CE1911 LABORATORY
SYSTEM DESIGN SPECIFICATIONS

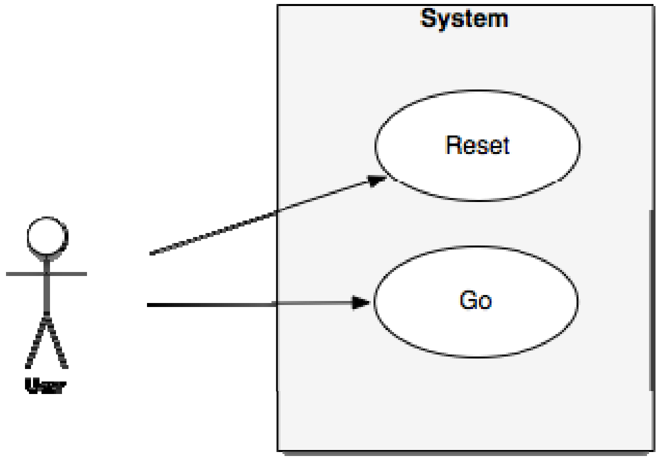
SYSTEM DESCRIPTION

This specification describes and defines the requirements of the CE1911 special purpose processor. The processor uses a set of input toggle switches to provide values for the nibble-sized algebraic variables X and Y. It uses two toggle switches to choose one of four possible algebraic functions. The processor completes the chosen algebraic function as an 8-bit calculation and produces the 8-bit result. The 8-bit result is displayed as hexadecimal on 7-segment displays.

REQUIREMENTS

1. The system must present power-up system reset messaging.
2. The system must wait for a go signal provided by the user.
3. The system must use values provided on toggle switches.
4. The system must display the output value in hexadecimal on 7-segment displays.
5. The system must operate from DC power or USB supplied power.

USE CASE DIAGRAM



USE CASE EVENTS

1. Reset

    A. The user pushes the reset button.
    B. The system displays a power-up reset message.
    C. The system waits for the **GO** command to complete a calculation.

2. Go

    A. The user pushes the go button.
    B. The system calculates the chosen function and displays the result.

SPECIFICATION OF SYSTEM INPUTS AND OUTPUTS

1. System Inputs

   A. The system uses an active-low reset pushbutton.

   B. The system uses an active-low pushbutton to request a calculation.

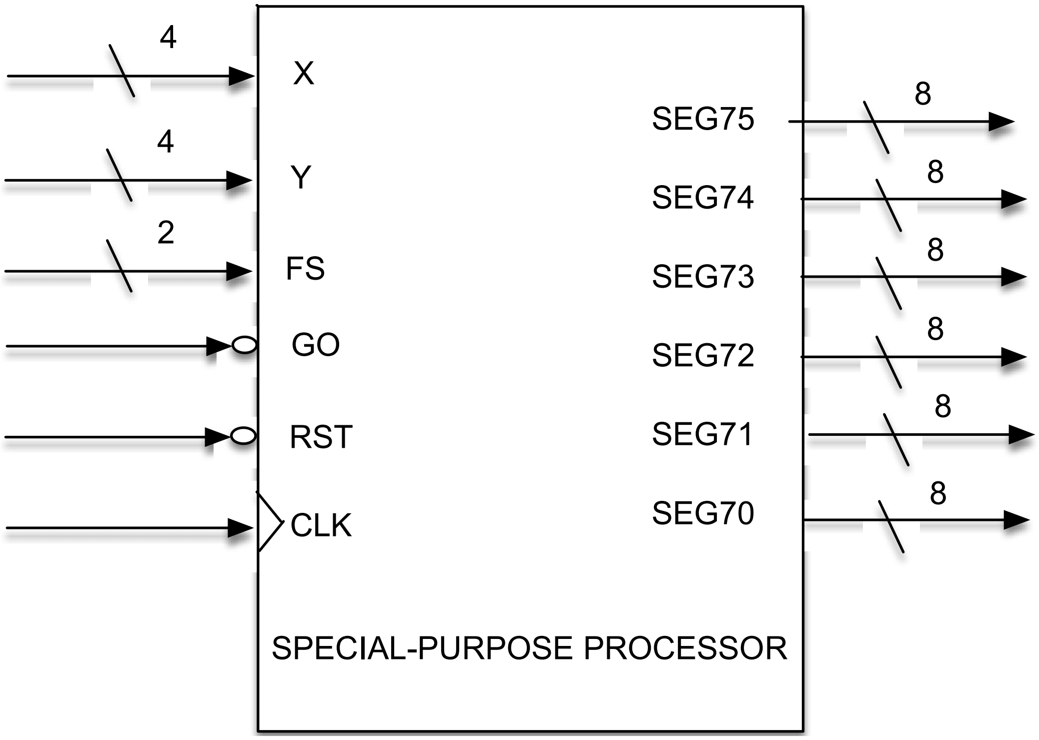
   C. The system uses eight toggle switches to provide nibble values X and Y.

   D. The system uses two toggle switches to request an algebraic function.

2. System Outputs

   A. The system uses six 7-segment displays.

3. Entity Input and Output Diagram



CE1911 LABORATORY
SYSTEM DESIGN SPECIFICATIONS

SYSTEM FUNCTIONAL SPECIFICATION

A. The system is a finite state machine controlling an arithmetic data path.

- i. The machine enters a reset state when the reset pushbutton is pushed.
- ii. The machine moves through power-on reset messaging states when the reset pushbutton is released.
- iii The machine waits in a hold state until commanded to begin calculation.
- iii. The machine moves through a calculation timeline when the go pushbutton is pushed.
- iv. The machine returns to the hold state after calculation.

B. Power-on reset messaging algorithm

- i. Center and display "bPc on" using the DSEG7 font.
- ii. Wait one second
- iii. Center and display "rEAdy" using the DSEG7 font.
- iv. Wait one second

C. Calculation algorithm

```
DO
    WAIT UNTIL GO COMMAND
    SWITCH(FS)
        CASE 0:    CALCULATE 8X+4Y
        CASE 1:    CALCULATE 5X-Y
        CASE 2:    CALCULATE 12Y+6
        CASE 3:    CALCULATE 2X+3Y-2
    DISPLAY 8-BIT RESULT AS TWO HEX NIBBLES
FOREVER
```

D. ALU Operations

i. An ALU will be created with the following operations selected by a control signal called ALUS.

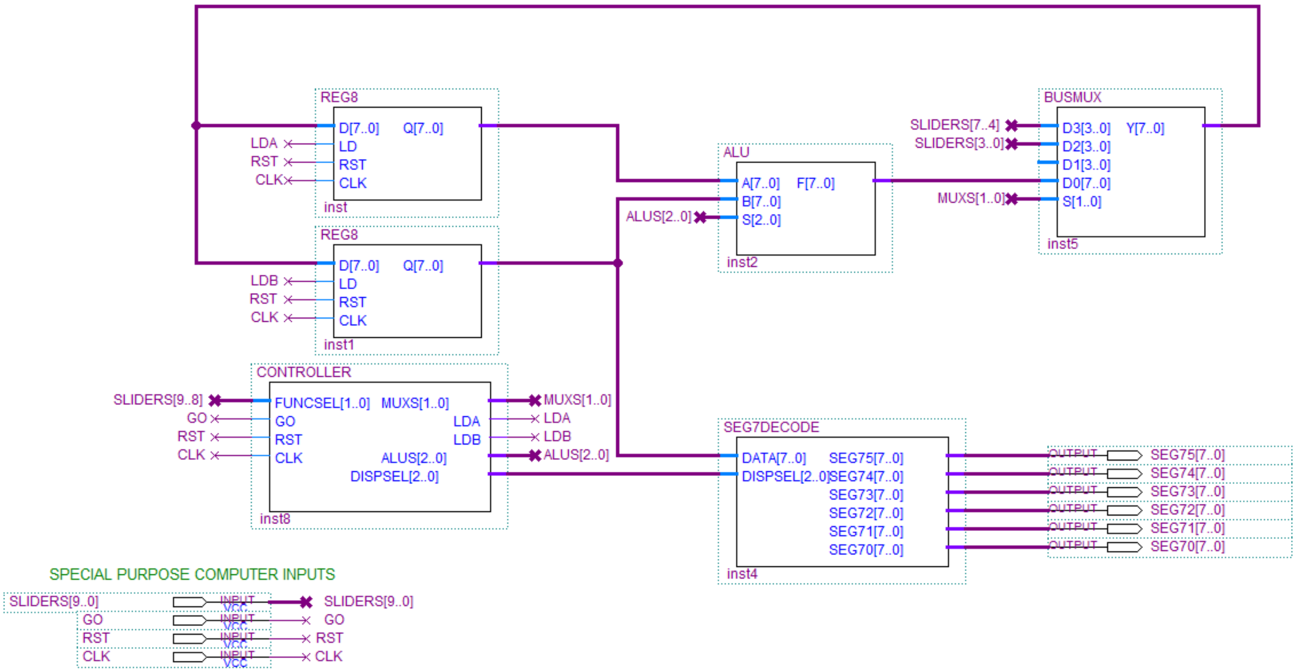
ALUS	F(A,B,S)
0	F=0
1	F=1
2	F=B-1
3	F=A+B
4	F=A-B
5	F=A+A
6	F=A AND B
7	F=A OR B

USER INTERFACE SPECIFICATION

bPc on rEAdy

and calculated numerical results

SCHEMATIC DIAGRAM



NOTE: BUSMUX as a name conflicts in schematic projects. Use the name MYBUSMUX instead of BUSMUX in your work!

IMPLEMENTATION TECHNOLOGY

1. The system is described as dataflow, behavioral, and structural VHDL entities.
2. The system is implemented in an off-the-shelf Altera DE10-Lite FPGA board.

Table 1: Project Files

PROJECT NAME: LW9		
FILENAME	**FORMAT**	**CIRCUIT**
reg.vhd	behavioral VHDL	8-bit register with synchronous reset and synchronous load
alu.vhd	with-select dataflow	ALU implementing the operations from the functional specification
busmux.vhd	with-select dataflow	bus multiplexer
seg7decode.vhd	with-select dataflow	seven segment decoder for messages and numbers
controller.vhd	behavioral VHDL	Moore finite state machine to control the data path
lw9.vhd	structural VHDL	top-level schematic implemented as structural VHDL

TEST AND VERIFICATION PLAN

1. Simulation verifies the reset and operational use cases.

   A. Simulation includes the reset use case as the first event.
   B. Simulation overwrites arbitrary values on the X and Y data inputs.
   C. Simulation overwrites an arbitrary value on the function select inputs.
   C. Simulation asserts the go signal for one clock cycle to request calculation.
   D. Visual inspection verifies the calculation result.
   E. Simulation is repeated for all four functions.

2. Laboratory testing verifies the reset and operational use cases.

HINTS

1. Implement the ALU using **ieee.std_logic_unsigned** arithmetic. Use a single with-select statement to assign function values to F.

```
with ALUS select
F <= B"00000000" when B"000",
     B"00000001" when B"001",
     B-1 when B”010”,
       etc.
```

2. The requirements specify inputs X and Y as 4-bit numbers. Yet, the schematic shows 8-bit ALU arithmetic. The four-bit X and Y inputs can be converted to 8-bit values inside the bus multiplexer:

```
with S select
Y <= D0 when B"00",   -- all 8 ALU output bits transfer: 8-bit input to 8-bit output
     B"0000"&D2 when B"10", -- zero valued upper nibble, D2 in lower nibble
     B"0000"&D1 when B"01", -- zero valued upper nibble, D1 in lower nibble
     B"0000"&D3 when others;
```

Cute, huh?! This allows you to convert your four-bit input numbers to eight-bit output numbers when multiplexed into the circuit.

4. The hexadecimal decoder simply uses with-select to decode each 7-segment display output. The display select bus allows the controller to choose what is on screen. You can choose the selection value that chooses bPc on, the selection value that chooses rEAdy, and the value that chooses normal numeric decoding.