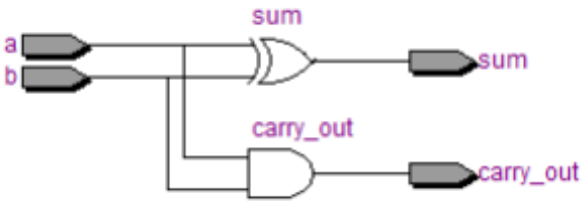
DSD Exercise



Dataflow-Style Combinatorial Designs in VHDL

2-9-2015/PHM

GOALS

Designs without memory are also known as combinatorial. Combinatorial designs can be expressed with truth-tables and this is also how they are implemented in the FPGA. The goals for this exercise are:

- Get experience with “with-select” and “when” statements
- Learn to identify latches in your design and how to remove them
-

PREREQUISITES

- Have Quartus II up and running
- That you have read THE DSD EXERCISE GUIDELINES!!!

THE EXERCISE ITSELF

This exercise will get around different combinatorial code constructions and it starts out with “With-Select” statements.

1) BINARY TO 7-SEGMENT DECODER USING “WITH-SELECT”

In this step it is your task to implement a binary to seven segment decoder using with-select.

1) Create a binary-to-7-segment converter component with the interface depicted in figure 3. The component must be implemented using a with-select statement. See the “DE-2 User Manual” for details about the seven-segment displays (named HEX displays in the text). Also note that the segments are active-low.

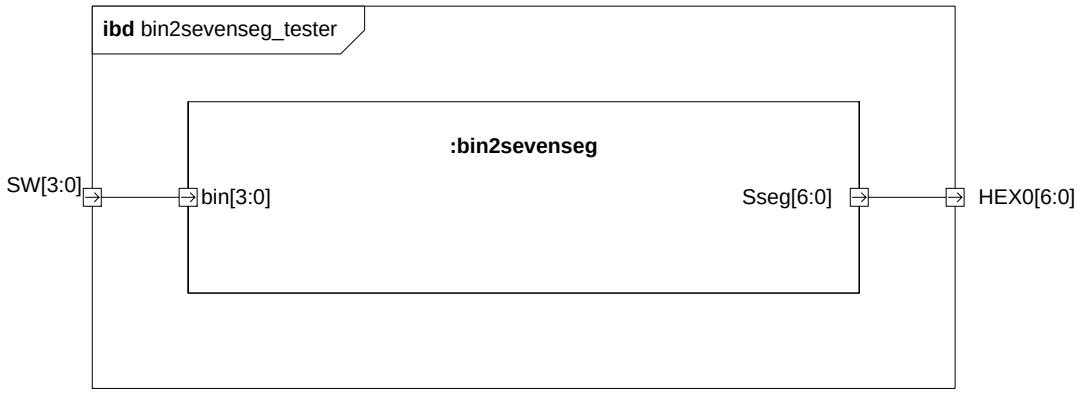


FIGURE 1: BINARY TO 7-SEGMENT DECODER

2) Compile and test the multiplier on the DE2-board. *How is the design implemented according to the RTL-Viewer?*

2) DEMULTIPLEXING USING "WHEN"

"When" statements are used to multiplex or de-multiplex signals. Like "With-select" these statements are written as concurrent statements. In this exercise you will use "when" to switch between sources to be displayed on the HEX displays.

1) Copy the tester from previous exercise to a new name and implement the content shown in figure 3 using "when" statements. The three HEX displays must show "On " when no keys are pushed, "Err" when KEY(1) is pushed and the hex-value of SW(11 downto 0) when KEY(0) is pushed. Note that KEY is active low.

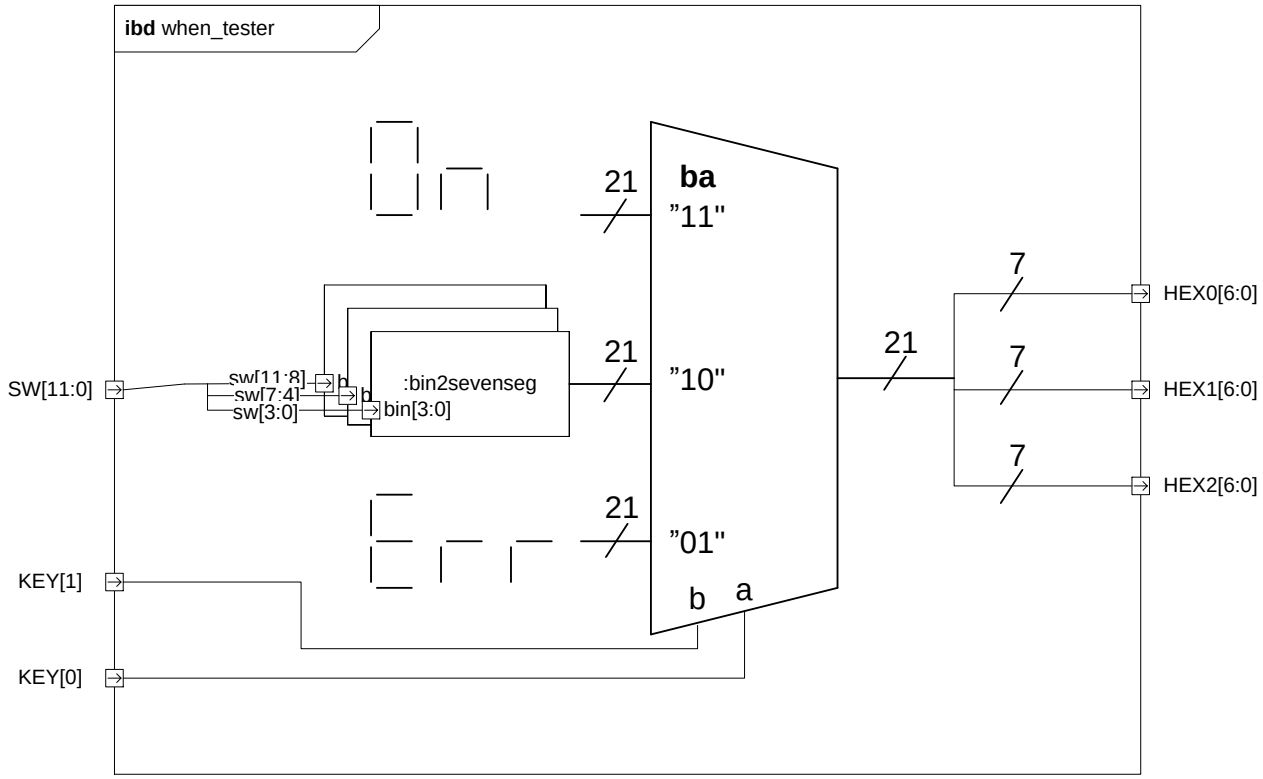


FIGURE 2: WHEN-TESTER

2) Compile and test the multiplier on the DE2-board. *Do you have any compile warnings about inferred latches? How could your implementation result in inferred latches? How do you fix this?*

3) TABLE LOOKUP

With-Select can be used to create look-up tables. Another way is to create a constant array with pre-defined values and use its index to look-up values. In this exercise you will use this approach.

a	b	c	x
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	-
1	1	0	-
1	1	1	1

FIGURE 3 LOOK-UP TABLE

1) Design a component implementing the truth table in Figure 3, to the right, using only a lookup table. Perform a functional simulation to verify the correctness of the design. See listing 4.9.1 for more information.

2) Test the component on the DE2 board. Use SW[2:0] as inputs and LEDR[0] as output.

4) BIDIRECTIONAL PORTS (OPTIONAL)

VHDL supports bidirectional ports using the “inout” port type. Bidirectional ports are typically used in top-level designs to connect to external bidirectional hardware interfaces. Internal interfaces in structural VHDL designs are typically unidirectional, as this is easier for the synthesis tool to route and optimize. In this exercise you will gain a little experience with controlling bidirectional ports.

The design that you must implement is illustrated in Figure 4. This illustration also shows the switches and LEDs to be used on the DE-2 board.

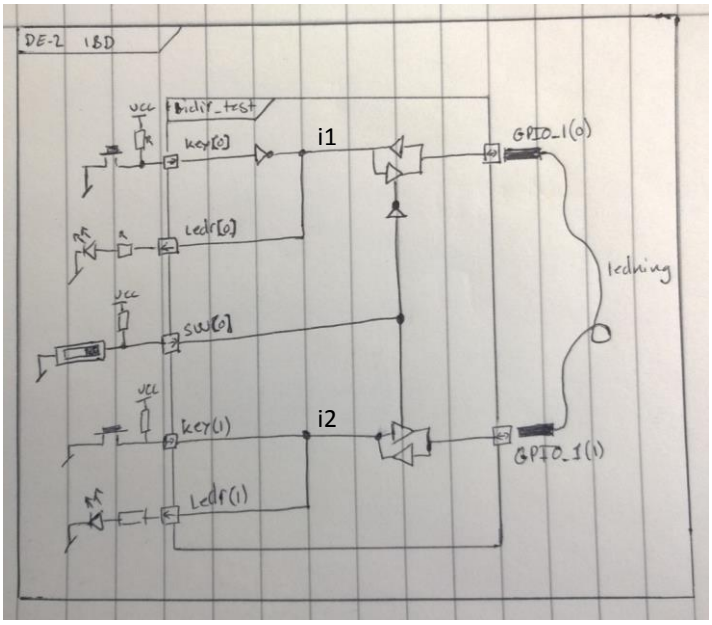


FIGURE 4 BIDIR_IBD

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

entity bidir_test is
  port(KEY    : in     std_logic_vector(1 downto 0);
       SW     : in     std_logic_vector(0 downto 0);
       LEDR   : out    std_logic_vector(1 downto 0);
       GPIO_1 : inout  std_logic_vector(1 downto 0));
end bidir_test;
```

FIGURE 5 BIDIR_TEST INTERFACE

1) Create a component with an interface as shown in Figure 5 and implement the function illustrated in Figure 4. (Can be implemented by 6-lines of dataflow-style code!!)

2) Connect GPIO_1 pin 0 and pin 1 with a wire as shown in Figure 6

FIGURE 6 GPIO1 CONNECTIONS

3) Test your code on the DE2-board. *Explain how it works. What happens if you remove the wire?*

4) *How would you simulate a design as this? (Hint std_logic provides values beside '0', '1')*