EGRE 365

Homework #1

Due: 09/18/17

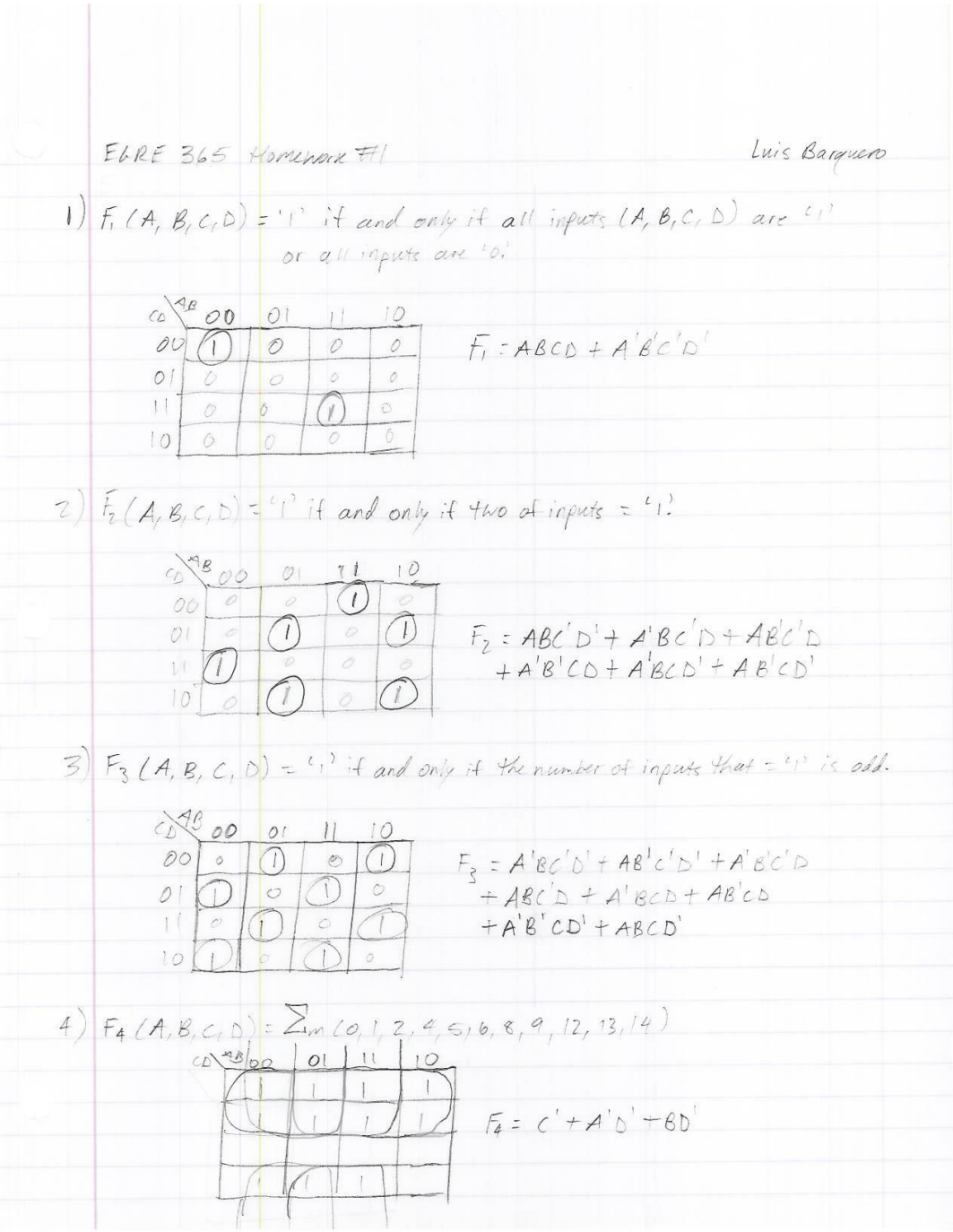

Figure 1 – Figure one contains question 1's K-Maps and their corresponding output function.

```vhdl
entity homework1 is
  port(A, B, C, D : in bit;
       F1: out bit);

end homework1;
 architecture simple of homework1 is
begin
   F1 <= (A AND B AND C AND D) OR ((NOT A) AND (NOT B) AND (NOT C) AND (NOT D));
end simple;

```

Figure 2 – Figure 2 contains the code for the F1's VHDL Model.

```vhdl
entity hw1A_testbench is
end hw1A_testbench;

architecture behavior of hw1A_testbench is

signal input : bit_vector (0 to 3);
signal output: bit;

begin

stimulus : process
   begin
        input <= "0000" after 100 ns,
                 "0001" after 200 ns,
                 "0010" after 300 ns,
                 "0011" after 400 ns,
                 "0100" after 500 ns,
                 "0101" after 600 ns,
                 "0110" after 700 ns,
                 "0111" after 800 ns,
                 "1000" after 900 ns,
                 "1001" after 1000 ns,
                 "1010" after 1100 ns,
                 "1011" after 1200 ns,
                 "1100" after 1300 ns,
                 "1101" after 1400 ns,
                 "1110" after 1500 ns,
                 "1111" after 1600 ns;

      wait;
    end process stimulus;

      DUT: entity work.homework1(simple)
           port map(A => input(0),
                    B => input(1),
                    C => input(2),
                    D => input(3),
                    F1 => output);

      monitor : process
      begin
         wait;
      end process monitor;
end behavior;
```

Figure 3 – Figure 3 contains the code for F1's Testbench.

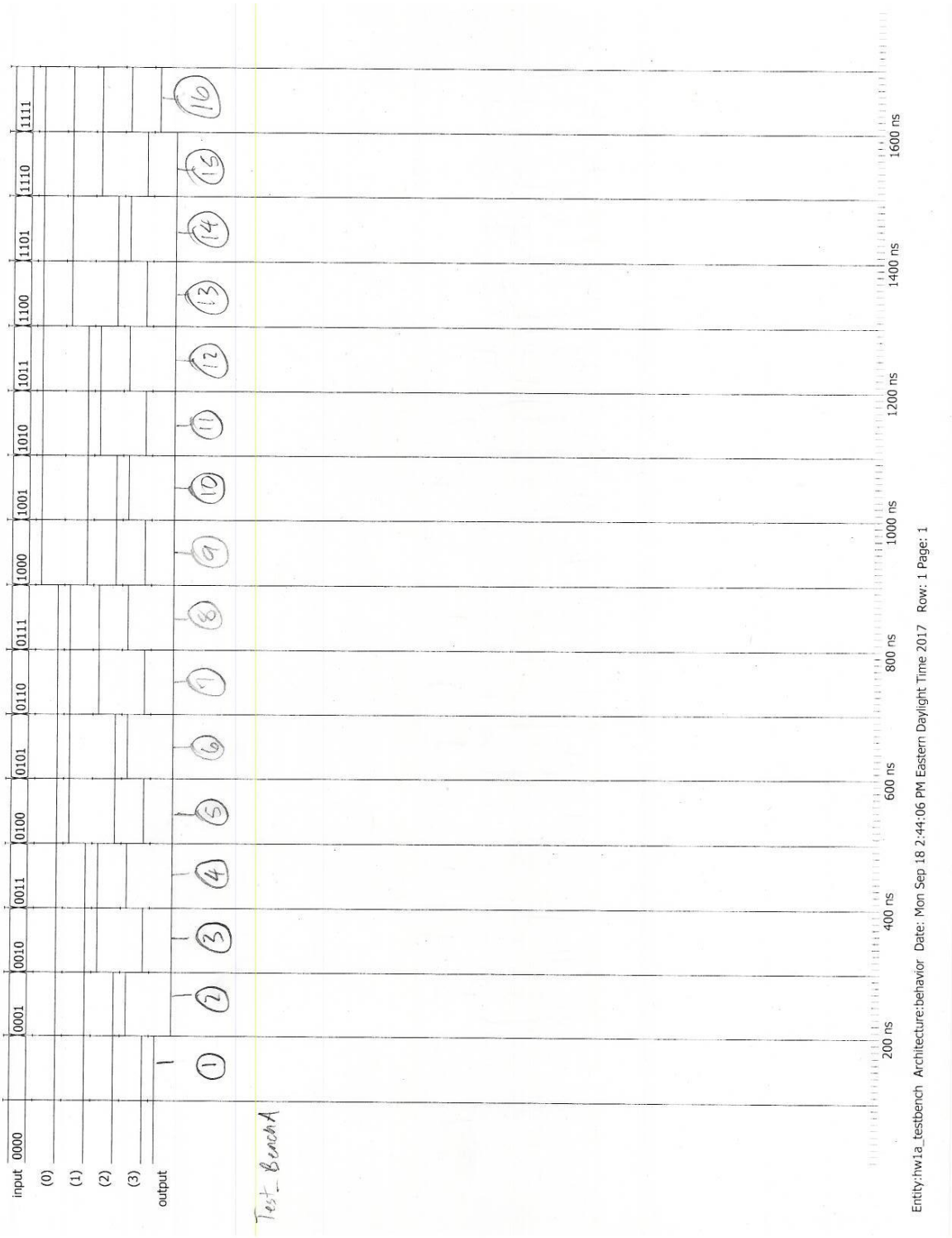

Figure 4 – Figure 4 contains the simulation result from F1's testbench.

```
entity hw1_A is
   port (A, B, C, D: in bit;
         F2: out bit);
end hw1_A;
architecture simple of hw1_A is
begin
     F2 <= (A AND B AND (NOT C) AND (NOT D))
           OR ((NOT A) AND B AND (NOT C) AND D)
           OR (A AND (NOT B) AND (NOT C) AND D)
           OR ((NOT A) AND (NOT B) AND C AND D)
           OR ((NOT A) AND B AND C AND (NOT D))
           OR (A AND (NOT B) AND C AND (NOT D));
end simple;
```

Figure 5 – Figure 5 contains the code for F2's VHDL Model

```
entity hw1B_testbench is
end hw1B_testbench;

architecture behavior of hw1B_testbench is

signal input : bit_vector (0 to 3);
signal output: bit;

begin

stimulus : process
   begin
        input <= "0000" after 100 ns,
                 "0001" after 200 ns,
                 "0010" after 300 ns,
                 "0011" after 400 ns,
                 "0100" after 500 ns,
                 "0101" after 600 ns,
                 "0110" after 700 ns,
                 "0111" after 800 ns,
                 "1000" after 900 ns,
                 "1001" after 1000 ns,
                 "1010" after 1100 ns,
                 "1011" after 1200 ns,
                 "1100" after 1300 ns,
                 "1101" after 1400 ns,
                 "1110" after 1500 ns,
                 "1111" after 1600 ns;

       wait;
     end process stimulus;

       DUT: entity work.hw1_A(simple)
            port map(A => input(0),
                     B => input(1),
                     C => input(2),
                     D => input(3),
                     F2 => output);

       monitor : process
       begin
          wait;
       end process monitor;
end behavior;
```

Figure 6 – Figure 6 contains F2's VHDL Testbench code.

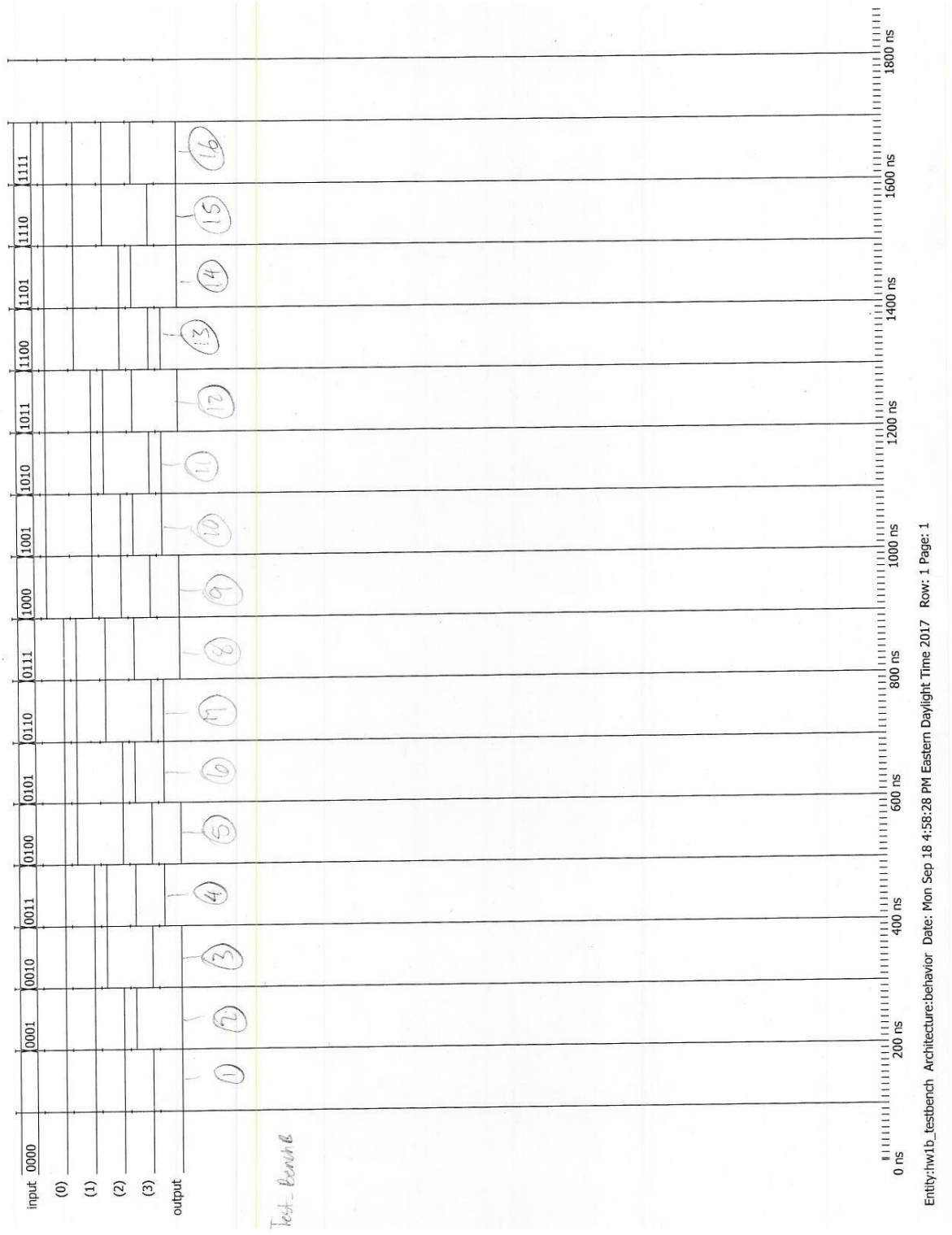

Figure 7 – Figure 7 contains the simulation result for F2's VHDL Testbench.

```
entity hw1_B is
   port (A, B, C, D: in bit;
         F3: out bit);
end hw1_B;
architecture simple of hw1_B is
begin
     F3 <= ((NOT A) AND B AND (NOT C) AND (NOT D)) OR (A AND (NOT B) AND (NOT C) AND (NOT D))
        OR ((NOT A) AND (NOT B) AND (NOT C) AND D) OR (A AND B AND (NOT C) AND D)
        OR ((NOT A) AND B AND C AND D) OR (A AND (NOT B) AND C AND D)
        OR ((NOT A) AND (NOT B) AND C AND (NOT D)) OR (A AND B AND C AND (NOT D));
end simple;
```

Figure 8 – Figure 8 contains F3's VHDL Model code.

```
entity hw1C_testbench is
end hw1C_testbench;

architecture behavior of hw1C_testbench is

signal input : bit_vector (0 to 3);
signal output: bit;

begin

stimulus : process
   begin
        input <= "0000" after 100 ns,
                 "0001" after 200 ns,
                 "0010" after 300 ns,
                 "0011" after 400 ns,
                 "0100" after 500 ns,
                 "0101" after 600 ns,
                 "0110" after 700 ns,
                 "0111" after 800 ns,
                 "1000" after 900 ns,
                 "1001" after 1000 ns,
                 "1010" after 1100 ns,
                 "1011" after 1200 ns,
                 "1100" after 1300 ns,
                 "1101" after 1400 ns,
                 "1110" after 1500 ns,
                 "1111" after 1600 ns;

       wait;
     end process stimulus;

       DUT: entity work.hw1_B(simple)
            port map(A => input(0),
                     B => input(1),
                     C => input(2),
                     D => input(3),
                     F3 => output);

       monitor : process
       begin
          wait;
       end process monitor;
end behavior;
```

Figure 9 – Figure 9 contains the F3's VHDL Testbench Model code.

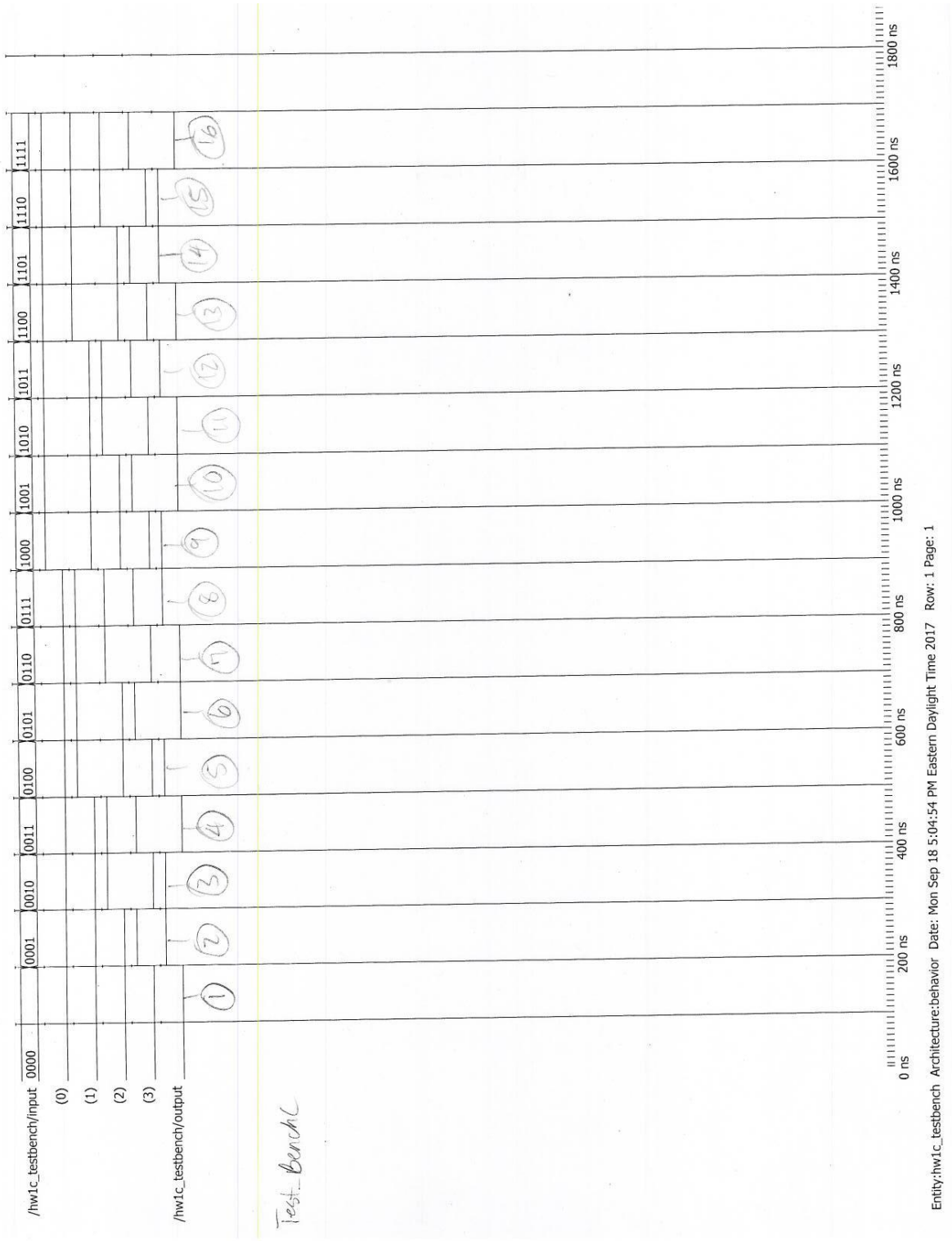

Figure 10 – Figure 10 contains the simulation result for F3's VHDL Testbench.

```
entity hw1_C is
  port (A,B,C,D: in bit;
        F4: out bit);
end hw1_C;
architecture simple of hw1_C is
begin
       F4 <= (NOT C) OR ((NOT A) AND (NOT D)) OR (B AND (NOT D));
end simple;
```

Figure 11 – Figure 11 contains the code for F4's VHDL model.

```
entity hw1D_testbench is
end hw1D_testbench;

architecture behavior of hw1D_testbench is

signal input : bit_vector (0 to 3);
signal output: bit;

begin

stimulus : process
   begin
        input <= "0000" after 100 ns,
                 "0001" after 200 ns,
                 "0010" after 300 ns,
                 "0011" after 400 ns,
                 "0100" after 500 ns,
                 "0101" after 600 ns,
                 "0110" after 700 ns,
                 "0111" after 800 ns,
                 "1000" after 900 ns,
                 "1001" after 1000 ns,
                 "1010" after 1100 ns,
                 "1011" after 1200 ns,
                 "1100" after 1300 ns,
                 "1101" after 1400 ns,
                 "1110" after 1500 ns,
                 "1111" after 1600 ns;

       wait;
     end process stimulus;

       DUT: entity work.hw1_C(simple)
            port map(A => input(0),
                     B => input(1),
                     C => input(2),
                     D => input(3),
                     F4 => output);

       monitor : process
       begin
          wait;
       end process monitor;
end behavior;
```

Figure 12 – Figure 12 contains F4's VHDL Testbench Model code.

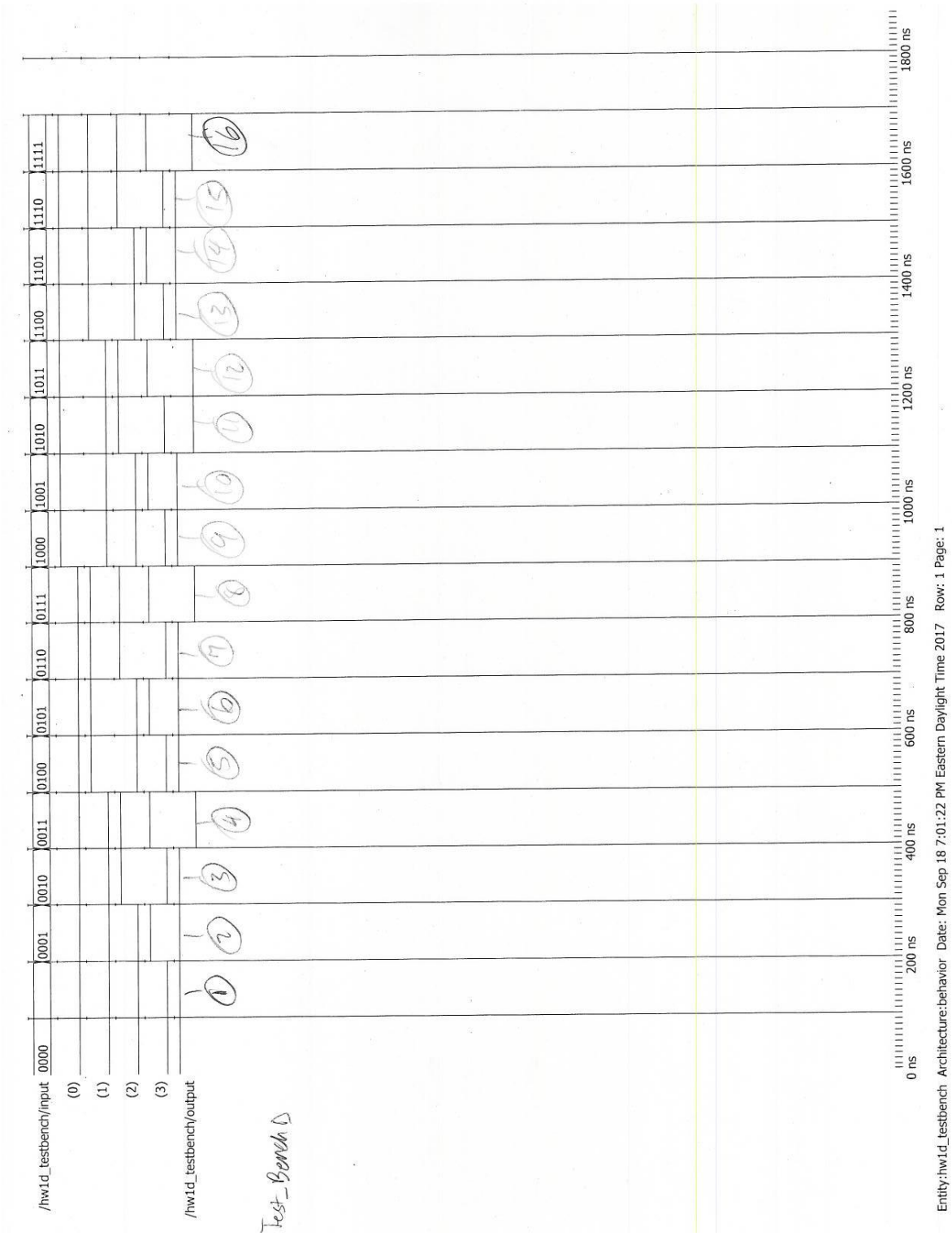

Figure 13 – Figure 13 contains the simulation result for F4's VHDL Testbench.

```
1  entity two_to_one_mux is
2    port(a, b, sel: in bit;
3         y: out bit);
4
5  end two_to_one_mux;
6   architecture simple of two_to_one_mux
7   is
8  begin
9     process(a,b, sel)
10 begin
11    if (sel <= '0') then
12         y <= a;
13    else
14         y <= b;
15    end if;
16  end process;
17 end simple;
```

Figure 14 – Figure 14 contains the 2-to-1 Mux's VHDL Model code.

```
1  entity two_to_one_mux_test is
2  end two_to_one_mux_test;
3
4  architecture behavior of two_to_one_mux_test is
5
6  signal input : bit_vector (0 to 2);
7  signal output: bit;
8  begin
9
10 stimulus : process
11    begin
12         input <= "000" after 100 ns,
13                  "001" after 200 ns,
14                  "010" after 300 ns,
15                  "011" after 400 ns,
16                  "100" after 500 ns,
17                  "101" after 600 ns,
18                  "110" after 700 ns,
19                  "111" after 800 ns;
20
21         wait;
22       end process stimulus;
23
24         DUT: entity work.two_to_one_mux(simple)
25              port map(a => input(0),
26                       b => input(1),
27                       sel => input(2),
28                       y => output);
29
30
31 end behavior;
```

Figure 15 – Figure 15 contains the 2-to-1 Mux's VHDL Testbench Model code.

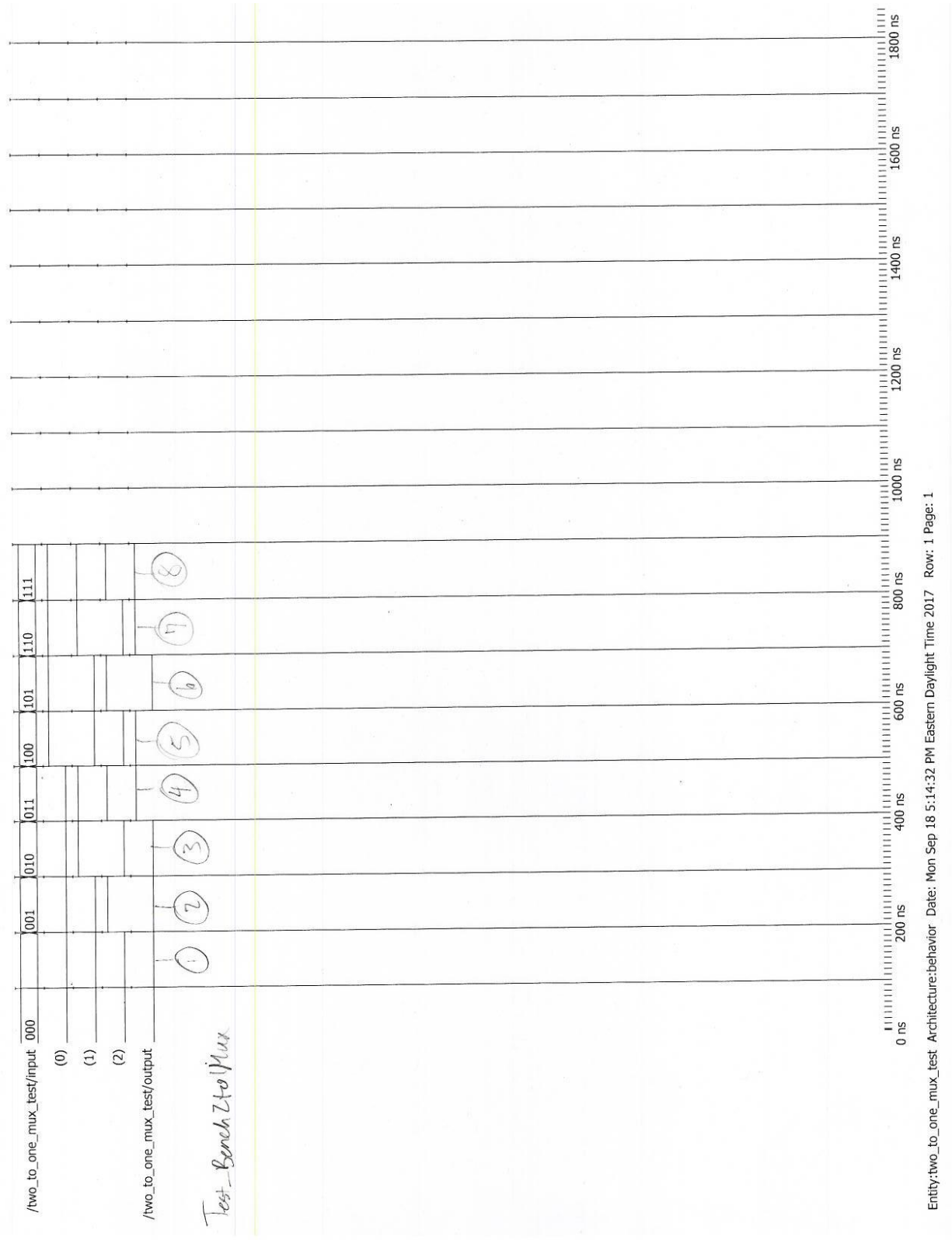

Figure 16 – Figure 16 shows the simulation result for the 2-to-1 Mux's Testbench.