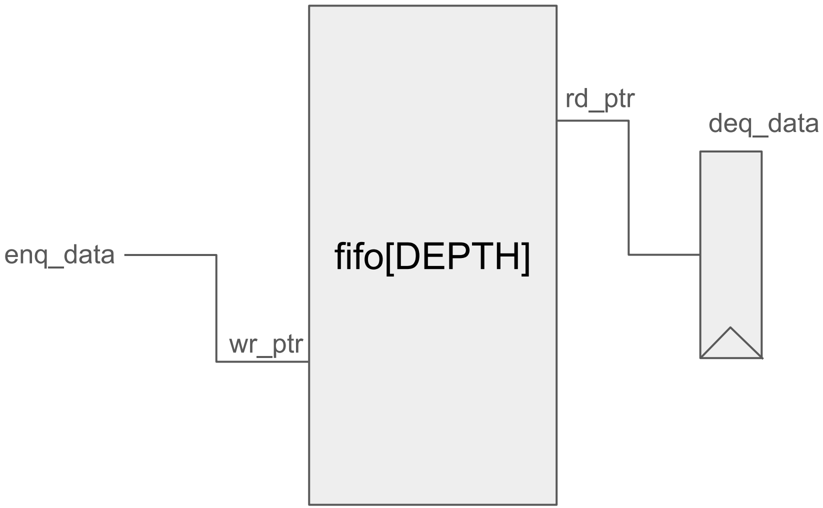
# mp_ooo CP1

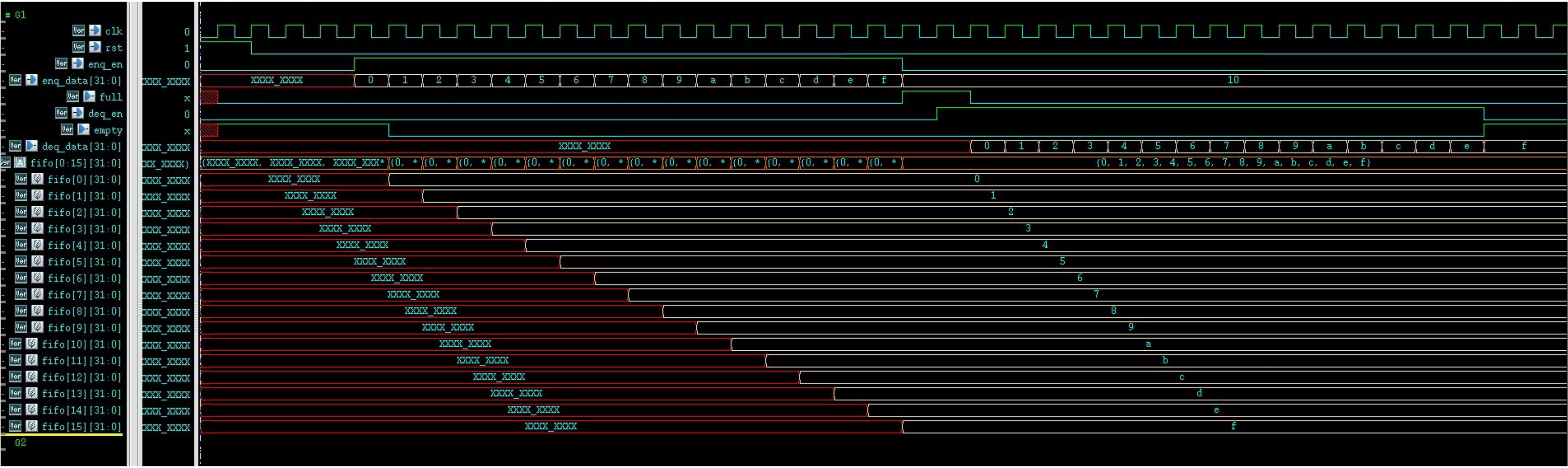
Team I_Wanna_Drop

# Instruction Queue (FIFO) w/ bit-extension technique

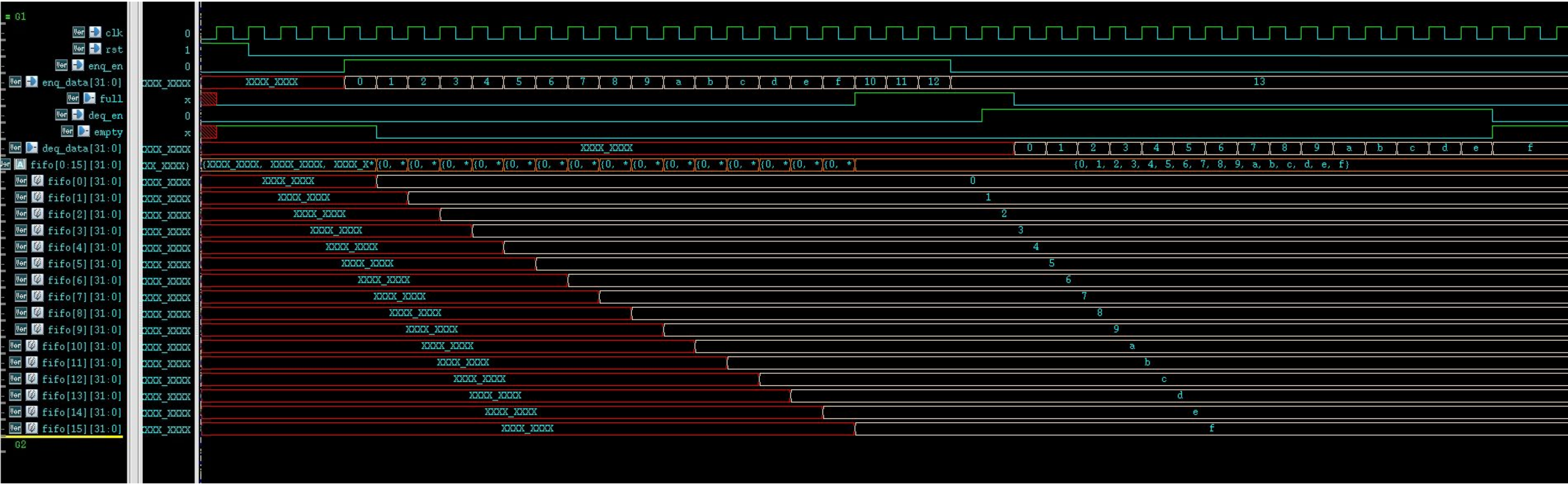
```
always_ff @(posedge clk) begin
    if (rst) begin
        wr_ptr <= '0;
        rd_ptr <= '0;
    end else begin
        if (enq_en && ~full) begin
            fifo[wr_ptr_actual] <= enq_data;
            wr_ptr <= (ADDR_IDX+1)'(wr_ptr + 1);
        end
        if (deq_en && ~empty) begin
            deq_data <= fifo[rd_ptr_actual];
            rd_ptr <= (ADDR_IDX+1)'(rd_ptr + 1);
        end
    end
end
assign empty = (wr_ptr == rd_ptr);
assign full = (wr_ptr_actual == rd_ptr_actual) &&
(wr_ptr_flag == ~rd_ptr_flag);
```



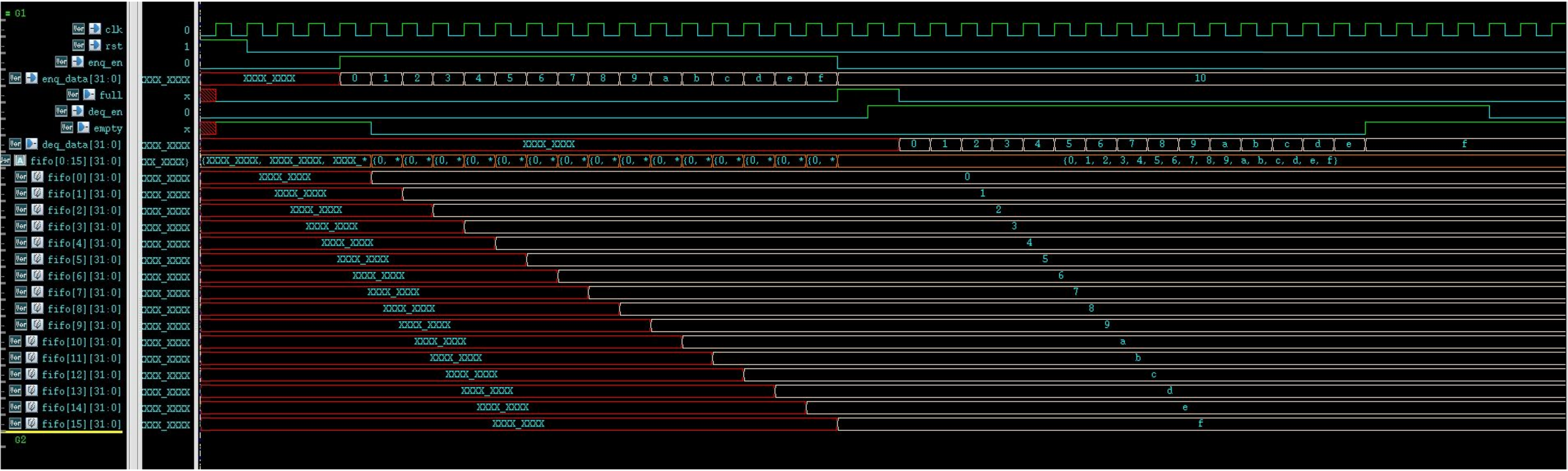
# FIFO Test 1 - Enque and then deque everything


G1
clk 0
rst 1
enq_en 0
enq_data[31:0] xxx_xxxx
XXXX_XXXX 0 1 2 3 4 5 6 7 8 9 a b c d e f 10
full x
deq_en 0
empty x
deq_data[31:0] xxx_xxxx
XXXX_XXXX 0 1 2 3 4 5 6 7 8 9 a b c d e f
fifo[0:15][31:0] xx_xxxx}
{XXXX_XXXX, XXXX_XXXX, XXXX_XXX*
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f}
fifo[0][31:0] XXXX_XXXX 0
fifo[1][31:0] XXXX_XXXX 1
fifo[2][31:0] XXXX_XXXX 2
fifo[3][31:0] XXXX_XXXX 3
fifo[4][31:0] XXXX_XXXX 4
fifo[5][31:0] XXXX_XXXX 5
fifo[6][31:0] XXXX_XXXX 6
fifo[7][31:0] XXXX_XXXX 7
fifo[8][31:0] XXXX_XXXX 8
fifo[9][31:0] XXXX_XXXX 9
fifo[10][31:0] XXXX_XXXX a
fifo[11][31:0] XXXX_XXXX b
fifo[12][31:0] XXXX_XXXX c
fifo[13][31:0] XXXX_XXXX d
fifo[14][31:0] XXXX_XXXX e
fifo[15][31:0] XXXX_XXXX f
G2

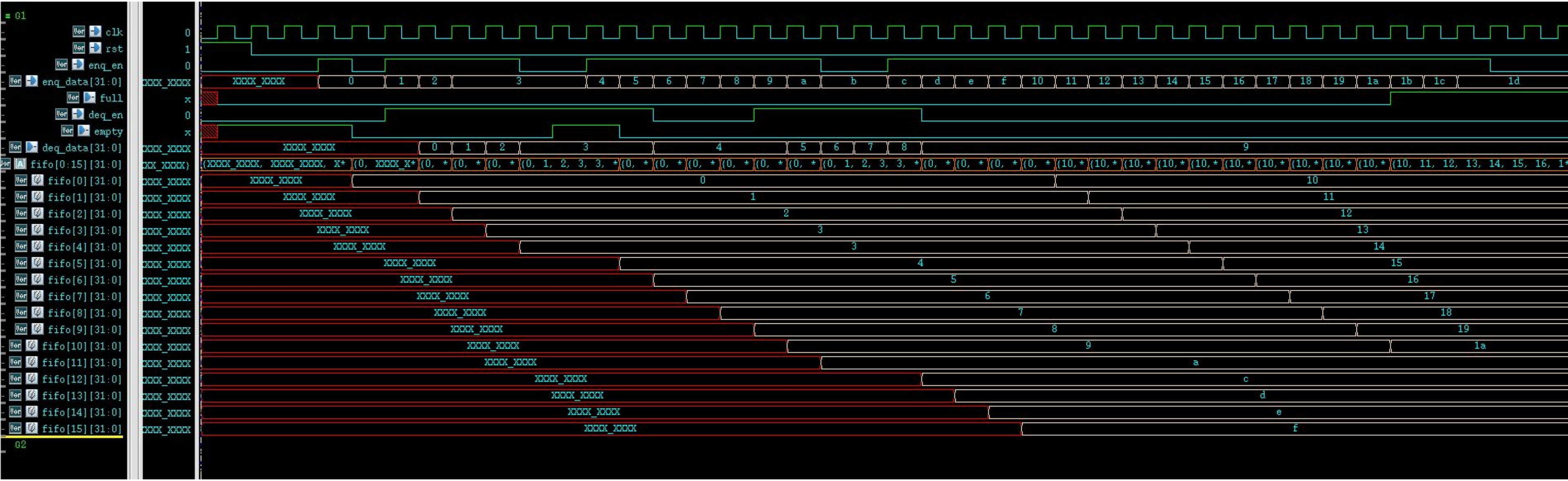
# FIFO Test 2 - FIFO already full


G1
clk 0
rst 1
enq_en 0
enq_data[31:0] xxx_xxxx
XXXX_XXXX 0 1 2 3 4 5 6 7 8 9 a b c d e f 10 11 12 13
full x
deq_en 0
empty x
deq_data[31:0] xxx_xxxx
XXXX_XXXX 0 1 2 3 4 5 6 7 8 9 a b c d e f
fifo[0:15][31:0] xx_xxxx}
{XXXX_XXXX, XXXX_XXXX, XXXX_X* {0, * {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f}
fifo[0][31:0] XXXX_XXXX 0
fifo[1][31:0] XXXX_XXXX 1
fifo[2][31:0] XXXX_XXXX 2
fifo[3][31:0] XXXX_XXXX 3
fifo[4][31:0] XXXX_XXXX 4
fifo[5][31:0] XXXX_XXXX 5
fifo[6][31:0] XXXX_XXXX 6
fifo[7][31:0] XXXX_XXXX 7
fifo[8][31:0] XXXX_XXXX 8
fifo[9][31:0] XXXX_XXXX 9
fifo[10][31:0] XXXX_XXXX a
fifo[11][31:0] XXXX_XXXX b
fifo[12][31:0] XXXX_XXXX c
fifo[13][31:0] XXXX_XXXX d
fifo[14][31:0] XXXX_XXXX e
fifo[15][31:0] XXXX_XXXX f
G2

# FIFO Test 3 - FIFO already empty

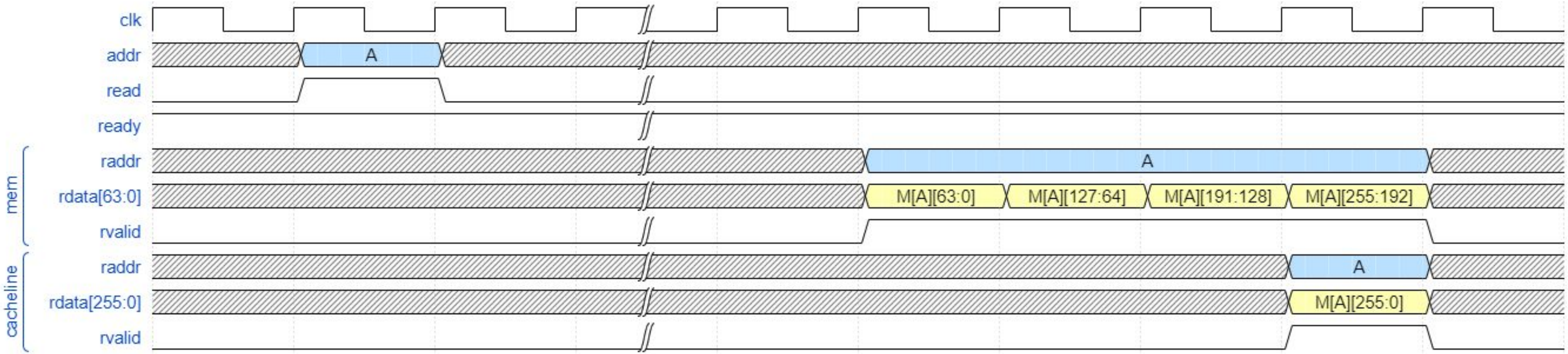

G1
clk 0
rst 1
enq_en 0
enq_data[31:0] xxx_xxxx
XXXX_XXXX 0 1 2 3 4 5 6 7 8 9 a b c d e f 10
full x
deq_en 0
empty x
deq_data[31:0] xxx_xxxx
XXXX_XXXX 0 1 2 3 4 5 6 7 8 9 a b c d e f
fifo[0:15][31:0] xx_xxxx}
{XXXX_XXXX, XXXX_XXXX, XXXX_*
{0, *
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f}
fifo[0][31:0] xxx_xxxx XXXX_XXXX 0
fifo[1][31:0] xxx_xxxx XXXX_XXXX 1
fifo[2][31:0] xxx_xxxx XXXX_XXXX 2
fifo[3][31:0] xxx_xxxx XXXX_XXXX 3
fifo[4][31:0] xxx_xxxx XXXX_XXXX 4
fifo[5][31:0] xxx_xxxx XXXX_XXXX 5
fifo[6][31:0] xxx_xxxx XXXX_XXXX 6
fifo[7][31:0] xxx_xxxx XXXX_XXXX 7
fifo[8][31:0] xxx_xxxx XXXX_XXXX 8
fifo[9][31:0] xxx_xxxx XXXX_XXXX 9
fifo[10][31:0] xxx_xxxx XXXX_XXXX a
fifo[11][31:0] xxx_xxxx XXXX_XXXX b
fifo[12][31:0] xxx_xxxx XXXX_XXXX c
fifo[13][31:0] xxx_xxxx XXXX_XXXX d
fifo[14][31:0] xxx_xxxx XXXX_XXXX e
fifo[15][31:0] xxx_xxxx XXXX_XXXX f
G2

# FIFO Test 4 - Mixed enque and deque

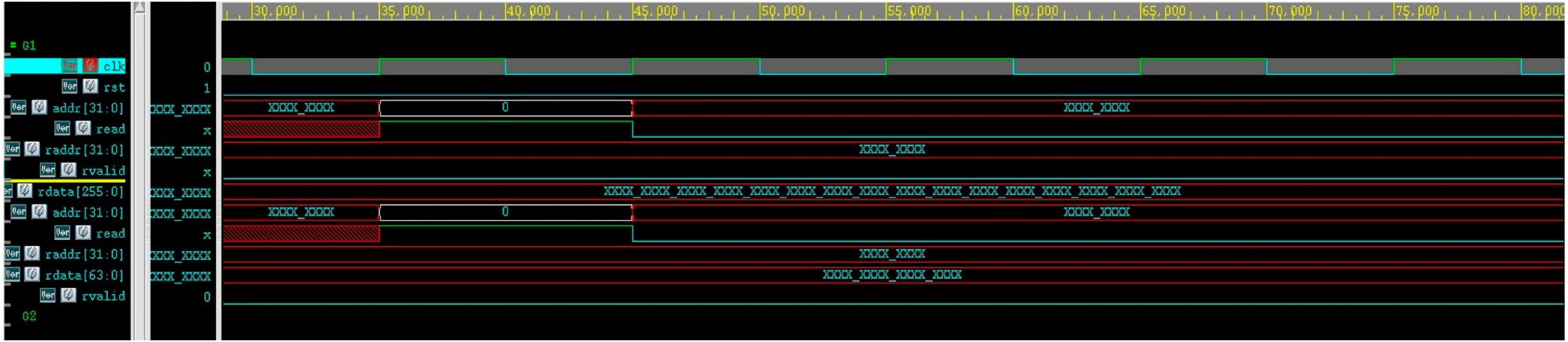

G1
clk 0
rst 1
enq_en 0
enq_data[31:0] xxx_xxxx
XXXX_XXXX 0 1 2 3 4 5 6 7 8 9 a b c d e f 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d
full x
deq_en 0
empty x
deq_data[31:0] xxx_xxxx
XXXX_XXXX 0 1 2 3 4 5 6 7 8 9
fifo[0:15][31:0] xx_xxxx}
{XXXX_XXXX, XXXX_XXXX, X* {0, XXXX_X* {0, * {0, 1, 2, 3, 3, * {10, * {10, 11, 12, 13, 14, 15, 16, 1*
fifo[0][31:0] XXXX_XXXX 0 10
fifo[1][31:0] XXXX_XXXX 1 11
fifo[2][31:0] XXXX_XXXX 2 12
fifo[3][31:0] XXXX_XXXX 3 13
fifo[4][31:0] XXXX_XXXX 3 14
fifo[5][31:0] XXXX_XXXX 4 15
fifo[6][31:0] XXXX_XXXX 5 16
fifo[7][31:0] XXXX_XXXX 6 17
fifo[8][31:0] XXXX_XXXX 7 18
fifo[9][31:0] XXXX_XXXX 8 19
fifo[10][31:0] XXXX_XXXX 9 1a
fifo[11][31:0] XXXX_XXXX a
fifo[12][31:0] XXXX_XXXX c
fifo[13][31:0] XXXX_XXXX d
fifo[14][31:0] XXXX_XXXX e
fifo[15][31:0] XXXX_XXXX f
G2

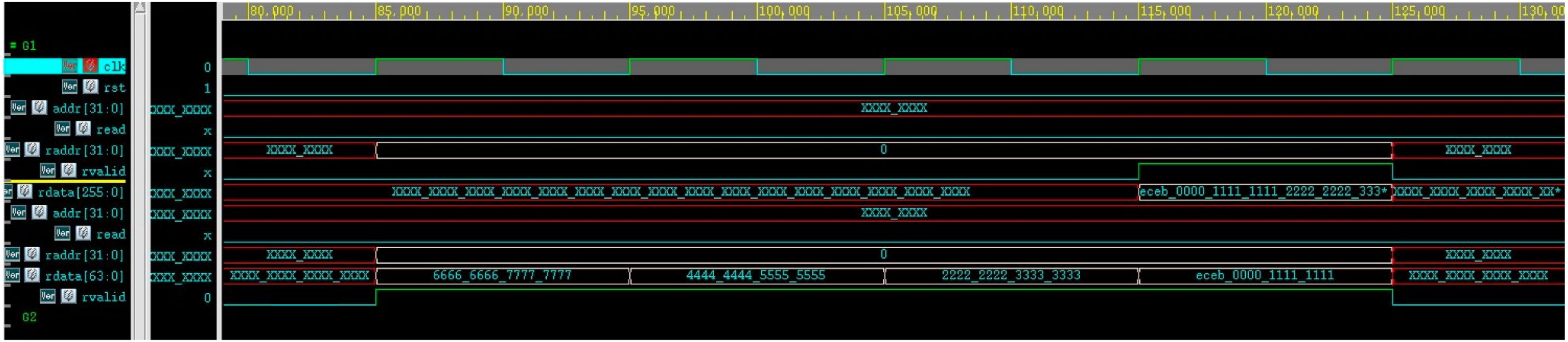
# Cacheline Adapter

Translate from burst interface to non-burst interface.
Giving cache the illusion that it's accessing a non-burst memory.
However we did not translate back to the interface in mp_cache, instead we preserve the read/write-ready handshake.
Cache code is slightly modified to adapt to the new read/write-ready handshake protocol though.

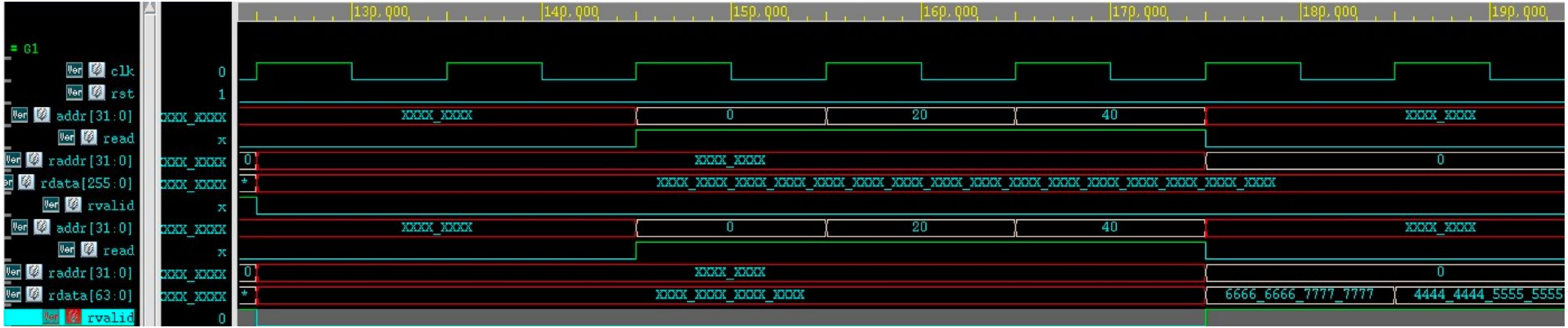


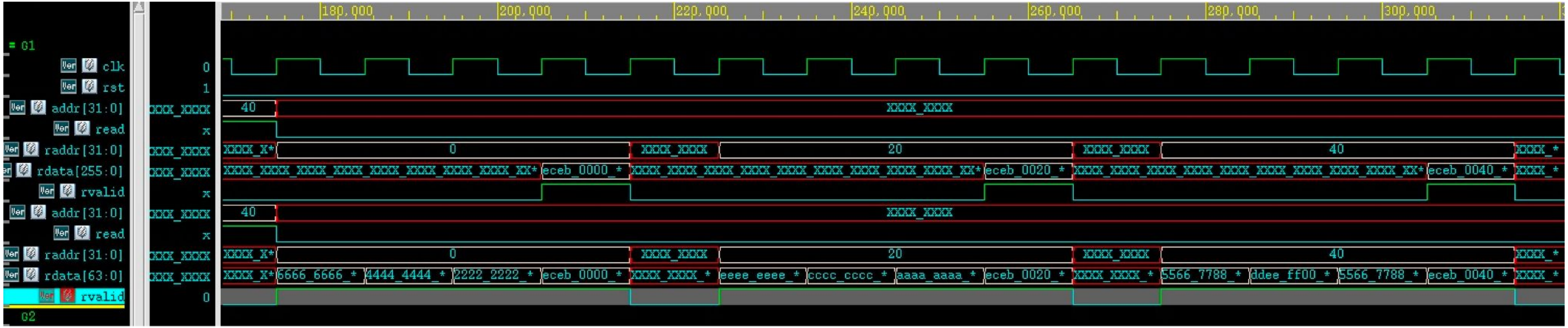
# Cacheline Adapter Test - Single read



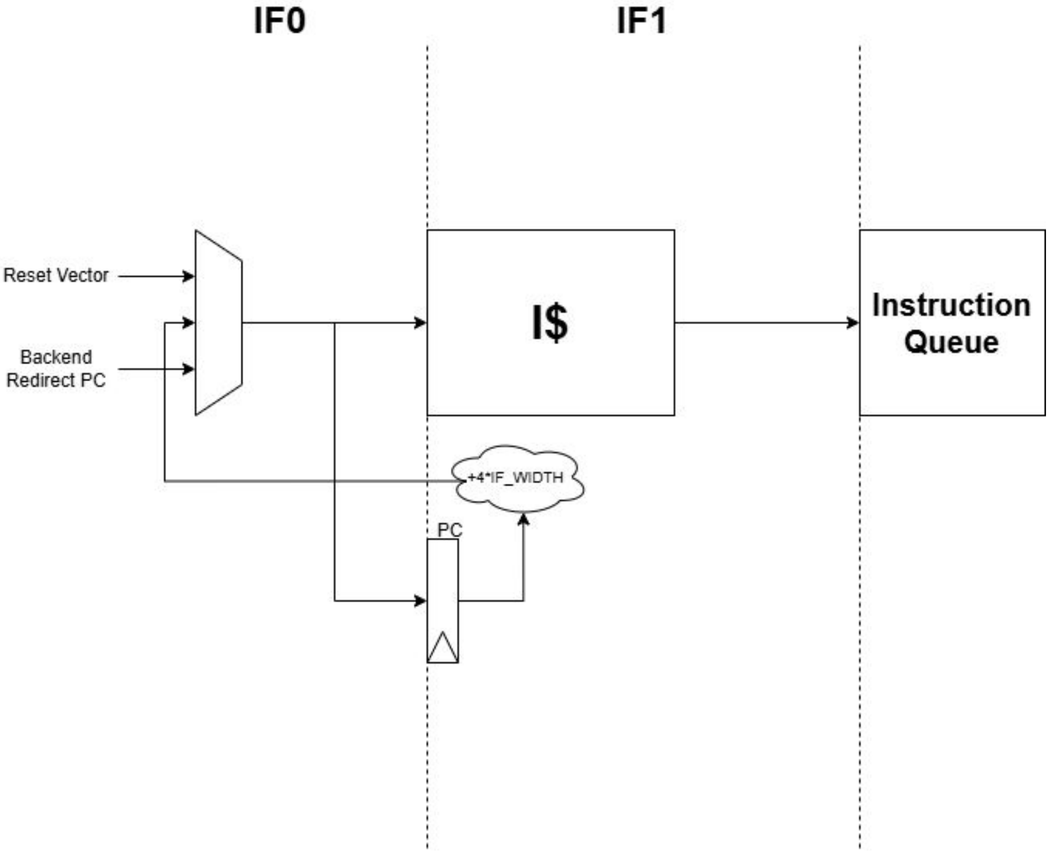


# Cacheline Adapter Test - Consecutive read

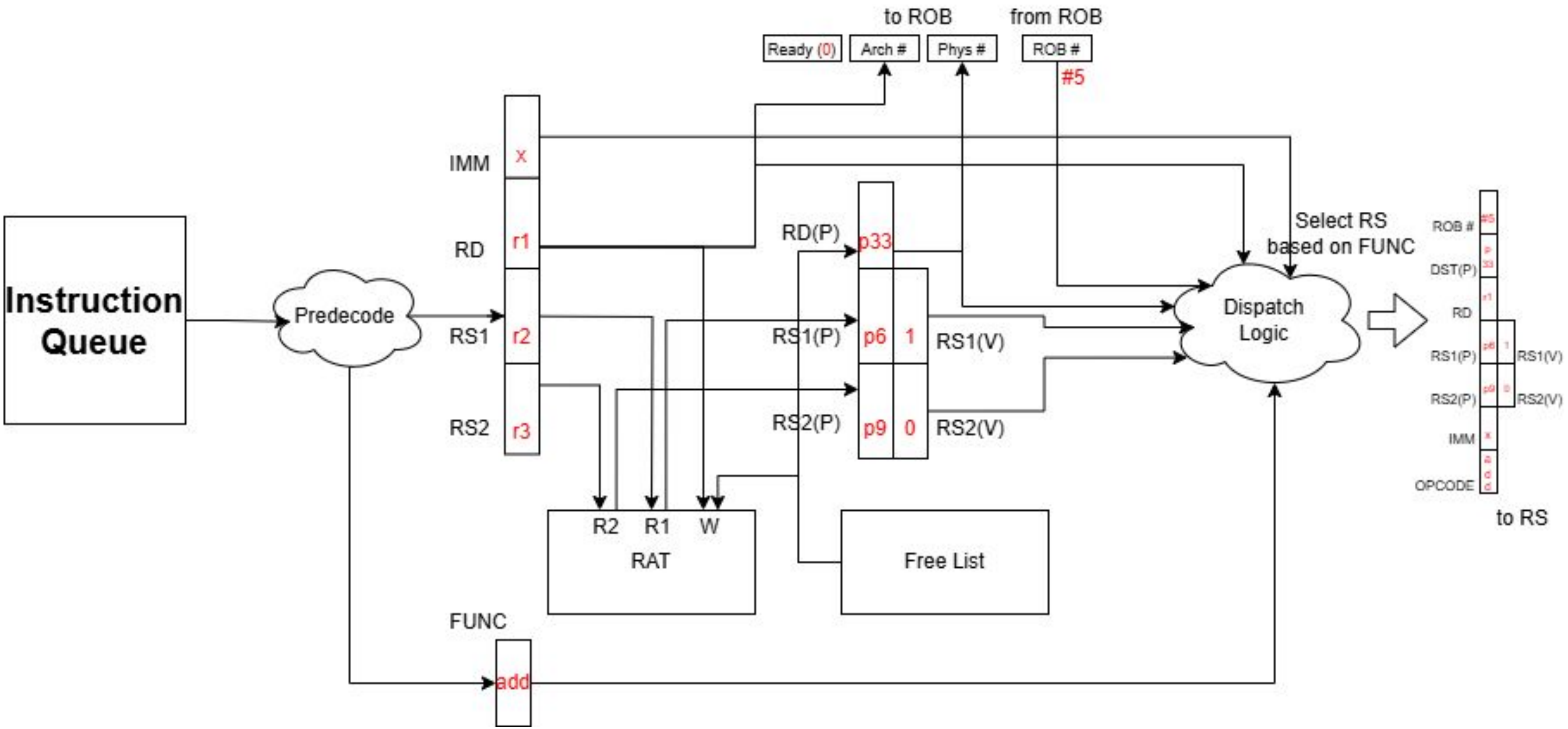




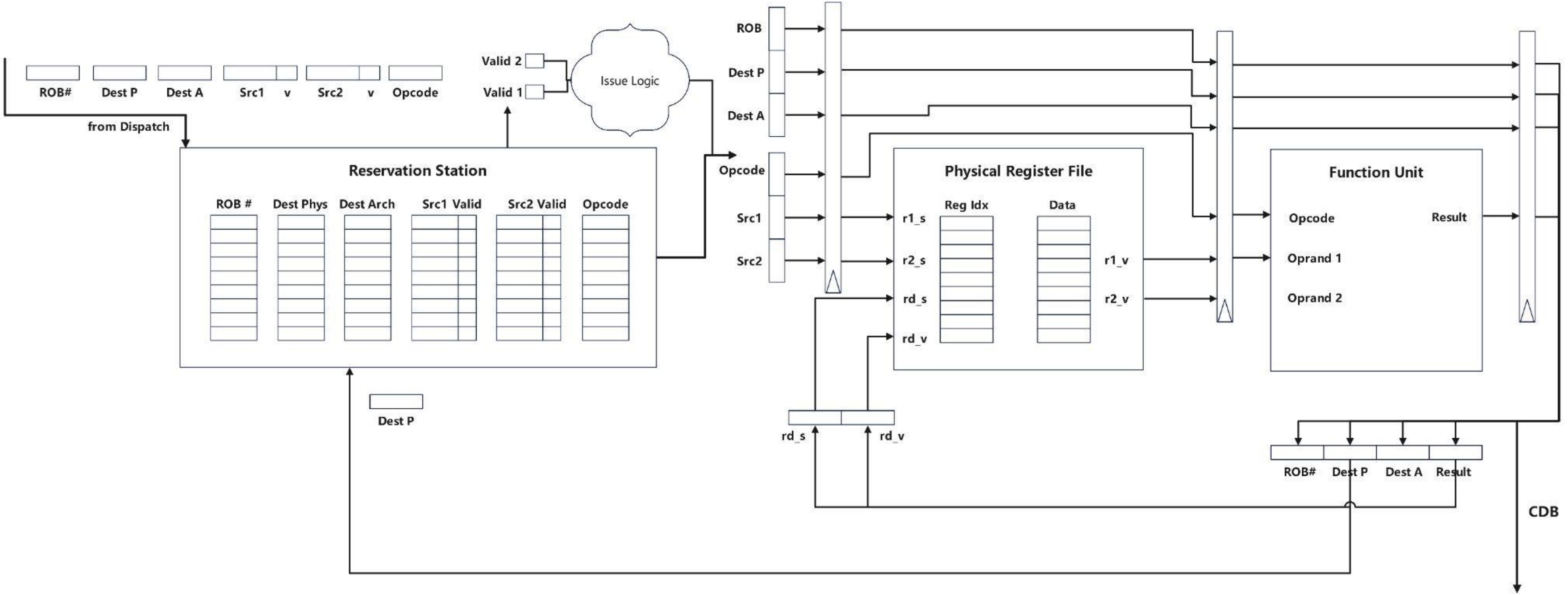
# Frontend


IF0
IF1
Reset Vector
Backend
Redirect PC
I$
Instruction
Queue
+4*IF_WIDTH
PC

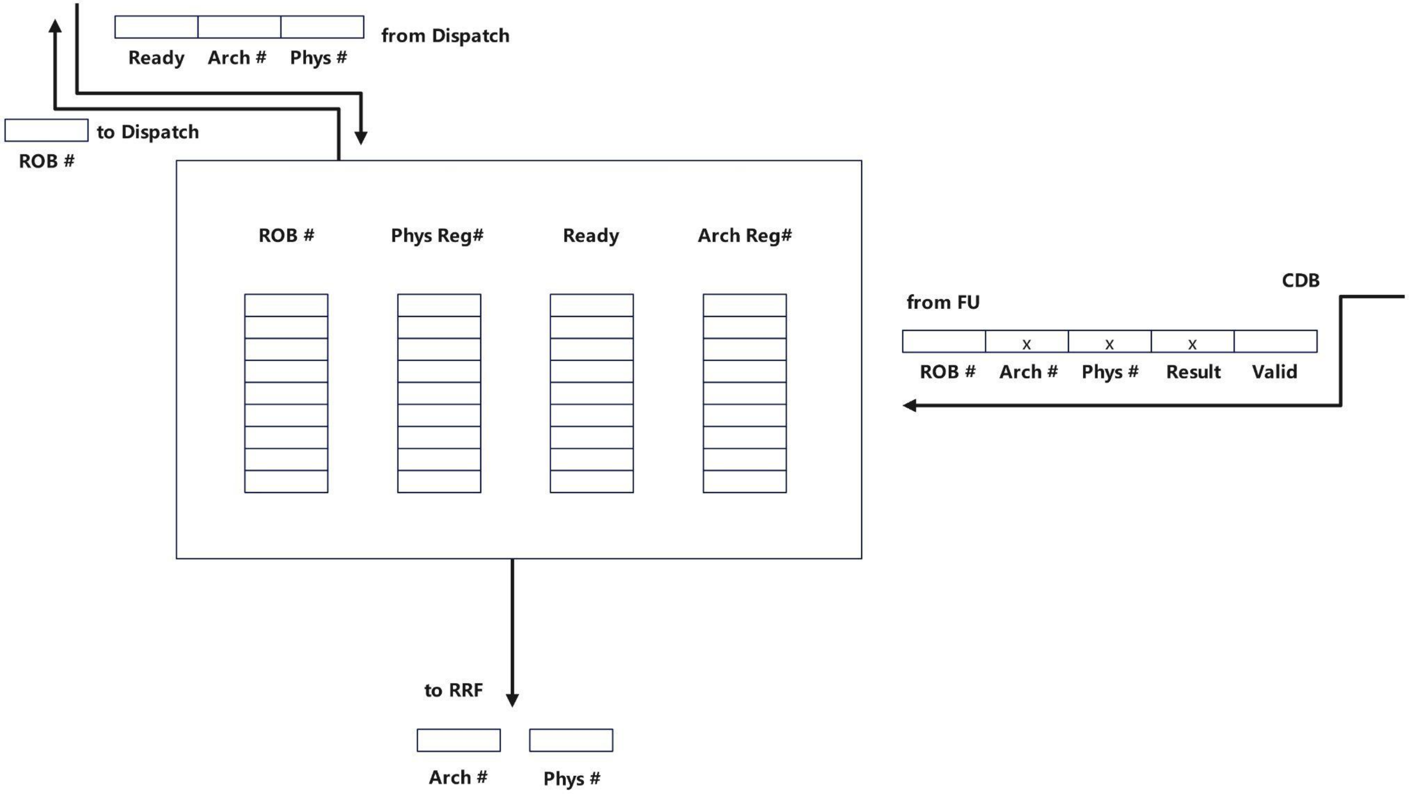
# Decode & Rename & Dispatch



# Issue & Execute & Writeback (wakeup)



# ROB


from Dispatch
Ready
Arch #
Phys #
to Dispatch
ROB #
ROB #
Phys Reg#
Ready
Arch Reg#
CDB
from FU
x
x
x
ROB #
Arch #
Phys #
Result
Valid
to RRF
Arch #
Phys #