NATIONAL YANG MING CHIAO TUNG UNIVERSITY

INTERNATIONAL COLLEGE OF

SEMICONDUCTOR TECHNOLOGY

NGUYEN TIEN DAT 313593001

EXERCISE 2

Report of Memory Circuits and System

INTENSE Program

(Master Degree)

Hsinchu - 2025

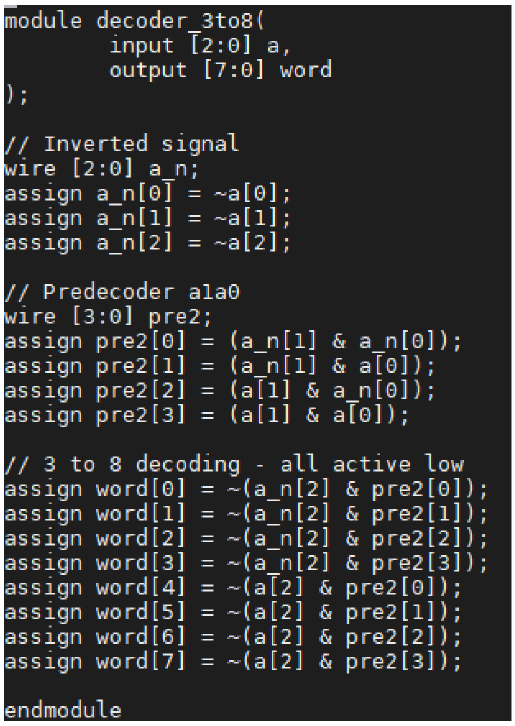
Contents

List of Figures

List of Tables

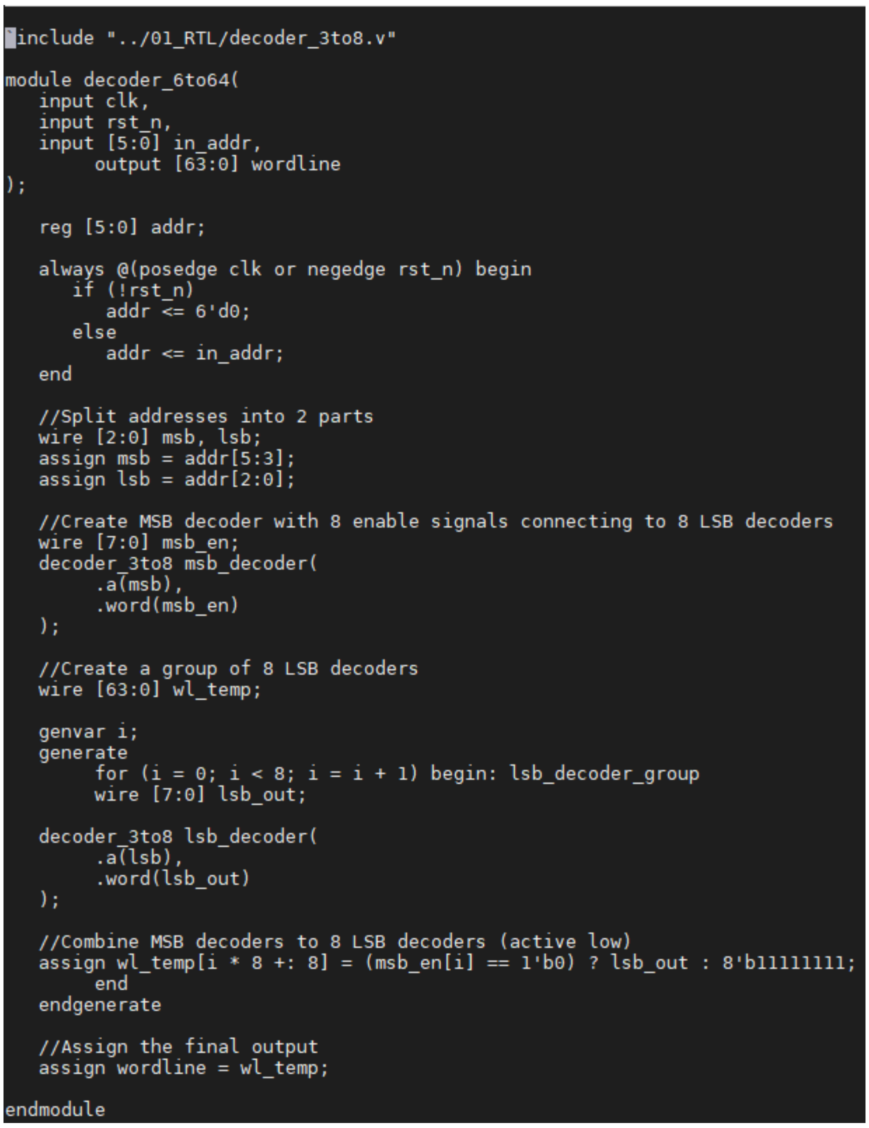
Chapter 1 Circuit architecture

A decoder is essential for selecting a specific word line for reading or writing operations. It accomplishes this by reading and decoding the row address, thereby determining the appropriate row to target. This functionality significantly reduces the need for multiple select signals. For instance, in a system with 64 word lines, instead of requiring a 64-bit signal for row selection, a mere 6-bit signal suffices to accurately identify the target row. To decode from 6bit to 64bit, i use 9 block of 3to8 decoder, the architecture shownd below. I use Verilog language to write 3to8 decoder and 6to64 decoder.



```
module decoder_3to8(
        input [2:0] a,
        output [7:0] word
);

// Inverted signal
wire [2:0] a_n;
assign a_n[0] = ~a[0];
assign a_n[1] = ~a[1];
assign a_n[2] = ~a[2];

// Predecoder a1a0
wire [3:0] pre2;
assign pre2[0] = (a_n[1] & a_n[0]);
assign pre2[1] = (a_n[1] & a[0]);
assign pre2[2] = (a[1] & a_n[0]);
assign pre2[3] = (a[1] & a[0]);

// 3 to 8 decoding - all active low
assign word[0] = ~(a_n[2] & pre2[0]);
assign word[1] = ~(a_n[2] & pre2[1]);
assign word[2] = ~(a_n[2] & pre2[2]);
assign word[3] = ~(a_n[2] & pre2[3]);
assign word[4] = ~(a[2] & pre2[0]);
assign word[5] = ~(a[2] & pre2[1]);
assign word[6] = ~(a[2] & pre2[2]);
assign word[7] = ~(a[2] & pre2[3]);

endmodule
```

Figure 1.1: 3to8 decoder block.



```
`include "../01_RTL/decoder_3to8.v"

module decoder_6to64(
   input clk,
   input rst_n,
   input [5:0] in_addr,
        output [63:0] wordline
);

   reg [5:0] addr;

   always @(posedge clk or negedge rst_n) begin
      if (!rst_n)
         addr <= 6'd0;
      else
         addr <= in_addr;
   end

   //Split addresses into 2 parts
   wire [2:0] msb, lsb;
   assign msb = addr[5:3];
   assign lsb = addr[2:0];

   //Create MSB decoder with 8 enable signals connecting to 8 LSB decoders
   wire [7:0] msb_en;
   decoder_3to8 msb_decoder(
        .a(msb),
        .word(msb_en)
   );

   //Create a group of 8 LSB decoders
   wire [63:0] wl_temp;

   genvar i;
   generate
        for (i = 0; i < 8; i = i + 1) begin: lsb_decoder_group
        wire [7:0] lsb_out;

   decoder_3to8 lsb_decoder(
        .a(lsb),
        .word(lsb_out)
   );

   //Combine MSB decoders to 8 LSB decoders (active low)
   assign wl_temp[i * 8 +: 8] = (msb_en[i] == 1'b0) ? lsb_out : 8'b11111111;
        end
   endgenerate

   //Assign the final output
   assign wordline = wl_temp;

endmodule
```

Figure 1.2: 6to64 decoder block.

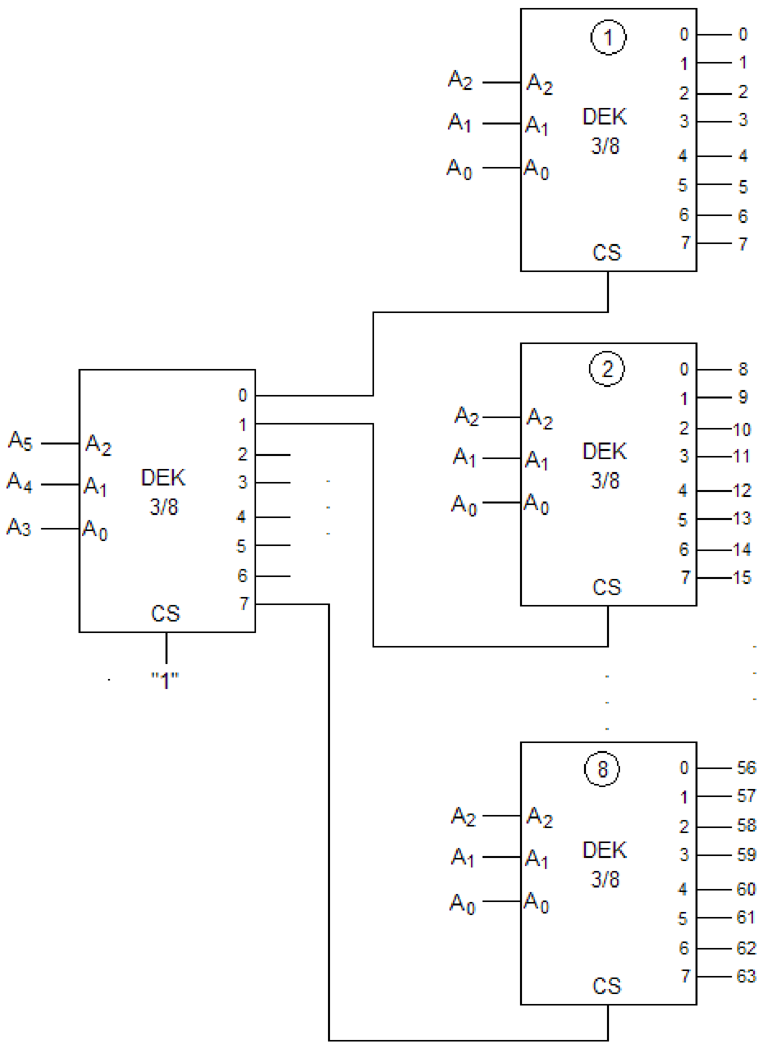
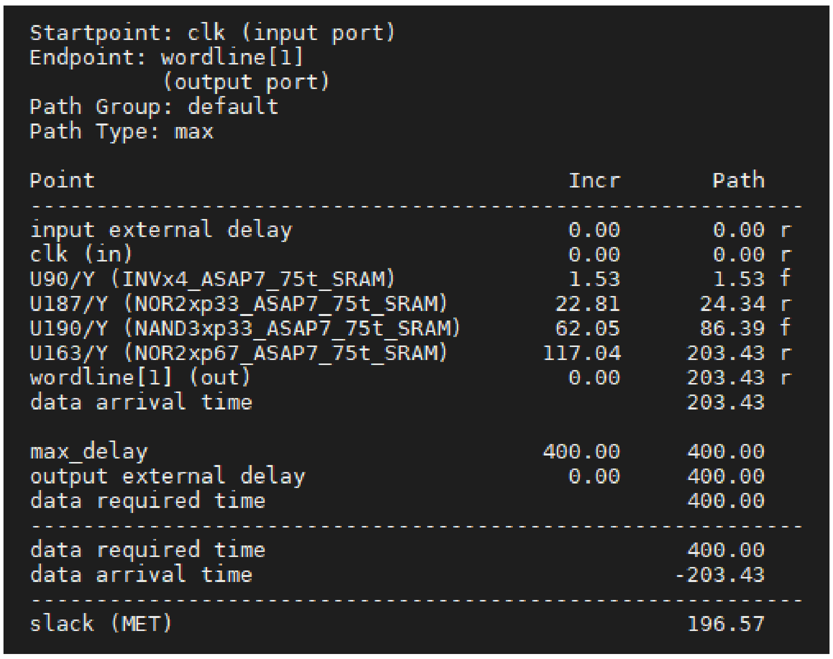
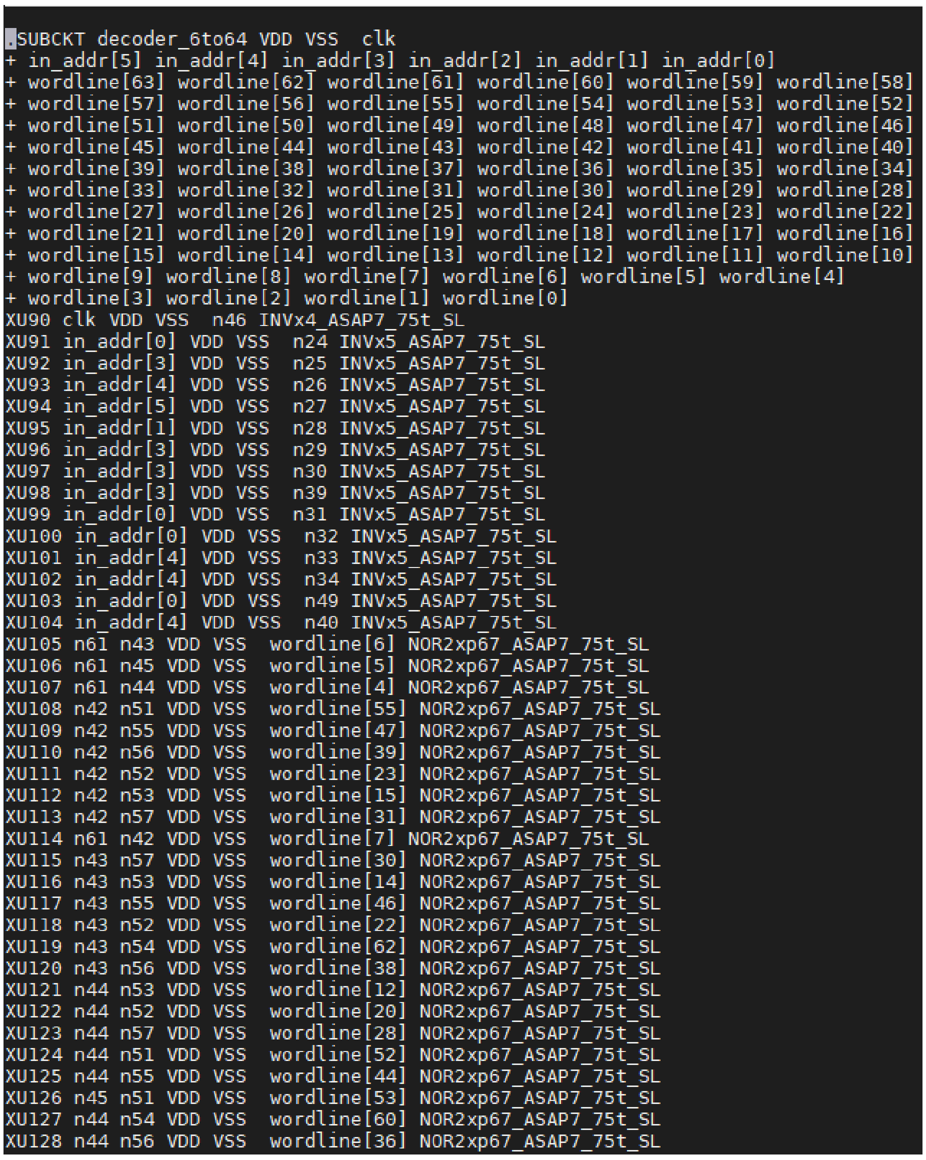


Figure 1.3: ”6to64 decoder block” from 9 ”3to8 block”.



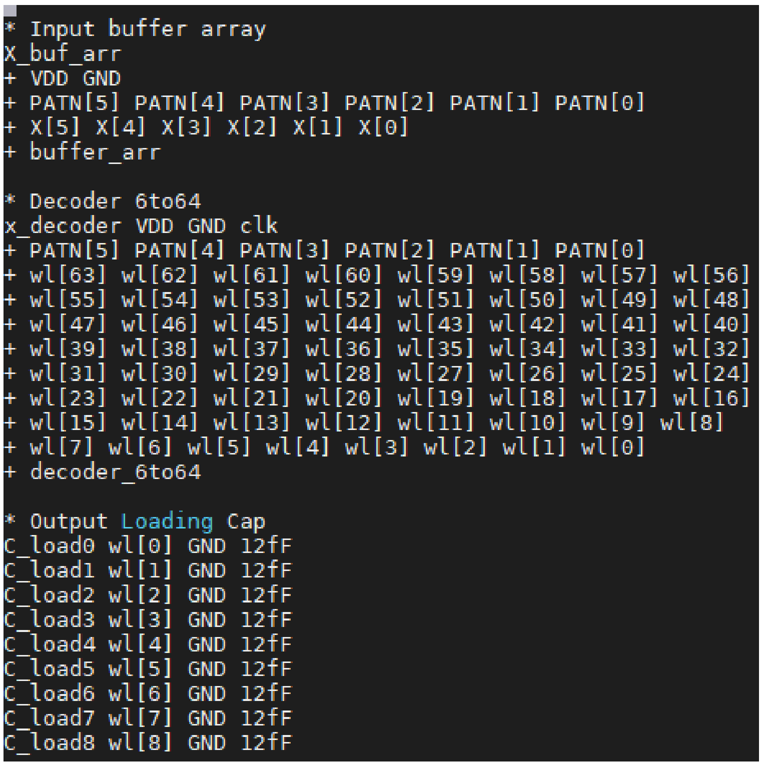
```
Startpoint: clk (input port)
Endpoint: wordline[1]
          (output port)
Path Group: default
Path Type: max

Point                                          Incr       Path
---------------------------------------------------------------
input external delay                           0.00       0.00 r
clk (in)                                       0.00       0.00 r
U90/Y (INVx4_ASAP7_75t_SRAM)                   1.53       1.53 f
U187/Y (NOR2xp33_ASAP7_75t_SRAM)              22.81      24.34 r
U190/Y (NAND3xp33_ASAP7_75t_SRAM)             62.05      86.39 f
U163/Y (NOR2xp67_ASAP7_75t_SRAM)             117.04     203.43 r
wordline[1] (out)                              0.00     203.43 r
data arrival time                                       203.43

max_delay                                    400.00     400.00
output external delay                          0.00     400.00
data required time                                      400.00
---------------------------------------------------------------
data required time                                      400.00
data arrival time                                      -203.43
---------------------------------------------------------------
slack (MET)                                             196.57
```

Figure 1.4: Synthesize complete.



```
.SUBCKT decoder_6to64 VDD VSS  clk
+ in_addr[5] in_addr[4] in_addr[3] in_addr[2] in_addr[1] in_addr[0]
+ wordline[63] wordline[62] wordline[61] wordline[60] wordline[59] wordline[58]
+ wordline[57] wordline[56] wordline[55] wordline[54] wordline[53] wordline[52]
+ wordline[51] wordline[50] wordline[49] wordline[48] wordline[47] wordline[46]
+ wordline[45] wordline[44] wordline[43] wordline[42] wordline[41] wordline[40]
+ wordline[39] wordline[38] wordline[37] wordline[36] wordline[35] wordline[34]
+ wordline[33] wordline[32] wordline[31] wordline[30] wordline[29] wordline[28]
+ wordline[27] wordline[26] wordline[25] wordline[24] wordline[23] wordline[22]
+ wordline[21] wordline[20] wordline[19] wordline[18] wordline[17] wordline[16]
+ wordline[15] wordline[14] wordline[13] wordline[12] wordline[11] wordline[10]
+ wordline[9] wordline[8] wordline[7] wordline[6] wordline[5] wordline[4]
+ wordline[3] wordline[2] wordline[1] wordline[0]
XU90 clk VDD VSS  n46 INVx4_ASAP7_75t_SL
XU91 in_addr[0] VDD VSS  n24 INVx5_ASAP7_75t_SL
XU92 in_addr[3] VDD VSS  n25 INVx5_ASAP7_75t_SL
XU93 in_addr[4] VDD VSS  n26 INVx5_ASAP7_75t_SL
XU94 in_addr[5] VDD VSS  n27 INVx5_ASAP7_75t_SL
XU95 in_addr[1] VDD VSS  n28 INVx5_ASAP7_75t_SL
XU96 in_addr[3] VDD VSS  n29 INVx5_ASAP7_75t_SL
XU97 in_addr[3] VDD VSS  n30 INVx5_ASAP7_75t_SL
XU98 in_addr[3] VDD VSS  n39 INVx5_ASAP7_75t_SL
XU99 in_addr[0] VDD VSS  n31 INVx5_ASAP7_75t_SL
XU100 in_addr[0] VDD VSS  n32 INVx5_ASAP7_75t_SL
XU101 in_addr[4] VDD VSS  n33 INVx5_ASAP7_75t_SL
XU102 in_addr[4] VDD VSS  n34 INVx5_ASAP7_75t_SL
XU103 in_addr[0] VDD VSS  n49 INVx5_ASAP7_75t_SL
XU104 in_addr[4] VDD VSS  n40 INVx5_ASAP7_75t_SL
XU105 n61 n43 VDD VSS  wordline[6] NOR2xp67_ASAP7_75t_SL
XU106 n61 n45 VDD VSS  wordline[5] NOR2xp67_ASAP7_75t_SL
XU107 n61 n44 VDD VSS  wordline[4] NOR2xp67_ASAP7_75t_SL
XU108 n42 n51 VDD VSS  wordline[55] NOR2xp67_ASAP7_75t_SL
XU109 n42 n55 VDD VSS  wordline[47] NOR2xp67_ASAP7_75t_SL
XU110 n42 n56 VDD VSS  wordline[39] NOR2xp67_ASAP7_75t_SL
XU111 n42 n52 VDD VSS  wordline[23] NOR2xp67_ASAP7_75t_SL
XU112 n42 n53 VDD VSS  wordline[15] NOR2xp67_ASAP7_75t_SL
XU113 n42 n57 VDD VSS  wordline[31] NOR2xp67_ASAP7_75t_SL
XU114 n61 n42 VDD VSS  wordline[7] NOR2xp67_ASAP7_75t_SL
XU115 n43 n57 VDD VSS  wordline[30] NOR2xp67_ASAP7_75t_SL
XU116 n43 n53 VDD VSS  wordline[14] NOR2xp67_ASAP7_75t_SL
XU117 n43 n55 VDD VSS  wordline[46] NOR2xp67_ASAP7_75t_SL
XU118 n43 n52 VDD VSS  wordline[22] NOR2xp67_ASAP7_75t_SL
XU119 n43 n54 VDD VSS  wordline[62] NOR2xp67_ASAP7_75t_SL
XU120 n43 n56 VDD VSS  wordline[38] NOR2xp67_ASAP7_75t_SL
XU121 n44 n53 VDD VSS  wordline[12] NOR2xp67_ASAP7_75t_SL
XU122 n44 n52 VDD VSS  wordline[20] NOR2xp67_ASAP7_75t_SL
XU123 n44 n57 VDD VSS  wordline[28] NOR2xp67_ASAP7_75t_SL
XU124 n44 n51 VDD VSS  wordline[52] NOR2xp67_ASAP7_75t_SL
XU125 n44 n55 VDD VSS  wordline[44] NOR2xp67_ASAP7_75t_SL
XU126 n45 n51 VDD VSS  wordline[53] NOR2xp67_ASAP7_75t_SL
XU127 n44 n54 VDD VSS  wordline[60] NOR2xp67_ASAP7_75t_SL
XU128 n44 n56 VDD VSS  wordline[36] NOR2xp67_ASAP7_75t_SL
```

Figure 1.5: 6to64 decoder Hspice code.



```
* Input buffer array
X_buf_arr
+ VDD GND
+ PATN[5] PATN[4] PATN[3] PATN[2] PATN[1] PATN[0]
+ X[5] X[4] X[3] X[2] X[1] X[0]
+ buffer_arr

* Decoder 6to64
x_decoder VDD GND clk
+ PATN[5] PATN[4] PATN[3] PATN[2] PATN[1] PATN[0]
+ wl[63] wl[62] wl[61] wl[60] wl[59] wl[58] wl[57] wl[56]
+ wl[55] wl[54] wl[53] wl[52] wl[51] wl[50] wl[49] wl[48]
+ wl[47] wl[46] wl[45] wl[44] wl[43] wl[42] wl[41] wl[40]
+ wl[39] wl[38] wl[37] wl[36] wl[35] wl[34] wl[33] wl[32]
+ wl[31] wl[30] wl[29] wl[28] wl[27] wl[26] wl[25] wl[24]
+ wl[23] wl[22] wl[21] wl[20] wl[19] wl[18] wl[17] wl[16]
+ wl[15] wl[14] wl[13] wl[12] wl[11] wl[10] wl[9] wl[8]
+ wl[7] wl[6] wl[5] wl[4] wl[3] wl[2] wl[1] wl[0]
+ decoder_6to64

* Output Loading Cap
C_load0 wl[0] GND 12fF
C_load1 wl[1] GND 12fF
C_load2 wl[2] GND 12fF
C_load3 wl[3] GND 12fF
C_load4 wl[4] GND 12fF
C_load5 wl[5] GND 12fF
C_load6 wl[6] GND 12fF
C_load7 wl[7] GND 12fF
C_load8 wl[8] GND 12fF
```

Figure 1.6: Hspice Testbed with input buffer, loading capacitance.

Chapter 2 Wordline Capacitances

We have an equation of wordline capacitance:

$$C_{WL} = 2C_g + C_{wireload} \tag{2.1}$$

FINFET parameters (From 7nn model and the paper [1][2]):

- H_{fin} = 18nm
- W_{fin} = 6.5nm
- t_{ox} = 1.15nm
- C_{gs} = C_{gd} = 1.54×10^{-10} F/m
- L_{eff} = 2.10^{-8} m
- $\varepsilon_o = 8.85418 \times 10^{-12}$ F/m
- $\varepsilon_r = 3.9$
- $C_{wireload} = 0.18fF/\mu m$

Calculate required parameters:

$$W_{eff} = (2 \times H_{fin} + W_{fin}) \times N_{fin} = 8.5 \times 10^{-8} m \tag{2.2}$$

$$C_{ox} = \frac{\varepsilon_o \times \varepsilon_r}{t_{ox}} = 0.03F/m^2 \tag{2.3}$$

$$C_o = C_{ox} \times W_{eff} \times L_{eff} = 0.05fF \tag{2.4}$$

$$\longrightarrow C_g = C_o + C_{gso} \times W_{eff} = 0.07fF \tag{2.5}$$

1 pass transistor has $C_g = 0.07fF \rightarrow$ 64 SRAM cells have $\sum C_g = 8.96fF$.

Following SRAM 7nm guidance [3], 1:2:2 layout have fin pitch = 0.027 μm. $\rightarrow$ Wordline's length of 1 cell = $0.027 \times 10 = 0.27\mu$m. $\rightarrow$ Wordline's length of 1 row = 0.27×64 = 17.28μm.

$\longrightarrow$ Wireload of 1 row = $C_{wireload} = 0.18 \times 17.28 \approx 3fF$

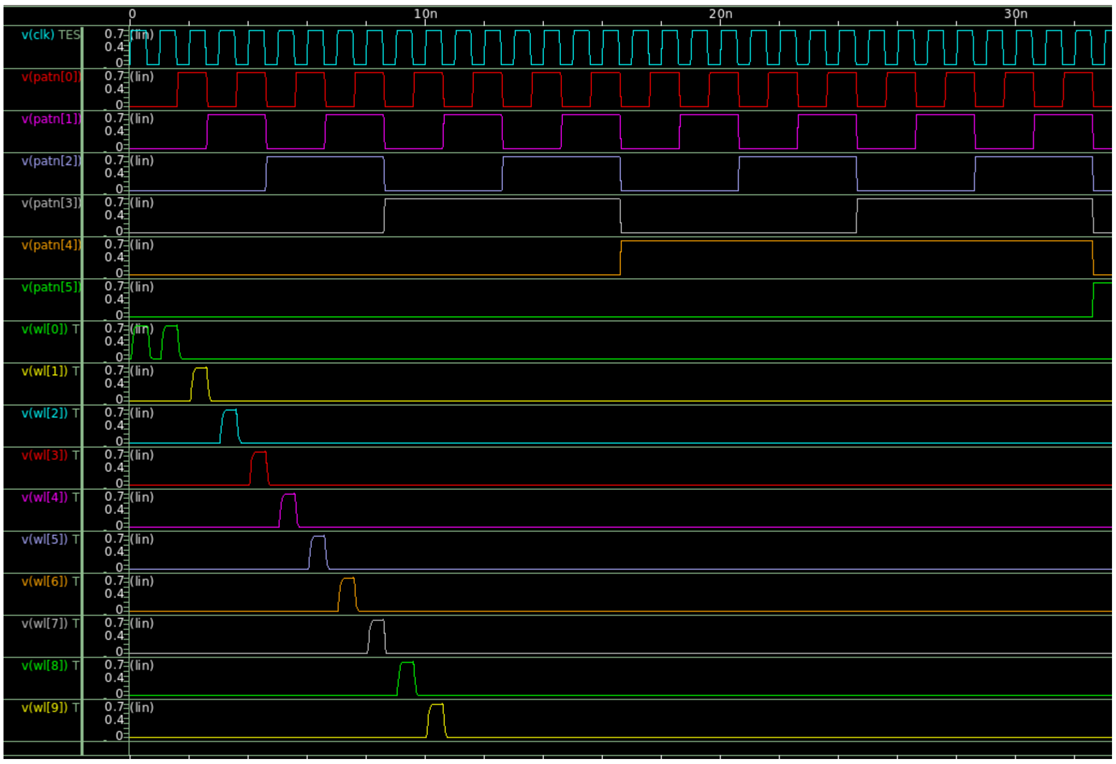
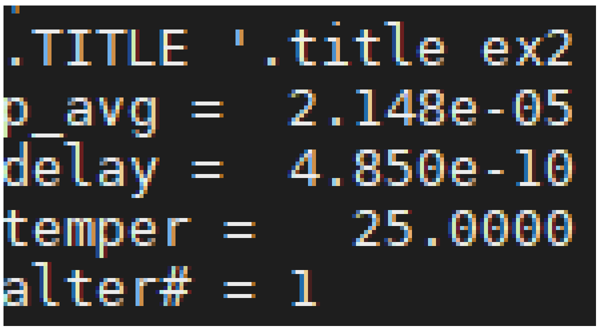


Figure 2.1: 6to64 decoder block waveform run at 1GHz.

As Figure 1.4, the critical path start from clk to WL[1], so I use the .MEAS command to measure the delay from PATN[0] rise to WL[1] rise.



```
.TITLE '.title ex2
p_avg =  2.148e-05
delay =  4.850e-10
temper =   25.0000
alter# = 1
```

Figure 2.2: Average power consumtion and delay in Critical path.

Refferences

[1] Lazzaz Abdelaziz, Bousbahi Khaled, and Ghamnia Mustapha. “Parameters optimization to minimize the power dissipation of FiNFET 7 nm”. In: *2024 16th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*. IEEE. 2024, pp. 1–4.

[2] Serkan Kincal, Mathew C Abraham, and Klaus Schuegraf. “*RC* performance evaluation of interconnect architecture options beyond the 10-nm logic node”. In: *IEEE Transactions on Electron Devices* 61.6 (2014), pp. 1914–1919.

[3] Lawrence T Clark et al. “ASAP7: A 7-nm finFET predictive process design kit”. In: *Microelectronics Journal* 53 (2016), pp. 105–115.