March 23, 2006

CSE2306 Digital Logic CSE1308

Practical 3: Canonical forms. Decoders.

3.1 About this practical

The objective of this practical is to reinforce your knowledge of canonical forms of logic functions. We also study decoders as generators of minterms and maxterms.
In addition we introduce a concept of simulation scripts and examine an automatically generated VHDL description of a logic circuit.

Contents

3.2 Designing your own 2-to4 decoder.

Follow the instructions from **prac1** to create a new project with a new block diagram that will be used in simulation.

- Start Designer Manager invoking **FPGAdv** tools.
- In the **Getting Started** wizard select **Create a new Project** button and click **OK**
- In a **Creating a New Project** wizard specify:

Name of new project:	`P3`**you**
Directory in which your project folder will be created:	`DigDes`

where **you** should be replaced with **your initials**.

- Open a Block Diagram window by selecting in the Design Manager
  File → New → Graphical view → Block Diagram
- In the Block Diagram create a 2-to-4 decoder as in Figure 1

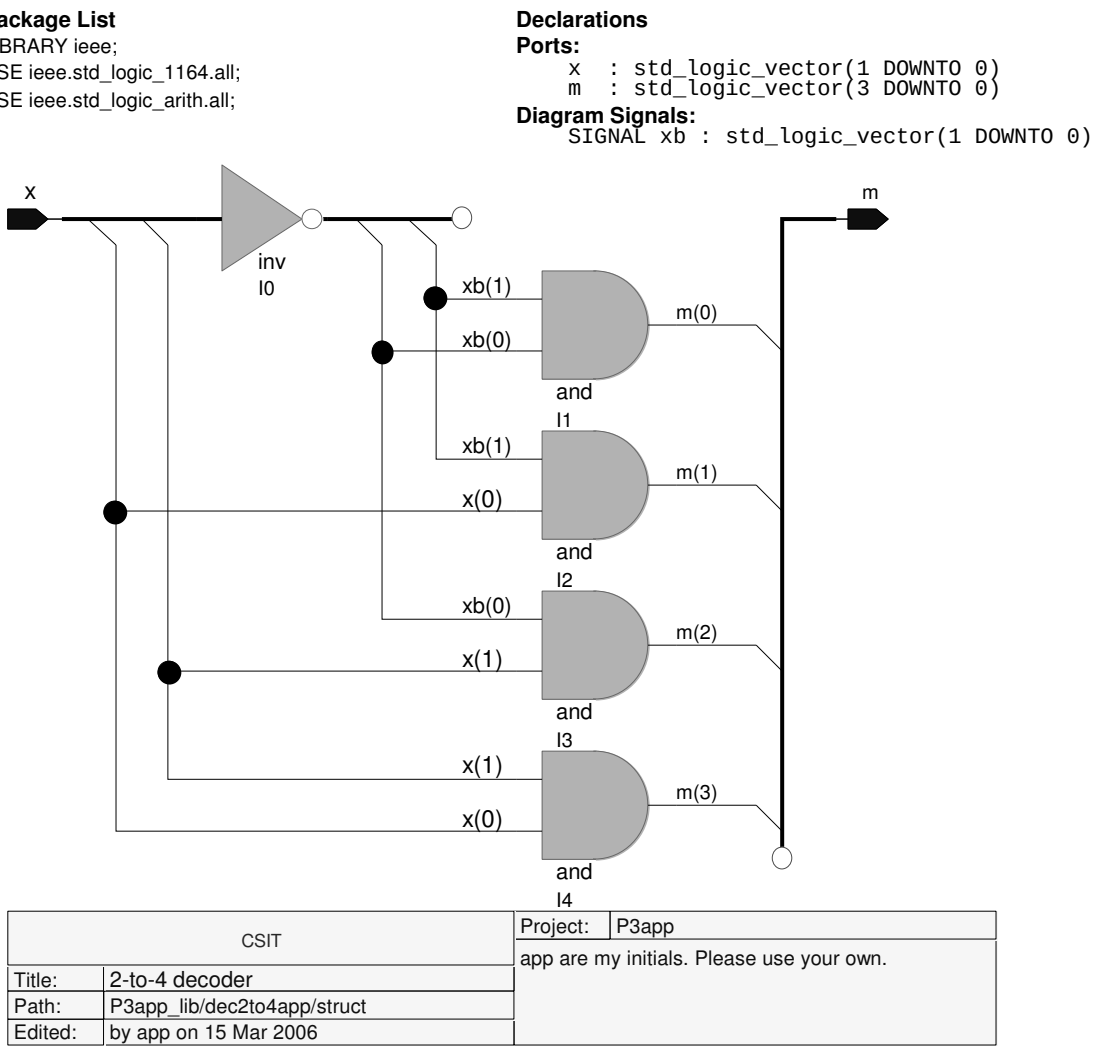

Figure 1: Logic diagram of a 2-to4 decoder

- Save the diagram as: `dec2to4`**you**
  where **you** should be replaced with your initials.
- In the Design Manager expand the design unit `P3you_lib` and find the symbol for `dec2to4you` . Double-click on the "symbol" to open the Symbol window.
- Modify the symbol to look similar to that in Figure 2
  Such a symbol can be now use in you future designs.

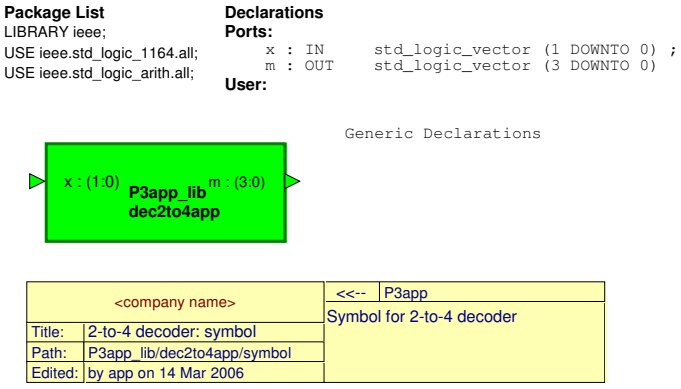

Figure 2: Symbol of a 2-to4 decoder

3.3 Simulation of a 2-to-4 decoder

- To start ModelSim simulator select in the Block Diagram window
  Tasks → ModelSim Flow → Run Single
- Accept defaults in the **Start ModelSim** window clicking OK.
- In the ModelSim window, in its left pane, select first an instance `dec2to4`**you** , and then from the pull-down menu: **View → Signals**. A window **signals** is created in which all input/output signals (ports) are listed.
- In the **signals** window select **Add → Wave → Signals in design**. This opens a **wave – default** window with all signals ready to be monitored.
- Observe the command window of the simulator
- In the **signals** window click on '+' against the `x` bus to see the individual signals `x(1)`,`x(0)`
- Select `x(1)` and then **Edit → Clock...**. A dialog window **Define Clock** pops up.
  Specify: Duty — 50[%], Period — 10[ns], FirstEdge — Falling and click OK.
- Similarly select `x(0)` and then **Edit → Clock...**.
  Define clock as: Duty — 40[%], Period — 5[ns], FirstEdge — Falling and click OK.
- Run simulation for 20ns.
  The result displayed in the **wave – default** window should be similar to that in Figure 3.

3.3.1 Creating a simulation script

In order to simplify the repetitive simulation tasks we can create and run the simulation script. One way of doing so is to use the transcript file created in your previous simulation.

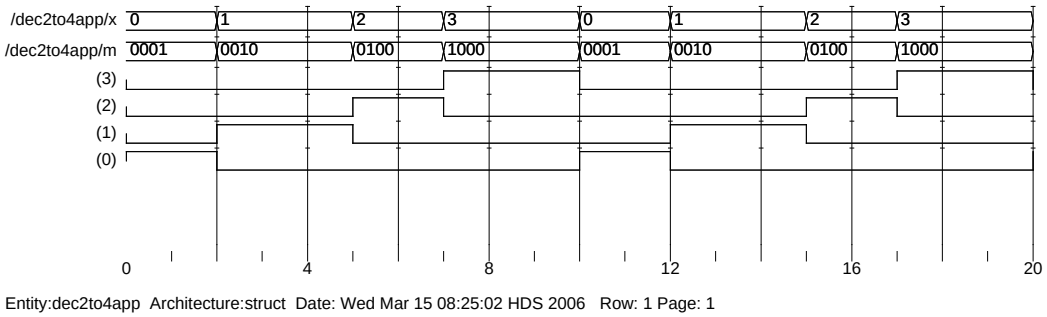

Figure 3: Simulation waveforms for a 2-to-4 decoder

- In the ModelSim window execute **File → Transcript → Save Transcript As ...**

  Save transcript in a file `dec2to4.sim` . Note that this file will be created most likely in the directory `P3you_lib` .

- Use your favourite text editor and edit the `dec2to4.sim` file to look like the following

```
# dec2to4.sim
restart -force -nowave
view signals
add wave -uns x
add wave  m
force -freeze x(0) 0 0, 1 {2 ns} -r 5
force -freeze x(1) 0 0, 1 {5 ns} -r 10
run 20
.wave.tree zoomfull
```

- If you have observed the command window you can interpret most of the commands. However sooner or later you have to go to **Help → HTML Documentation** and study details of each command.

- Before we run the script execute the command `pwd` to verify that you are in the directory `P3you_lib/work` .

- To execute the simulation script type in: `do ../dec2to4.sim`

  This should re-run the simulation for 20ns

3.4 The VHDL source code for the 2-to-4 decoder

ModelSim generates the VHDL code describing your circuit. Execute `view source` command to obtain the source window.
The VHDL code should be similar to the following:

```
-- VHDL Entity P3app_lib.dec2to4app.symbol
--
-- Created: by - app.UNKNOWN (ANDROO)
--          at - 08:24:07 15/03/2006
-- Generated by Mentor Graphics' HDL Designer(TM) 2004.1 (Build 41)
--
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;

ENTITY dec2to4app IS
  PORT(
     x : IN     std_logic_vector (1 DOWNTO 0);
     m : OUT    std_logic_vector (3 DOWNTO 0)
  );
END dec2to4app ;

-- VHDL Architecture P3app_lib.dec2to4app.struct
-- Created: by - app.UNKNOWN (ANDROO)
--          at - 08:24:07 15/03/2006
-- Generated by Mentor Graphics' HDL Designer(TM) 2004.1 (Build 41)
--
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;

ARCHITECTURE struct OF dec2to4app IS
  SIGNAL xb : std_logic_vector(1 DOWNTO 0);
BEGIN
  m(0) <= xb(1) AND xb(0);
  m(1) <= xb(1) AND x(0);
  m(2) <= xb(0) AND x(1);
  m(3) <= x(1)  AND x(0);
  xb   <= NOT(x);
END struct;
```

We will discuss details of the code in class.

3.5 Implementing a logic function using a 3-to-8 decoder

The objective of this exercise is to implement a logic function in four different forms using a 3-to-8 decoder and four gates, OR, AND, NOR, NAND as in Figure 4
We will use a decoder from the **ModuleWare** library.

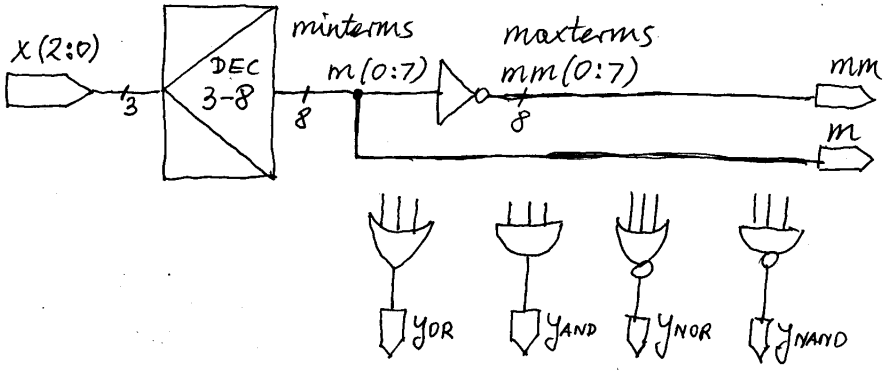

Figure 4: Implementing a logic function in four equivalent ways

3.5.1 Selecting your logic function to be implemented

There are five logic functions to choose from. The choice is based on the last digit of your **student ID number** as follows:

ID #	function
$0, 5$	$\sum(0, 4, 6, 7)$
$1, 6$	$\sum(0, 2, 4, 7)$
$2, 7$	$\sum(1, 2, 3, 4)$
$3, 8$	$\sum(3, 4, 5, 6)$
$4, 9$	$\sum(1, 2, 5, 7)$

3.5.2 Building the block diagram of the canonical implementations

- You should start with **saving and closing** all windows but the Design Manager. You can even exit the **FPGAdv** tools and restart it continuing the `P3you` project.
- Open a new Block Diagram window by selecting in the Design Manager
  File → **New** → **Graphical view** → **Block Diagram** and save it as **you**`LgcFun` .
- **Add ModuleWare**: Go to the moduleware library to the **combinatorial** group and instance the **Decoder (combined output)** in your block diagram.
- You have to set up parameters of the decoder to be 3-to-8. Double click on the decoder symbol to open the **ModuleWare Parameters** window.
- Change the **Value** parameters for **din** and **dout** signals to be 3 and 8, respectively. Click OK.
- You should aim at a block diagram as in Figure 5.
- The number of inverters (or the number of input/output signals in the inverter) will be adjusted automatically when you connect by a bus the decoder output with the inverter input.

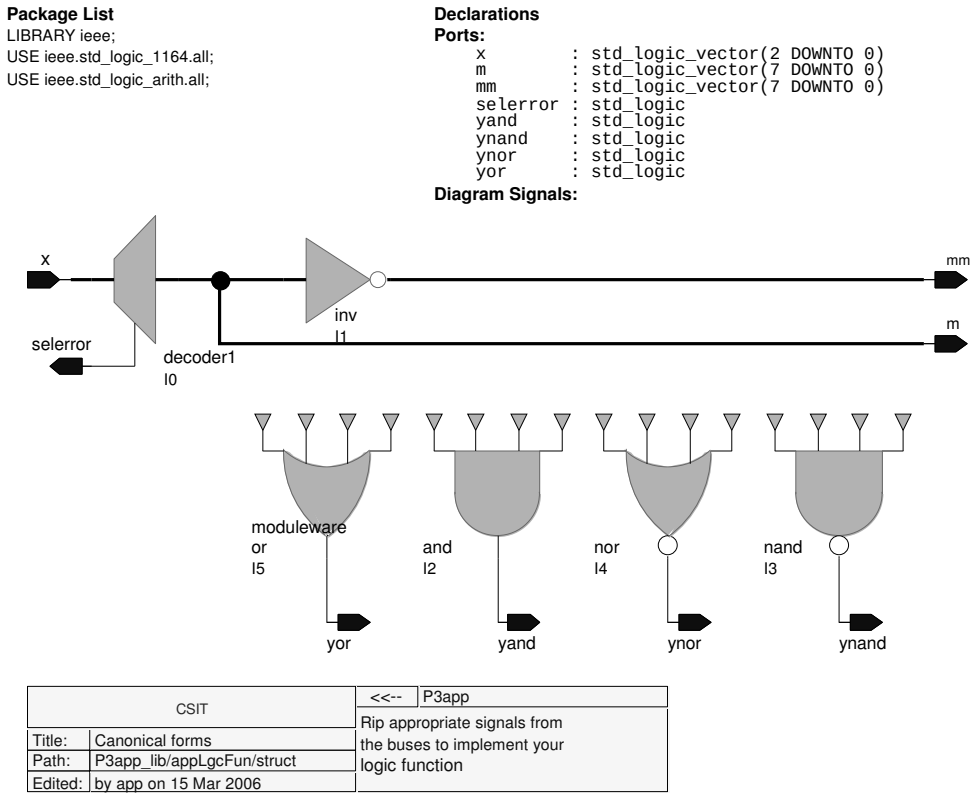

Figure 5: Four canonical implementations of a logic function (incomplete).

- To get the number of inputs to the gates increased to 4, double click on the gate and in the **ModuleWare Parameters** window, in the options section, increase the **Dynamic number of ports** to 4.
- To rotate the symbol of the gate right-click on it and select appropriate action from the pop-up menu.
- From the same pop-up menu you can show/hide the description (text) associated with each gate (or element).
- When you have a block diagram similar to that in Figure 5 the final step is to **connect the gates** to the appropriate signals from two busses and implement your logic function in four equivalent canonical ways.

3.6 Simulation of your logic function implementations

Using the simulation script similar to the following

```
# LgcF.sim
restart -force -nowave
view signals
add wave -uns x
add wave -dec m  mm
add wave  yor  yand  ynor  ynand
force -freeze x(0) 0 0, 1 {2 ns} -r 4
force -freeze x(1) 0 0, 1 {4 ns} -r 8
force -freeze x(2) 0 0, 1 {8 ns} -r 16
run 32
.wave.tree zoomfull
```

you should obtain the simulation waveforms similar to those in Figure 6.

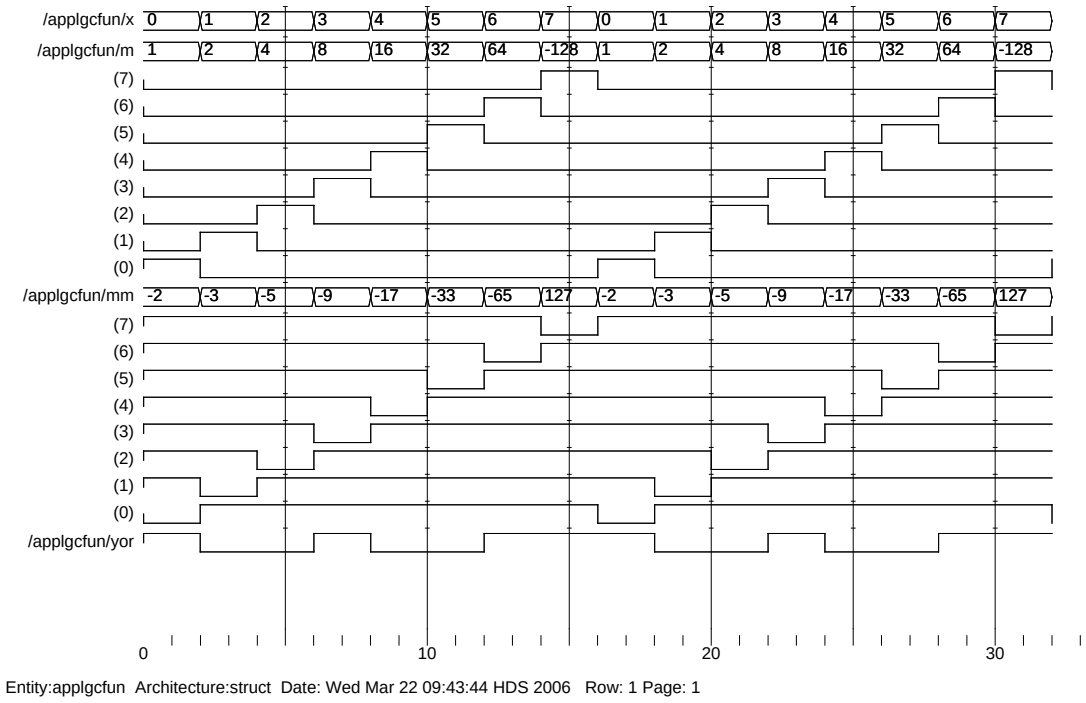

Figure 6: Simulation waveforms.

3.7 The report

In your report (due after prac 4) include the results in the form of:

- block/logic diagrams,
- VHDL programs (if available),
- simulation scripts (if available),
- simulation waveforms,
- **short** description of the above.

Wherever possible publish the results selecting in the **Block Diagram** window **File** → **HTML Export ...**. Specify the export target directory to be `...\DigDes\Reports`.