FPGA laboratory report

Diego Figueroa – 2485776

July 2, 2025

1 Convolutional encoder

1.1 Symbol schematic implementation

1.2 VHDL implementation

```
library ieee;
use ieee.std_logic_1164.all;

entity convolutional_code_VHDL is
  port(data_in, clk, rst: in std_logic;
       data_out0, data_out1: out std_logic);
end convolutional_code_VHDL;

architecture state_machine of convolutional_code_VHDL is
  type state is (s00, s01, s10, s11);
  signal curr_st, next_st: state;
  signal d_tmp, d_out0, d_out1: std_logic;
  begin
    process(curr_st, d_tmp) -- state logic
    begin
      case curr_st is
      when s00 =>
        if d_tmp = '0' then
          d_out0 <= '0';
          d_out1 <= '0';
          next_st <= s00;
        else
          d_out0 <= '1';
          d_out1 <= '1';
          next_st <= s01;
        end if;
      when s01 =>
        if d_tmp = '0' then
          d_out0 <= '0';
          d_out1 <= '1';
          next_st <= s10;
        else
          d_out0 <= '1';
```

```
34            d_out1 <= '0';
35            next_st <= s11;
36          end if;
37        when s10 =>
38          if d_tmp = '0' then
39            d_out0 <= '1';
40            d_out1 <= '1';
41            next_st <= s00;
42          else
43            d_out0 <= '0';
44            d_out1 <= '0';
45            next_st <= s01;
46          end if;
47        when s11 =>
48          if d_tmp = '0' then
49            d_out0 <= '1';
50            d_out1 <= '0';
51            next_st <= s10;
52          else
53            d_out0 <= '0';
54            d_out1 <= '1';
55            next_st <= s11;
56          end if;
57        end case;
58      end process;
59
60      process(clk, rst) -- go to next state
61      begin
62        if rst = '1' then
63          curr_st <= s00;
64          d_tmp <= '0';
65        elsif (clk = '1' and clk'event) then
66          curr_st <= next_st;
67          d_tmp <= data_in;
68        end if;
69      end process;
70
71      -- concurrent assigment of the output
72      data_out0 <= d_out0;
73      data_out1 <= d_out1;
74    end state_machine;
```

Listing 1: xd

1.3 AHDL implementation

```
1 subdesign convolutional_code_AHDL(
2   data_in, clk, rst: input;
3   data_out0, data_out1: output
4 )
5 variable
6   conv_code_state: MACHINE
```

```
7   WITH STATES (
8     s00 = b"00",
9     s01 = b"01",
10    s10 = b"10",
11    s11 = b"11");
12  d_in: node;
13
14 begin
15  conv_code_state.clk = clk;
16  conv_code_state.reset = rst;
17
18  d_in = DFF(data_in, clk, VCC, VCC);
19
20  TABLE
21  conv_code_state, d_in => data_out1, data_out0,
       conv_code_state;
22  s00, 0 => 0, 0, s00;
23  s00, 1 => 1, 1, s01;
24  s01, 0 => 1, 0, s10;
25  s01, 1 => 0, 1, s11;
26  s10, 0 => 1, 1, s00;
27  s10, 1 => 0, 0, s01;
28  s11, 0 => 0, 1, s10;
29  s11, 1 => 1, 0, s11;
30  END TABLE;
31 end;
```

Listing 2: xd

2 Running light

2.1 Functional description

2.2 Digital Design

2.3 VHDL implementation

2.4 Results

3 FIR filters

3.1 Low pass

3.2 High pass

3.3 Band pass

3.4 Band stop

3.5 Project with the 4 filters