DSD EXERCISE

DATAFLOW-STYLE COMBINATORIAL DESIGNS IN VHDL

AARHUS UNIVERSITY
SCHOOL OF ENGINEERNG
PHM, CEF

JUNE 2018

Document History

2015-09-25:	PHM, initial version.
2017-08-23:	CEF, fixed figure 8, `bi_port` idb.
2017-10-02:	CEF, updated exercise.
2017-10-04:	CEF, updated exercise 1, made questions 5) and 6) optional. Updated text. Updated wrong enumerations in exe 2.
2017-10-09:	CEF, fixed index in exe 1.
2018-01-26:	CEF, converted to LaTeX.
2018-01-27:	CEF, removed "Array Element Operations" excercise.
2018-02-09:	CEF, fixed comma error in Bi-dir source code table.
2018-06-07:	CEF, fixed figure placements.
2019-10-01:	CEF, renamed exe 1 and 2.

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:

- Understand bit-ordering in terms of downto/to vector sizes.
- Get experience with “with-select” and “when” statements.
- Learn to identify latches in your design and how to remove them.

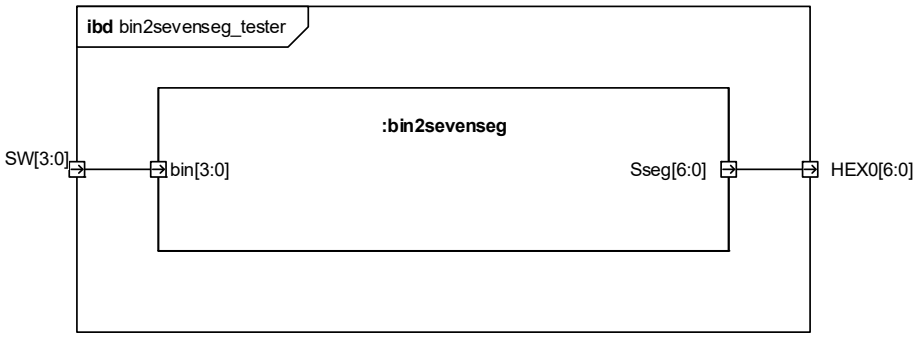

Fig. 1: *Binary to 7-segment-decoder.*

1 Binary to 7-Segment Decoder Using Selected Signal Assignment ("WITH-SELECT")

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

a) Create a binary-to-7-segment converter component with the interface depicted in figure 1. The component must be implemented using a with-select statement. See the "DE-2 User Manual" for details about the sevensegment displays (named HEX displays in the text). The output must cover the hexadecimal values 0 to F. Also note that the segments are active-low.

b) Compile and test the multiplexer on the DE2-board. How is the design implemented according to the RTLViewer?

2 The Conditional Signal Assignment ("WHEN-ELSE")

"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.

a) Implement the content shown in figure 2 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.

b) Create a tester as shown in figure 3.

c) Compile and test the design 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) Design a component implementing the truth table in table 1, 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.

b) 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.

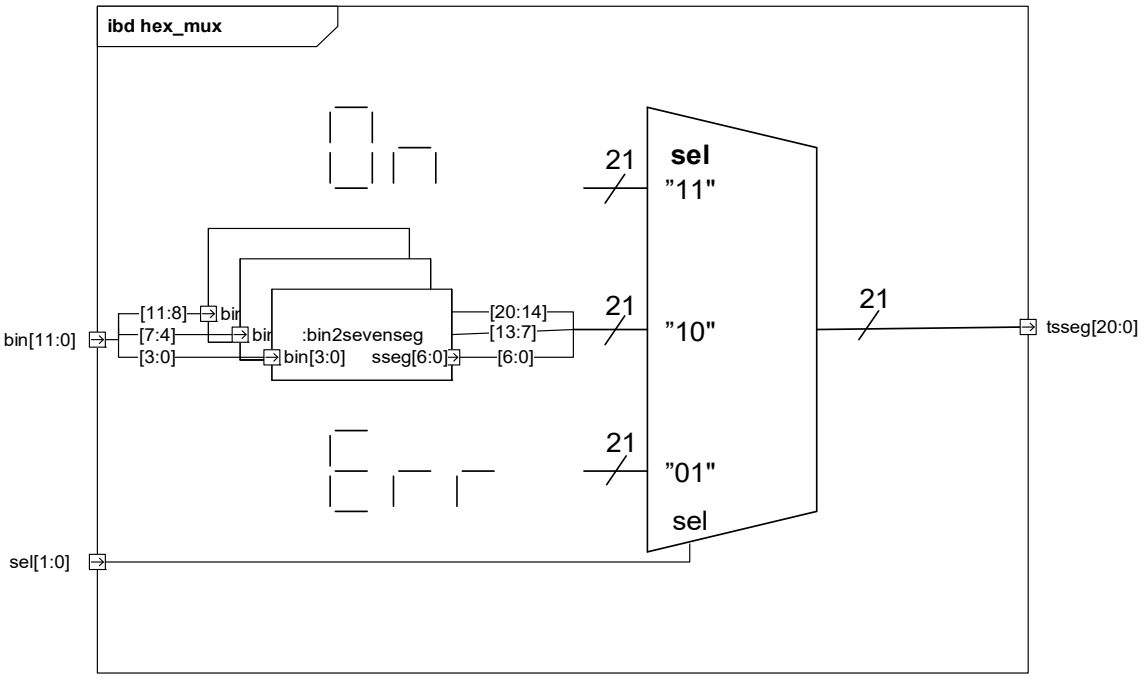

Fig. 2: *Hex MUX.*

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

Tab. 1: *Lookup truth table.*

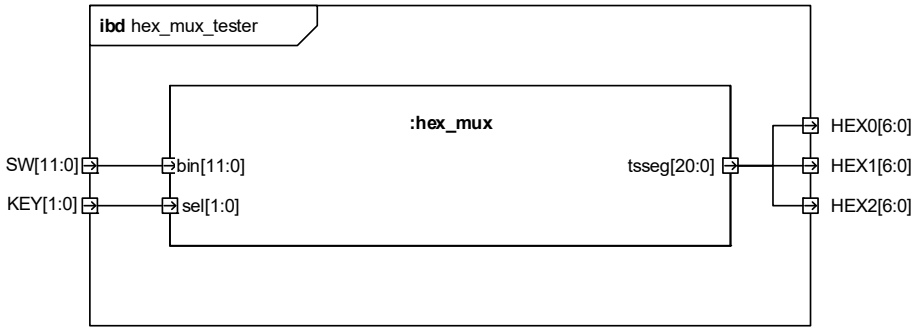

Fig. 3: *When tester.*

a) Create a component with an interface as shown in source code table 2 and implement the function illustrated in figure 4. (Can be implemented by 6-lines of dataflow-style code!!)

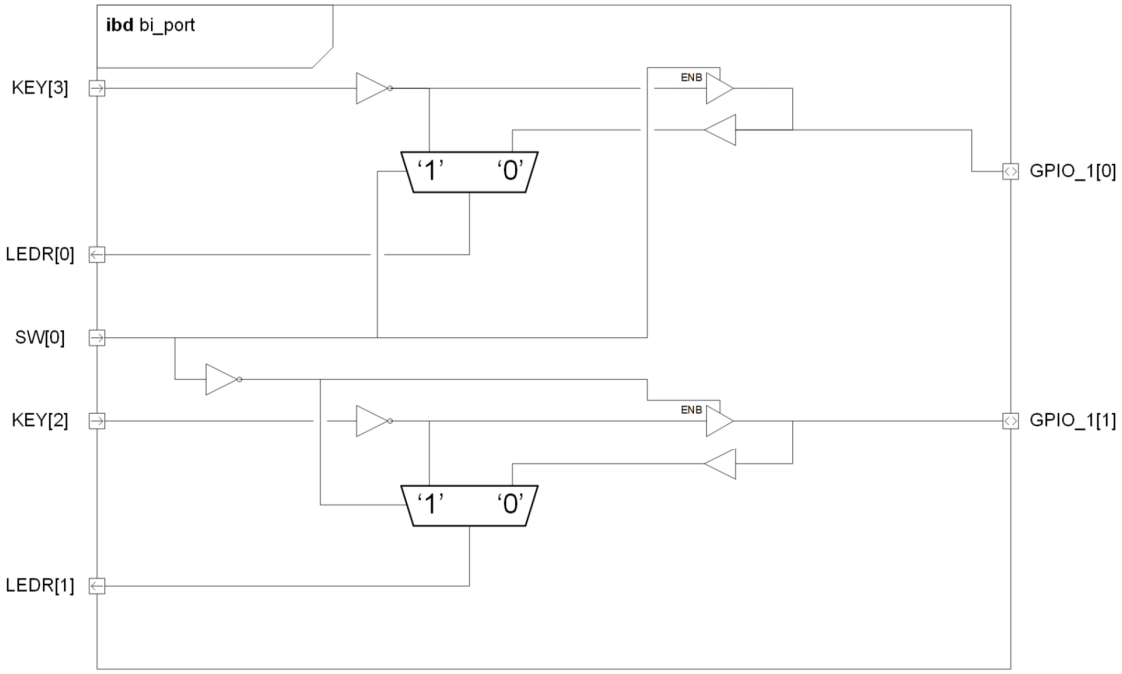

Fig. 4: *Bi port IBD.*

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity bi_port is
5    port(
6      KEY    : in    std_logic_vector(3 downto 2);
7      SW     : in    std_logic_vector(0 downto 0);
8      LEDR   : out   std_logic_vector(1 downto 0);
9      GPIO_1 : inout std_logic_vector(1 downto 0)
10   );
11 end;
```

Tab. 2: *Bi-dir test interface.*

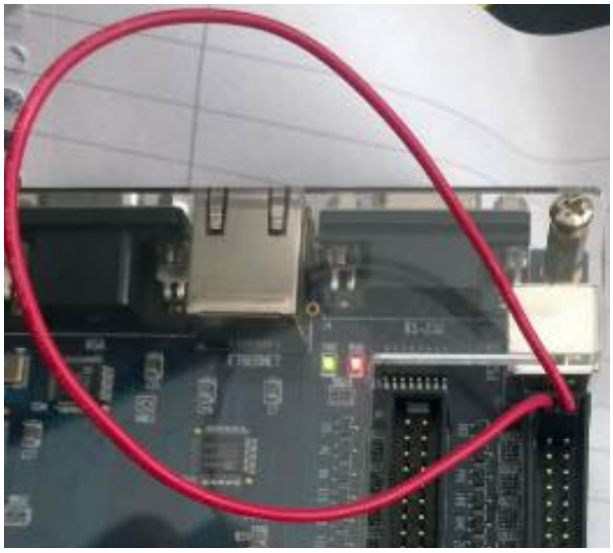

Fig. 5: *GPIO1 connections.*

b) Connect `GPIO_1` pin 0 and pin 1 with a wire as shown in figure 5.

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

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