INDIAN INSTITUTE OF INFORMATION TECHNOLOGY DESIGN AND MANUFACTURING KANCHEEPURAM, 600127

A MINI PROJECT REPORT

ON

TOMASULO's ALGORITHM IMPLEMENTATION

(ON 3X3 MATRIX ADDITION)

SUBMITTED BY

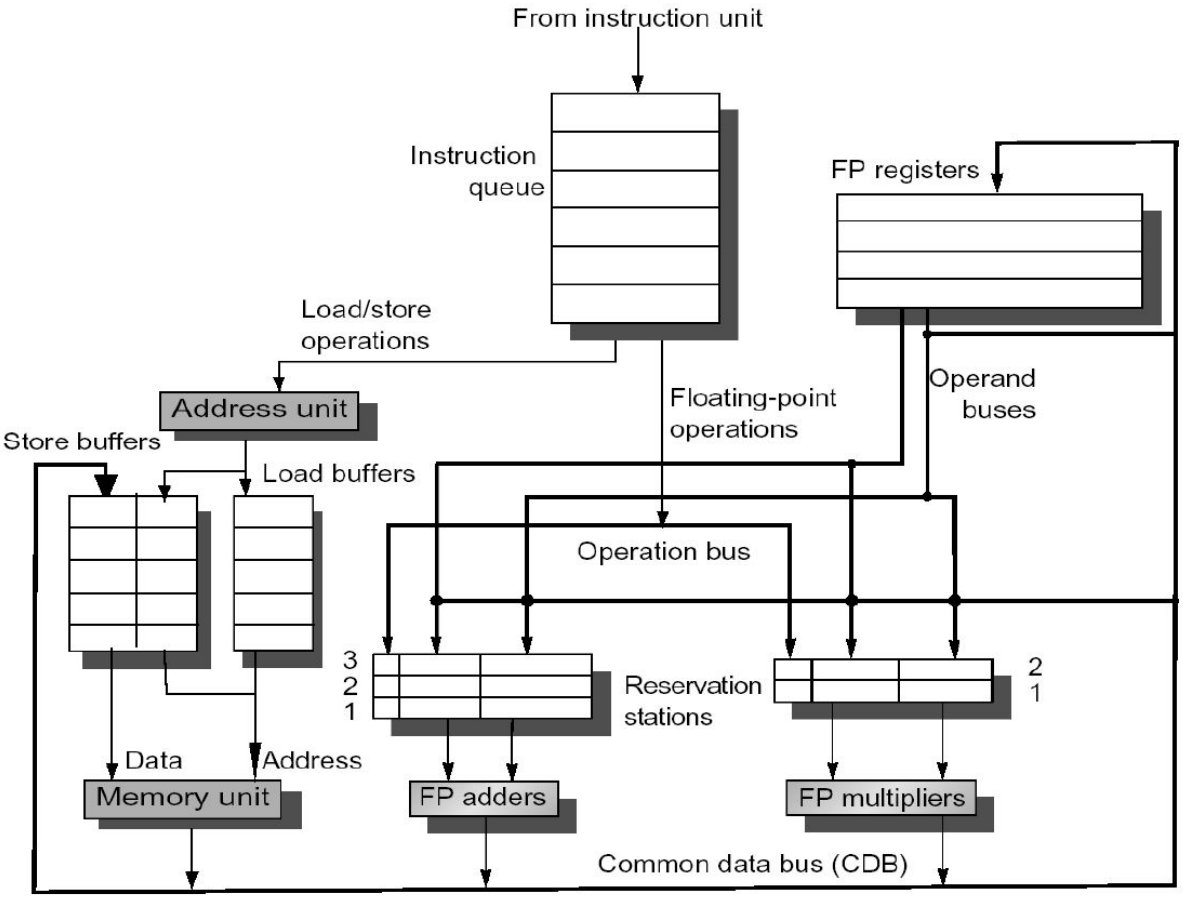
AMAR KUMAR(CED17I029)

TO

DR. NOOR MAHAMMAD SK

1. Introduction -

Tomasulo's algorithm is a computer architecture hardware algorithm for dynamic scheduling of instructions that allows out-of-order execution and enables more efficient use of multiple execution units. It was developed by Robert Tomasulo at IBM in 1967 and was first implemented in the IBM System/360 Model 91's floating point unit.



The major innovations of Tomasulo's algorithm include register renaming in hardware, reservation stations for all execution units, and a common data bus (CDB) on which computed values broadcast to all reservation stations that may need them. These developments allow for improved parallel execution of instructions that would otherwise stall under the use of scoreboarding or other earlier algorithms.

The concepts of reservation stations, register renaming, and the common data bus in Tomasulo's algorithm presents significant advancements in the design of high-performance computers.

Reservation stations take on the responsibility of waiting for operands in the presence of data dependencies and other inconsistencies such as varying storage access time and circuit speeds, thus freeing up the functional units. This improvement overcomes long floating point delays and memory accesses. In particular the algorithm is more tolerant of cache misses. Additionally, programmers are freed from

implementing optimized code. This is a result of the common data bus and reservation station working together to preserve dependencies as well as encouraging concurrency.

By tracking operands for instructions in the reservation stations and register renaming in hardware the algorithm minimizes read-after-write (RAW) and eliminates write-after-write (WAW) and Write-after-Read (WAR) computer architecture hazards. This improves performance by reducing wasted time that would otherwise be required for stalls.

2. Implementation -

2.1 Input File Format -

1. Integer Adder properties have to be set in the following format:
    - int_adder [num_of_rs] [cycles_in_ex] [num_of_fus]
2. Floating Point Adder properties have to be set in the following format:
    - fp_adder [num_of_rs] [cycles_in_ex] [num_of_fus]
3. Floating Point Multiplier properties have to be set in the following format:
    - fp_multiplier [num_of_rs] [cycles_in_ex] [num_of_fus]
4. Load Store Unit properties have to be set in the following format:
    - load_store_unit [num_of_rs] [cycles_in_ex] [cycles_in_mem][num_of_fus]
5. Register values have to be set in the following format:
    - reg [reg_name] [reg_value]
6. Memory values have to be set in the following format:
    - mem [mem_address_in_bytes] [mem_value]
7. Individual instructions are parsed by space, do not use commas
8. Input file is insensitive to order between properties/instructions/etc.
9. Input file is case-insensitive
10. Input file can handle comments that begin with “# “ and empty lines

2.2 Input Format Snippet -

```
#FU name #_of_rs #_of_cycles_in_EX #_of_FUs
int_adder 1 1 1
fp_adder 3 3 1
fp_multiplier 2 20 1
#FU name #_of_rs #_of_cycles_in_EX #_of_cycles_in_mem
#_of_FUs
load_store_unit 3 1 4 1
#rob entries #_of_rob_entries
rob_entries 10
#reg #reg_name #reg_value
reg R1 0
reg R2 3
reg R3 2
reg R4 0
reg F1 0
reg F2 1.1
reg F3 2.5
reg F4 0.0
#instruction
mult.d F1 F2 F3
mult.d F4 F1 F2
add R1 R2 R3
sub R4 R2 R3
```

3. Test Implementation -

Implementing the tomasulo algorithm performing the 3x3 Matrix Addition and analyzing number of stalls ,hazards, status and clock cycles.

3.1 Pseudo Code -

```
C[0][0] = A[0][0] + B[0][0]
C[0][1] = A[0][1] + B[0][1]
C[0][2] = A[0][2] + B[0][2]
C[1][0] = A[1][0] + B[1][0]
C[1][1] = A[1][1] + B[1][1]
C[1][2] = A[1][2] + B[1][2]
C[2][0] = A[2][0] + B[2][0]
C[2][1] = A[2][1] + B[2][1]
C[2][2] = A[2][2] + B[2][2]
```

3.2 Assembly Code-

```
int_adder 2 6 1
fp_adder 3 21 1
fp_multiplier 2 24 1
load_store_unit 3 2 1 1
rob_entries 10
mem 4 3.6           // A[0][0]
mem 8 7.5           // A[0][1]
mem 12 3.5          // A[0][2]
mem 16 6.2          // A[1][0]
mem 20 9.1          // A[1][1]
mem 24 2.7          // A[1][2]
mem 28 0.9          // A[2][0]
mem 32 3.6          // A[2][1]
mem 36 0.6          // A[2][2]
mem 40 2.6          // B[0][0]
mem 44 1.8          // B[0][1]
mem 48 7.9          // B[0][2]
mem 52 2.0          // B[1][0]
mem 56 2.3          // B[1][1]
mem 60 7.5          // B[1][2]
mem 64 9.2          // B[2][0]
mem 68 2.8          // B[2][1]
mem 72 9.7          // B[2][2]
mem 76 0
```

```
ld F2 1(R0)          // A[0][0]
ld F3 10(R0)         // B[0][0]
add.d F1 F2 F3       // C[0][0] = A[0][0] + B[0][0]
ld F3 2(R0)          // A[0][1]
ld F4 11(R0)         // B[0][1]
add.d F2 F3 F4       // C[0][1] = A[0][1] + B[0][1]
ld F4 3(R0)          // A[0][2]
ld F5 12(R0)         // B[0][2]
add.d F3 F4 F5       //C[0][2] = A[0][2] + B[0][2]
ld F5 4(R0)          // A[1][0]
ld F6 13(R0)         // B[1][0]
add.d F4 F5 F6       // C[1][0] = A[1][0] + B[1][0]
ld F6 5(R0)          // A[1][1]
ld F7 14(R0)         // B[1][1]
add.d F5 F6 F7       // C[1][1] = A[1][1] + B[1][1]
ld F7 6(R0)          // A[1][2]
ld F8 15(R0)         // B[1][2]
add.d F6 F7 F8       // C[1][2] = A[1][2] + B[1][2]
ld F8 7(R0)          // A[2][0]
ld F9 16(R0)         // B[2][0]
add.d F7 F8 F9       // C[2][0] = A[2][0] + B[2][0]
ld F9 8(R0)          // A[2][1]
ld F10 17(R0)        // B[2][1]
add.d F8 F9 F10      // C[2][1] = A[2][1] + B[2][1]
ld F10 9(R0)         // A[2][2]
ld F11 18(R0)        // B[2][2]
add.d F9 F10 F11     // C[2][2] = A[2][2] + B[2][2]
ld F10 19(R0)
ld F11 19(R0)
```

4. Output of Implementation -

```
--------------------------------------------------------------------------------------------------------------------------------
                                                    INTEGER ADDER PROPERTIES
--------------------------------------------------------------------------------------------------------------------------------
Number of reservation stations: 2
Number of cycles in execution stage: 6
Number of function Units: 1
--------------------------------------------------------------------------------------------------------------------------------
                                                    FP ADDER PROPERTIES
--------------------------------------------------------------------------------------------------------------------------------
Number of reservation stations: 2
Number of cycles in execution stage: 6
Number of function Units: 1
--------------------------------------------------------------------------------------------------------------------------------
                                                    FP MULTIPLIER PROPERTIES
--------------------------------------------------------------------------------------------------------------------------------
Number of reservation stations: 2
Number of cycles in execution stage: 24
Number of function Units: 1
--------------------------------------------------------------------------------------------------------------------------------
                                                  LOAD - STORE UNIT PROPERTIES
--------------------------------------------------------------------------------------------------------------------------------
Number of reservation stations: 3
Number of cycles in execution stage: 2
Number of cycles in memory stage: 1
Number of function Units: 1
--------------------------------------------------------------------------------------------------------------------------------
                                                    ROB PROPERTIES
--------------------------------------------------------------------------------------------------------------------------------
Number of rob entries: 10
```

All the reservation stations for integer adder, floating point adder, floating point multiplier, load-store unit are initiated , functional units assigned and execution time cycle is initiated.

1. Instruction - LD F2 1(R0) -

PC	INSTRUCTION	ISSUE	EX_S	EX_F	MEM_S	MEM_F	WB	COMMIT_S	COMMIT_F
0	LD F2 1(R0)	1	2	3	4	4	5	6	6

Load instruction was issued at 1CC and added to the Load-Store queue , its occupies **ROB0** entry to perform then load , as mentioned in the input it takes 2CC for execution and 1 CC in memory state, the changes are committed to F2 and **ROB0** is freed so other instruction can use it.(F2 = 3.6).

2. Instruction - LD F3 10(R0) -

PC	INSTRUCTION	ISSUE	EX_S	EX_F	MEM_S	MEM_F	WB	COMMIT_S	COMMIT_F
0	LD F2 1(R0)	1	2	3	4	4	5	6	6
4	LD F3 10(R0)	2	4	5	6	6	7	8	8

Load instruction was issued at 1CC and added to the Load-Store queue , its occupies **ROB1** entry to perform then load , as mentioned in the input it takes 2CC for execution and there is stall because of only one functional unit which is presently occupied by 1st instruction this reduces the chances of Structural hazard and 1 CC in memory state, the changes are committed to F3 and **ROB1** is freed so other instruction can use it.(F3 = 2.6)

3. Instruction - ADD.D F1 F2 F3 -

TIMING TABLE

PC	INSTRUCTION	ISSUE	EX_S	EX_F	MEM_S	MEM_F	WB	COMMIT_S	COMMIT_F
0	LD F2 1(R0)	1	2	3	4	4	5	6	6
4	LD F3 10(R0)	2	4	5	6	6	7	8	8
8	ADD.D F1 F2 F3	3	8	28	-	-	29	30	30

Add instruction is issued at 3CC and added to the Floating point adder reservation station then at 6CC

ROB

	BUSY	INSTRUCTION	STATE	DESTINATION	VALUE
ROB0	no	-	-	-	-
ROB1	yes	LD F3 10(R0)	MEM	F3	-
ROB2	yes	ADD.D F1 F2 F3	ISSUE	F1	-
ROB3	yes	LD F3 2(R0)	EX	F3	-
ROB4	yes	LD F4 11(R0)	ISSUE	F4	-

There the add instruction doesn’t execute as the previous instruction has not provided the operand yet so it can't read from F3 it creates a stalls until the operand is available here **RAW** Data-hazard is eliminated.Then it executes for 21 CC in Floating point Adder unit writes back into F1 at 29 CC and commits the **ROB2** entry to F1 and **ROB2** is freed.(F1 = 6.2)

FLOATING POINT REGISTER FILE

F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15
0	6.2	3.6	2.6	0	0	0	0	0	0	0	0	0	0	0	0

4. Instruction - LD F3 2(R0) -

PC	INSTRUCTION	ISSUE	EX_S	EX_F	MEM_S	MEM_F	WB	COMMIT_S	COMMIT_F
0	LD F2 1(R0)	1	2	3	4	4	5	6	6
4	LD F3 10(R0)	2	4	5	6	6	7	8	8
8	ADD.D F1 F2 F3	3	8	28	-	-	29	30	30
12	LD F3 2(R0)	4	6	7	8	8	9	31	31

At 4CC this instruction is issued it stalls for 1CC after issue because the functional unit is occupied by LD F3 instruction avoiding structural hazard it executes for 2CC access memory in 1CC writebacks in 1CC but commits at 31CC because since ADD instruction previous to it was using the operand F3 so no changes must be made until that instruction completes these stalls in pipeline helps in avoiding the data-hazard. After completion of instruction **ROB3** is freed.(F3 = 7.5)

5. Instruction - LD F4 11(R0) -

PC	INSTRUCTION	ISSUE	EX_S	EX_F	MEM_S	MEM_F	WB	COMMIT_S	COMMIT_F
0	LD F2 1(R0)	1	2	3	4	4	5	6	6
4	LD F3 10(R0)	2	4	5	6	6	7	8	8
8	ADD.D F1 F2 F3	3	8	28	-	-	29	30	30
12	LD F3 2(R0)	4	6	7	8	8	9	31	31
16	LD F4 11(R0)	6	8	9	10	10	11	32	32

This instruction is issued at 6CC because the number of reservation stations available for load-store instruction is 3 therefore one of the entries is freed at 6CC when 1st load instruction completes this stall ensures that.it executes for 2CC accesses memory 1CC then write backs and **ROB4** is freed.(F4= 1.8).

6. Instruction - ADD.D F2 F3 F4 -

PC	INSTRUCTION	ISSUE	EX_S	EX_F	MEM_S	MEM_F	WB	COMMIT_S	COMMIT_F
0	LD F2 1(R0)	1	2	3	4	4	5	6	6
4	LD F3 10(R0)	2	4	5	6	6	7	8	8
8	ADD.D F1 F2 F3	3	8	28	-	-	29	30	30
12	LD F3 2(R0)	4	6	7	8	8	9	31	31
16	LD F4 11(R0)	6	8	9	10	10	11	32	32
20	ADD.D F2 F3 F4	7	12	32	-	-	33	34	34

At 7CC add instruction is issued and added to the floating point adder reservation station. It stalls till 12CC because F4 is still to be fetched by previous instruction; this stall helps in eliminating **RAW** hazard.At 28CC

FLOATING POINT ADDER RS

BUSY	OP	DEST	Vj	Vk	Qj	Qk
no	ADD.D	ROB2	3.6	2.6	-	-
no	ADD.D	ROB5	7.5	1.8	-	-

FLOATING POINT RAT

F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15
F0	F1	ROB5	ROB8	ROB1	ROB9	ROB2	ROB3	F8	F9	F10	F11	F12	F13	F14	F15

Above pictures shows the dependency of operand on **ROB8** and **ROB1** in Register Alias Table.After 33CC

FLOATING POINT REGISTER FILE

F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15
0	6.2	3.6	7.5	1.8	0	0	0	0	0	0	0	0	0	0	0

Now both the operands are ready now it executes for 21CC and result is written back and committed.(F2 = 9.3)

7. Instruction - LD F4 3(R0) -

PC	INSTRUCTION	ISSUE	EX_S	EX_F	MEM_S	MEM_F	WB	COMMIT_S	COMMIT_F
0	LD F2 1(R0)	1	2	3	4	4	5	6	6
4	LD F3 10(R0)	2	4	5	6	6	7	8	8
8	ADD.D F1 F2 F3	3	8	28	-	-	29	30	30
12	LD F3 2(R0)	4	6	7	8	8	9	31	31
16	LD F4 11(R0)	6	8	9	10	10	11	32	32
20	ADD.D F2 F3 F4	7	12	32	-	-	33	34	34
24	LD F4 3(R0)	8	10	11	12	12	13	35	35

Load Instruction is issued at 8CC it stalls till 10CC because previous instruction of load with F4 has not finished executing in pipeline it executes for 2CC and commits only after the previous instruction of add finishes commit so **WAR** hazard is avoided.

8. Instruction - LD F5 12(R0) -

PC	INSTRUCTION	ISSUE	EX_S	EX_F	MEM_S	MEM_F	WB	COMMIT_S	COMMIT_F
0	LD F2 1(R0)	1	2	3	4	4	5	6	6
4	LD F3 10(R0)	2	4	5	6	6	7	8	8
8	ADD.D F1 F2 F3	3	8	28	-	-	29	30	30
12	LD F3 2(R0)	4	6	7	8	8	9	31	31
16	LD F4 11(R0)	6	8	9	10	10	11	32	32
20	ADD.D F2 F3 F4	7	12	32	-	-	33	34	34
24	LD F4 3(R0)	8	10	11	12	12	13	35	35
28	LD F5 12(R0)	10	12	13	14	14	15	36	36

Load instruction is issued at 10CC it stalls till 12CC because the reservation of the station of the load store is now full .It takes 2CC for execution and after committing it frees the ROB entry in the Reorder Buffer.

9. Instruction - ADD.D F3 F4 F5 -

PC	INSTRUCTION	ISSUE	EX_S	EX_F	MEM_S	MEM_F	WB	COMMIT_S	COMMIT_F
0	LD F2 1(R0)	1	2	3	4	4	5	6	6
4	LD F3 10(R0)	2	4	5	6	6	7	8	8
8	ADD.D F1 F2 F3	3	8	28	-	-	29	30	30
12	LD F3 2(R0)	4	6	7	8	8	9	31	31
16	LD F4 11(R0)	6	8	9	10	10	11	32	32
20	ADD.D F2 F3 F4	7	12	32	-	-	33	34	34
24	LD F4 3(R0)	8	10	11	12	12	13	35	35
28	LD F5 12(R0)	10	12	13	14	14	15	36	36
32	ADD.D F3 F4 F5	11	16	36	-	-	37	38	38

This ADD instruction is issued at 11CC it starts execution at 16CC because at 15CC the last load instruction completes and then it starts executing this stall ensures elimination of **RAW** hazard in the pipeline. Then it executes for 21CC then write backs and commits at 38CC.

FLOATING POINT REGISTER FILE

F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15
0	6.2	9.3	7.5	3.5	7.9	0	0	0	0	0	0	0	0	0	0

FLOATING POINT ADDER RS

BUSY	OP	DEST	Vj	Vk	Qj	Qk
no	ADD.D	ROB1	6.2	2.0	-	-
yes	ADD.D	ROB4	9.1	-	-	ROB3

ROB

	BUSY	INSTRUCTION	STATE	DESTINATION	VALUE
ROB0	no	LD F6 13(R0)	WB	F6	2.0
ROB1	yes	ADD.D F4 F5 F6	EX	F4	-
ROB2	no	LD F6 5(R0)	WB	F6	9.1
ROB3	yes	LD F7 14(R0)	MEM	F7	-
ROB4	yes	ADD.D F5 F6 F7	ISSUE	F5	-
ROB5	yes	LD F7 6(R0)	EX	F7	-
ROB6	yes	LD F8 15(R0)	ISSUE	F8	-
ROB7	no	-	-	-	-
ROB8	no	ADD.D F3 F4 F5	WB	F3	11.4

10. Instruction - LD F5 4(R0) -

PC	INSTRUCTION	ISSUE	EX_S	EX_F	MEM_S	MEM_F	WB	COMMIT_S	COMMIT_F
0	LD F2 1(R0)	1	2	3	4	4	5	6	6
4	LD F3 10(R0)	2	4	5	6	6	7	8	8
8	ADD.D F1 F2 F3	3	8	28	-	-	29	30	30
12	LD F3 2(R0)	4	6	7	8	8	9	31	31
16	LD F4 11(R0)	6	8	9	10	10	11	32	32
20	ADD.D F2 F3 F4	7	12	32	-	-	33	34	34
24	LD F4 3(R0)	8	10	11	12	12	13	35	35
28	LD F5 12(R0)	10	12	13	14	14	15	36	36
32	ADD.D F3 F4 F5	11	16	36	-	-	37	38	38
36	LD F5 4(R0)	12	14	15	16	16	17	39	39

Load instruction is issued at 12CC starts executing at 14CC and takes 2CC to execute it commits only after the previous instruction of ADD which has one of the operands as F5 completes so these stalls avoid the hazard.It commits and the ROB entry is freed for other units to use it.

11. Instruction - LD F6 13(R0) -

PC	INSTRUCTION	ISSUE	EX_S	EX_F	MEM_S	MEM_F	WB	COMMIT_S	COMMIT_F
0	LD F2 1(R0)	1	2	3	4	4	5	6	6
4	LD F3 10(R0)	2	4	5	6	6	7	8	8
8	ADD.D F1 F2 F3	3	8	28	-	-	29	30	30
12	LD F3 2(R0)	4	6	7	8	8	9	31	31
16	LD F4 11(R0)	6	8	9	10	10	11	32	32
20	ADD.D F2 F3 F4	7	12	32	-	-	33	34	34
24	LD F4 3(R0)	8	10	11	12	12	13	35	35
28	LD F5 12(R0)	10	12	13	14	14	15	36	36
32	ADD.D F3 F4 F5	11	16	36	-	-	37	38	38
36	LD F5 4(R0)	12	14	15	16	16	17	39	39
40	LD F6 13(R0)	14	16	17	18	18	19	40	40

This Load instruction is issued at 14CC it executes at 16CC for 2CC and then accesses the memory for the value then commits to F6 at 40CC.

12. Instruction - ADD.D F4 F5 F6 -

PC	INSTRUCTION	ISSUE	EX_S	EX_F	MEM_S	MEM_F	WB	COMMIT_S	COMMIT_F
0	LD F2 1(R0)	1	2	3	4	4	5	6	6
4	LD F3 10(R0)	2	4	5	6	6	7	8	8
8	ADD.D F1 F2 F3	3	8	28	-	-	29	30	30
12	LD F3 2(R0)	4	6	7	8	8	9	31	31
16	LD F4 11(R0)	6	8	9	10	10	11	32	32
20	ADD.D F2 F3 F4	7	12	32	-	-	33	34	34
24	LD F4 3(R0)	8	10	11	12	12	13	35	35
28	LD F5 12(R0)	10	12	13	14	14	15	36	36
32	ADD.D F3 F4 F5	11	16	36	-	-	37	38	38
36	LD F5 4(R0)	12	14	15	16	16	17	39	39
40	LD F6 13(R0)	14	16	17	18	18	19	40	40
44	ADD.D F4 F5 F6	30	31	51	-	-	52	53	53

This ADD instruction is issued at 30CC because the Floating point then it executes for 21CC then writes back at 52CC and commits at 53CC no hazards or stalls are created at this instruction.

13. Instruction - LD F6 5(R0) -

PC	INSTRUCTION	ISSUE	EX_S	EX_F	MEM_S	MEM_F	WB	COMMIT_S	COMMIT_F
0	LD F2 1(R0)	1	2	3	4	4	5	6	6
4	LD F3 10(R0)	2	4	5	6	6	7	8	8
8	ADD.D F1 F2 F3	3	8	28	-	-	29	30	30
12	LD F3 2(R0)	4	6	7	8	8	9	31	31
16	LD F4 11(R0)	6	8	9	10	10	11	32	32
20	ADD.D F2 F3 F4	7	12	32	-	-	33	34	34
24	LD F4 3(R0)	8	10	11	12	12	13	35	35
28	LD F5 12(R0)	10	12	13	14	14	15	36	36
32	ADD.D F3 F4 F5	11	16	36	-	-	37	38	38
36	LD F5 4(R0)	12	14	15	16	16	17	39	39
40	LD F6 13(R0)	14	16	17	18	18	19	40	40
44	ADD.D F4 F5 F6	30	31	51	-	-	52	53	53
48	LD F6 5(R0)	31	32	33	34	34	35	54	54

This load instruction is issued at 31CC execution starts at 32CC in pipeline and executes for 2CC then memory access in 1CC then write back and commit at 54CC and the ROB entry is freed.

14. Instruction - LD F7 14(R0) -

PC	INSTRUCTION	ISSUE	EX_S	EX_F	MEM_S	MEM_F	WB	COMMIT_S	COMMIT_F
0	LD F2 1(R0)	1	2	3	4	4	5	6	6
4	LD F3 10(R0)	2	4	5	6	6	7	8	8
8	ADD.D F1 F2 F3	3	8	28	-	-	29	30	30
12	LD F3 2(R0)	4	6	7	8	8	9	31	31
16	LD F4 11(R0)	6	8	9	10	10	11	32	32
20	ADD.D F2 F3 F4	7	12	32	-	-	33	34	34
24	LD F4 3(R0)	8	10	11	12	12	13	35	35
28	LD F5 12(R0)	10	12	13	14	14	15	36	36
32	ADD.D F3 F4 F5	11	16	36	-	-	37	38	38
36	LD F5 4(R0)	12	14	15	16	16	17	39	39
40	LD F6 13(R0)	14	16	17	18	18	19	40	40
44	ADD.D F4 F5 F6	30	31	51	-	-	52	53	53
48	LD F6 5(R0)	31	32	33	34	34	35	54	54
52	LD F7 14(R0)	32	34	35	36	36	38	55	55

This load instruction is issued at 32CC execution starts at 34CC in pipeline because the reservation stations of load store is full and it stalls until the reservation station is not available and executes for 2CC then memory access in 1CC then write back and commit at 55CC and the ROB entry is freed.

15. Instruction - ADD.D F5 F6 F7 -

PC	INSTRUCTION	ISSUE	EX_S	EX_F	MEM_S	MEM_F	WB	COMMIT_S	COMMIT_F
0	LD F2 1(R0)	1	2	3	4	4	5	6	6
4	LD F3 10(R0)	2	4	5	6	6	7	8	8
8	ADD.D F1 F2 F3	3	8	28	-	-	29	30	30
12	LD F3 2(R0)	4	6	7	8	8	9	31	31
16	LD F4 11(R0)	6	8	9	10	10	11	32	32
20	ADD.D F2 F3 F4	7	12	32	-	-	33	34	34
24	LD F4 3(R0)	8	10	11	12	12	13	35	35
28	LD F5 12(R0)	10	12	13	14	14	15	36	36
32	ADD.D F3 F4 F5	11	16	36	-	-	37	38	38
36	LD F5 4(R0)	12	14	15	16	16	17	39	39
40	LD F6 13(R0)	14	16	17	18	18	19	40	40
44	ADD.D F4 F5 F6	30	31	51	-	-	52	53	53
48	LD F6 5(R0)	31	32	33	34	34	35	54	54
52	LD F7 14(R0)	32	34	35	36	36	38	55	55
56	ADD.D F5 F6 F7	34	39	59	-	-	60	61	61

This ADD instruction is issued at 34CC it stalls till 39CC because F7 is still fetched this stalls ensures elimination of **RAW** Hazard at ADD.it then executes for 21CC and gets committed at 61CC.

ROB

	BUSY	INSTRUCTION	STATE	DESTINATION	VALUE
ROB0	no	LD F6 13(R0)	WB	F6	2.0
ROB1	yes	ADD.D F4 F5 F6	EX	F4	-
ROB2	no	LD F6 5(R0)	WB	F6	9.1
ROB3	yes	LD F7 14(R0)	MEM	F7	-
ROB4	yes	ADD.D F5 F6 F7	ISSUE	F5	-
ROB5	yes	LD F7 6(R0)	EX	F7	-
ROB6	yes	LD F8 15(R0)	ISSUE	F8	-
ROB7	no	-	-	-	-
ROB8	no	ADD.D F3 F4 F5	WB	F3	11.4
ROB9	no	LD F5 4(R0)	WB	F5	6.2

ReOrder Buffer status of all the Rob entries at 37CC.

FLOATING POINT REGISTER FILE

F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15
0	6.2	9.3	7.5	3.5	7.9	0	0	0	0	0	0	0	0	0	0

FLOATING POINT RAT

F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15
F0	F1	F2	ROB8	ROB1	ROB4	ROB2	ROB5	ROB6	F9	F10	F11	F12	F13	F14	F15

16. Instruction - LD F7 6(R0) -

PC	INSTRUCTION	ISSUE	EX_S	EX_F	MEM_S	MEM_F	WB	COMMIT_S	COMMIT_F
0	LD F2 1(R0)	1	2	3	4	4	5	6	6
4	LD F3 10(R0)	2	4	5	6	6	7	8	8
8	ADD.D F1 F2 F3	3	8	28	-	-	29	30	30
12	LD F3 2(R0)	4	6	7	8	8	9	31	31
16	LD F4 11(R0)	6	8	9	10	10	11	32	32
20	ADD.D F2 F3 F4	7	12	32	-	-	33	34	34
24	LD F4 3(R0)	8	10	11	12	12	13	35	35
28	LD F5 12(R0)	10	12	13	14	14	15	36	36
32	ADD.D F3 F4 F5	11	16	36	-	-	37	38	38
36	LD F5 4(R0)	12	14	15	16	16	17	39	39
40	LD F6 13(R0)	14	16	17	18	18	19	40	40
44	ADD.D F4 F5 F6	30	31	51	-	-	52	53	53
48	LD F6 5(R0)	31	32	33	34	34	35	54	54
52	LD F7 14(R0)	32	34	35	36	36	38	55	55
56	ADD.D F5 F6 F7	34	39	59	-	-	60	61	61
60	LD F7 6(R0)	35	36	37	38	38	39	62	62

The load instruction is issued at 35CC then it starts executing from 36CC for 2CC then it accesses the memory,writebacks and commits at 62CC,**ROB5** is freed.

17. Instruction - LD F8 15(R0) -

PC	INSTRUCTION	ISSUE	EX_S	EX_F	MEM_S	MEM_F	WB	COMMIT_S	COMMIT_F
0	LD F2 1(R0)	1	2	3	4	4	5	6	6
4	LD F3 10(R0)	2	4	5	6	6	7	8	8
8	ADD.D F1 F2 F3	3	8	28	-	-	29	30	30
12	LD F3 2(R0)	4	6	7	8	8	9	31	31
16	LD F4 11(R0)	6	8	9	10	10	11	32	32
20	ADD.D F2 F3 F4	7	12	32	-	-	33	34	34
24	LD F4 3(R0)	8	10	11	12	12	13	35	35
28	LD F5 12(R0)	10	12	13	14	14	15	36	36
32	ADD.D F3 F4 F5	11	16	36	-	-	37	38	38
36	LD F5 4(R0)	12	14	15	16	16	17	39	39
40	LD F6 13(R0)	14	16	17	18	18	19	40	40
44	ADD.D F4 F5 F6	30	31	51	-	-	52	53	53
48	LD F6 5(R0)	31	32	33	34	34	35	54	54
52	LD F7 14(R0)	32	34	35	36	36	38	55	55
56	ADD.D F5 F6 F7	34	39	59	-	-	60	61	61
60	LD F7 6(R0)	35	36	37	38	38	39	62	62
64	LD F8 15(R0)	36	38	39	40	40	41	63	63

The load instruction is issued at 36CC it stalls till 38CC because the reservation station is full for Load-Store stations.then it executes for 2CC ,then accesses the memory for 1CC then write backs and commits, **ROB6** is freed.

18. Instruction - ADD.D F6 F7 F8 -

PC	INSTRUCTION	ISSUE	EX_S	EX_F	MEM_S	MEM_F	WB	COMMIT_S	COMMIT_F
0	LD F2 1(R0)	1	2	3	4	4	5	6	6
4	LD F3 10(R0)	2	4	5	6	6	7	8	8
8	ADD.D F1 F2 F3	3	8	28	-	-	29	30	30
12	LD F3 2(R0)	4	6	7	8	8	9	31	31
16	LD F4 11(R0)	6	8	9	10	10	11	32	32
20	ADD.D F2 F3 F4	7	12	32	-	-	33	34	34
24	LD F4 3(R0)	8	10	11	12	12	13	35	35
28	LD F5 12(R0)	10	12	13	14	14	15	36	36
32	ADD.D F3 F4 F5	11	16	36	-	-	37	38	38
36	LD F5 4(R0)	12	14	15	16	16	17	39	39
40	LD F6 13(R0)	14	16	17	18	18	19	40	40
44	ADD.D F4 F5 F6	30	31	51	-	-	52	53	53
48	LD F6 5(R0)	31	32	33	34	34	35	54	54
52	LD F7 14(R0)	32	34	35	36	36	38	55	55
56	ADD.D F5 F6 F7	34	39	59	-	-	60	61	61
60	LD F7 6(R0)	35	36	37	38	38	39	62	62
64	LD F8 15(R0)	36	38	39	40	40	41	63	63
68	ADD.D F6 F7 F8	38	42	62	-	-	63	64	64

This ADD instruction is issued at 38CC; it stalls until 42CC until F8 is fetched; these stalls ensure that there isn't any **RAW** hazard.then it executes for 21CC after which writebacks to F6.

ROB

	BUSY	INSTRUCTION	STATE	DESTINATION	VALUE
ROB0	yes	ADD.D F7 F8 F9	EX	F7	-
ROB1	no	LD F9 8(R0)	WB	F9	3.6
ROB2	no	LD F10 17(R0)	WB	F10	2.8
ROB3	yes	ADD.D F8 F9 F10	EX	F8	-
ROB4	yes	LD F10 9(R0)	EX	F10	-
ROB5	yes	LD F11 18(R0)	ISSUE	F11	-
ROB6	no	-	-	-	-
ROB7	no	ADD.D F6 F7 F8	WB	F6	10.2
ROB8	no	LD F8 7(R0)	WB	F8	0.9
ROB9	no	LD F9 16(R0)	WB	F9	9.2

FLOATING POINT ADDER RS

BUSY	OP	DEST	Vj	Vk	Qj	Qk
no	ADD.D	ROB0	0.9	9.2	-	-
no	ADD.D	ROB3	3.6	2.8	-	-

LOAD STORE QUEUE

DEST	TYPE	VSD	QSD	VAddr	QAddr	CONST	ADDR	VAL	FWD
ROB4	LD	-	-	0	-	9	-	-	-
ROB5	LD	-	-	0	-	18	-	-	-

INTEGER REGISTER FILE

R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	R13	R14	R15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FLOATING POINT REGISTER FILE

F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15
0	6.2	9.3	11.4	8.2	11.399999999999999	9.1	2.7	7.5	0	0	0	0	0	0	0

INTEGER RAT

R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	R13	R14	R15
R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	R13	R14	R15

FLOATING POINT RAT

F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15
F0	F1	F2	F3	F4	F5	ROB7	ROB0	ROB3	ROB1	ROB4	ROB5	F12	F13	F14	F15

The Floating Register file and RAT of Floating Register file at the end of 63CC.

19. Instruction - LD F8 7(R0) -

PC	INSTRUCTION	ISSUE	EX_S	EX_F	MEM_S	MEM_F	WB	COMMIT_S	COMMIT_F
0	LD F2 1(R0)	1	2	3	4	4	5	6	6
4	LD F3 10(R0)	2	4	5	6	6	7	8	8
8	ADD.D F1 F2 F3	3	8	28	-	-	29	30	30
12	LD F3 2(R0)	4	6	7	8	8	9	31	31
16	LD F4 11(R0)	6	8	9	10	10	11	32	32
20	ADD.D F2 F3 F4	7	12	32	-	-	33	34	34
24	LD F4 3(R0)	8	10	11	12	12	13	35	35
28	LD F5 12(R0)	10	12	13	14	14	15	36	36
32	ADD.D F3 F4 F5	11	16	36	-	-	37	38	38
36	LD F5 4(R0)	12	14	15	16	16	17	39	39
40	LD F6 13(R0)	14	16	17	18	18	19	40	40
44	ADD.D F4 F5 F6	30	31	51	-	-	52	53	53
48	LD F6 5(R0)	31	32	33	34	34	35	54	54
52	LD F7 14(R0)	32	34	35	36	36	38	55	55
56	ADD.D F5 F6 F7	34	39	59	-	-	60	61	61
60	LD F7 6(R0)	35	36	37	38	38	39	62	62
64	LD F8 15(R0)	36	38	39	40	40	41	63	63
68	ADD.D F6 F7 F8	38	42	62	-	-	63	64	64
72	LD F8 7(R0)	39	40	41	42	42	43	65	65

The load instruction was issued at 39CC then it executes for 2CC then it accesses the memory for 1CC then writebacks and commits.**ROB8** is freed.

20. Instruction - LD F9 16(R0) -

PC	INSTRUCTION	ISSUE	EX_S	EX_F	MEM_S	MEM_F	WB	COMMIT_S	COMMIT_F
0	LD F2 1(R0)	1	2	3	4	4	5	6	6
4	LD F3 10(R0)	2	4	5	6	6	7	8	8
8	ADD.D F1 F2 F3	3	8	28	-	-	29	30	30
12	LD F3 2(R0)	4	6	7	8	8	9	31	31
16	LD F4 11(R0)	6	8	9	10	10	11	32	32
20	ADD.D F2 F3 F4	7	12	32	-	-	33	34	34
24	LD F4 3(R0)	8	10	11	12	12	13	35	35
28	LD F5 12(R0)	10	12	13	14	14	15	36	36
32	ADD.D F3 F4 F5	11	16	36	-	-	37	38	38
36	LD F5 4(R0)	12	14	15	16	16	17	39	39
40	LD F6 13(R0)	14	16	17	18	18	19	40	40
44	ADD.D F4 F5 F6	30	31	51	-	-	52	53	53
48	LD F6 5(R0)	31	32	33	34	34	35	54	54
52	LD F7 14(R0)	32	34	35	36	36	38	55	55
56	ADD.D F5 F6 F7	34	39	59	-	-	60	61	61
60	LD F7 6(R0)	35	36	37	38	38	39	62	62
64	LD F8 15(R0)	36	38	39	40	40	41	63	63
68	ADD.D F6 F7 F8	38	42	62	-	-	63	64	64
72	LD F8 7(R0)	39	40	41	42	42	43	65	65
76	LD F9 16(R0)	40	42	43	44	44	45	66	66

The load instruction was issued at 40CC then it stalls for 2CC because the load store reservation station is full and it is available after the last instruction execution finishes at 41CC.It executes for 2CC then it accesses the memory for 1CC then writebacks and commits.**ROB9** is freed.

21. Instruction - ADD.D F7 F8 F9 -

PC	INSTRUCTION	ISSUE	EX_S	EX_F	MEM_S	MEM_F	WB	COMMIT_S	COMMIT_F
0	LD F2 1(R0)	1	2	3	4	4	5	6	6
4	LD F3 10(R0)	2	4	5	6	6	7	8	8
8	ADD.D F1 F2 F3	3	8	28	-	-	29	30	30
12	LD F3 2(R0)	4	6	7	8	8	9	31	31
16	LD F4 11(R0)	6	8	9	10	10	11	32	32
20	ADD.D F2 F3 F4	7	12	32	-	-	33	34	34
24	LD F4 3(R0)	8	10	11	12	12	13	35	35
28	LD F5 12(R0)	10	12	13	14	14	15	36	36
32	ADD.D F3 F4 F5	11	16	36	-	-	37	38	38
36	LD F5 4(R0)	12	14	15	16	16	17	39	39
40	LD F6 13(R0)	14	16	17	18	18	19	40	40
44	ADD.D F4 F5 F6	30	31	51	-	-	52	53	53
48	LD F6 5(R0)	31	32	33	34	34	35	54	54
52	LD F7 14(R0)	32	34	35	36	36	38	55	55
56	ADD.D F5 F6 F7	34	39	59	-	-	60	61	61
60	LD F7 6(R0)	35	36	37	38	38	39	62	62
64	LD F8 15(R0)	36	38	39	40	40	41	63	63
68	ADD.D F6 F7 F8	38	42	62	-	-	63	64	64
72	LD F8 7(R0)	39	40	41	42	42	43	65	65
76	LD F9 16(R0)	40	42	43	44	44	45	66	66
80	ADD.D F7 F8 F9	53	54	74	-	-	75	76	76

This ADD instruction is issued at 53CC; it doesn’t stall because F8 and F9 is already fetched before the issue of this instruction. Hence there is not any stall and hazard for this instruction. then it executes for 21CC after which writebacks to F7.

22. Instruction - LD F9 8(R0) -

PC	INSTRUCTION	ISSUE	EX_S	EX_F	MEM_S	MEM_F	WB	COMMIT_S	COMMIT_F
0	LD F2 1(R0)	1	2	3	4	4	5	6	6
4	LD F3 10(R0)	2	4	5	6	6	7	8	8
8	ADD.D F1 F2 F3	3	8	28	-	-	29	30	30
12	LD F3 2(R0)	4	6	7	8	8	9	31	31
16	LD F4 11(R0)	6	8	9	10	10	11	32	32
20	ADD.D F2 F3 F4	7	12	32	-	-	33	34	34
24	LD F4 3(R0)	8	10	11	12	12	13	35	35
28	LD F5 12(R0)	10	12	13	14	14	15	36	36
32	ADD.D F3 F4 F5	11	16	36	-	-	37	38	38
36	LD F5 4(R0)	12	14	15	16	16	17	39	39
40	LD F6 13(R0)	14	16	17	18	18	19	40	40
44	ADD.D F4 F5 F6	30	31	51	-	-	52	53	53
48	LD F6 5(R0)	31	32	33	34	34	35	54	54
52	LD F7 14(R0)	32	34	35	36	36	38	55	55
56	ADD.D F5 F6 F7	34	39	59	-	-	60	61	61
60	LD F7 6(R0)	35	36	37	38	38	39	62	62
64	LD F8 15(R0)	36	38	39	40	40	41	63	63
68	ADD.D F6 F7 F8	38	42	62	-	-	63	64	64
72	LD F8 7(R0)	39	40	41	42	42	43	65	65
76	LD F9 16(R0)	40	42	43	44	44	45	66	66
80	ADD.D F7 F8 F9	53	54	74	-	-	75	76	76
84	LD F9 8(R0)	54	55	56	57	57	58	77	77

The load instruction was issued at 54CC then it executes for 2CC then it accesses the memory for 1CC then writebacks and commits.**ROB1** is freed.

23. Instruction - LD F10 17(R0) -

PC	INSTRUCTION	ISSUE	EX_S	EX_F	MEM_S	MEM_F	WB	COMMIT_S	COMMIT_F
0	LD F2 1(R0)	1	2	3	4	4	5	6	6
4	LD F3 10(R0)	2	4	5	6	6	7	8	8
8	ADD.D F1 F2 F3	3	8	28	-	-	29	30	30
12	LD F3 2(R0)	4	6	7	8	8	9	31	31
16	LD F4 11(R0)	6	8	9	10	10	11	32	32
20	ADD.D F2 F3 F4	7	12	32	-	-	33	34	34
24	LD F4 3(R0)	8	10	11	12	12	13	35	35
28	LD F5 12(R0)	10	12	13	14	14	15	36	36
32	ADD.D F3 F4 F5	11	16	36	-	-	37	38	38
36	LD F5 4(R0)	12	14	15	16	16	17	39	39
40	LD F6 13(R0)	14	16	17	18	18	19	40	40
44	ADD.D F4 F5 F6	30	31	51	-	-	52	53	53
48	LD F6 5(R0)	31	32	33	34	34	35	54	54
52	LD F7 14(R0)	32	34	35	36	36	38	55	55
56	ADD.D F5 F6 F7	34	39	59	-	-	60	61	61
60	LD F7 6(R0)	35	36	37	38	38	39	62	62
64	LD F8 15(R0)	36	38	39	40	40	41	63	63
68	ADD.D F6 F7 F8	38	42	62	-	-	63	64	64
72	LD F8 7(R0)	39	40	41	42	42	43	65	65
76	LD F9 16(R0)	40	42	43	44	44	45	66	66
80	ADD.D F7 F8 F9	53	54	74	-	-	75	76	76
84	LD F9 8(R0)	54	55	56	57	57	58	77	77
88	LD F10 17(R0)	55	57	58	59	59	61	78	78

ROB2	yes	LD F10 17(R0)	EX	F10	-

The load instruction was issued at 55CC then it stalls for 2CC because the previous load instruction was accessing memory at 56cc. It executes for 2CC then it accesses the memory for 1CC then writebacks and commits.**ROB2** is freed.

24. Instruction - ADD.D F8 F9 F10 -

PC	INSTRUCTION	ISSUE	EX_S	EX_F	MEM_S	MEM_F	WB	COMMIT_S	COMMIT_F
0	LD F2 1(R0)	1	2	3	4	4	5	6	6
4	LD F3 10(R0)	2	4	5	6	6	7	8	8
8	ADD.D F1 F2 F3	3	8	28	-	-	29	30	30
12	LD F3 2(R0)	4	6	7	8	8	9	31	31
16	LD F4 11(R0)	6	8	9	10	10	11	32	32
20	ADD.D F2 F3 F4	7	12	32	-	-	33	34	34
24	LD F4 3(R0)	8	10	11	12	12	13	35	35
28	LD F5 12(R0)	10	12	13	14	14	15	36	36
32	ADD.D F3 F4 F5	11	16	36	-	-	37	38	38
36	LD F5 4(R0)	12	14	15	16	16	17	39	39
40	LD F6 13(R0)	14	16	17	18	18	19	40	40
44	ADD.D F4 F5 F6	30	31	51	-	-	52	53	53
48	LD F6 5(R0)	31	32	33	34	34	35	54	54
52	LD F7 14(R0)	32	34	35	36	36	38	55	55
56	ADD.D F5 F6 F7	34	39	59	-	-	60	61	61
60	LD F7 6(R0)	35	36	37	38	38	39	62	62
64	LD F8 15(R0)	36	38	39	40	40	41	63	63
68	ADD.D F6 F7 F8	38	42	62	-	-	63	64	64
72	LD F8 7(R0)	39	40	41	42	42	43	65	65
76	LD F9 16(R0)	40	42	43	44	44	45	66	66
80	ADD.D F7 F8 F9	53	54	74	-	-	75	76	76
84	LD F9 8(R0)	54	55	56	57	57	58	77	77
88	LD F10 17(R0)	55	57	58	59	59	61	78	78
92	ADD.D F8 F9 F10	61	62	82	-	-	83	84	84

This ADD instruction is issued at 61CC; it doesn't stalls because F9 and F10 is already fetched before the issue of this instruction i.e. at 56CC and 57CC respectively. Hence there is not any stall and hazard for this instruction. then it executes for 21CC after which writebacks to F8.

25. Instruction - LD F10 9(R0) -

PC	INSTRUCTION	ISSUE	EX_S	EX_F	MEM_S	MEM_F	WB	COMMIT_S	COMMIT_F
0	LD F2 1(R0)	1	2	3	4	4	5	6	6
4	LD F3 10(R0)	2	4	5	6	6	7	8	8
8	ADD.D F1 F2 F3	3	8	28	-	-	29	30	30
12	LD F3 2(R0)	4	6	7	8	8	9	31	31
16	LD F4 11(R0)	6	8	9	10	10	11	32	32
20	ADD.D F2 F3 F4	7	12	32	-	-	33	34	34
24	LD F4 3(R0)	8	10	11	12	12	13	35	35
28	LD F5 12(R0)	10	12	13	14	14	15	36	36
32	ADD.D F3 F4 F5	11	16	36	-	-	37	38	38
36	LD F5 4(R0)	12	14	15	16	16	17	39	39
40	LD F6 13(R0)	14	16	17	18	18	19	40	40
44	ADD.D F4 F5 F6	30	31	51	-	-	52	53	53
48	LD F6 5(R0)	31	32	33	34	34	35	54	54
52	LD F7 14(R0)	32	34	35	36	36	38	55	55
56	ADD.D F5 F6 F7	34	39	59	-	-	60	61	61
60	LD F7 6(R0)	35	36	37	38	38	39	62	62
64	LD F8 15(R0)	36	38	39	40	40	41	63	63
68	ADD.D F6 F7 F8	38	42	62	-	-	63	64	64
72	LD F8 7(R0)	39	40	41	42	42	43	65	65
76	LD F9 16(R0)	40	42	43	44	44	45	66	66
80	ADD.D F7 F8 F9	53	54	74	-	-	75	76	76
84	LD F9 8(R0)	54	55	56	57	57	58	77	77
88	LD F10 17(R0)	55	57	58	59	59	61	78	78
92	ADD.D F8 F9 F10	61	62	82	-	-	83	84	84
96	LD F10 9(R0)	62	63	64	65	65	66	85	85

ROB4	no	LD F10 9(R0)	WB	F10	0.6

The load instruction was issued at 62CC then it starts executing at 63CC(after issue). It executes for 2CC then it accesses the memory for 1CC then writebacks and commits.**ROB4** is freed.

26. Instruction - LD F11 18(R0) -

PC	INSTRUCTION	ISSUE	EX_S	EX_F	MEM_S	MEM_F	WB	COMMIT_S	COMMIT_F
0	LD F2 1(R0)	1	2	3	4	4	5	6	6
4	LD F3 10(R0)	2	4	5	6	6	7	8	8
8	ADD.D F1 F2 F3	3	8	28	-	-	29	30	30
12	LD F3 2(R0)	4	6	7	8	8	9	31	31
16	LD F4 11(R0)	6	8	9	10	10	11	32	32
20	ADD.D F2 F3 F4	7	12	32	-	-	33	34	34
24	LD F4 3(R0)	8	10	11	12	12	13	35	35
28	LD F5 12(R0)	10	12	13	14	14	15	36	36
32	ADD.D F3 F4 F5	11	16	36	-	-	37	38	38
36	LD F5 4(R0)	12	14	15	16	16	17	39	39
40	LD F6 13(R0)	14	16	17	18	18	19	40	40
44	ADD.D F4 F5 F6	30	31	51	-	-	52	53	53
48	LD F6 5(R0)	31	32	33	34	34	35	54	54
52	LD F7 14(R0)	32	34	35	36	36	38	55	55
56	ADD.D F5 F6 F7	34	39	59	-	-	60	61	61
60	LD F7 6(R0)	35	36	37	38	38	39	62	62
64	LD F8 15(R0)	36	38	39	40	40	41	63	63
68	ADD.D F6 F7 F8	38	42	62	-	-	63	64	64
72	LD F8 7(R0)	39	40	41	42	42	43	65	65
76	LD F9 16(R0)	40	42	43	44	44	45	66	66
80	ADD.D F7 F8 F9	53	54	74	-	-	75	76	76
84	LD F9 8(R0)	54	55	56	57	57	58	77	77
88	LD F10 17(R0)	55	57	58	59	59	61	78	78
92	ADD.D F8 F9 F10	61	62	82	-	-	83	84	84
96	LD F10 9(R0)	62	63	64	65	65	66	85	85
100	LD F11 18(R0)	63	65	66	67	67	68	86	86

ROB5	no	LD F11 18(R0)	WB	F11	9.7

The load instruction was issued at 63CC then it stalls for 2CC because the previous load instruction was accessing memory at 64cc. It executes for 2CC then it accesses the memory for 1CC then writebacks and commits.**ROB5** is freed.

27. Instruction - ADD.D F9 F10 F11 -

PC	INSTRUCTION	ISSUE	EX_S	EX_F	MEM_S	MEM_F	WB	COMMIT_S	COMMIT_F
0	LD F2 1(R0)	1	2	3	4	4	5	6	6
4	LD F3 10(R0)	2	4	5	6	6	7	8	8
8	ADD.D F1 F2 F3	3	8	28	-	-	29	30	30
12	LD F3 2(R0)	4	6	7	8	8	9	31	31
16	LD F4 11(R0)	6	8	9	10	10	11	32	32
20	ADD.D F2 F3 F4	7	12	32	-	-	33	34	34
24	LD F4 3(R0)	8	10	11	12	12	13	35	35
28	LD F5 12(R0)	10	12	13	14	14	15	36	36
32	ADD.D F3 F4 F5	11	16	36	-	-	37	38	38
36	LD F5 4(R0)	12	14	15	16	16	17	39	39
40	LD F6 13(R0)	14	16	17	18	18	19	40	40
44	ADD.D F4 F5 F6	30	31	51	-	-	52	53	53
48	LD F6 5(R0)	31	32	33	34	34	35	54	54
52	LD F7 14(R0)	32	34	35	36	36	38	55	55
56	ADD.D F5 F6 F7	34	39	59	-	-	60	61	61
60	LD F7 6(R0)	35	36	37	38	38	39	62	62
64	LD F8 15(R0)	36	38	39	40	40	41	63	63
68	ADD.D F6 F7 F8	38	42	62	-	-	63	64	64
72	LD F8 7(R0)	39	40	41	42	42	43	65	65
76	LD F9 16(R0)	40	42	43	44	44	45	66	66
80	ADD.D F7 F8 F9	53	54	74	-	-	75	76	76
84	LD F9 8(R0)	54	55	56	57	57	58	77	77
88	LD F10 17(R0)	55	57	58	59	59	61	78	78
92	ADD.D F8 F9 F10	61	62	82	-	-	83	84	84
96	LD F10 9(R0)	62	63	64	65	65	66	85	85
100	LD F11 18(R0)	63	65	66	67	67	68	86	86
104	ADD.D F9 F10 F11	64	69	89	-	-	90	91	91

This ADD instruction is issued at 64CC; it stalls until 69CC until F11 is fetched(at 68CC); these stalls ensure that there isn't any **RAW** hazard.then it executes for 21CC after which writebacks to F9.

5. Hazards -

Total RAW hazards - 18
Total WAW hazards - 9
Total WAR hazards - 9

RAW, WAR , WAW are possible but handled in instructions,We will stall the instruction where ever any of these three situations arise:

1. RAW

Writing to a FP register when someone occupied it for reading

2. WAR

Reading from a register which has been occupied for writing by another instruction

3. WAW

Writing to a register when some other instruction is writing to it.

We handle all these by giving a flag that if the destination register is already occupied or not.

6. Number of Clock Cycles -

Total number of cycle tomasulo's algorithm takes is 91CC to add two 3X3 matrices.
Without the implementation of tomasulo's algorithm it would have taken 333CC.

6. Conclusion -

Tomasulo's Algorithm reduces the execution time drastically and completely eliminates the WAW and WAR hazards and minimizes the RAW hazards with stalls