|  |
| --- |
| EXPERIMENT: A First Computing Machine - Calculator |

References

1. *Digital Design Using Digilent FPGA Boards – VHDL / Active-HDL Edition*
   1. Example 10: *Generic Multiplexers*
   2. Example 44: *N-Bit Register*
   3. Example 48: *Clock Pulse*
   4. Example 51: *N-Bit Counter*
   5. Example 52: *Clock Divider*
   6. Chapter 5: *Combinational Logic*
   7. Chapter 6: *Arithmetic Circuits*

LAB

Part 1

The function unit, *Funit*, shown in Fig. 1 is a type of ALU that is used in a microprocessor core that executes instructions from the Forth programming language. The names of these Forth instructions and their functions are shown in Table 1.

y(15:0) x(15:0)

fcode(3:0)

***Funit*** z(15:0)

Figure 1: Funit

1. Write a VHDL program called *funit.vhd* that will implement this function unit. Simulate the program with a test bench that uses fixed values of *x* = X"FFFFFEDC", *y* = X"0000001F" and *fcode* from X"0" through X"F".
2. Demonstrate your simulation to your lab instructor.

**Table 1 Instructions for the Function Unit**

|  |  |  |
| --- | --- | --- |
| fcode | Name | Function |
| 0 | + | Add x to *y* (*x*+*y*) |
| 1 | - | Subtract *y* from *x* (*x*-*y*) |
| 2 | 1- | Subtract 1 from *x* (*x*-1) |
| 3 | 1+ | Add 1 to x (x+1) |
| 4 | INVERT | Complement all bits of *x* (not *x*) |
| 5 | AND | AND all bits of *x* to *y* (*x* and *y*) |
| 6 | OR | OR all bits of *x* to *y* (*x* or *y*) |
| 7 | XOR | XOR all bits of *x* to *y* (*x* xor *y*) |
| 8 | 2\* | Logic shift left *x* |
| 9 | U2/ | Logic shift right *x* |
| A | 2/ | Arithmetic shift right *x* |
| B | NEGATE | Output the 2's complement of x |
| C | 2\*\*x | Output 2 to the power of x |
| D | FALSE | Set all bits in *z* to '0' |
| E | TRUE | Set all bits in *z* to '1' |
| F | Y | Pass through y |

**Deliverables:** *funit.vhd*, *funit\_tb.vhd*, functional simulation of *funit*

Part 2

Now, you will use your *funit* component from Part 1 to create a calculator using the top-level design shown in Fig. 3. **Do not modify the *funit* component**. We will use this calculator to enter two values *a* and *b* from the switches into the two registers *Treg* and *Nreg*, and then calculate the value of

The Forth-type instructions shown in Table 2 can be used to make this calculation using hex switch values of X"1234" and X"4263" for *a* and *b* respectively. Note, the instruction “S@” stands for Switch Fetch, or load from switches.

**Table 2**

|  |  |  |
| --- | --- | --- |
| Instruction | Treg | Nreg |
| S@ | 4263 (b) | ---- |
| S@ | 1234 (a) | 4263 (b) |
| + | 5497 (a+b) | 4263 (b) |
| DUP | 5497 (a+b) | 5497 (a+b) |
| 2\* | A92E 2\*(a+b) | 5497 (a+b) |
| + | FDC5 3\*(a+b) | 5497 (a+b) |
| U2/ | 7EE2 1.5\*(a+b) | 5497 (a+b) |
| 1+ | 7EE3 1.5\*(a+b)+1 | 5497 (a+b) |

1. Listing 1 shows a VHDL program for an 8-word ROM. Modify this program so that the data values encode the eight instructions shown in Table 2 using the format shown in Fig. 4.
2. Write a VHDL program called *calc\_top.vhd* that will implement the top-level design shown in Fig. 3. Use the version of *x7segb8.vhd* given on Moodle and the clock divider component *clkdiv* from Listing 7.18 using only 190 Hz output. The program counter component, *pc*, is just a 3-bit counter and the *clock\_pulse* component is given in Listing 7.13.
3. Create a testbench called *calc\_tb.vhd* that simulates loading X"4261" and X"1230" onto the switches and simulate long enough to obtain the result. Add the 't' and 'n' signals to the waveform so that the Treg and Nreg outputs are visible in the simulation.
4. Synthesize the design and download the .*bit* file to the Nexys board. Demonstrate the lab to your lab instructor by pressing *btn*(0) to step through all instructions, so that we may see the result after each instruction is executed.

**Deliverables:** *calc\_top.vhd*, *rom8.vhd*, *calc\_tb.vhd*, functional simulation of calc\_top

**Listing 1**

**library** IEEE;

**use** IEEE.std\_logic\_1164.**all**;

**use** IEEE.std\_logic\_unsigned.**all**;

**entity** rom8 **is**

**port** (

addr: **in** STD\_LOGIC\_VECTOR (2 **downto** 0);

M: **out** STD\_LOGIC\_VECTOR (7 **downto** 0)

);

**end** rom8;

**architecture** rom8 **of** rom8 **is**

**constant** data0: STD\_LOGIC\_VECTOR (7 **downto** 0) := "00000000";

**constant** data1: STD\_LOGIC\_VECTOR (7 **downto** 0) := "11001000";

**constant** data2: STD\_LOGIC\_VECTOR (7 **downto** 0) := X"F9";

**constant** data3: STD\_LOGIC\_VECTOR (7 **downto** 0) := X"AF";

**constant** data4: STD\_LOGIC\_VECTOR (7 **downto** 0) := X"64";

**constant** data5: STD\_LOGIC\_VECTOR (7 **downto** 0) := X"95";

**constant** data6: STD\_LOGIC\_VECTOR (7 **downto** 0) := "01101100";

**constant** data7: STD\_LOGIC\_VECTOR (7 **downto** 0) := "11010100";

**type** rom\_array **is** array (NATURAL range <>) **of**

STD\_LOGIC\_VECTOR (7 **downto** 0);

**constant** rom: rom\_array := (

data0, data1, data2, data3,

data4, data5, data6, data7);

**begin**

**process**(addr)

**variable** j: integer;

**begin**

j := conv\_integer(addr);

M <= rom(j);

**end process**;

**end** rom8;

X"0000" & sw(15:0)

**calc\_top**

**pc**

mclk

clkp

addr(2:0)

**rom8**

ld(7:0)

clr

clr

**clkdiv**

**mux**

msel

M(4)

tin(15:0)

clkp

tld

M(6)

clk190

clr

**clock**\_**pulse**

**Treg**

M(7:0)

clr

inp =

btn(0)

clkp

t(15:0)

clkp

**Nreg**

clr

mclk

nld

M(5)

n(15:0)

**x7segb8**

x

**Funit**

y

x

z

fcode(3:0)

M(3:0)

an(7:0)

a\_to\_g(6:0)

dp

z1(15:0)



Fig. 3 Circuit diagram of *calc\_top.vhd*



Fig. 4 Instruction format of data in *rom8.vhd*

Part 3

Create a table similar to Table 2 that will calculate the function

where **b is the first value entered from the switches and *a* is the second value entered from the switches**. Make a ROM for your program. You may need to increase the size of your ROM. Create a test bench and simulate your program for the values of *a=*X"3456" and *b=*X"1220", making sure to add the 't' and 'n' signals to the waveform as before. Synthesize and implement the program on the Nexys board. Demonstrate the operation to your lab instructor.

**Deliverables:** *rom8\_2.vhd* (from part 3), *calc\_top2.vhd* (from part 3), *calc\_tb\_2.vhd* (from part 3), functional simulation of calc\_top2