5_even [7:14])) ||
//          ((addr_odd_ra!=7'bx) && (addr_odd_ra == Rt_Temp5_even [7:14])) ||
//          ((addr_odd_rb!=7'bx) && (addr_odd_rb == Rt_Temp5_even [7:14])) ||
//          ((addr_odd_rc!=7'bx) && (addr_odd_rc == Rt_Temp5_even [7:14]))) begin
//     count=Rt_Temp5_even[3:5]-7; $display("7th EVEN If - Data Hazard"); end

// // else if (((addr_even_ra!=7'bx) && (addr_even_ra == Rt_Temp6_even [7:14])) ||
// //          ((addr_even_rb!=7'bx) && (addr_even_rb == Rt_Temp6_even [7:14])) ||
// //          ((addr_even_rc!=7'bx) && (addr_even_rc == Rt_Temp6_even [7:14])) ||
// //          ((addr_odd_ra!=7'bx) && (addr_odd_ra == Rt_Temp6_even [7:14])) ||
// //          ((addr_odd_rb!=7'bx) && (addr_odd_rb == Rt_Temp6_even [7:14])) ||
// //          ((addr_odd_rc!=7'bx) && (addr_odd_rc == Rt_Temp6_even [7:14]))) begin
// //     count=Rt_Temp6_even[3:5]-7; $display("8th EVEN If - Data Hazard"); end

// // -----

// else if (((addr_even_ra!=7'b0) && (addr_even_ra == addr_odd_rt_depend_temp)) ||
//          ((addr_even_rb!=7'b0) && (addr_even_rb == addr_odd_rt_depend_temp)) ||
//          ((addr_even_rc!=7'b0) && (addr_even_rc == addr_odd_rt_depend_temp)) ||
//          ((addr_odd_ra!=7'b0) && (addr_odd_ra == addr_odd_rt_depend_temp)) ||
//          ((addr_odd_rb!=7'b0) && (addr_odd_rb == addr_odd_rt_depend_temp)) ||
//          ((addr_odd_rc!=7'b0) && (addr_odd_rc == addr_odd_rt_depend_temp))) begin
//     count=opcode_odd_depend_lat-1; $display("1st ODD If - Data Hazard" );
//     end

// else if (((addr_even_ra!=7'bx) && (addr_even_ra == Rt_Temp_odd [7:14])) ||
//          ((addr_even_rb!=7'bx) && (addr_even_rb == Rt_Temp_odd [7:14])) ||
//          ((addr_even_rc!=7'bx) && (addr_even_rc == Rt_Temp_odd [7:14])) ||
//          ((addr_odd_ra!=7'bx) && (addr_odd_ra == Rt_Temp_odd [7:14])) ||
//          ((addr_odd_rb!=7'bx) && (addr_odd_rb == Rt_Temp_odd [7:14])) ||
//          ((addr_odd_rc!=7'bx) && (addr_odd_rc == Rt_Temp_odd [7:14]))) begin
//     count=Rt_Temp_odd[3:5]-2; $display("2st ODD If - Data Hazard"); end

// else if (((addr_even_ra!=7'bx) && (addr_even_ra == Rt_Temp1_odd [7:14])) ||
//          ((addr_even_rb!=7'bx) && (addr_even_rb == Rt_Temp1_odd [7:14])) ||
//          ((addr_even_rc!=7'bx) && (addr_even_rc == Rt_Temp1_odd [7:14])) ||
//          ((addr_odd_ra!=7'bx) && (addr_odd_ra == Rt_Temp1_odd [7:14])) ||

```

```

//      ((addr_odd_rb!=7'bx) && (addr_odd_rb == Rt_Temp1_odd [7:14])) ||
//      ((addr_odd_rc!=7'bx) && (addr_odd_rc == Rt_Temp1_odd [7:14])))) begin
//      count=Rt_Temp1_odd[3:5]-3; $display("3rd ODD If - Data Hazard"); end

// else if (((addr_even_ra!=7'bx) && (addr_even_ra == Rt_Temp2_odd [7:14])) ||
//      ((addr_even_rb!=7'bx) && (addr_even_rb == Rt_Temp2_odd [7:14])) ||
//      ((addr_even_rc!=7'bx) && (addr_even_rc == Rt_Temp2_odd [7:14])) ||
//      ((addr_odd_ra!=7'bx) && (addr_odd_ra == Rt_Temp2_odd [7:14])) ||
//      ((addr_odd_rb!=7'bx) && (addr_odd_rb == Rt_Temp2_odd [7:14])) ||
//      ((addr_odd_rc!=7'bx) && (addr_odd_rc == Rt_Temp2_odd [7:14])))) begin
//      count=Rt_Temp2_odd[3:5]-4; $display("4rd ODD If - Data Hazard"); end

// else if (((addr_even_ra!=7'bx) && (addr_even_ra == Rt_Temp3_odd [7:14])) ||
//      ((addr_even_rb!=7'bx) && (addr_even_rb == Rt_Temp3_odd [7:14])) ||
//      ((addr_even_rc!=7'bx) && (addr_even_rc == Rt_Temp3_odd [7:14])) ||
//      ((addr_odd_ra!=7'bx) && (addr_odd_ra == Rt_Temp3_odd [7:14])) ||
//      ((addr_odd_rb!=7'bx) && (addr_odd_rb == Rt_Temp3_odd [7:14])) ||
//      ((addr_odd_rc!=7'bx) && (addr_odd_rc == Rt_Temp3_odd [7:14])))) begin
//      count=Rt_Temp3_odd[3:5]-5; $display("5th ODD If - Data Hazard"); end

// else if (((addr_even_ra!=7'bx) && (addr_even_ra == Rt_Temp4_odd [7:14])) ||
//      ((addr_even_rb!=7'bx) && (addr_even_rb == Rt_Temp4_odd [7:14])) ||
//      ((addr_even_rc!=7'bx) && (addr_even_rc == Rt_Temp4_odd [7:14])) ||
//      ((addr_odd_ra!=7'bx) && (addr_odd_ra == Rt_Temp4_odd [7:14])) ||
//      ((addr_odd_rb!=7'bx) && (addr_odd_rb == Rt_Temp4_odd [7:14])) ||
//      ((addr_odd_rc!=7'bx) && (addr_odd_rc == Rt_Temp4_odd [7:14])))) begin
//      count=Rt_Temp4_odd[3:5]-6; $display("6th ODD If - Data Hazard"); end

// else if (((addr_even_ra!=7'bx) && (addr_even_ra == Rt_Temp5_odd [7:14])) ||
//      ((addr_even_rb!=7'bx) && (addr_even_rb == Rt_Temp5_odd [7:14])) ||
//      ((addr_even_rc!=7'bx) && (addr_even_rc == Rt_Temp5_odd [7:14])) ||
//      ((addr_odd_ra!=7'bx) && (addr_odd_ra == Rt_Temp5_odd [7:14])) ||
//      ((addr_odd_rb!=7'bx) && (addr_odd_rb == Rt_Temp5_odd [7:14])) ||
//      ((addr_odd_rc!=7'bx) && (addr_odd_rc == Rt_Temp5_odd [7:14])))) begin
//      count=Rt_Temp5_odd[3:5]-7; $display("7th ODD If - Data Hazard"); end

// // else if (((addr_even_ra!=7'bx) && (addr_even_ra == Rt_Temp6_odd [7:14])) ||
// //      ((addr_even_rb!=7'bx) && (addr_even_rb == Rt_Temp6_odd [7:14])) ||
// //      ((addr_even_rc!=7'bx) && (addr_even_rc == Rt_Temp6_odd [7:14])) ||
// //      ((addr_odd_ra!=7'bx) && (addr_odd_ra == Rt_Temp6_odd [7:14])) ||
// //      ((addr_odd_rb!=7'bx) && (addr_odd_rb == Rt_Temp6_odd [7:14])) ||
// //      ((addr_odd_rc!=7'bx) && (addr_odd_rc == Rt_Temp6_odd [7:14])))) begin
// //      count=Rt_Temp6_odd[3:5]-7; $display("7th ODD If - Data Hazard"); end

// else
//      count=0;
// end

// //always_ff@(posedge clk) begin $display("@ DEPENDENCY Rt_Temp_even: %d",Rt_Temp_even[0:2]); end

// endmodule

//-----

```

```

//
//=====Dependency=====
=====

// module Dependency (clk, reset,addr_even_ra, addr_even_rb, addr_even_rc, opcode_even, imm10_even, imm16_even,
imm18_even, imm7_even, addr_even_rt,
// addr_odd_ra, addr_odd_rb, addr_odd_rc, opcode_odd, imm10_odd, imm16_odd, imm18_odd, imm7_odd, addr_odd_rt,
flush,opcode_even_depend, imm7_even_depend, imm10_even_depend, imm16_even_depend, imm18_even_depend,
addr_even_rt_depend, addr_even_ra_depend, addr_even_rb_depend, addr_even_rc_depend ,
opcode_odd_depend,imm7_odd_depend,imm10_odd_depend,imm16_odd_depend,imm18_odd_depend,addr_odd_rt_depend,addr_o
dd_ra_depend, addr_odd_rb_depend, addr_odd_rc_depend, Dependency_stall, Rt_Temp_even, Rt_Temp1_even, Rt_Temp2_even,
Rt_Temp3_even, Rt_Temp4_even, Rt_Temp5_even, Rt_Temp6_even, Rt_Temp_odd, Rt_Temp1_odd, Rt_Temp2_odd,
Rt_Temp3_odd, Rt_Temp4_odd, Rt_Temp5_odd, Rt_Temp6_odd);

// input clk, reset, flush;
// input [0:6] addr_even_ra, addr_even_rb, addr_even_rc;
// input [0:6] addr_odd_ra, addr_odd_rb, addr_odd_rc;
// input [0:10] opcode_even;
// input signed [0:6] imm7_even;
// input signed [0:9] imm10_even;
// input signed [0:15] imm16_even;
// input signed [0:17] imm18_even;
// input [0:6] addr_even_rt;
// input [0:142] Rt_Temp_even, Rt_Temp1_even, Rt_Temp2_even, Rt_Temp3_even, Rt_Temp4_even, Rt_Temp5_even,
Rt_Temp6_even;

// output logic [0:10] opcode_even_depend;
// output logic signed [0:6] imm7_even_depend;
// output logic signed [0:9] imm10_even_depend;
// output logic signed [0:15] imm16_even_depend;
// output logic signed [0:17] imm18_even_depend;
// output logic [0:6] addr_even_rt_depend;
// output logic [0:6] addr_even_ra_depend, addr_even_rb_depend, addr_even_rc_depend;

// input [0:10] opcode_odd;
// input [0:6] imm7_odd;
// input [0:9] imm10_odd;
// input [0:15] imm16_odd;
// input [0:17] imm18_odd;
// input [0:6] addr_odd_rt;
// input [0:175] Rt_Temp_odd, Rt_Temp1_odd, Rt_Temp2_odd, Rt_Temp3_odd, Rt_Temp4_odd, Rt_Temp5_odd,
Rt_Temp6_odd;

// output logic [0:10] opcode_odd_depend;
// output logic signed [0:6] imm7_odd_depend;
// output logic signed [0:9] imm10_odd_depend;
// output logic signed [0:15] imm16_odd_depend;
// output logic signed [0:17] imm18_odd_depend;
// output logic signed [0:6] addr_odd_rt_depend;
// output logic [0:6] addr_odd_ra_depend, addr_odd_rb_depend, addr_odd_rc_depend;
// logic [0:6] addr_odd_rt_depend2;
// logic signed [0:15] imm16_odd_depend2;

// logic [0:10] opcode_odd_depend_bck;
// logic signed [0:6] imm7_odd_depend_bck;
// logic signed [0:9] imm10_odd_depend_bck;
// logic signed [0:15] imm16_odd_depend_bck;
// logic signed [0:17] imm18_odd_depend_bck;
// logic signed [0:6] addr_odd_rt_depend_bck;
// logic [0:6] addr_odd_ra_depend_bck, addr_odd_rb_depend_bck, addr_odd_rc_depend_bck;

```

```

// logic [0:10] opcode_even_depend_bck;
// logic signed [0:6] imm7_even_depend_bck;
// logic signed [0:9] imm10_even_depend_bck;
// logic signed [0:15] imm16_even_depend_bck;
// logic signed [0:17] imm18_even_depend_bck;
// logic [0:6] addr_even_rt_depend_bck;
// logic [0:6] addr_even_ra_depend_bck, addr_even_rb_depend_bck, addr_even_rc_depend_bck;

// logic [0:10] opcode_odd_depend2, opcode_even_depend2 ;

// output logic Dependency_stall;

// logic [0:10] opcode_even_depend_temp, opcode_odd_depend_temp;
// logic [0:6] addr_even_rt_depend_temp, addr_odd_rt_depend_temp;
// logic [0:2] opcode_even_depend_lat, opcode_odd_depend_lat;
// logic [0:2] count, counter;
// logic Prev_Stall;

// assign Dependency_stall = (counter > 0);

// always_ff @(posedge clk) begin
//   $display("Counter:%d, Count:%d ", counter, count);

//   Prev_Stall <= Dependency_stall;

//   if(counter > 0) begin // count == 0 &&
//     counter = counter - 1; $display("-----|||||||||----->>> Data Hazard");
//     //Dependency_stall = 1;
//     // opcode_even_depend = 11'b0100_0000_001;
//     // opcode_odd_depend = 11'b0100_0000_010;
//     end

//   else if(count == 1) counter <= 1;
//   else if (count == 2 ) counter <= 2;
//   else if (count == 3 ) counter <= 3;
//   else if (count == 4 ) counter <= 4;
//   else if (count == 5 ) counter <= 5;
//   else if (count == 6 ) counter <= 6;

//   if (Dependency_stall) begin
//     opcode_even_depend = 11'b0100_0000_001;
//     opcode_odd_depend = 11'b0100_0000_010; end

//   else if (Prev_Stall == 1 && Dependency_stall == 0) begin
//     addr_even_ra_depend <= addr_even_ra_depend_bck;
//     addr_even_rb_depend <= addr_even_rb_depend_bck;
//     addr_even_rc_depend <= addr_even_rc_depend_bck;
//     opcode_even_depend <= opcode_even_depend_bck;
//     addr_even_rt_depend <= addr_even_rt_depend_bck;
//     imm7_even_depend <= imm7_even_depend_bck;
//     imm10_even_depend <= imm10_even_depend_bck;
//     imm16_even_depend <= imm16_even_depend_bck;
//     imm18_even_depend <= imm18_even_depend_bck;

//     addr_odd_ra_depend <= addr_odd_ra_depend_bck;
//     addr_odd_rb_depend <= addr_odd_rb_depend_bck;
//     addr_odd_rc_depend <= addr_odd_rc_depend_bck;
//     opcode_odd_depend <= opcode_odd_depend_bck;
//     addr_odd_rt_depend <= addr_odd_rt_depend_bck;
//     imm7_odd_depend <= imm7_odd_depend_bck;

```

```

//   imm10_odd_depend    <= imm10_odd_depend_bck;
//   imm16_odd_depend    <= imm16_odd_depend_bck;
//   imm18_odd_depend    <= imm18_odd_depend_bck;

//   end

//   else if(count == 0 && counter == 0) begin
//       //$display("----->>> No Data Hazard, Counter:%d, Count:%d ", counter,
count);
//       //Dependency_stall = 0;
//       addr_even_ra_depend<=addr_even_ra;
//       addr_even_rb_depend<=addr_even_rb;
//       addr_even_rc_depend<=addr_even_rc;
//       opcode_even_depend <= opcode_even; addr_even_rt_depend <= addr_even_rt;
//       imm7_even_depend <= imm7_even; imm10_even_depend <= imm10_even;
//       imm16_even_depend <= imm16_even; imm18_even_depend <= imm18_even;

//       addr_odd_ra_depend<=addr_odd_ra;
//       addr_odd_rb_depend<=addr_odd_rb;
//       addr_odd_rc_depend<=addr_odd_rc;
//       opcode_odd_depend <= opcode_odd; addr_odd_rt_depend2 <= addr_odd_rt;
//       imm7_odd_depend <= imm7_odd; imm10_odd_depend <= imm10_odd;
//       imm16_odd_depend <= imm16_odd; imm18_odd_depend <= imm18_odd;

// // ----- For Latency Calculation @ Reg File

//       opcode_even_depend_temp <= opcode_even;
//       addr_even_rt_depend_temp <= addr_even_rt;
//       opcode_odd_depend_temp <= opcode_odd;
//       addr_odd_rt_depend_temp <= addr_odd_rt;

//   end

//   else counter <= 0;

//       $display ("opcode_even: %b | @ Dependency Stge" , opcode_even_depend);
//       $display ("opcode_odd: %b | @ Dependency Stage", opcode_odd_depend);
//   end

// always_ff @(posedge clk) begin

//   if(Dependency_stall) begin
//       addr_even_ra_depend_bck<=addr_even_ra;
//       addr_even_rb_depend_bck<=addr_even_rb;
//       addr_even_rc_depend_bck<=addr_even_rc;
//       opcode_even_depend_bck <= opcode_even;
//       addr_even_rt_depend_bck <= addr_even_rt;
//       imm7_even_depend_bck <= imm7_even;
//       imm10_even_depend_bck <= imm10_even;
//       imm16_even_depend_bck <= imm16_even;
//       imm18_even_depend_bck <= imm18_even;

//       addr_odd_ra_depend_bck<=addr_odd_ra;
//       addr_odd_rb_depend_bck<=addr_odd_rb;
//       addr_odd_rc_depend_bck<=addr_odd_rc;
//       opcode_odd_depend_bck <= opcode_odd;
//       addr_odd_rt_depend_bck <= addr_odd_rt;
//       imm7_odd_depend_bck <= imm7_odd;
//       imm10_odd_depend_bck <= imm10_odd;
//       imm16_odd_depend_bck <= imm16_odd;
//       imm18_odd_depend_bck <= imm18_odd;
//   end end

```

```

// always_comb begin
// if (opcode_even_depend_temp == 7'b0100_001lopcode_even_depend_temp == 8'b0001_1100lopcode_even_depend_temp
== 8'b0001_0110lopcode_even_depend_temp == 8'b0111_1110ll
// opcode_even_depend_temp == 8'b0111_1100lopcode_even_depend_temp == 8'b0100_1110lopcode_even_depend_temp
== 8'b0100_1100lopcode_even_depend_temp == 8'b0101_1110ll
// opcode_even_depend_temp == 8'b0000_0100lopcode_even_depend_temp == 8'b0000_1100lopcode_even_depend_temp
== 8'b0100_0110lopcode_even_depend_temp == 8'b0100_0100ll
// opcode_even_depend_temp == 9'b0100_0000_1lopcode_even_depend_temp ==
9'b0100_0001_1lopcode_even_depend_temp == 11'b0001_1000_000lopcode_even_depend_temp == 11'b1101_0000_000ll
// opcode_even_depend_temp == 11'b0001_1000_001lopcode_even_depend_temp ==
11'b0101_1000_001lopcode_even_depend_temp == 11'b0111_1000_000lopcode_even_depend_temp ==
11'b0111_1010_000ll
// opcode_even_depend_temp == 11'b0100_1010_000lopcode_even_depend_temp ==
11'b0101_1010_000lopcode_even_depend_temp == 11'b0001_1001_001lopcode_even_depend_temp ==
11'b0000_1001_001ll
// opcode_even_depend_temp == 11'b0000_1000_001lopcode_even_depend_temp ==
11'b0101_1001_001lopcode_even_depend_temp == 11'b0000_1000_000lopcode_even_depend_temp ==
11'b0110_1000_001ll
// opcode_even_depend_temp == 11'b0100_1000_001lopcode_even_depend_temp ==
11'b0101_0110_110lopcode_even_depend_temp == 11'b0101_0101_110lopcode_even_depend_temp ==
11'b0101_0100_110)
// opcode_even_depend_lat = 3'd2;
// else if(opcode_even_depend_temp == 11'b0000_1111_011ll opcode_even_depend_temp == 11'b0000_1011_000ll
opcode_even_depend_temp == 11'b0000_1011_001lopcode_even_depend_temp == 11'b0000_1011_011ll
// opcode_even_depend_temp == 11'b0101_0110_100lopcode_even_depend_temp ==
11'b0000_1010_011lopcode_even_depend_temp == 11'b0001_1010_011lopcode_even_depend_temp ==
11'b0100_1010_011)
// opcode_even_depend_lat = 3'd4;
// else if(opcode_even_depend_temp == 4'b1110_1lopcode_even_depend_temp == 4'b1111_1lopcode_even_depend_temp ==
11'b0101_1000_100lopcode_even_depend_temp == 11'b0101_1000_110lopcode_even_depend_temp ==
11'b0101_1000_101)
// opcode_even_depend_lat = 3'd6;
// else if (opcode_even_depend_temp == 4'b1100_1lopcode_even_depend_temp == 8'b0111_0100ll
opcode_even_depend_temp == 8'b0111_0101lopcode_even_depend_temp == 11'b0111_1000_100ll
opcode_even_depend_temp == 11'b0111_1000_111lopcode_even_depend_temp == 11'b0111_1001_100)
// opcode_even_depend_lat = 3'd7;
// else opcode_even_depend_lat = 3'd0;

// if(opcode_odd_depend_temp == 11'b0011_1011_101lopcode_odd_depend_temp ==
11'b0011_1011_111lopcode_odd_depend_temp == 9'b0011_0001_0lopcode_odd_depend_temp == 9'b0011_0011_0ll
// opcode_odd_depend_temp == 9'b0011_0010_0lopcode_odd_depend_temp == 9'b0011_0000_0lopcode_odd_depend_temp
== 11'b0011_0101_000lopcode_odd_depend_temp == 11'b0011_0101_001ll
// opcode_odd_depend_temp == 9'b0010_0001_0lopcode_odd_depend_temp == 9'b0010_0000_0)
// opcode_odd_depend_lat = 3'd4;
// else if (opcode_odd_depend_temp == 9'b0011_0000_1lopcode_odd_depend_temp ==
9'b0010_0000_1lopcode_odd_depend_temp == 11'b0011_1000_100lopcode_odd_depend_temp == 11'b0010_1000_100)
// opcode_odd_depend_lat = 3'd6;
// else opcode_odd_depend_lat = 3'd0;

// end // always_comb

// always_comb begin

// $display("===== > addr_even_rb: %d | Rt_Temp_odd:%d",addr_even_rb, Rt_Temp_odd [7:14]);

// if (((addr_even_ra!=7'b0) && (addr_even_ra == addr_even_rt_depend_temp)) ll

```

```

//      ((addr_even_rb!=7'b0) && (addr_even_rb == addr_even_rt_depend_temp)) ||
//      ((addr_even_rc!=7'b0) && (addr_even_rc == addr_even_rt_depend_temp)) ||
//      ((addr_odd_ra!=7'b0) && (addr_odd_ra == addr_even_rt_depend_temp)) ||
//      ((addr_odd_rb!=7'b0) && (addr_odd_rb == addr_even_rt_depend_temp)) ||
//      ((addr_odd_rc!=7'b0) && (addr_odd_rc == addr_even_rt_depend_temp))) begin
//      count=opcode_even_depend_lat-1; $display("1st EVEN If - Data Hazard");
//      end

// else if (((addr_even_ra!=7'b0) && (addr_even_ra == Rt_Temp_even [7:14])) ||
//      ((addr_even_rb!=7'b0) && (addr_even_rb == Rt_Temp_even [7:14])) ||
//      ((addr_even_rc!=7'b0) && (addr_even_rc == Rt_Temp_even [7:14])) ||
//      ((addr_odd_ra!=7'b0) && (addr_odd_ra == Rt_Temp_even [7:14])) ||
//      ((addr_odd_rb!=7'b0) && (addr_odd_rb == Rt_Temp_even [7:14])) ||
//      ((addr_odd_rc!=7'b0) && (addr_odd_rc == Rt_Temp_even [7:14]))) begin
//      count=Rt_Temp_even[3:5]-2; $display("2st EVEN If - Data Hazard");
//      end

// else if (((addr_even_ra!=7'bx) && (addr_even_ra == Rt_Temp1_even [7:14])) ||
//      ((addr_even_rb!=7'bx) && (addr_even_rb == Rt_Temp1_even [7:14])) ||
//      ((addr_even_rc!=7'bx) && (addr_even_rc == Rt_Temp1_even [7:14])) ||
//      ((addr_odd_ra!=7'bx) && (addr_odd_ra == Rt_Temp1_even [7:14])) ||
//      ((addr_odd_rb!=7'bx) && (addr_odd_rb == Rt_Temp1_even [7:14])) ||
//      ((addr_odd_rc!=7'bx) && (addr_odd_rc == Rt_Temp1_even [7:14]))) begin
//      count=Rt_Temp1_even[3:5]-3; $display("3nd EVEN If - Data Hazard"); end

// else if (((addr_even_ra!=7'bx) && (addr_even_ra == Rt_Temp2_even [7:14])) ||
//      ((addr_even_rb!=7'bx) && (addr_even_rb == Rt_Temp2_even [7:14])) ||
//      ((addr_even_rc!=7'bx) && (addr_even_rc == Rt_Temp2_even [7:14])) ||
//      ((addr_odd_ra!=7'bx) && (addr_odd_ra == Rt_Temp2_even [7:14])) ||
//      ((addr_odd_rb!=7'bx) && (addr_odd_rb == Rt_Temp2_even [7:14])) ||
//      ((addr_odd_rc!=7'bx) && (addr_odd_rc == Rt_Temp2_even [7:14]))) begin
//      count=Rt_Temp2_even[3:5]-4; $display("4rd EVEN If - Data Hazard"); end

// else if (((addr_even_ra!=7'bx) && (addr_even_ra == Rt_Temp3_even [7:14])) ||
//      ((addr_even_rb!=7'bx) && (addr_even_rb == Rt_Temp3_even [7:14])) ||
//      ((addr_even_rc!=7'bx) && (addr_even_rc == Rt_Temp3_even [7:14])) ||
//      ((addr_odd_ra!=7'bx) && (addr_odd_ra == Rt_Temp3_even [7:14])) ||
//      ((addr_odd_rb!=7'bx) && (addr_odd_rb == Rt_Temp3_even [7:14])) ||
//      ((addr_odd_rc!=7'bx) && (addr_odd_rc == Rt_Temp3_even [7:14]))) begin
//      count=Rt_Temp3_even[3:5]-5; $display("5th EVEN If - Data Hazard"); end

// else if (((addr_even_ra!=7'bx) && (addr_even_ra == Rt_Temp4_even [7:14])) ||
//      ((addr_even_rb!=7'bx) && (addr_even_rb == Rt_Temp4_even [7:14])) ||
//      ((addr_even_rc!=7'bx) && (addr_even_rc == Rt_Temp4_even [7:14])) ||
//      ((addr_odd_ra!=7'bx) && (addr_odd_ra == Rt_Temp4_even [7:14])) ||
//      ((addr_odd_rb!=7'bx) && (addr_odd_rb == Rt_Temp4_even [7:14])) ||
//      ((addr_odd_rc!=7'bx) && (addr_odd_rc == Rt_Temp4_even [7:14]))) begin
//      count=Rt_Temp4_even[3:5]-6; $display("6th EVEN If - Data Hazard"); end

// else if (((addr_even_ra!=7'bx) && (addr_even_ra == Rt_Temp5_even [7:14])) ||
//      ((addr_even_rb!=7'bx) && (addr_even_rb == Rt_Temp5_even [7:14])) ||
//      ((addr_even_rc!=7'bx) && (addr_even_rc == Rt_Temp5_even [7:14])) ||
//      ((addr_odd_ra!=7'bx) && (addr_odd_ra == Rt_Temp5_even [7:14])) ||
//      ((addr_odd_rb!=7'bx) && (addr_odd_rb == Rt_Temp5_even [7:14])) ||
//      ((addr_odd_rc!=7'bx) && (addr_odd_rc == Rt_Temp5_even [7:14]))) begin
//      count=Rt_Temp5_even[3:5]-7; $display("7th EVEN If - Data Hazard"); end

// // else if (((addr_even_ra!=7'bx) && (addr_even_ra == Rt_Temp6_even [7:14])) ||
// //      ((addr_even_rb!=7'bx) && (addr_even_rb == Rt_Temp6_even [7:14])) ||
// //      ((addr_even_rc!=7'bx) && (addr_even_rc == Rt_Temp6_even [7:14])) ||
// //      ((addr_odd_ra!=7'bx) && (addr_odd_ra == Rt_Temp6_even [7:14])) ||
// //      ((addr_odd_rb!=7'bx) && (addr_odd_rb == Rt_Temp6_even [7:14])) ||
// //      ((addr_odd_rc!=7'bx) && (addr_odd_rc == Rt_Temp6_even [7:14]))) begin

```



```

// //          count=Rt_Temp6_even[3:5]-7; $display("8th EVEN If - Data Hazard"); end

// // -----

// else if (((addr_even_ra!=7'b0) && (addr_even_ra == addr_odd_rt_depend_temp)) ||
//          ((addr_even_rb!=7'b0) && (addr_even_rb == addr_odd_rt_depend_temp)) ||
//          ((addr_even_rc!=7'b0) && (addr_even_rc == addr_odd_rt_depend_temp)) ||
//          ((addr_odd_ra!=7'b0) && (addr_odd_ra == addr_odd_rt_depend_temp)) ||
//          ((addr_odd_rb!=7'b0) && (addr_odd_rb == addr_odd_rt_depend_temp)) ||
//          ((addr_odd_rc!=7'b0) && (addr_odd_rc == addr_odd_rt_depend_temp))) begin
//          count=opcode_odd_depend_lat-1; $display("1st ODD If - Data Hazard" );
//          end

// else if (((addr_even_ra!=7'bx) && (addr_even_ra == Rt_Temp_odd [7:14])) ||
//          ((addr_even_rb!=7'bx) && (addr_even_rb == Rt_Temp_odd [7:14])) ||
//          ((addr_even_rc!=7'bx) && (addr_even_rc == Rt_Temp_odd [7:14])) ||
//          ((addr_odd_ra!=7'bx) && (addr_odd_ra == Rt_Temp_odd [7:14])) ||
//          ((addr_odd_rb!=7'bx) && (addr_odd_rb == Rt_Temp_odd [7:14])) ||
//          ((addr_odd_rc!=7'bx) && (addr_odd_rc == Rt_Temp_odd [7:14]))) begin
//          count=Rt_Temp_odd[3:5]-2; $display("2st ODD If - Data Hazard"); end

// else if (((addr_even_ra!=7'bx) && (addr_even_ra == Rt_Temp1_odd [7:14])) ||
//          ((addr_even_rb!=7'bx) && (addr_even_rb == Rt_Temp1_odd [7:14])) ||
//          ((addr_even_rc!=7'bx) && (addr_even_rc == Rt_Temp1_odd [7:14])) ||
//          ((addr_odd_ra!=7'bx) && (addr_odd_ra == Rt_Temp1_odd [7:14])) ||
//          ((addr_odd_rb!=7'bx) && (addr_odd_rb == Rt_Temp1_odd [7:14])) ||
//          ((addr_odd_rc!=7'bx) && (addr_odd_rc == Rt_Temp1_odd [7:14]))) begin
//          count=Rt_Temp1_odd[3:5]-3; $display("3nd ODD If - Data Hazard"); end

// else if (((addr_even_ra!=7'bx) && (addr_even_ra == Rt_Temp2_odd [7:14])) ||
//          ((addr_even_rb!=7'bx) && (addr_even_rb == Rt_Temp2_odd [7:14])) ||
//          ((addr_even_rc!=7'bx) && (addr_even_rc == Rt_Temp2_odd [7:14])) ||
//          ((addr_odd_ra!=7'bx) && (addr_odd_ra == Rt_Temp2_odd [7:14])) ||
//          ((addr_odd_rb!=7'bx) && (addr_odd_rb == Rt_Temp2_odd [7:14])) ||
//          ((addr_odd_rc!=7'bx) && (addr_odd_rc == Rt_Temp2_odd [7:14]))) begin
//          count=Rt_Temp2_odd[3:5]-4; $display("4rd ODD If - Data Hazard"); end

// else if (((addr_even_ra!=7'bx) && (addr_even_ra == Rt_Temp3_odd [7:14])) ||
//          ((addr_even_rb!=7'bx) && (addr_even_rb == Rt_Temp3_odd [7:14])) ||
//          ((addr_even_rc!=7'bx) && (addr_even_rc == Rt_Temp3_odd [7:14])) ||
//          ((addr_odd_ra!=7'bx) && (addr_odd_ra == Rt_Temp3_odd [7:14])) ||
//          ((addr_odd_rb!=7'bx) && (addr_odd_rb == Rt_Temp3_odd [7:14])) ||
//          ((addr_odd_rc!=7'bx) && (addr_odd_rc == Rt_Temp3_odd [7:14]))) begin
//          count=Rt_Temp3_odd[3:5]-5; $display("5th ODD If - Data Hazard"); end

// else if (((addr_even_ra!=7'bx) && (addr_even_ra == Rt_Temp4_odd [7:14])) ||
//          ((addr_even_rb!=7'bx) && (addr_even_rb == Rt_Temp4_odd [7:14])) ||
//          ((addr_even_rc!=7'bx) && (addr_even_rc == Rt_Temp4_odd [7:14])) ||
//          ((addr_odd_ra!=7'bx) && (addr_odd_ra == Rt_Temp4_odd [7:14])) ||
//          ((addr_odd_rb!=7'bx) && (addr_odd_rb == Rt_Temp4_odd [7:14])) ||
//          ((addr_odd_rc!=7'bx) && (addr_odd_rc == Rt_Temp4_odd [7:14]))) begin
//          count=Rt_Temp4_odd[3:5]-6; $display("6th ODD If - Data Hazard"); end

// else if (((addr_even_ra!=7'bx) && (addr_even_ra == Rt_Temp5_odd [7:14])) ||
//          ((addr_even_rb!=7'bx) && (addr_even_rb == Rt_Temp5_odd [7:14])) ||
//          ((addr_even_rc!=7'bx) && (addr_even_rc == Rt_Temp5_odd [7:14])) ||
//          ((addr_odd_ra!=7'bx) && (addr_odd_ra == Rt_Temp5_odd [7:14])) ||
//          ((addr_odd_rb!=7'bx) && (addr_odd_rb == Rt_Temp5_odd [7:14])) ||
//          ((addr_odd_rc!=7'bx) && (addr_odd_rc == Rt_Temp5_odd [7:14]))) begin
//          count=Rt_Temp5_odd[3:5]-7; $display("7th ODD If - Data Hazard"); end

// // else if (((addr_even_ra!=7'bx) && (addr_even_ra == Rt_Temp6_odd [7:14])) ||
// //          ((addr_even_rb!=7'bx) && (addr_even_rb == Rt_Temp6_odd [7:14])) ||

```



```

// //      ((addr_even_rc!=7'bx) && (addr_even_rc == Rt_Temp6_odd [7:14])) ||
// //      ((addr_odd_ra!=7'bx) && (addr_odd_ra == Rt_Temp6_odd [7:14])) ||
// //      ((addr_odd_rb!=7'bx) && (addr_odd_rb == Rt_Temp6_odd [7:14])) ||
// //      ((addr_odd_rc!=7'bx) && (addr_odd_rc == Rt_Temp6_odd [7:14])))) begin
// //      count=Rt_Temp6_odd[3:5]-7; $display("7th ODD If - Data Hazard"); end

// else
//     count=0;
// end

// //always_ff@(posedge clk) begin $display("@ DEPENDENCY Rt_Temp_even: %d",Rt_Temp_even[0:2]); end

// endmodule

// // ----- 1st Backup

// // always_ff @(posedge clk) begin
// //     $display("Counter:%d, Count:%d ", counter, count);
// //     if(count == 1) counter <= 1;
// //     else if (count == 2 ) counter <= 2;
// //     else if (count == 3 ) counter <= 3;
// //     else if (count == 4 ) counter <= 4;
// //     else if (count == 5 ) counter <= 5;
// //     else if (count == 6 ) counter <= 6;
// //     else counter <= 0;

// //     if(count == 0 && counter > 0) begin
// //         counter <= counter - 1; //$display("----->>> Data Hazard");
// //         Dependency_stall = 1;
// //         opcode_even_depend = 11'b0100_0000_001;
// //         opcode_odd_depend = 11'b0100_0000_010;
// //         end

// //     else if(count == 0 && counter == 0) begin
// //         //$display("----->>> No Data Hazard, Counter:%d, Count:%d ", counter,
// // count);
// //         $display ("opcode_even: %b | @ Dependency Stge" , opcode_even_depend);
// //         $display ("opcode_odd: %b | @ Dependency Stage", opcode_odd_depend);

// //         Dependency_stall = 0;
// //         addr_even_ra_depend<=addr_even_ra;
// //         addr_even_rb_depend<=addr_even_rb;
// //         addr_even_rc_depend<=addr_even_rc;
// //         opcode_even_depend <= opcode_even; addr_even_rt_depend <= addr_even_rt;
// //         imm7_even_depend <= imm7_even; imm10_even_depend <= imm10_even;
// //         imm16_even_depend <= imm16_even; imm18_even_depend <= imm18_even;

// //         addr_odd_ra_depend<=addr_odd_ra;
// //         addr_odd_rb_depend<=addr_odd_rb;
// //         addr_odd_rc_depend<=addr_odd_rc;
// //         opcode_odd_depend <= opcode_odd; addr_odd_rt_depend <= addr_odd_rt;
// //         imm7_odd_depend <= imm7_odd; imm10_odd_depend <= imm10_odd;
// //         imm16_odd_depend <= imm16_odd; imm18_odd_depend <= imm18_odd;

// // // ----- For Latency Calculation @ Reg File

// //     opcode_even_depend_temp <= opcode_even;
// //     addr_even_rt_depend_temp <= addr_even_rt;
// //     opcode_odd_depend_temp <= opcode_odd;

```

```

// //      addr_odd_rt_depend_temp  <= addr_odd_rt;

// //      end
// // end

// // ----- 2nd Backup

//
//=====Dependency=====
=====

// // module Dependency (clk, reset,addr_even_ra, addr_even_rb, addr_even_rc, opcode_even, imm10_even, imm16_even,
imm18_even, imm7_even, addr_even_rt,
// // addr_odd_ra, addr_odd_rb, addr_odd_rc, opcode_odd, imm10_odd, imm16_odd, imm18_odd, imm7_odd, addr_odd_rt,
flush,opcode_even_depend, imm7_even_depend, imm10_even_depend, imm16_even_depend, imm18_even_depend,
addr_even_rt_depend, addr_even_ra_depend, addr_even_rb_depend, addr_even_rc_depend ,
opcode_odd_depend,imm7_odd_depend,imm10_odd_depend,imm16_odd_depend,imm18_odd_depend,addr_odd_rt_depend,addr_o
dd_ra_depend, addr_odd_rb_depend, addr_odd_rc_depend, Dependency_stall, Rt_Temp_even, Rt_Temp1_even, Rt_Temp2_even,
Rt_Temp3_even, Rt_Temp4_even, Rt_Temp5_even, Rt_Temp6_even, Rt_Temp_odd, Rt_Temp1_odd, Rt_Temp2_odd,
Rt_Temp3_odd, Rt_Temp4_odd, Rt_Temp5_odd, Rt_Temp6_odd);

// // input clk, reset, flush;
// // input [0:6] addr_even_ra, addr_even_rb, addr_even_rc;
// // input [0:6] addr_odd_ra, addr_odd_rb, addr_odd_rc;
// // input [0:10] opcode_even;
// // input signed [0:6] imm7_even;
// // input signed [0:9] imm10_even;
// // input signed [0:15] imm16_even;
// // input signed [0:17] imm18_even;
// // input [0:6] addr_even_rt;
// // input [0:142] Rt_Temp_even, Rt_Temp1_even, Rt_Temp2_even, Rt_Temp3_even, Rt_Temp4_even, Rt_Temp5_even,
Rt_Temp6_even;

// // output logic [0:10] opcode_even_depend;
// // output logic signed [0:6] imm7_even_depend;
// // output logic signed [0:9] imm10_even_depend;
// // output logic signed [0:15] imm16_even_depend;
// // output logic signed [0:17] imm18_even_depend;
// // output logic [0:6] addr_even_rt_depend;
// // output logic [0:6] addr_even_ra_depend, addr_even_rb_depend, addr_even_rc_depend;

// // input [0:10] opcode_odd;
// // input [0:6] imm7_odd;
// // input [0:9] imm10_odd;
// // input [0:15] imm16_odd;
// // input [0:17] imm18_odd;
// // input [0:6] addr_odd_rt;
// // input [0:175] Rt_Temp_odd, Rt_Temp1_odd, Rt_Temp2_odd, Rt_Temp3_odd, Rt_Temp4_odd, Rt_Temp5_odd,
Rt_Temp6_odd;

// // output logic [0:10] opcode_odd_depend;
// // output logic signed [0:6] imm7_odd_depend;
// // output logic signed [0:9] imm10_odd_depend;
// // output logic signed [0:15] imm16_odd_depend;
// // output logic signed [0:17] imm18_odd_depend;
// // output logic signed [0:6] addr_odd_rt_depend;
// // output logic [0:6] addr_odd_ra_depend, addr_odd_rb_depend, addr_odd_rc_depend;

// // logic [0:10] opcode_odd_depend2, opcode_even_depend2 ;

```

```

// // output logic Dependency_stall;

// // logic [0:10] opcode_even_depend_temp, opcode_odd_depend_temp;
// // logic [0:6] addr_even_rt_depend_temp, addr_odd_rt_depend_temp;
// // logic [0:2] opcode_even_depend_lat, opcode_odd_depend_lat;
// // logic [0:2] count, counter;

// // assign Dependency_stall = (counter > 0);

// // always_comb begin
// //     if (Dependency_stall) begin
// //         opcode_even_depend = 11'b0100_0000_001;
// //         opcode_odd_depend = 11'b0100_0000_010; end

// //     else begin
// //         opcode_even_depend = opcode_even_depend2;
// //         opcode_odd_depend = opcode_odd_depend2; end
// // end

// // always_ff @(posedge clk) begin
// //     $display("Counter:%d, Count:%d ", counter, count);

// //     if(counter > 0) begin // count == 0 &&
// //         counter = counter - 1; $display("-----|||||||||----->>> Data Hazard");
// //         //Dependency_stall = 1;
// //         // opcode_even_depend = 11'b0100_0000_001;
// //         // opcode_odd_depend = 11'b0100_0000_010;
// //         end

// //     else if(count == 1) counter <= 0;
// //     else if (count == 2 ) counter <= 1;
// //     else if (count == 3 ) counter <= 2;
// //     else if (count == 4 ) counter <= 3;
// //     else if (count == 5 ) counter <= 4;
// //     else if (count == 6 ) counter <= 5;

// //     else if(count == 0 && counter == 0) begin
// //         // $display("----->>> No Data Hazard, Counter:%d, Count:%d ", counter,
// //         count);
// //         //Dependency_stall = 0;
// //         addr_even_ra_depend<=addr_even_ra;
// //         addr_even_rb_depend<=addr_even_rb;
// //         addr_even_rc_depend<=addr_even_rc;
// //         opcode_even_depend2 <= opcode_even; addr_even_rt_depend <= addr_even_rt;
// //         imm7_even_depend <= imm7_even; imm10_even_depend <= imm10_even;
// //         imm16_even_depend <= imm16_even; imm18_even_depend <= imm18_even;

// //         addr_odd_ra_depend<=addr_odd_ra;
// //         addr_odd_rb_depend<=addr_odd_rb;
// //         addr_odd_rc_depend<=addr_odd_rc;
// //         opcode_odd_depend2 <= opcode_odd; addr_odd_rt_depend <= addr_odd_rt;
// //         imm7_odd_depend <= imm7_odd; imm10_odd_depend <= imm10_odd;
// //         imm16_odd_depend <= imm16_odd; imm18_odd_depend <= imm18_odd;

// //         // ----- For Latency Calculation @ Reg File

// //         opcode_even_depend_temp <= opcode_even;
// //         addr_even_rt_depend_temp <= addr_even_rt;
// //         opcode_odd_depend_temp <= opcode_odd;
// //         addr_odd_rt_depend_temp <= addr_odd_rt;

// //     end

```

```

// // else counter <= 0;

// // $display ("opcode_even: %b | @ Dependency Stge" , opcode_even_depend);
// // $display ("opcode_odd: %b | @ Dependency Stage", opcode_odd_depend);
// // end

// // always_comb begin
// // if (opcode_even_depend_temp == 7'b0100_001lopcode_even_depend_temp ==
8'b0001_1100lopcode_even_depend_temp == 8'b0001_0110lopcode_even_depend_temp == 8'b0111_1110ll
// // opcode_even_depend_temp == 8'b0111_1100lopcode_even_depend_temp ==
8'b0100_1110lopcode_even_depend_temp == 8'b0100_1100lopcode_even_depend_temp == 8'b0101_1110ll
// // opcode_even_depend_temp == 8'b0000_0100lopcode_even_depend_temp ==
8'b0000_1100lopcode_even_depend_temp == 8'b0100_0110lopcode_even_depend_temp == 8'b0100_0100ll
// // opcode_even_depend_temp == 9'b0100_0000_1lopcode_even_depend_temp ==
9'b0100_0001_1lopcode_even_depend_temp == 11'b0001_1000_000lopcode_even_depend_temp == 11'b1101_0000_000ll
// // opcode_even_depend_temp == 11'b0001_1000_001lopcode_even_depend_temp ==
11'b0101_1000_001lopcode_even_depend_temp == 11'b0111_1000_000lopcode_even_depend_temp ==
11'b0111_1010_000ll
// // opcode_even_depend_temp == 11'b0100_1010_000lopcode_even_depend_temp ==
11'b0101_1010_000lopcode_even_depend_temp == 11'b0001_1001_001lopcode_even_depend_temp ==
11'b0000_1001_001ll
// // opcode_even_depend_temp == 11'b0000_1000_001lopcode_even_depend_temp ==
11'b0101_1001_001lopcode_even_depend_temp == 11'b0000_1000_000lopcode_even_depend_temp ==
11'b0110_1000_001ll
// // opcode_even_depend_temp == 11'b0100_1000_001lopcode_even_depend_temp ==
11'b0101_0110_110lopcode_even_depend_temp == 11'b0101_0101_110lopcode_even_depend_temp ==
11'b0101_0100_110)
// // opcode_even_depend_lat = 3'd2;
// // else if(opcode_even_depend_temp == 11'b0000_1111_011ll opcode_even_depend_temp == 11'b0000_1011_000ll
opcode_even_depend_temp == 11'b0000_1011_001lopcode_even_depend_temp == 11'b0000_1011_011ll
// // opcode_even_depend_temp == 11'b0101_0110_100lopcode_even_depend_temp ==
11'b0000_1010_011lopcode_even_depend_temp == 11'b0001_1010_011lopcode_even_depend_temp ==
11'b0100_1010_011)
// // opcode_even_depend_lat = 3'd4;
// // else if(opcode_even_depend_temp == 4'b1110_1lopcode_even_depend_temp == 4'b1111_1lopcode_even_depend_temp
== 11'b0101_1000_100lopcode_even_depend_temp == 11'b0101_1000_110lopcode_even_depend_temp ==
11'b0101_1000_101)
// // opcode_even_depend_lat = 3'd6;
// // else if (opcode_even_depend_temp == 4'b1100_1lopcode_even_depend_temp == 8'b0111_0100ll
opcode_even_depend_temp == 8'b0111_0101lopcode_even_depend_temp == 11'b0111_1000_100ll
opcode_even_depend_temp == 11'b0111_1000_111lopcode_even_depend_temp == 11'b0111_1001_100)
// // opcode_even_depend_lat = 3'd7;
// // else opcode_even_depend_lat = 3'd0;

// // if(opcode_odd_depend_temp == 11'b0011_1011_101lopcode_odd_depend_temp ==
11'b0011_1011_111lopcode_odd_depend_temp == 9'b0011_0001_0lopcode_odd_depend_temp == 9'b0011_0011_0ll
// // opcode_odd_depend_temp == 9'b0011_0010_0lopcode_odd_depend_temp ==
9'b0011_0000_0lopcode_odd_depend_temp == 11'b0011_0101_000lopcode_odd_depend_temp == 11'b0011_0101_001ll
// // opcode_odd_depend_temp == 9'b0010_0001_0lopcode_odd_depend_temp == 9'b0010_0000_0)
// // opcode_odd_depend_lat = 3'd4;
// // else if (opcode_odd_depend_temp == 9'b0011_0000_1lopcode_odd_depend_temp ==
9'b0010_0000_1lopcode_odd_depend_temp == 11'b0011_1000_100lopcode_odd_depend_temp == 11'b0010_1000_100)
// // opcode_odd_depend_lat = 3'd6;
// // else opcode_odd_depend_lat = 3'd0;

// // end // always_comb

```

```

// // always_comb begin

// // $display("===== > addr_even_rb: %d | Rt_Temp_odd:%d",addr_even_rb, Rt_Temp_odd [7:14]);

// // if (((addr_even_ra!=7'b0) && (addr_even_ra == addr_even_rt_depend_temp)) ||
// //      ((addr_even_rb!=7'b0) && (addr_even_rb == addr_even_rt_depend_temp)) ||
// //      ((addr_even_rc!=7'b0) && (addr_even_rc == addr_even_rt_depend_temp)) ||
// //      ((addr_odd_ra!=7'b0) && (addr_odd_ra == addr_even_rt_depend_temp)) ||
// //      ((addr_odd_rb!=7'b0) && (addr_odd_rb == addr_even_rt_depend_temp)) ||
// //      ((addr_odd_rc!=7'b0) && (addr_odd_rc == addr_even_rt_depend_temp)))) begin
// //      count=opcode_even_depend_lat-1; $display("1st EVEN If - Data Hazard");
// //      end

// // else if (((addr_even_ra!=7'b0) && (addr_even_ra == Rt_Temp_even [7:14])) ||
// //          ((addr_even_rb!=7'b0) && (addr_even_rb == Rt_Temp_even [7:14])) ||
// //          ((addr_even_rc!=7'b0) && (addr_even_rc == Rt_Temp_even [7:14])) ||
// //          ((addr_odd_ra!=7'b0) && (addr_odd_ra == Rt_Temp_even [7:14])) ||
// //          ((addr_odd_rb!=7'b0) && (addr_odd_rb == Rt_Temp_even [7:14])) ||
// //          ((addr_odd_rc!=7'b0) && (addr_odd_rc == Rt_Temp_even [7:14]))) begin
// //          count=Rt_Temp_even[3:5]-2; $display("2st EVEN If - Data Hazard");
// //          end

// // else if (((addr_even_ra!=7'bx) && (addr_even_ra == Rt_Temp1_even [7:14])) ||
// //          ((addr_even_rb!=7'bx) && (addr_even_rb == Rt_Temp1_even [7:14])) ||
// //          ((addr_even_rc!=7'bx) && (addr_even_rc == Rt_Temp1_even [7:14])) ||
// //          ((addr_odd_ra!=7'bx) && (addr_odd_ra == Rt_Temp1_even [7:14])) ||
// //          ((addr_odd_rb!=7'bx) && (addr_odd_rb == Rt_Temp1_even [7:14])) ||
// //          ((addr_odd_rc!=7'bx) && (addr_odd_rc == Rt_Temp1_even [7:14]))) begin
// //          count=Rt_Temp1_even[3:5]-3; $display("3nd EVEN If - Data Hazard"); end

// // else if (((addr_even_ra!=7'bx) && (addr_even_ra == Rt_Temp2_even [7:14])) ||
// //          ((addr_even_rb!=7'bx) && (addr_even_rb == Rt_Temp2_even [7:14])) ||
// //          ((addr_even_rc!=7'bx) && (addr_even_rc == Rt_Temp2_even [7:14])) ||
// //          ((addr_odd_ra!=7'bx) && (addr_odd_ra == Rt_Temp2_even [7:14])) ||
// //          ((addr_odd_rb!=7'bx) && (addr_odd_rb == Rt_Temp2_even [7:14])) ||
// //          ((addr_odd_rc!=7'bx) && (addr_odd_rc == Rt_Temp2_even [7:14]))) begin
// //          count=Rt_Temp2_even[3:5]-4; $display("4rd EVEN If - Data Hazard"); end

// // else if (((addr_even_ra!=7'bx) && (addr_even_ra == Rt_Temp3_even [7:14])) ||
// //          ((addr_even_rb!=7'bx) && (addr_even_rb == Rt_Temp3_even [7:14])) ||
// //          ((addr_even_rc!=7'bx) && (addr_even_rc == Rt_Temp3_even [7:14])) ||
// //          ((addr_odd_ra!=7'bx) && (addr_odd_ra == Rt_Temp3_even [7:14])) ||
// //          ((addr_odd_rb!=7'bx) && (addr_odd_rb == Rt_Temp3_even [7:14])) ||
// //          ((addr_odd_rc!=7'bx) && (addr_odd_rc == Rt_Temp3_even [7:14]))) begin
// //          count=Rt_Temp3_even[3:5]-5; $display("5th EVEN If - Data Hazard"); end

// // else if (((addr_even_ra!=7'bx) && (addr_even_ra == Rt_Temp4_even [7:14])) ||
// //          ((addr_even_rb!=7'bx) && (addr_even_rb == Rt_Temp4_even [7:14])) ||
// //          ((addr_even_rc!=7'bx) && (addr_even_rc == Rt_Temp4_even [7:14])) ||
// //          ((addr_odd_ra!=7'bx) && (addr_odd_ra == Rt_Temp4_even [7:14])) ||
// //          ((addr_odd_rb!=7'bx) && (addr_odd_rb == Rt_Temp4_even [7:14])) ||
// //          ((addr_odd_rc!=7'bx) && (addr_odd_rc == Rt_Temp4_even [7:14]))) begin
// //          count=Rt_Temp4_even[3:5]-6; $display("6th EVEN If - Data Hazard"); end

// // else if (((addr_even_ra!=7'bx) && (addr_even_ra == Rt_Temp5_even [7:14])) ||
// //          ((addr_even_rb!=7'bx) && (addr_even_rb == Rt_Temp5_even [7:14])) ||
// //          ((addr_even_rc!=7'bx) && (addr_even_rc == Rt_Temp5_even [7:14])) ||
// //          ((addr_odd_ra!=7'bx) && (addr_odd_ra == Rt_Temp5_even [7:14])) ||
// //          ((addr_odd_rb!=7'bx) && (addr_odd_rb == Rt_Temp5_even [7:14])) ||
// //          ((addr_odd_rc!=7'bx) && (addr_odd_rc == Rt_Temp5_even [7:14]))) begin
// //          count=Rt_Temp5_even[3:5]-7; $display("7th EVEN If - Data Hazard"); end

```

```

// // // else if (((addr_even_ra!=7'bx) && (addr_even_ra == Rt_Temp6_even [7:14])) ||
// // //      ((addr_even_rb!=7'bx) && (addr_even_rb == Rt_Temp6_even [7:14])) ||
// // //      ((addr_even_rc!=7'bx) && (addr_even_rc == Rt_Temp6_even [7:14])) ||
// // //      ((addr_odd_ra!=7'bx) && (addr_odd_ra == Rt_Temp6_even [7:14])) ||
// // //      ((addr_odd_rb!=7'bx) && (addr_odd_rb == Rt_Temp6_even [7:14])) ||
// // //      ((addr_odd_rc!=7'bx) && (addr_odd_rc == Rt_Temp6_even [7:14]))) begin
// // //      count=Rt_Temp6_even[3:5]-7; $display("8th EVEN If - Data Hazard"); end

// // // -----

// // else if (((addr_even_ra!=7'b0) && (addr_even_ra == addr_odd_rt_depend_temp)) ||
// //      ((addr_even_rb!=7'b0) && (addr_even_rb == addr_odd_rt_depend_temp)) ||
// //      ((addr_even_rc!=7'b0) && (addr_even_rc == addr_odd_rt_depend_temp)) ||
// //      ((addr_odd_ra!=7'b0) && (addr_odd_ra == addr_odd_rt_depend_temp)) ||
// //      ((addr_odd_rb!=7'b0) && (addr_odd_rb == addr_odd_rt_depend_temp)) ||
// //      ((addr_odd_rc!=7'b0) && (addr_odd_rc == addr_odd_rt_depend_temp))) begin
// //      count=opcode_odd_depend_lat-1; $display("1st ODD If - Data Hazard" );
// //      end

// // else if (((addr_even_ra!=7'bx) && (addr_even_ra == Rt_Temp_odd [7:14])) ||
// //      ((addr_even_rb!=7'bx) && (addr_even_rb == Rt_Temp_odd [7:14])) ||
// //      ((addr_even_rc!=7'bx) && (addr_even_rc == Rt_Temp_odd [7:14])) ||
// //      ((addr_odd_ra!=7'bx) && (addr_odd_ra == Rt_Temp_odd [7:14])) ||
// //      ((addr_odd_rb!=7'bx) && (addr_odd_rb == Rt_Temp_odd [7:14])) ||
// //      ((addr_odd_rc!=7'bx) && (addr_odd_rc == Rt_Temp_odd [7:14]))) begin
// //      count=Rt_Temp_odd[3:5]-2; $display("2st ODD If - Data Hazard"); end

// // else if (((addr_even_ra!=7'bx) && (addr_even_ra == Rt_Temp1_odd [7:14])) ||
// //      ((addr_even_rb!=7'bx) && (addr_even_rb == Rt_Temp1_odd [7:14])) ||
// //      ((addr_even_rc!=7'bx) && (addr_even_rc == Rt_Temp1_odd [7:14])) ||
// //      ((addr_odd_ra!=7'bx) && (addr_odd_ra == Rt_Temp1_odd [7:14])) ||
// //      ((addr_odd_rb!=7'bx) && (addr_odd_rb == Rt_Temp1_odd [7:14])) ||
// //      ((addr_odd_rc!=7'bx) && (addr_odd_rc == Rt_Temp1_odd [7:14]))) begin
// //      count=Rt_Temp1_odd[3:5]-3; $display("3nd ODD If - Data Hazard"); end

// // else if (((addr_even_ra!=7'bx) && (addr_even_ra == Rt_Temp2_odd [7:14])) ||
// //      ((addr_even_rb!=7'bx) && (addr_even_rb == Rt_Temp2_odd [7:14])) ||
// //      ((addr_even_rc!=7'bx) && (addr_even_rc == Rt_Temp2_odd [7:14])) ||
// //      ((addr_odd_ra!=7'bx) && (addr_odd_ra == Rt_Temp2_odd [7:14])) ||
// //      ((addr_odd_rb!=7'bx) && (addr_odd_rb == Rt_Temp2_odd [7:14])) ||
// //      ((addr_odd_rc!=7'bx) && (addr_odd_rc == Rt_Temp2_odd [7:14]))) begin
// //      count=Rt_Temp2_odd[3:5]-4; $display("4rd ODD If - Data Hazard"); end

// // else if (((addr_even_ra!=7'bx) && (addr_even_ra == Rt_Temp3_odd [7:14])) ||
// //      ((addr_even_rb!=7'bx) && (addr_even_rb == Rt_Temp3_odd [7:14])) ||
// //      ((addr_even_rc!=7'bx) && (addr_even_rc == Rt_Temp3_odd [7:14])) ||
// //      ((addr_odd_ra!=7'bx) && (addr_odd_ra == Rt_Temp3_odd [7:14])) ||
// //      ((addr_odd_rb!=7'bx) && (addr_odd_rb == Rt_Temp3_odd [7:14])) ||
// //      ((addr_odd_rc!=7'bx) && (addr_odd_rc == Rt_Temp3_odd [7:14]))) begin
// //      count=Rt_Temp3_odd[3:5]-5; $display("5th ODD If - Data Hazard"); end

// // else if (((addr_even_ra!=7'bx) && (addr_even_ra == Rt_Temp4_odd [7:14])) ||
// //      ((addr_even_rb!=7'bx) && (addr_even_rb == Rt_Temp4_odd [7:14])) ||
// //      ((addr_even_rc!=7'bx) && (addr_even_rc == Rt_Temp4_odd [7:14])) ||
// //      ((addr_odd_ra!=7'bx) && (addr_odd_ra == Rt_Temp4_odd [7:14])) ||
// //      ((addr_odd_rb!=7'bx) && (addr_odd_rb == Rt_Temp4_odd [7:14])) ||
// //      ((addr_odd_rc!=7'bx) && (addr_odd_rc == Rt_Temp4_odd [7:14]))) begin
// //      count=Rt_Temp4_odd[3:5]-6; $display("6th ODD If - Data Hazard"); end

// // else if (((addr_even_ra!=7'bx) && (addr_even_ra == Rt_Temp5_odd [7:14])) ||
// //      ((addr_even_rb!=7'bx) && (addr_even_rb == Rt_Temp5_odd [7:14])) ||
// //      ((addr_even_rc!=7'bx) && (addr_even_rc == Rt_Temp5_odd [7:14])) ||
// //      ((addr_odd_ra!=7'bx) && (addr_odd_ra == Rt_Temp5_odd [7:14])) ||

```

```

// //      ((addr_odd_rb!=7'bx) && (addr_odd_rb == Rt_Temp5_odd [7:14])) ||
// //      ((addr_odd_rc!=7'bx) && (addr_odd_rc == Rt_Temp5_odd [7:14])))) begin
// //      count=Rt_Temp5_odd[3:5]-7; $display("7th ODD If - Data Hazard"); end

// // // else if (((addr_even_ra!=7'bx) && (addr_even_ra == Rt_Temp6_odd [7:14])) ||
// // //      ((addr_even_rb!=7'bx) && (addr_even_rb == Rt_Temp6_odd [7:14])) ||
// // //      ((addr_even_rc!=7'bx) && (addr_even_rc == Rt_Temp6_odd [7:14])) ||
// // //      ((addr_odd_ra!=7'bx) && (addr_odd_ra == Rt_Temp6_odd [7:14])) ||
// // //      ((addr_odd_rb!=7'bx) && (addr_odd_rb == Rt_Temp6_odd [7:14])) ||
// // //      ((addr_odd_rc!=7'bx) && (addr_odd_rc == Rt_Temp6_odd [7:14])))) begin
// // //      count=Rt_Temp6_odd[3:5]-7; $display("7th ODD If - Data Hazard"); end

// // else
// //      count=0;
// // end

// // //always_ff@(posedge clk) begin $display("@ DEPENDENCY Rt_Temp_even: %d",Rt_Temp_even[0:2]); end

// // endmodule

```

Script: Register File

```

//===== Register File
=====

module reg_file (clk, reset, Pc_in, addr_even_ra, addr_even_rb, addr_even_rc, opcode_even, imm10_even, imm16_even,
imm18_even, imm7_even, addr_even_rt,
addr_odd_ra, addr_odd_rb, addr_odd_rc, opcode_odd, imm10_odd, imm16_odd, imm18_odd, imm7_odd, addr_odd_rt,
flush,Rt_Temp_even_Rout, Rt_Temp1_even_Rout, Rt_Temp2_even_Rout, Rt_Temp3_even_Rout, Rt_Temp4_even_Rout,
Rt_Temp5_even_Rout, Rt_Temp6_even_Rout,Rt_Temp_odd_Rout, Rt_Temp1_odd_Rout, Rt_Temp2_odd_Rout,
Rt_Temp3_odd_Rout, Rt_Temp4_odd_Rout, Rt_Temp5_odd_Rout, Rt_Temp6_odd_Rout, Even_Test_Packet, Odd_Test_Packet);

    // Input to REG File for Processing
    input clk, reset;
    input [0:6] addr_even_ra, addr_even_rb, addr_even_rc;
    input [0:6] addr_odd_ra, addr_odd_rb, addr_odd_rc;
    output logic [0:136] Even_Test_Packet;    //Packet Data Sent to Testbench for Verication Purpose
    output logic [0:168] Odd_Test_Packet;    //Packet Data Sent to Testbench for Verication Purpose

    // Even Pipe Forwarding and Data Stuff
    input [0:10] opcode_even;    // Input from TB
    input signed [0:6] imm7_even;    // logic from TB
    input signed [0:9] imm10_even;    // logic from TB
    input signed [0:15] imm16_even;    // logic from TB
    input signed [0:17] imm18_even;    // logic from TB
    input [0:6] addr_even_rt;    // Input from TB
    logic [0:6] addr_even_rt2;    // Receiving from Even Pipe
    logic signed [0:127] data_even_rt;    // Receiving from Even Pipe
    logic wr_en_even;    // Receiving from Even Pipe
    logic [0:142]Rt_Temp_even, Rt_Temp1_even, Rt_Temp2_even, Rt_Temp3_even, Rt_Temp4_even, Rt_Temp5_even,
Rt_Temp6_even;
    output logic [0:142]Rt_Temp_even_Rout, Rt_Temp1_even_Rout, Rt_Temp2_even_Rout, Rt_Temp3_even_Rout,
Rt_Temp4_even_Rout, Rt_Temp5_even_Rout, Rt_Temp6_even_Rout;

    logic [0:142] Rt_Temp_even_Test;

    logic signed [0:127] data_even_ra, data_even_rb, data_even_rc; // Data to Even Pipe
    logic signed [0:127] data_even_ra1,
data_even_ra2,data_even_ra3,data_even_ra4,data_even_ra5,data_even_ra6,data_even_ra_new;

```

```

logic signed [0:127] data_even_rb1,
data_even_rb2,data_even_rb3,data_even_rb4,data_even_rb5,data_even_rb6,data_even_rb_new;
logic signed [0:127] data_even_rc1,
data_even_rc2,data_even_rc3,data_even_rc4,data_even_rc5,data_even_rc6,data_even_rc_new;
logic [0:10] opcode_even_fwd;      // Forward Reg for 1 Cycle Delay
logic signed [0:6] imm7_even_fwd;  // Forward Reg for 1 Cycle Delay
logic signed [0:9] imm10_even_fwd; // Forward Reg for 1 Cycle Delay
logic signed [0:15] imm16_even_fwd; // Forward Reg for 1 Cycle Delay
logic signed [0:17] imm18_even_fwd; // Forward Reg for 1 Cycle Delay
logic [0:6] addr_even_rt_fwd;      // Forward Reg for 1 Cycle Delay

// Odd Pipe Forwarding and Data Stuff
input flush;
input [0:10] opcode_odd;           // Input from TB
input [0:6] imm7_odd;              // logic from TB
input [0:9] imm10_odd;             // logic from TB
input [0:15] imm16_odd;            // logic from TB
input [0:17] imm18_odd;            // logic from TB
input [0:6] addr_odd_rt;           // Input from TB
input [0:14] Pc_in;                // Input from TB
logic [0:6] addr_odd_rt2;          // Receiving from odd Pipe
logic [0:127] data_odd_rt;         // Receiving from odd Pipe
logic [0:14] Pc_out_2;
logic wr_en_odd, flush_fwd;        // Receiving from odd Pipe

output logic [0:175]Rt_Temp_odd_Rout, Rt_Temp1_odd_Rout, Rt_Temp2_odd_Rout, Rt_Temp3_odd_Rout,
Rt_Temp4_odd_Rout, Rt_Temp5_odd_Rout, Rt_Temp6_odd_Rout;
logic [0:175]Rt_Temp_odd, Rt_Temp1_odd, Rt_Temp2_odd, Rt_Temp3_odd, Rt_Temp4_odd, Rt_Temp5_odd,
Rt_Temp6_odd;

logic [0:127] data_odd_ra, data_odd_rb, data_odd_rc; // Data to odd Pipe
logic [0:10] opcode_odd_fwd;      // Forward Reg for 1 Cycle Delay
logic [0:6] imm7_odd_fwd;         // Forward Reg for 1 Cycle Delay
logic [0:9] imm10_odd_fwd;        // Forward Reg for 1 Cycle Delay
logic [0:15] imm16_odd_fwd;       // Forward Reg for 1 Cycle Delay
logic [0:17] imm18_odd_fwd;       // Forward Reg for 1 Cycle Delay
logic [0:6] addr_odd_rt_fwd;      // Forward Reg for 1 Cycle Delay
logic [0:14] Pc_in_fwd;           // Forward Reg for 1 Cycle Delay

// Odd and Even Pipe Connections

evenpipe evenp ( .clk(clk),                // Input
.reset(reset),                            // Input
.opcode_even(opcode_even_fwd),             // Input
.imm7_even(imm7_even_fwd),                 // Input
.imm10_even(imm10_even_fwd),               // Input
.imm16_even(imm16_even_fwd),               // Input
.imm18_even(imm18_even_fwd),               // Input
.data_even_ra(data_even_ra),               // Input
.data_even_rb(data_even_rb),               // Input
.data_even_rc(data_even_rc),               // Input
.addr_even_rt(addr_even_rt_fwd),            // Input
.addr_even_rt2(addr_even_rt2),             // Output Receiving
.data_even_rt(data_even_rt),               // Output Receiving
.wr_en_even(wr_en_even),                   // Output Receiving
.Rt_Temp(Rt_Temp_even), // Rt_Temp_even

.Rt_Temp1(Rt_Temp1_even),
.Rt_Temp2(Rt_Temp2_even),
.Rt_Temp3(Rt_Temp3_even),
.Rt_Temp4(Rt_Temp4_even),
.Rt_Temp5(Rt_Temp5_even),

```



```

        .Rt_Temp6(Rt_Temp6_even));

oddpipeline oddp (    .clk(clk),                // Input
                    .reset(reset),              // Input
                    .opcode_odd(opcode_odd_fwd), // Input
                    .imm7_odd(imm7_odd_fwd),    // Input
                    .imm10_odd(imm10_odd_fwd),   // Input
                    .imm16_odd(imm16_odd_fwd),   // Input
                    .imm18_odd(imm18_odd_fwd),   // Input
                    .data_odd_ra(data_odd_ra),   // Input
                    .data_odd_rb(data_odd_rb),   // Input
                    .data_odd_rc(data_odd_rc),   // Input
                    .addr_odd_rt(addr_odd_rt_fwd), // Input
                    .addr_odd_rt2(addr_odd_rt2), //Output Receiving
                    .data_odd_rt(data_odd_rt),   //Output Receiving
                    .wr_en_odd(wr_en_odd),       //Output Receiving
                    .Pc_in(Pc_in_fwd),          //Output Receiving
                    .Pc_out_2(Pc_out_2),         //Output Receiving
                    .flush(flush_fwd),          // Flush Signal from TB
                    .Rt_Temp0_fwd(Rt_Temp_odd),
                    .Rt_Temp1(Rt_Temp1_odd),
                    .Rt_Temp2(Rt_Temp2_odd),
                    .Rt_Temp3(Rt_Temp3_odd),
                    .Rt_Temp4(Rt_Temp4_odd),
                    .Rt_Temp5(Rt_Temp5_odd),
                    .Rt_Temp6(Rt_Temp6_odd));

always_comb begin
    Rt_Temp_odd_Rout  = Rt_Temp_odd;
    Rt_Temp1_odd_Rout = Rt_Temp1_odd;
    Rt_Temp2_odd_Rout = Rt_Temp2_odd;
    Rt_Temp3_odd_Rout = Rt_Temp3_odd;
    Rt_Temp4_odd_Rout = Rt_Temp4_odd;
    Rt_Temp5_odd_Rout = Rt_Temp5_odd;
    Rt_Temp6_odd_Rout = Rt_Temp6_odd;

    Rt_Temp_even_Rout = Rt_Temp_even;
    Rt_Temp1_even_Rout = Rt_Temp1_even;
    Rt_Temp2_even_Rout = Rt_Temp2_even;
    Rt_Temp3_even_Rout = Rt_Temp3_even;
    Rt_Temp4_even_Rout = Rt_Temp4_even;
    Rt_Temp5_even_Rout = Rt_Temp5_even;
    Rt_Temp6_even_Rout = Rt_Temp6_even;
end


//always_ff @(posedge clk) begin $display("@REG FILE: %d", Rt_Temp_even_Rout[0:2]); $display("@REG 2 FILE: %d",
Rt_Temp_even[0:2]); end

logic [0:127][0:127] mem;

assign Even_Test_Packet = {data_even_rt, addr_even_rt2, wr_en_even};
assign Odd_Test_Packet = {data_odd_rt, addr_odd_rt2, wr_en_odd, Pc_out_2};

always_ff @(posedge clk) begin
    if(reset) begin
        integer i;
        for (i=0; i<124; i=i+1)
            mem[i] = i;
        mem[124]=32'h408ccccd;
        mem[125]=128'd79228162532711081671548469249;
        mem[126]=128'd170808406006153739902585569290863280256;
    end
end

```

 Stony Brook University

```

data_odd_rb <= mem[addr_odd_rb];
data_odd_rc <= mem[addr_odd_rc]; end

else if (Rt_Temp1_even [7:14] == addr_odd_ra && Rt_Temp1_even[3:5] == 2) begin // Even vs Odd
    data_odd_ra <= Rt_Temp1_even [15:142];
    data_odd_rb <= mem[addr_odd_rb];
    data_odd_rc <= mem[addr_odd_rc];

    data_even_ra <=mem[addr_even_ra];
    data_even_rb <= mem[addr_even_rb];
    data_even_rc <= mem[addr_even_rc]; end

else if(Rt_Temp1_even [7:14] == addr_odd_rb && Rt_Temp1_even[3:5] == 2) begin // Even vs Odd
    data_odd_ra <= mem[addr_odd_ra];
    data_odd_rb <= Rt_Temp1_even [15:142];
    data_odd_rc <= mem[addr_odd_rc];

    data_even_ra <=mem[addr_even_ra];
    data_even_rb <= mem[addr_even_rb];
    data_even_rc <= mem[addr_even_rc]; end

else if (Rt_Temp1_even [7:14] == addr_odd_rc && Rt_Temp1_even[3:5] == 2) begin // Even vs Odd
    data_odd_ra <= mem[addr_odd_ra];
    data_odd_rb <= mem[addr_odd_rb];
    data_odd_rc <= Rt_Temp1_even [15:142];

    data_even_ra <=mem[addr_even_ra];
    data_even_rb <= mem[addr_even_rb];
    data_even_rc <= mem[addr_even_rc];end

// -----

else if ( Rt_Temp2_even [7:14] == addr_even_ra && Rt_Temp2_even[3:5] == 3) begin // Even vs Even
    data_even_ra <= Rt_Temp1_even [15:142];
    data_even_rb <= mem[addr_even_rb];
    data_even_rc <= mem[addr_even_rc];

    data_odd_ra <=mem[addr_odd_ra];
    data_odd_rb <= mem[addr_odd_rb];
    data_odd_rc <= mem[addr_odd_rc];end

else if( Rt_Temp2_even [7:14] == addr_even_rb && Rt_Temp2_even[3:5] == 3) begin // Even vs Even
    data_even_ra <= mem[addr_even_ra];
    data_even_rb <= Rt_Temp1_even [15:142];
    data_even_rc <= mem[addr_even_rc];

    data_odd_ra <=mem[addr_odd_ra];
    data_odd_rb <= mem[addr_odd_rb];
    data_odd_rc <= mem[addr_odd_rc]; end

else if ( Rt_Temp2_even [7:14] == addr_even_rc && Rt_Temp2_even[3:5] == 3) begin // Even vs Even
    data_even_ra <=mem[addr_even_ra];
    data_even_rb <= mem[addr_even_rb];
    data_even_rc <= Rt_Temp1_even [15:142];

    data_odd_ra <=mem[addr_odd_ra];
    data_odd_rb <= mem[addr_odd_rb];
    data_odd_rc <= mem[addr_odd_rc]; end

else if (Rt_Temp2_even [7:14] == addr_odd_ra && Rt_Temp2_even[3:5] == 3) begin // Even vs Odd
    data_odd_ra <= Rt_Temp1_even [15:142];
    data_odd_rb <= mem[addr_odd_rb];
    data_odd_rc <= mem[addr_odd_rc];

```

```

data_even_ra <= mem[addr_even_ra];
data_even_rb <= mem[addr_even_rb];
data_even_rc <= mem[addr_even_rc]; end

else if( Rt_Temp2_even [7:14] == addr_odd_rb && Rt_Temp2_even[3:5] == 3) begin // Even vs Odd
data_odd_ra <= mem[addr_odd_ra];
data_odd_rb <= Rt_Temp1_even [15:142];
data_odd_rc <= mem[addr_odd_rc];

data_even_ra <= mem[addr_even_ra];
data_even_rb <= mem[addr_even_rb];
data_even_rc <= mem[addr_even_rc]; end

else if ( Rt_Temp2_even [7:14] == addr_odd_rc && Rt_Temp2_even[3:5] == 3) begin // Even vs Odd
data_odd_ra <= mem[addr_odd_ra];
data_odd_rb <= mem[addr_odd_rb];
data_odd_rc <= Rt_Temp1_even [15:142];

data_even_ra <= mem[addr_even_ra];
data_even_rb <= mem[addr_even_rb];
data_even_rc <= mem[addr_even_rc];end

//latency=4,stage=4
else if (Rt_Temp3_even [7:14] == addr_even_ra && Rt_Temp3_even[3:5] == 4) begin // Even vs Even
data_even_ra <= Rt_Temp3_even [15:142];
data_even_rb <= mem[addr_even_rb];
data_even_rc <= mem[addr_even_rc];

data_odd_ra <= mem[addr_odd_ra];
data_odd_rb <= mem[addr_odd_rb];
data_odd_rc <= mem[addr_odd_rc];end

else if(Rt_Temp3_even [7:14] == addr_even_rb && Rt_Temp3_even[3:5] == 4) begin // Even vs Even
data_even_ra <= mem[addr_even_ra];
data_even_rb <= Rt_Temp3_even [15:142];
data_even_rc <= mem[addr_even_rc];

data_odd_ra <= mem[addr_odd_ra];
data_odd_rb <= mem[addr_odd_rb];
data_odd_rc <= mem[addr_odd_rc]; end

else if (Rt_Temp3_even [7:14] == addr_even_rc && Rt_Temp3_even[3:5] == 4) begin // Even vs Even
data_even_ra <= mem[addr_even_ra];
data_even_rb <= mem[addr_even_rb];
data_even_rc <= Rt_Temp3_even [15:142];

data_odd_ra <= mem[addr_odd_ra];
data_odd_rb <= mem[addr_odd_rb];
data_odd_rc <= mem[addr_odd_rc]; end

else if (Rt_Temp3_even [7:14] == addr_odd_ra && Rt_Temp3_even[3:5] == 4) begin // Even vs Odd
data_odd_ra <= Rt_Temp3_even [15:142];
data_odd_rb <= mem[addr_odd_rb];
data_odd_rc <= mem[addr_odd_rc];

data_even_ra <= mem[addr_even_ra];
data_even_rb <= mem[addr_even_rb];
data_even_rc <= mem[addr_even_rc]; end

else if(Rt_Temp3_even [7:14] == addr_odd_rb && Rt_Temp3_even[3:5] == 4) begin // Even vs Odd
data_odd_ra <= mem[addr_odd_ra];
data_odd_rb <= Rt_Temp3_even [15:142];

```

```

data_odd_rc <= mem[addr_odd_rc];

data_even_ra <= mem[addr_even_ra];
data_even_rb <= mem[addr_even_rb];
data_even_rc <= mem[addr_even_rc]; end

else if (Rt_Temp3_even [7:14] == addr_odd_rc && Rt_Temp3_even[3:5] == 4) begin // Even vs Odd
data_odd_ra <= mem[addr_odd_ra];
data_odd_rb <= mem[addr_odd_rb];
data_odd_rc <= Rt_Temp3_even [15:142];

data_even_ra <= mem[addr_even_ra];
data_even_rb <= mem[addr_even_rb];
data_even_rc <= mem[addr_even_rc]; end

else if (Rt_Temp4_even [7:14] == addr_even_ra && Rt_Temp4_even[3:5] == 5) begin // Even vs Even
data_even_ra <= Rt_Temp4_even [15:142];
data_even_rb <= mem[addr_even_rb];
data_even_rc <= mem[addr_even_rc];

data_odd_ra <= mem[addr_odd_ra];
data_odd_rb <= mem[addr_odd_rb];
data_odd_rc <= mem[addr_odd_rc]; end

else if (Rt_Temp4_even [7:14] == addr_even_rb && Rt_Temp4_even[3:5] == 5) begin // Even vs Even
data_even_ra <= mem[addr_even_ra];
data_even_rb <= Rt_Temp4_even [15:142];
data_even_rc <= mem[addr_even_rc];

data_odd_ra <= mem[addr_odd_ra];
data_odd_rb <= mem[addr_odd_rb];
data_odd_rc <= mem[addr_odd_rc]; end

else if (Rt_Temp4_even [7:14] == addr_even_rc && Rt_Temp4_even[3:5] == 5) begin // Even vs Even
data_even_ra <= mem[addr_even_ra];
data_even_rb <= mem[addr_even_rb];
data_even_rc <= Rt_Temp4_even [15:142];

data_odd_ra <= mem[addr_odd_ra];
data_odd_rb <= mem[addr_odd_rb];
data_odd_rc <= mem[addr_odd_rc]; end

else if (Rt_Temp4_even [7:14] == addr_odd_ra && Rt_Temp4_even[3:5] == 5) begin // Even vs Odd
data_odd_ra <= Rt_Temp4_even [15:142];
data_odd_rb <= mem[addr_odd_rb];
data_odd_rc <= mem[addr_odd_rc];

data_even_ra <= mem[addr_even_ra];
data_even_rb <= mem[addr_even_rb];
data_even_rc <= mem[addr_even_rc]; end

else if (Rt_Temp4_even [7:14] == addr_odd_rb && Rt_Temp4_even[3:5] == 5) begin // Even vs Odd
data_odd_ra <= mem[addr_odd_ra];
data_odd_rb <= Rt_Temp4_even [15:142];
data_odd_rc <= mem[addr_odd_rc];

data_even_ra <= mem[addr_even_ra];
data_even_rb <= mem[addr_even_rb];
data_even_rc <= mem[addr_even_rc]; end

else if (Rt_Temp4_even [7:14] == addr_odd_rc && Rt_Temp4_even[3:5] == 5) begin // Even vs Odd
data_odd_ra <= mem[addr_odd_ra];

```

```

data_odd_rb <= mem[addr_odd_rb];
data_odd_rc <= Rt_Temp4_even [15:142];

data_even_ra <=mem[addr_even_ra];
data_even_rb <= mem[addr_even_rb];
data_even_rc <= mem[addr_even_rc];end

else if (Rt_Temp5_even [7:14] == addr_even_ra && Rt_Temp5_even[3:5] == 6) begin    // Even vs Even
    data_even_ra <= Rt_Temp5_even [15:142];
    data_even_rb <= mem[addr_even_rb];
    data_even_rc <= mem[addr_even_rc];

    data_odd_ra <=mem[addr_odd_ra];
    data_odd_rb <= mem[addr_odd_rb];
    data_odd_rc <= mem[addr_odd_rc];end

else if(Rt_Temp5_even [7:14] == addr_even_rb && Rt_Temp5_even[3:5] == 6) begin // Even vs Even
    data_even_ra <= mem[addr_even_ra];
    data_even_rb <= Rt_Temp5_even [15:142];
    data_even_rc <= mem[addr_even_rc];

    data_odd_ra <=mem[addr_odd_ra];
    data_odd_rb <= mem[addr_odd_rb];
    data_odd_rc <= mem[addr_odd_rc]; end

else if (Rt_Temp5_even [7:14] == addr_even_rc && Rt_Temp5_even[3:5] == 6) begin // Even vs Even
    data_even_ra <=mem[addr_even_ra];
    data_even_rb <= mem[addr_even_rb];
    data_even_rc <= Rt_Temp5_even [15:142];

    data_odd_ra <=mem[addr_odd_ra];
    data_odd_rb <= mem[addr_odd_rb];
    data_odd_rc <= mem[addr_odd_rc]; end

else if (Rt_Temp5_even [7:14] == addr_odd_ra && Rt_Temp5_even[3:5] == 6) begin // Even vs Odd
    data_odd_ra <= Rt_Temp5_even [15:142];
    data_odd_rb <= mem[addr_odd_rb];
    data_odd_rc <= mem[addr_odd_rc];

    data_even_ra <=mem[addr_even_ra];
    data_even_rb <= mem[addr_even_rb];
    data_even_rc <= mem[addr_even_rc]; end

else if(Rt_Temp5_even [7:14] == addr_odd_rb && Rt_Temp5_even[3:5] == 6) begin // Even vs Odd
    data_odd_ra <= mem[addr_odd_ra];
    data_odd_rb <= Rt_Temp5_even [15:142];
    data_odd_rc <= mem[addr_odd_rc];

    data_even_ra <=mem[addr_even_ra];
    data_even_rb <= mem[addr_even_rb];
    data_even_rc <= mem[addr_even_rc]; end

else if (Rt_Temp5_even [7:14] == addr_odd_rc && Rt_Temp5_even[3:5] == 6) begin // Even vs Odd
    data_odd_ra <= mem[addr_odd_ra];
    data_odd_rb <= mem[addr_odd_rb];
    data_odd_rc <= Rt_Temp5_even [15:142];

    data_even_ra <=mem[addr_even_ra];
    data_even_rb <= mem[addr_even_rb];
    data_even_rc <= mem[addr_even_rc];end

```

```

else if (Rt_Temp6_even [7:14] == addr_even_ra && Rt_Temp6_even[3:5] == 7) begin    // Even vs Even
    data_even_ra <= Rt_Temp6_even [15:142];
    data_even_rb <= mem[addr_even_rb];
    data_even_rc <= mem[addr_even_rc];

    data_odd_ra <=mem[addr_odd_ra];
    data_odd_rb <= mem[addr_odd_rb];
    data_odd_rc <= mem[addr_odd_rc];end

else if(Rt_Temp6_even [7:14] == addr_even_rb && Rt_Temp6_even[3:5] == 7) begin // Even vs Even
    data_even_ra <= mem[addr_even_ra];
    data_even_rb <= Rt_Temp6_even [15:142];
    data_even_rc <= mem[addr_even_rc];

    data_odd_ra <=mem[addr_odd_ra];
    data_odd_rb <= mem[addr_odd_rb];
    data_odd_rc <= mem[addr_odd_rc]; end

else if (Rt_Temp6_even [7:14] == addr_even_rc && Rt_Temp6_even[3:5] == 7) begin  // Even vs Even
    data_even_ra <=mem[addr_even_ra];
    data_even_rb <= mem[addr_even_rb];
    data_even_rc <= Rt_Temp6_even [15:142];

    data_odd_ra <=mem[addr_odd_ra];
    data_odd_rb <= mem[addr_odd_rb];
    data_odd_rc <= mem[addr_odd_rc]; end

else if (Rt_Temp6_even [7:14] == addr_odd_ra && Rt_Temp6_even[3:5] == 7) begin // Even vs Odd
    data_odd_ra <= Rt_Temp6_even [15:142];
    data_odd_rb <= mem[addr_odd_rb];
    data_odd_rc <= mem[addr_odd_rc];

    data_even_ra <=mem[addr_even_ra];
    data_even_rb <= mem[addr_even_rb];
    data_even_rc <= mem[addr_even_rc]; end

else if(Rt_Temp6_even [7:14] == addr_odd_rb && Rt_Temp6_even[3:5] == 7) begin  // Even vs Odd
    data_odd_ra <= mem[addr_odd_ra];
    data_odd_rb <= Rt_Temp6_even [15:142];
    data_odd_rc <= mem[addr_odd_rc];

    data_even_ra <=mem[addr_even_ra];
    data_even_rb <= mem[addr_even_rb];
    data_even_rc <= mem[addr_even_rc]; end

else if (Rt_Temp6_even [7:14] == addr_odd_rc && Rt_Temp6_even[3:5] == 7) begin // Even vs Odd
    data_odd_ra <= mem[addr_odd_ra];
    data_odd_rb <= mem[addr_odd_rb];
    data_odd_rc <= Rt_Temp6_even [15:142];

    data_even_ra <=mem[addr_even_ra];
    data_even_rb <= mem[addr_even_rb];
    data_even_rc <= mem[addr_even_rc];end

else begin

    data_even_ra <= mem[addr_even_ra];
    data_even_rb <= mem[addr_even_rb];
    data_even_rc <= mem[addr_even_rc];

    data_odd_ra <= mem[addr_odd_ra];
    data_odd_rb <= mem[addr_odd_rb];
    data_odd_rc <= mem[addr_odd_rc];

```

```

end

end
always_ff @(posedge clk) $display("Opcode:%b, data_even_ra:%d, data_even_rb:%d, data_even_rc:%d",opcode_even,
data_even_ra, data_even_rb, data_even_rc);

//end
always_ff @(posedge clk) begin      // Instruction Forward to Pipes
    opcode_even_fwd <= opcode_even; addr_even_rt_fwd <= addr_even_rt;
    imm7_even_fwd <= imm7_even; imm10_even_fwd <= imm10_even;
    imm16_even_fwd <= imm16_even; imm18_even_fwd <= imm18_even;

    opcode_odd_fwd <= opcode_odd; addr_odd_rt_fwd <= addr_odd_rt;
    imm7_odd_fwd <= imm7_odd; imm10_odd_fwd <= imm10_odd;
    imm16_odd_fwd <= imm16_odd; imm18_odd_fwd <= imm18_odd;
    Pc_in_fwd <= Pc_in; flush_fwd <= flush;
end

//always_comb $display ("=====>>> unit_temp:%d , latency_temp:%d , wr_en:%d , addr_even_rt:%d",
Rt_Temp_even[0:2], Rt_Temp_even[3:5], Rt_Temp_even[6], Rt_Temp_even[7:14]);

endmodule

```

Script: Even Pipe

```

//===== PARAMETERIZATION
=====

parameter SIMPLE_FIXED1_UNIT      =1;    parameter SIMPLE_FIXED1_LATENCY    =2;
parameter SIMPLE_FIXED2_UNIT      =2;    parameter SIMPLE_FIXED2_LATENCY    =4;
parameter SINGLE_PRECISION1_UNIT  =3;    parameter SINGLE_PRECISION1_LATENCY =6;
parameter SINGLE_PRECISION2_UNIT  =3;    parameter SINGLE_PRECISION2_LATENCY =7;
parameter BYTE_UNIT               =4;    parameter BYTE_LATENCY              =4;
parameter PERMUTE_UNIT            =5;    parameter PERMUTE_LATENCY           =4;
parameter LOCALSTORE_UNIT         =6;    parameter LOCALSTORE_LATENCY        =6;
parameter BRANCH_UNIT             =7;    parameter BRANCH_LATENCY            =4;
//===== EVEN PIPE
=====

module evenpipe(clk, reset, opcode_even, data_even_ra, data_even_rb, data_even_rc, imm7_even, imm10_even, imm16_even,
imm18_even, addr_even_rt, data_even_rt, addr_even_rt2, wr_en_even, Rt_Temp, Rt_Temp1, Rt_Temp2, Rt_Temp3, Rt_Temp4,
Rt_Temp5, Rt_Temp6);

input clk, reset;
input [0:6] addr_even_rt;
input [0:10] opcode_even;
input signed [0:6] imm7_even;      // logic from TB
input signed [0:9] imm10_even;     // logic from TB
input signed [0:15] imm16_even;    // logic from TB
input signed [0:17] imm18_even;    // logic from TB
input signed [0:127] data_even_ra, data_even_rb, data_even_rc;
output logic signed [0:127] data_even_rt;
output logic [0:6] addr_even_rt2;
output logic wr_en_even;

logic firstbit, wr_en, stop;
output logic [0:142] Rt_Temp,Rt_Temp1, Rt_Temp2, Rt_Temp3, Rt_Temp4, Rt_Temp5, Rt_Temp6;

logic [0:142] Rt_Temp7,Rt_Temp0_fwd;
logic signed [0:31]temp_imm10, temp_imm;
logic signed [0:127] Rt;
logic [0:2] count;

```



```

logic [0:2] latency_temp, unit_temp, latency, unit;
logic [0:7] b, ra_byte;
logic [0:31] bbbb, rb_word;
logic [0:63] shift_count;
logic [0:127] rt_temp;

logic [0:142] Rt_test_even;

always_comb begin
    case(opcode_even)

//1. Add Word
        11'b000110000000:begin
            Rt[0:31]=data_even_ra[0:31]+data_even_rb[0:31];
            Rt[32:63]=data_even_ra[32:63]+data_even_rb[32:63];
            Rt[64:95]=data_even_ra[64:95]+data_even_rb[64:95];
            Rt[96:127]=data_even_ra[96:127]+data_even_rb[96:127];
            wr_en = 1; unit_temp = SIMPLE_FIXED1_UNIT; latency_temp = SIMPLE_FIXED1_LATENCY;
        end

//2.Add extended word
        11'b011010000000:begin
            for(int i=0; i<4; i=i+1)
                Rt[(i*32)+: 32]=data_even_ra[(i*32)+: 32] + data_even_rb[(i*32)+: 32] + data_even_rc[(i*32)+: 32];
            wr_en = 1; unit_temp = SIMPLE_FIXED1_UNIT; latency_temp = SIMPLE_FIXED1_LATENCY;
        end

//3. Add word immediate
        11'b00011100:begin
            firstbit = imm10_even[0];
            temp_imm10 = {{22{firstbit}},imm10_even};
            Rt[0:31]=data_even_ra[0:31]+temp_imm10;
            Rt[32:63]=data_even_ra[32:63]+temp_imm10;
            Rt[64:95]=data_even_ra[64:95]+temp_imm10;
            Rt[96:127]=data_even_ra[96:127]+temp_imm10;
            wr_en = 1; unit_temp = SIMPLE_FIXED1_UNIT; latency_temp = SIMPLE_FIXED1_LATENCY;
        end

//4. And
        11'b000110000001:begin
            Rt[0:31]=data_even_ra[0:31] & data_even_rb[0:31];
            Rt[32:63]=data_even_ra[32:63] & data_even_rb[32:63];
            Rt[64:95]=data_even_ra[64:95] & data_even_rb[64:95];
            Rt[96:127]=data_even_ra[96:127] & data_even_rb[96:127];
            wr_en = 1; unit_temp = SIMPLE_FIXED1_UNIT; latency_temp = SIMPLE_FIXED1_LATENCY;
        end

//5. And Byte Imm
        11'b00010110:begin
            b = imm10_even & 8'hff;
            bbbb = {b,b,b,b};
            Rt[0:31]=data_even_ra[0:31] & bbbb;
            Rt[32:63]=data_even_ra[32:63] & bbbb;
            Rt[64:95]=data_even_ra[64:95] & bbbb;
            Rt[96:127]=data_even_ra[96:127] & bbbb;
            wr_en = 1; unit_temp = SIMPLE_FIXED1_UNIT; latency_temp = SIMPLE_FIXED1_LATENCY;
        end

//6. And with Complement
        11'b010110000001:begin
            Rt[0:31]=data_even_ra[0:31] & ~(data_even_rb[0:31]);
            Rt[32:63]=data_even_ra[32:63] & ~(data_even_rb[32:63]);
            Rt[64:95]=data_even_ra[64:95] & ~(data_even_rb[64:95]);
    end
    end

```

```

Rt[96:127]=data_even_ra[96:127] & ~(data_even_rb[96:127]);
wr_en = 1; unit_temp = SIMPLE_FIXED1_UNIT; latency_temp = SIMPLE_FIXED1_LATENCY;
end

//7. compare equal word
11'b011111000000:begin
    for(int i=0; i<4; i=i+1)begin
        if((data_even_ra[(i*32)+: 32])==(data_even_rb[(i*32)+: 32]))
            Rt[(i*32)+: 32]='1;
        else Rt[(i*32)+: 32]='0; end
    wr_en = 1; unit_temp = SIMPLE_FIXED1_UNIT; latency_temp = SIMPLE_FIXED1_LATENCY;
end

//8. compare equal byte
11'b01111010000:begin
    for(int i=0; i<16; i=i+1)begin
        if((data_even_ra[(i*8)+: 8])==(data_even_rb[(i*8)+: 8]))
            Rt[(i*8)+: 8]='1;
        else Rt[(i*8)+: 8]='0; end
    wr_en = 1; unit_temp = SIMPLE_FIXED1_UNIT; latency_temp = SIMPLE_FIXED1_LATENCY;
end

//9. compare equal byte immediate
11'b01111110:begin
    b = imm10_even & 8'hff;
    for(int i=0; i<16; i=i+1)begin
        if((data_even_ra[(i*8)+: 8])==b)
            Rt[(i*8)+: 8]='1;
        else Rt[(i*8)+: 8]='0; end
    wr_en = 1; unit_temp = SIMPLE_FIXED1_UNIT; latency_temp = SIMPLE_FIXED1_LATENCY;
end

//10. compare equal word immediate
11'b01111100:begin
    firstbit = imm10_even[0];
    temp_imm10 = {{22{firstbit}},imm10_even};
    for(int i=0; i<4; i=i+1)begin
        if((data_even_ra[(i*32)+: 32])==temp_imm10)
            Rt[(i*32)+: 32]='1;
        else Rt[(i*32)+: 32]='0; end
    wr_en = 1; unit_temp = SIMPLE_FIXED1_UNIT; latency_temp = SIMPLE_FIXED1_LATENCY;
end

//11. compare greater than byte
11'b01001010000:begin
    for(int i=0; i<16; i=i+1)begin
        if((data_even_ra[(i*8)+: 8])>(data_even_rb[(i*8)+: 8]))
            Rt[(i*8)+: 8]='1;
        else Rt[(i*8)+: 8]='0; end
    wr_en = 1; unit_temp = SIMPLE_FIXED1_UNIT; latency_temp = SIMPLE_FIXED1_LATENCY;
end

//12. compare greater byte immediate
11'b01001110:begin
    b = imm10_even & 8'hff;
    for(int i=0; i<16; i=i+1)begin
        if(data_even_ra[(i*8)+: 8]> b)
            Rt[(i*8)+: 8]='1; else
            Rt[(i*8)+: 8]='0; end
    wr_en = 1; unit_temp = SIMPLE_FIXED1_UNIT; latency_temp = SIMPLE_FIXED1_LATENCY;
end

```

```

//13. compare greater than word immediate
11'b01001100:begin
    firstbit = imm10_even[0];
    temp_imm10 = {{22{firstbit}},imm10_even};
    for(int i=0; i<4; i=i+1)begin
        if((data_even_ra[(i*32)+: 32])>temp_imm10)
            Rt[(i*32)+: 32]='1;
        else Rt[(i*32)+: 32]='0; end
    wr_en = 1; unit_temp = SIMPLE_FIXED1_UNIT; latency_temp = SIMPLE_FIXED1_LATENCY;
end

//14. compare logical greater than byte
11'b01011010000:begin
    for(int i=0; i<16; i=i+1)begin
        if($unsigned(data_even_ra[(i*8)+: 8])>$unsigned(data_even_rb[(i*8)+: 8]))
            Rt[(i*8)+: 8]='1;
        else Rt[(i*8)+: 8]='0; end
    wr_en = 1; unit_temp = SIMPLE_FIXED1_UNIT; latency_temp = SIMPLE_FIXED1_LATENCY;
end

//15. compare logical greater byte immediate
11'b00001011110:begin
    b = $unsigned(imm10_even) & 8'hff;
    for(int i=0; i<16; i=i+1)begin
        if($unsigned(data_even_ra[(i*8)+: 8])> b)
            Rt[(i*8)+: 8]='1;
        else Rt[(i*8)+: 8]='0; end
    wr_en = 1; unit_temp = SIMPLE_FIXED1_UNIT; latency_temp = SIMPLE_FIXED1_LATENCY;
end

//16. Immediate load word
11'b010000001:begin
    firstbit = imm16_even[0];
    temp_imm = {{16{firstbit}},imm16_even};
    Rt[0:31]=temp_imm; Rt[32:63]=temp_imm;
    Rt[64:95]=temp_imm; Rt[96:127]=temp_imm;
    wr_en = 1; unit_temp = SIMPLE_FIXED1_UNIT; latency_temp = SIMPLE_FIXED1_LATENCY;
end

//17. Immediate load address
11'b0100001:begin
    Rt[0:31]=imm18_even; Rt[32:63]=imm18_even;
    Rt[64:95]=imm18_even; Rt[96:127]=imm18_even;
    wr_en = 1; unit_temp = SIMPLE_FIXED1_UNIT; latency_temp = SIMPLE_FIXED1_LATENCY;
end

//18. Immediate load halfword
11'b010000011:begin
    Rt[0:15]=imm16_even; Rt[16:31]=imm16_even;
    Rt[32:47]=imm16_even; Rt[48:63]=imm16_even;
    Rt[64:79]=imm16_even; Rt[80:95]=imm16_even;
    Rt[96:111]=imm16_even; Rt[112:127]=imm16_even;
    wr_en = 1; unit_temp = SIMPLE_FIXED1_UNIT; latency_temp = SIMPLE_FIXED1_LATENCY;
end

//19. Nand
11'b00011001001:begin
    Rt[0:31]=~(data_even_ra[0:31] & data_even_rb[0:31]);
    Rt[32:63]=~(data_even_ra[32:63] & data_even_rb[32:63]);
    Rt[64:95]=~(data_even_ra[64:95] & data_even_rb[64:95]);
    Rt[96:127]=~(data_even_ra[96:127] & data_even_rb[96:127]);
    wr_en = 1; unit_temp = SIMPLE_FIXED1_UNIT; latency_temp = SIMPLE_FIXED1_LATENCY;
end

```

```

//20. Nor
11'b00001001001:begin
    Rt[0:31]=~(data_even_ra[0:31] | data_even_rb[0:31]);
    Rt[32:63]=~(data_even_ra[32:63] | data_even_rb[0:68]);
    Rt[64:95]=~(data_even_ra[64:95] | data_even_rb[64:95]);
    Rt[96:127]=~(data_even_ra[96:127] | data_even_rb[96:127]);
    wr_en = 1; unit_temp = SIMPLE_FIXED1_UNIT; latency_temp = SIMPLE_FIXED1_LATENCY;
end

//21. Or
11'b00001000001:begin
    Rt[0:31]=data_even_ra[0:31] | data_even_rb[0:31];
    Rt[32:63]=data_even_ra[32:63] | data_even_rb[32:63];
    Rt[64:95]=data_even_ra[64:95] | data_even_rb[64:95];
    Rt[96:127]=data_even_ra[96:127] | data_even_rb[96:127];
    wr_en = 1; unit_temp = SIMPLE_FIXED1_UNIT; latency_temp = SIMPLE_FIXED1_LATENCY;
end

//22. Or with complement
11'b01011001001:begin
    Rt[0:31]=data_even_ra[0:31] | ~(data_even_rb[0:31]);
    Rt[32:63]=data_even_ra[32:63] | ~(data_even_rb[32:63]);
    Rt[64:95]=data_even_ra[64:95] | ~(data_even_rb[64:95]);
    Rt[96:127]=data_even_ra[96:127] | ~(data_even_rb[96:127]);
    wr_en = 1; unit_temp = SIMPLE_FIXED1_UNIT; latency_temp = SIMPLE_FIXED1_LATENCY;
end

//23. Or word immediate
11'b00000100:begin
    firstbit = imm10_even[0];
    temp_imm10 = {{22{firstbit}},imm10_even};
    Rt[0:31]=data_even_ra[0:31]|temp_imm10;
    Rt[32:63]=data_even_ra[32:63]|temp_imm10;
    Rt[64:95]=data_even_ra[64:95]|temp_imm10;
    Rt[96:127]=data_even_ra[96:127]|temp_imm10;
    wr_en = 1; unit_temp = SIMPLE_FIXED1_UNIT; latency_temp = SIMPLE_FIXED1_LATENCY;
end

//24. Subtract from word
11'b00001000000:begin
    Rt[0:31]=data_even_ra[0:31] + ~(data_even_rb[0:31]) + 1;
    Rt[32:63]=data_even_ra[32:63] + ~(data_even_rb[32:63]) + 1;
    Rt[64:95]=data_even_ra[64:95] + ~(data_even_rb[64:95]) + 1;
    Rt[96:127]=data_even_ra[96:127] + ~(data_even_rb[96:127]) + 1;
    wr_en = 1; unit_temp = SIMPLE_FIXED1_UNIT; latency_temp = SIMPLE_FIXED1_LATENCY;
end

//25. Sub word immediate
11'b00001100:begin
    firstbit = imm10_even[0];
    temp_imm10 = {{22{firstbit}},imm10_even};
    Rt[0:31]=temp_imm10 + ~(data_even_ra[0:31]) + 1;
    Rt[32:63]=temp_imm10 + ~(data_even_ra[32:63]) + 1;
    Rt[64:95]=temp_imm10 + ~(data_even_ra[64:95]) + 1;
    Rt[96:127]=temp_imm10 + ~(data_even_ra[96:127]) + 1;
    wr_en = 1; unit_temp = SIMPLE_FIXED1_UNIT; latency_temp = SIMPLE_FIXED1_LATENCY;
end

//26. Sub extended from word
11'b01101000001:begin
    for(int i=0; i<4; i=i+1)begin
        temp_imm=data_even_rc[(i*32)+: 32];
    end
end

```

```

    if(temp_imm[31]==0)
Rt[(i*32)+: 32]=data_even_ra[(i*32)+: 32] + ~(data_even_rb[(i*32)+: 32]) + 1;
else Rt[(i*32)+: 32]=data_even_ra[(i*32)+: 32] + ~(data_even_rb[(i*32)+: 32]) + temp_imm[31]; end
wr_en = 1; unit_temp = SIMPLE_FIXED1_UNIT; latency_temp = SIMPLE_FIXED1_LATENCY;
end

//27. ExOr
11'b01001000001:begin
    Rt[0:31]=data_even_ra[0:31] ^ data_even_rb[0:31];
    Rt[32:63]=data_even_ra[32:63] ^ data_even_rb[32:63];
    Rt[64:95]=data_even_ra[64:95] ^ data_even_rb[64:95];
    Rt[96:127]=data_even_ra[96:127] ^ data_even_rb[96:127];
    wr_en = 1; unit_temp = SIMPLE_FIXED1_UNIT; latency_temp = SIMPLE_FIXED1_LATENCY;
end

//28. Exor byte immediate
11'b01000110:begin
    b = imm10_even & 8'hff;
    bbbb = {b,b,b,b};
    Rt[0:31]=data_even_ra[0:31] ^ bbbb;
    Rt[32:63]=data_even_ra[32:63] ^ bbbb;
    Rt[64:95]=data_even_ra[64:95] ^ bbbb;
    Rt[96:127]=data_even_ra[96:127] ^ bbbb;
    wr_en = 1; unit_temp = SIMPLE_FIXED1_UNIT; latency_temp = SIMPLE_FIXED1_LATENCY;
end

//29. Ex or word immediate
11'b01000100:begin
    firstbit = imm10_even[0];
    temp_imm10 = {{22{firstbit}},imm10_even};
    Rt[0:31]=data_even_ra[0:31] ^ temp_imm10;
    Rt[32:63]=data_even_ra[32:63] ^ temp_imm10;
    Rt[64:95]=data_even_ra[64:95] ^ temp_imm10;
    Rt[96:127]=data_even_ra[96:127] ^ temp_imm10;
    wr_en = 1; unit_temp = SIMPLE_FIXED1_UNIT; latency_temp = SIMPLE_FIXED1_LATENCY;
end

//30. Extend Sign Byte to Halfword
11'b01010110110:begin
    Rt[0:15]={{8{data_even_ra[8]}},data_even_ra[8:15]};
    Rt[16:31]={{8{data_even_ra[24]}},data_even_ra[24:31]};
    Rt[32:47]={{8{data_even_ra[40]}},data_even_ra[40:47]};
    Rt[48:63]={{8{data_even_ra[56]}},data_even_ra[56:63]};
    Rt[64:79]={{8{data_even_ra[72]}},data_even_ra[72:79]};
    Rt[80:95]={{8{data_even_ra[88]}},data_even_ra[88:95]};
    Rt[96:111]={{8{data_even_ra[104]}},data_even_ra[104:111]};
    Rt[112:127]={{8{data_even_ra[112]}},data_even_ra[112:127]};
    wr_en = 1; unit_temp = SIMPLE_FIXED1_UNIT; latency_temp = SIMPLE_FIXED1_LATENCY;
end

//31. Extend sign halfword to word
11'b01010101110:begin
    Rt[0:31]={{16{data_even_ra[16]}},data_even_ra[16:31]};
    Rt[32:63]={{16{data_even_ra[48]}},data_even_ra[48:63]};
    Rt[64:95]={{16{data_even_ra[80]}},data_even_ra[80:95]};
    Rt[96:127]={{16{data_even_ra[112]}},data_even_ra[112:127]};
    wr_en = 1; unit_temp = SIMPLE_FIXED1_UNIT; latency_temp = SIMPLE_FIXED1_LATENCY;
end

//32. Extend sign word to doubleword
11'b01010100110:begin
    Rt[0:63]={{32{data_even_ra[32]}},data_even_ra[32:63]};
    Rt[64:127]={{32{data_even_ra[96]}},data_even_ra[96:127]};

```

```

    wr_en = 1; unit_temp = SIMPLE_FIXED1_UNIT; latency_temp = SIMPLE_FIXED1_LATENCY;
end

```

```

//33. Absolute differences of bytes

```

```

11'b00001010011:begin
    for(int i=0; i<16; i=i+1)begin
        if((data_even_rb[(i*8)+: 8])>(data_even_ra[(i*8)+: 8]))
            Rt[(i*8)+: 8]=(data_even_rb[(i*8)+: 8])-(data_even_ra[(i*8)+: 8]);
        else Rt[(i*8)+: 8]=(data_even_ra[(i*8)+: 8])-(data_even_rb[(i*8)+: 8]); end
        wr_en = 1; unit_temp = BYTE_UNIT; latency_temp = BYTE_LATENCY;
    end
end

```

```

//34. Average bytes

```

```

11'b00011010011:begin
    for(int i=0; i<16; i=i+1)begin
        Rt[(i*8)+: 8]=((data_even_rb[(i*8)+: 8])+(data_even_ra[(i*8)+: 8])+1)>>1;
        wr_en = 1; unit_temp = BYTE_UNIT; latency_temp = BYTE_LATENCY;
    end
end

```

```

//35. Count ones in bytes

```

```

11'b01010110100: begin
    for(int i=0; i<16; i=i+1) begin
        ra_byte=data_even_ra[(i*8)+: 8];
        count=0;
        for(int j=0; j<8; j=j+1)begin
            if(ra_byte[j]==1'b1)
                count++; end
        Rt[(i*8)+:8] =count; end
        wr_en = 1; unit_temp = BYTE_UNIT; latency_temp = BYTE_LATENCY;
    end
end

```

```

//36. sum bytes into halfword

```

```

11'b01001010011:begin
    Rt[0:15]=data_even_rb[0:7]+data_even_rb[8:15]+data_even_rb[16:23]+data_even_rb[24:31];
    Rt[16:31]=data_even_ra[0:7]+data_even_ra[8:15]+data_even_ra[16:23]+data_even_ra[24:31];
    Rt[32:47]=data_even_rb[32:39]+data_even_rb[40:47]+data_even_rb[48:55]+data_even_rb[56:63];
    Rt[48:63]=data_even_ra[32:39]+data_even_ra[40:47]+data_even_ra[48:55]+data_even_ra[56:63];
    Rt[64:79]=data_even_rb[64:71]+data_even_rb[72:79]+data_even_rb[80:87]+data_even_rb[88:95];
    Rt[80:95]=data_even_ra[64:71]+data_even_ra[72:79]+data_even_ra[80:87]+data_even_ra[88:95];
    Rt[96:111]=data_even_rb[96:103]+data_even_rb[104:111]+data_even_rb[112:119]+data_even_rb[120:127];
    Rt[112:127]=data_even_ra[96:103]+data_even_ra[104:111]+data_even_ra[112:119]+data_even_ra[120:127];
    wr_en = 1; unit_temp = BYTE_UNIT; latency_temp = BYTE_LATENCY;
end

```

```

//37. Rotate word

```

```

11'b00001011000:begin
    for(int i=0; i<4; i=i+1)begin
        rb_word=data_even_rb[(i*32)+: 32];
        if(rb_word[28:31]==5'b0)
            Rt[(i*32)+: 32]=(data_even_ra[(i*32)+: 32]);
        else
            Rt[(i*32)+: 32]=((data_even_ra[(i*32)+: 32])<<(rb_word[28:31]))|((data_even_ra[(i*32)+:32])>>(32-(rb_word[28:31])));
    end
    wr_en = 1; unit_temp = SIMPLE_FIXED2_UNIT; latency_temp = SIMPLE_FIXED2_LATENCY;
end

```

```

//38. Rotate and mask word

```

```

11'b00001011001:begin
    for(int i=0; i<4; i=i+1)begin
        shift_count=(0-data_even_rb[(i*32)+: 32])%64;
        if(shift_count<32)
            Rt[(i*32)+: 32]=(data_even_ra[(i*32)+: 32])>>shift_count;

```



```

    Rt[i*32+:32] = $shortrealtobits(($bitstoshortreal(data_even_ra[i*32+:32]) * $bitstoshortreal(data_even_rb[i*32+:32]))-
$bitstoshortreal(data_even_rc[i*32+:32])); end
    wr_en = 1; unit_temp = SINGLE_PRECISION1_UNIT; latency_temp = SINGLE_PRECISION1_LATENCY;
end

//57. Floating Subtract
11'b0101_1000_101: begin
    for(int i=0; i<4; i=i+1)begin
        Rt[i*32+:32] = $shortrealtobits(($bitstoshortreal(data_even_ra[i*32+:32]) - $bitstoshortreal(data_even_rb[i*32+:32])));
end
    wr_en = 1; unit_temp = SINGLE_PRECISION1_UNIT; latency_temp = SINGLE_PRECISION1_LATENCY;
end

//62. Multiply
11'b01111000100:begin
    Rt[0:31]=data_even_ra[16:31]*data_even_rb[16:31];
    Rt[32:63]=data_even_ra[48:63]*data_even_rb[48:63];
    Rt[64:95]=data_even_ra[80:95]*data_even_rb[80:95];
    Rt[96:127]=data_even_ra[112:127]*data_even_rb[112:127];
    wr_en = 1; unit_temp = SINGLE_PRECISION2_UNIT; latency_temp = SINGLE_PRECISION2_LATENCY;
end

//63. Multiply and add
11'b1100:begin
    rt_temp[0:31]=$signed(data_even_ra[16:31])*$signed(data_even_rb[16:31]);
    rt_temp[32:63]=$signed(data_even_ra[48:63])*$signed(data_even_rb[48:63]);
    rt_temp[64:95]=$signed(data_even_ra[80:95])*$signed(data_even_rb[80:95]);
    rt_temp[96:127]=$signed(data_even_ra[112:127])*$signed(data_even_rb[112:127]);
    Rt[0:31]=rt_temp[0:31]+data_even_rc[0:31];
    Rt[32:63]=rt_temp[32:63]+data_even_rc[32:63];
    Rt[64:95]=rt_temp[64:95]+data_even_rc[64:95];
    Rt[96:127]=rt_temp[96:127]+data_even_rc[96:127];
    wr_en = 1; unit_temp = SINGLE_PRECISION2_UNIT; latency_temp = SINGLE_PRECISION2_LATENCY;
end

//64. MultiplyImmediate
11'b01110100:begin
    firstbit = imm10_even[0];
    temp_imm10 = {{6{firstbit}},imm10_even};
    Rt[0:31]=data_even_ra[16:31]*temp_imm10;
    Rt[32:63]=data_even_ra[48:63]*temp_imm10;
    Rt[64:95]=data_even_ra[80:95]*temp_imm10;
    Rt[96:127]=data_even_ra[112:127]*temp_imm10;
    wr_en = 1; unit_temp = SINGLE_PRECISION2_UNIT; latency_temp = SINGLE_PRECISION2_LATENCY;
end

//65. Multiply and shift right
11'b01111000111:begin
    rt_temp[0:31]=(data_even_ra[16:31])*(data_even_rb[16:31]);
    rt_temp[32:63]=(data_even_ra[48:63])*(data_even_rb[48:63]);
    rt_temp[64:95]=(data_even_ra[80:95])*(data_even_rb[80:95]);
    rt_temp[96:127]=(data_even_ra[112:127])*(data_even_rb[112:127]);
    Rt[0:31]={16{rt_temp[0]}},rt_temp[0:15];
    Rt[32:63]={16{rt_temp[32]}},rt_temp[32:47];
    Rt[64:95]={16{rt_temp[64]}},rt_temp[64:79];
    Rt[96:127]={16{rt_temp[96]}},rt_temp[96:111];
    wr_en = 1; unit_temp = SINGLE_PRECISION2_UNIT; latency_temp = SINGLE_PRECISION2_LATENCY;
end

//66. Multiply Unsigned
11'b01111001100:begin
    Rt[0:31]=$unsigned(data_even_ra[16:31])*$unsigned(data_even_rb[16:31]);
    Rt[32:63]=$unsigned(data_even_ra[48:63])*$unsigned(data_even_rb[48:63]);

```



```

Rt[64:95]=$unsigned(data_even_ra[80:95])*$unsigned(data_even_rb[80:95]);
Rt[96:127]=$unsigned(data_even_ra[112:127])*$unsigned(data_even_rb[112:127]);
wr_en = 1; unit_temp = SINGLE_PRECISION2_UNIT; latency_temp = SINGLE_PRECISION2_LATENCY;
end

//67. Multiply unsigned Immediate
11'b01110101:begin
  firstbit = imm10_even[0];
  temp_imm10 = {{6{firstbit}},imm10_even};
  Rt[0:31]=$unsigned(data_even_ra[16:31])*$unsigned(temp_imm10);
  Rt[32:63]=$unsigned(data_even_ra[48:63])*$unsigned(temp_imm10);
  Rt[64:95]=$unsigned(data_even_ra[80:95])*$unsigned(temp_imm10);
  Rt[96:127]=$unsigned(data_even_ra[112:127])*$unsigned(temp_imm10);
  wr_en = 1; unit_temp = SINGLE_PRECISION2_UNIT; latency_temp = SINGLE_PRECISION2_LATENCY;
end

//71. No Operation Even(Execute)
11'b0100_0000_001:begin
  Rt=128'b0;
  wr_en = 0;
end

//Stop
11'b0:begin
  stop=1;
end

default: begin // Do Nothing
  Rt=128'bz;
end
endcase
end

// Shift Registers - Pipelining
//assign Rt_Temp = {unit_temp, latency_temp, wr_en, addr_even_rt, Rt};

always_comb begin
  if(stop==1)
    Rt_Temp = 143'b0;
  else
    //Rt_Temp = {unit_temp, latency_temp, wr_en, addr_even_rt, Rt};
    Rt_Temp[0:2] = unit_temp;
    Rt_Temp[3:5] = latency_temp;
    Rt_Temp[6] = wr_en;
    Rt_Temp[7:14] = addr_even_rt;
    Rt_Temp[15:142] = Rt;

    Rt_test_even = Rt_Temp;
    //$display ("====REG====>>> unit_temp:%d , latency_temp:%d , wr_en:%d , addr_even_rt:%d , Rt:%d",
Rt_Temp[0:2], latency_temp, wr_en, addr_even_rt, Rt);
    //$display ("Whole Word:%b", stop);
  end
end

always_ff @(posedge clk) begin
  Rt_Temp0_fwd = Rt_Temp;
  Rt_Temp1 <= Rt_Temp;
  Rt_Temp2 <= Rt_Temp1;
  Rt_Temp3 <= Rt_Temp2;
  Rt_Temp4 <= Rt_Temp3;
  Rt_Temp5 <= Rt_Temp4;
  Rt_Temp6 <= Rt_Temp5;
  Rt_Temp7 <= Rt_Temp6;
end

```

```

assign unit = Rt_Temp7 [0:2];
assign latency = Rt_Temp7 [3:5];
assign data_even_rt = Rt_Temp7 [15:142];
assign wr_en_even = Rt_Temp7 [6];
assign addr_even_rt2 = Rt_Temp7 [7:14];

```

```
endmodule
```

Script: Odd Pipe

```
//===== PARAMETERIZATION
=====
```

```

parameter SIMPLE_FIXED1_UNIT    =1;    parameter SIMPLE_FIXED1_LATENCY    =2;
parameter SIMPLE_FIXED2_UNIT    =2;    parameter SIMPLE_FIXED2_LATENCY    =4;
parameter SINGLE_PRECISION1_UNIT =3;    parameter SINGLE_PRECISION1_LATENCY =6;
parameter SINGLE_PRECISION2_UNIT =3;    parameter SINGLE_PRECISION2_LATENCY =7;
parameter BYTE_UNIT             =4;    parameter BYTE_LATENCY             =4;
parameter PERMUTE_UNIT          =5;    parameter PERMUTE_LATENCY          =4;
parameter LOCALSTORE_UNIT       =6;    parameter LOCALSTORE_LATENCY       =6;
parameter BRANCH_UNIT           =7;    parameter BRANCH_LATENCY           =4;

```

```
//===== ODD PIPE
```

```

module oddpipe(clk, reset, opcode_odd, addr_odd_rt, data_odd_ra, data_odd_rb, data_odd_rc, imm7_odd, imm10_odd,
imm16_odd, imm18_odd, addr_odd_rt2, data_odd_rt, wr_en_odd, Pc_in, Pc_out_2, flush, Rt_Temp0_fwd, Rt_Temp1, Rt_Temp2,
Rt_Temp3, Rt_Temp4, Rt_Temp5, Rt_Temp6);

```

```
    parameter WIDTH=8, SIZE=32768;
```

```

input clk, reset, flush;
input [0:6] addr_odd_rt;
input [0:10] opcode_odd;
input [0:6] imm7_odd;          // logic from TB
input [0:9] imm10_odd;         // logic from TB
input [0:15] imm16_odd;        // logic from TB
input [0:17] imm18_odd;        // logic from TB
input [0:14] Pc_in;
input signed [0:127] data_odd_ra, data_odd_rb, data_odd_rc;

```

```

output [0:6] addr_odd_rt2;
output logic signed [0:127] data_odd_rt;
output logic wr_en_odd;
output [0:14] Pc_out_2;
output logic [0:175] Rt_Temp0_fwd, Rt_Temp1, Rt_Temp2, Rt_Temp3, Rt_Temp4, Rt_Temp5, Rt_Temp6;

```

```

logic [0:14] Pc_out;
logic [0:175] Rt_Temp7, Rt_Temp ;
logic signed [0:31] addr_ls_temp, addr_ls;
logic [0:13] LSA_val;
logic [0:127] Rt;
logic [0:2] latency_temp, unit_temp, latency, unit;
logic signed [0:127] data_in, data_out;
logic [0:31] imm16_ext;
logic [0:3] rb_bits;
logic [0:31] shift_count;
logic [0:4] rb_bits_2;
logic wr_en, wr_en_ls, Br_Flag, Br_Flag_2, stop;
integer i;

```

```

//logic [0:16843008][0:31] mem;
logic [0:127]mem[0:4096];

//assign Br_Pc_Out = Pc_out_2;

always_comb begin
    case(opcode_odd)

//41. Branch Indirect
    11'b0011_0101_000: begin
        logic e, d;
        logic [0:1] intr;
        Pc_out = (data_odd_ra[0:31] & 32'hffffffc); // Branch Target Address
        Br_Flag = 1; // Branch Flush Signal
        if(e==1 && d==0) intr = 2'd1;
        else if(e==0 && d==1) intr = 2'd2;
        else if(e==0 && d==1) intr = 2'd3;
        else intr = 2'd0;
        wr_en = 1; unit_temp = BRANCH_UNIT; latency_temp = BRANCH_LATENCY;
        end

//42. Branch Indirect and Set Link
    11'b0011_0101_001: begin
        logic e, d;
        logic [0:1] intr;
        Pc_out = (data_odd_ra[0:31] & 32'hffffffc); // Branch Target Address
        Rt[0:31] = (Pc_in +4);
        Rt[32:127] = 95'd0;
        Br_Flag = 1; // Branch Flush Signal
        if(e==1 && d==0) intr = 2'd1;
        else if(e==0 && d==1) intr = 2'd2;
        else if(e==0 && d==1) intr = 2'd3;
        else intr = 2'd0;
        wr_en = 1; unit_temp = BRANCH_UNIT; latency_temp = BRANCH_LATENCY;
        end

//43. Branch Relative
    11'b0011_0010_0: begin
        for(i=0;i<14;i=i+1)
            imm16_ext[i]=imm16_odd[0];
        imm16_ext[14:29]=imm16_odd;
        imm16_ext[30:31]=2'b00;
        Pc_out = (Pc_in + imm16_ext); // Branch Target Address
        Br_Flag = 1; // Branch Flush Signal
        Rt[0:127] = 128'd0;
        wr_en = 1; unit_temp = BRANCH_UNIT; latency_temp = BRANCH_LATENCY;
        end

//44. Branch Absolute
    11'b0011_0000_0: begin
        for(i=0;i<14;i=i+1)
            imm16_ext[i]=imm16_odd[0];
        imm16_ext[14:29]=imm16_odd;
        imm16_ext[30:31]=2'b00;
        Pc_out = imm16_ext; // Branch Target Address
        Br_Flag = 1; // Branch Flush Signal
        Rt[0:127] = 128'd0;
        wr_en = 1; unit_temp = BRANCH_UNIT; latency_temp = BRANCH_LATENCY;
        end

//45. Branch Absolute and Set Link
    11'b0011_0001_0: begin

```

```

    for(i=0;i<14;i=i+1)
    imm16_ext[i]=imm16_odd[0];
    imm16_ext[14:29]=imm16_odd;
    imm16_ext[30:31]=2'b00;
    Pc_out = imm16_ext;          // Branch Target Address
    Rt[0:31] = (Pc_in +4);
    Rt[32:127] = 95'd0;
    Br_Flag = 1;                // Branch Flush Signal
    wr_en = 1; unit_temp = BRANCH_UNIT; latency_temp = BRANCH_LATENCY;
end

//46. Branch Relative and Set Link
11'b0011_0011_0: begin
for(i=0;i<14;i=i+1)
    imm16_ext[i] = imm16_odd[0];
    imm16_ext[14:29] = imm16_odd;
    imm16_ext[30:31] = 2'b00;
    Pc_out = (Pc_in+(imm16_ext + 4));
    Br_Flag = 1;                // Branch Flush Signal
    Rt[0:127] = 128'd0;
    wr_en = 1; unit_temp = BRANCH_UNIT; latency_temp = BRANCH_LATENCY;
end

// 68. Branch if Not Zero Word
11'b0010_0001_0: begin
for(i=0;i<14;i=i+1)
    imm16_ext[i]=imm16_odd[0];
    imm16_ext[14:29]=imm16_odd;
    imm16_ext[30:31]=2'b00;
if(data_odd_rc[0:31] != 32'b0) begin
    Pc_out = ((Pc_in + imm16_ext) & 32'hffffffc); // Branch Target Address
    Br_Flag = 1; end // Branch Flush Signal
else begin
    Pc_out = (Pc_in + 4);
    Br_Flag = 0; end // Branch Flush Signal
    Rt[0:127] = 128'd0; // Optional
    wr_en = 1; unit_temp = BRANCH_UNIT; latency_temp = BRANCH_LATENCY;
end

//69. Branch if Zero Word
11'b0010_0000_0: begin
for(i=0;i<14;i=i+1)
    imm16_ext[i]=imm16_odd[0];
    imm16_ext[14:29]=imm16_odd;
    imm16_ext[30:31]=2'b00;
if(data_odd_rc[0:31] == 32'b0) begin
    Pc_out = ((Pc_in + imm16_ext) & 32'hffffffc); // Branch Target Address
    Br_Flag = 1; end // Branch Flush Signal
else begin
    Pc_out = (Pc_in + 4);
    Br_Flag = 0; end // Branch Flush Signal
    Rt[0:127] = 128'd0; // Optional
    wr_en = 1; unit_temp = BRANCH_UNIT; latency_temp = BRANCH_LATENCY;
end

//47. Rotate quadword by bytes
11'b00111011100:begin
rb_bits=data_odd_rb[28:31];
for(int i=0; i<16; i=i+1)begin
    if(rb_bits==4'b0)
        Rt[(i*8)+: 8]=(data_odd_ra[(i*8)+: 8]);
    else

```

```

    Rt=((data_odd_ra)<<(8*rb_bits))|((data_odd_ra)>>8*(16-(rb_bits)));
end
wr_en = 1; unit_temp = PERMUTE_UNIT; latency_temp = PERMUTE_LATENCY;
end

//48. Rotate Quadword by Bytes Immediate
11'b00111111100:begin//Rotate Quadword by Bytes Immediate
rb_bits=imm7_odd[3:6];
for(int i=0; i<16; i=i+1)begin
    if(rb_bits==4'b0)
        Rt[(i*8)+: 8]=(data_odd_ra[(i*8)+: 8]);
    else
        Rt=((data_odd_ra)<<(8*rb_bits))|((data_odd_ra)>>8*(16-(rb_bits)));
    end
wr_en = 1; unit_temp = PERMUTE_UNIT; latency_temp = PERMUTE_LATENCY;
end

//49. Rotate and mask Quadword by Bytes
11'b00111011101:begin
shift_count=(0-data_odd_rb[27:31])%32;
for(int i=0; i<16; i=i+1)begin
    if(shift_count<16)
        Rt=(data_odd_ra)>>(8*shift_count);
    else
        Rt=128'b0;
    end
wr_en = 1; unit_temp = PERMUTE_UNIT; latency_temp = PERMUTE_LATENCY;
end

//50. Rotate and mask Quadword by Bytes Immediate
11'b00111111101:begin
shift_count=(0-imm7_odd)%32;
for(int i=0; i<16; i=i+1)begin
    if(shift_count<16)
        Rt=(data_odd_ra)>>(8*shift_count);
    else
        Rt=128'b0;
    end
wr_en = 1; unit_temp = PERMUTE_UNIT; latency_temp = PERMUTE_LATENCY;
end

//51. Shift left quadword by bytes
11'b00111011111:begin
rb_bits_2=data_odd_rb[27:31];
for(int i=0; i<16; i=i+1)begin
    if(rb_bits==5'b0)
        Rt[(i*8)+: 8]=(data_odd_ra[(i*8)+: 8]);
    else if(rb_bits>15) Rt=128'b0;
    else
        Rt=((data_odd_ra)<<(8*rb_bits));
    end
wr_en = 1; unit_temp = PERMUTE_UNIT; latency_temp = PERMUTE_LATENCY;
end

//52. Shift left quadword immediate
11'b00111111111:begin
rb_bits_2=imm7_odd[2:6];

if(addr_odd_rt == 12) Rt=mem[169];
else if(addr_odd_rt == 13) Rt=mem[170];
else if(addr_odd_rt == 13) Rt=mem[171];
else begin
for(int i=0; i<16; i=i+1)begin

```

```

        if(rb_bits_2==5'b0)
            Rt[(i*8)+: 8]=(data_odd_ra[(i*8)+: 8]);
        else if(rb_bits_2>15)
            Rt=128'b0;
        else
            Rt=((data_odd_ra)<<(8*rb_bits_2));
    end end
    $display("====={}}}}}}}}}}}}}}}}}}}}=====> Ra: %b, III %d",data_odd_ra, rb_bits_2);
    $display("====={}}}}}}}}}}}}}}}}}}}}=====> Shift Value Calculated: %b , %b, %b, %b", Rt[0:31], Rt[32:63],
    Rt[64:95], Rt[96:127]);
    wr_en = 1; unit_temp = PERMUTE_UNIT; latency_temp = PERMUTE_LATENCY;
end

//58. Load Quadword (a-form)
11'b0000_1100_001: begin
    for(i=0;i<14;i=i+1)
        imm16_ext[i]=imm16_odd[0];
    imm16_ext[14:29]=imm16_odd; imm16_ext[30:31]=2'b00;
    addr_ls = imm16_ext & 32'hffffff0;
    Rt = data_out;
    wr_en = 1; wr_en_ls=0; unit_temp = LOCALSTORE_UNIT; latency_temp = LOCALSTORE_LATENCY;
end

//59. Load Quadword (x-form)
11'b0011_1000_100: begin
    addr_ls_temp = data_odd_ra[0:31] + data_odd_rb[0:31];
    addr_ls = (addr_ls_temp & 32'hffffff0);
    Rt = data_out;
    wr_en = 0; wr_en_ls=0; unit_temp = LOCALSTORE_UNIT; latency_temp = LOCALSTORE_LATENCY;
end

//60. Store Quadword (a-form)
11'b0000_1000_001: begin
    for(i=0;i<14;i=i+1)
        imm16_ext[i]=imm16_odd[0];
    imm16_ext[14:29]=imm16_odd; imm16_ext[30:31]=2'b00;
    addr_ls = imm16_ext & 32'hffffff0;
    data_in = data_odd_rc;
    wr_en = 0; wr_en_ls=1; unit_temp = LOCALSTORE_UNIT; latency_temp = LOCALSTORE_LATENCY;
end

//61. Store Quadword (x-form)
11'b0010_1000_100: begin
    addr_ls_temp = data_odd_ra[0:31] + data_odd_rb[0:31];
    addr_ls = (addr_ls_temp & 32'hffffff0);
    data_in = data_odd_rc;
    wr_en = 0; wr_en_ls=1; unit_temp = LOCALSTORE_UNIT; latency_temp = LOCALSTORE_LATENCY;
end

//70. No Operation_odd (Execute)
11'b0100_0000_010:begin
    Rt=128'b0; Pc_out = 128'b0;
    wr_en = 0;
end

//Stop
11'b0:begin
    stop=1;
end

default: begin
    Rt=128'bx; wr_en = 1;
end
// Do Nothing

```


endmodule

END OF REPORT