Laboratory 2

(Due date: **005**: February 15th, **006**: February 16th)

OBJECTIVES

- ✓ Implement a large combinational circuit using the Structural Description in VHDL.
- ✓ Introduce floating point and fixed point representations for VHDL implementation and Vivado simulation.

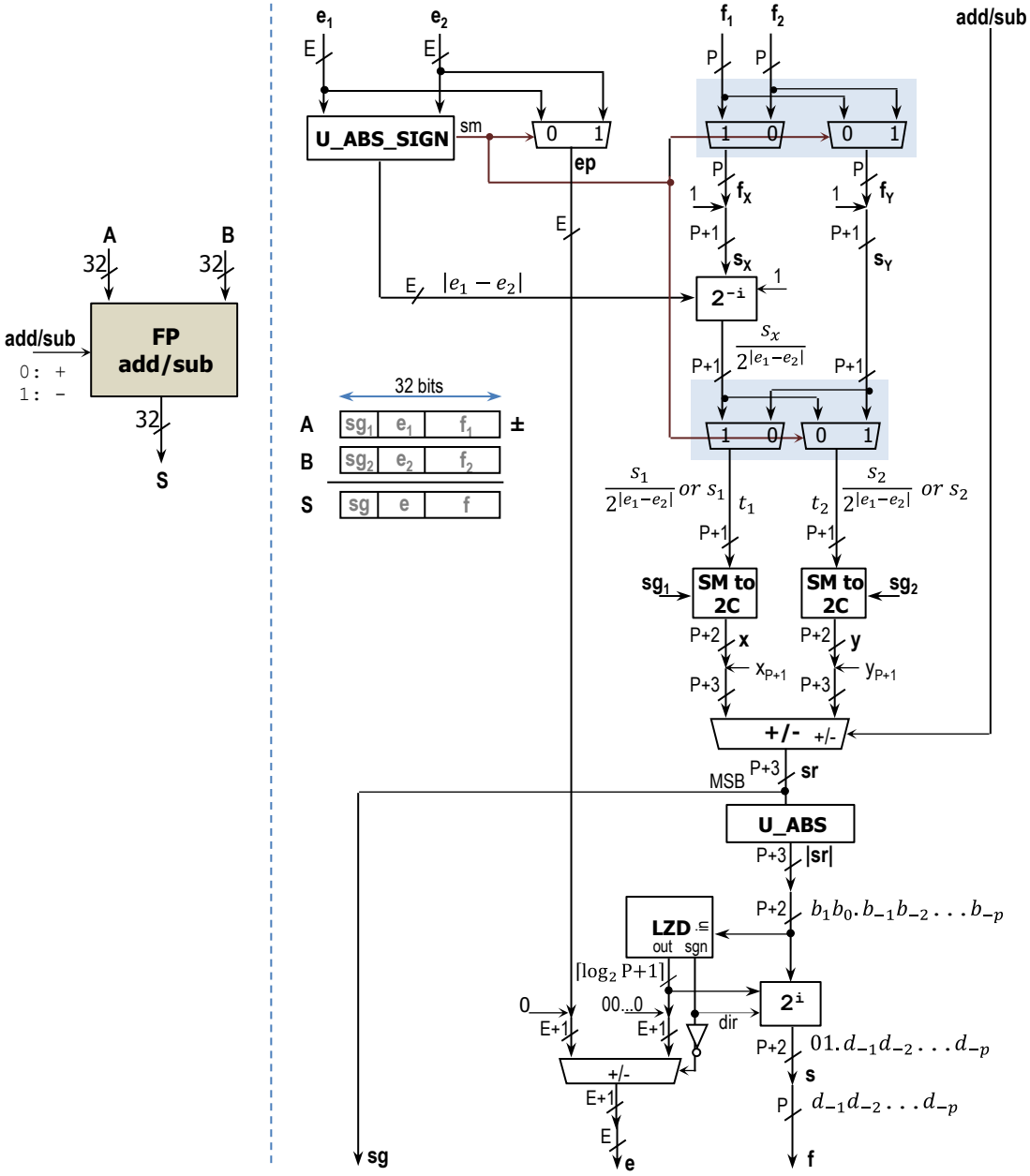
VHDL CODING

- ✓ Refer to the Tutorial: VHDL for FPGAs for parametric code for: adder/subtractor and busmux.

FIRST ACTIVITY: FLOATING POINT ADDER/SUBTRACTOR (100/100)

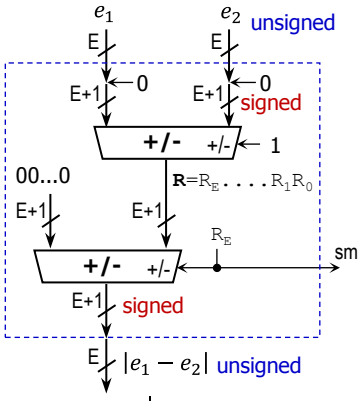
DESIGN PROBLEM

- Implement the following single-precision (E=8, P=23), floating point adder/subtractor. The circuit only works for ordinary numbers generating ordinary numbers (e.g.: the cases $S = A \pm B = 0$, $A = 0$, or $B = 0$ are not considered by this circuit). The exponents in the circuit are biased exponents, so they always are positive numbers.



Instructor: Daniel Llamocca
TAs: David Stern, Luke Nuculaj

- **U_ABS_SIGN**: This circuit computes the absolute value of the difference of two unsigned numbers (e_1 and e_2). It also generates the signal sm.
  - ✓ $e_1, e_2 \in [0, 2^E - 1] \rightarrow e_1 - e_2 \in [-(2^E - 1), 2^E - 1] \rightarrow |e_1 - e_2| \in [0, 2^E - 1]$. So, $|e_1 - e_2|$ only needs E bits as an unsigned number.
  - ✓ The figure (right) depicts a suggested circuit.
    - It uses the parametric adder/subtractor (`my_addsub.vhd`).
    - The result $|e_1 - e_2|$ is an unsigned number. Thus, it only needs E LSBs (taken from the adder/subtractor output).
  - ✓ The output sm tells us whether $e_1 \geq e_2$. This determines what f_x and f_y are.
    - $e_1 \geq e_2 \rightarrow sm = 0, ep = e_1, f_x = f_2, f_y = f_1$
    - $e_1 < e_2 \rightarrow sm = 1, ep = e_2, f_x = f_1, f_y = f_2$

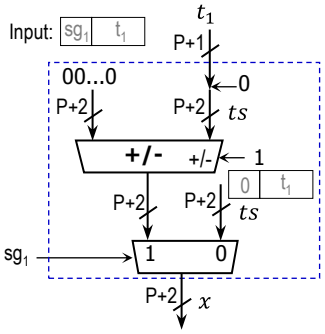


sm	ep	s_x	s_y	t_1	t_2
0	e_1	$s_2 = 1.f_2$	$s_1 = 1.f_1$	s_1	$\frac{s_2}{2^{\lvert e_1 - e_2 \rvert}}$
1	e_2	$s_1 = 1.f_1$	$s_2 = 1.f_2$	$\frac{s_1}{2^{\lvert e_1 - e_2 \rvert}}$	s_2

s_x: operand that gets divided by $2^{|e_1 - e_2|}$

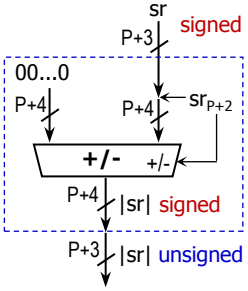
- **Barrel shifter 2^{-i}**: This circuit performs alignment of s_x, where we always shift to the right by $|e_1 - e_2|$ bits. Use the VHDL code `mybarrelshift_gen.vhd` with `SHIFTTYPE = "LOGICAL"` (unsigned input), `SW=E`, and `dir = '1'`.

- **SM to 2C**: This block converts a $(P + 2)$-bit number represented as a sign-and-magnitude into a 2C number. Note: the input is composed of the sign and the $(P + 1)$-bit magnitude.
  - ✓ The figure (right) depicts a suggested circuit (Inputs: sg_1 and t_1. Output: x). It uses the parametric adder/subtractor (`my_addsub.vhd`) and a multiplexor.
    - If the sign is '0', then the output result is equal to the sign-and-magnitude input.
    - If the sign is '1', then we convert the negative sign-and-magnitude input into its 2C representation. This is accomplished by appending a '0' to the magnitude of the input and applying the 2C operation.



- **Main adder/subtractor**: This circuit operates in 2C arithmetic. In order to avoid overflow, the $(P + 2)$-bit outputs from the SM to 2C blocks are sign extended to $P + 3$ bits.
  - ✓ Input operands (**x, y**) $\in [-2^{P+1} + 1, 2^{P+1} - 1]$, Output result ($sr$) $\in [-2^{P+2} + 2, 2^{P+2} - 2]$.

- **U_ABS block**: It takes the absolute value of sr, which is a number represented in 2C arithmetic. The output ($|sr|$) is provided as an unsigned number.
  - ✓ The figure (right) depicts a suggested circuit.
    - It uses the parametric adder/subtractor (`my_addsub.vhd`).
    - The unsigned result $|sr|$ only needs $P + 3$ LSBs (taken from the adder/subtractor output).
  - ✓ Note: In the floating-point adder/subtractor architecture, we only use the $P + 2$ LSBs from the **U_ABS** output. This is because $sr \in [-2^{P+2} + 2, 2^{P+2} - 2]$. Thus, $|sr| \in [0, 2^{P+2} - 2]$; this only requires $P + 2$ bits in unsigned representation.



- **Leading Zero Detector (LZD)**: This circuit outputs an integer number that indicates the amount of shifting required to normalize the result of the summation. It is also used to adjust the exponent. This circuit is commonly implemented using a priority encoder. `result` $\in [-1, p]$. The result is provided as sign and magnitude. Use the following code: `myLZD.vhd`.

Operands		Bitwidth
Input	`in`: $b_1 b_0 . b_{-1} \ldots b_{-p}$	P+2
Outputs	`sgn`	1
	`out`	$\lceil \log_2 P + 1 \rceil$

`myLZD.vhd` parameters:
`inputWidth = P+2`
`outputWidth =` $\lceil \log_2$`(P+1)`$\rceil$

  - ✓ The following table details how the expected result is encoded into the signals `out` and `sgn`.

`result`	`output`	`sign`	Actions
$[0, p]$	$sh \in [0, p]$	0	The barrel shifter needs to shift to the left by sh bits. Exponent adder/subtractor needs to subtract sh from the exponent ep.
-1	$sh = 1$	1	The barrel shifter needs to shift to the right by 1 bit. Exponent adder/subtractor needs to add 1 to the exponent ep.

Instructor: Daniel Llamocca
TAs: David Stern, Luke Nuculaj

- **Exponent adder/subtractor**: This circuit operates in 2C arithmetic. The result is a biased exponent.
  - ✓ Input operands: signed with $E+1$ bits.
    - ep: for ordinary numbers, $ep \in [1, 2^E - 2]$; this is an unsigned number with E bits. We zero-extend ep to $E+1$ bits to turn it into a signed number.
    - out: from LZD. This unsigned number needs to be zero-extended to $E+1$ bits.
  - ✓ Output result (biased exponent: $ep \pm out$): Signed number $E+1$ bits.
    - However, we note that the biased exponent cannot be negative: at most we subtract p from ep, or add 1. As a result, the biased exponent (e) can be represented as an E-bit unsigned number (we only use E LSBs from the output)

- **Barrel shifter 2^i**: It performs normalization of the final summation. We shift to the left (from 0 to P bits) or to the right (1 bit). Use the VHDL code `mybarrelshift_GEN.vhd` with `SHIFTTYPE="LOGICAL"` (unsigned input), `SW =` $\lceil \log_2(P+1) \rceil$, and `dir=sign(LZD)`.

- **VIVADO DESIGN FLOW FOR FPGAs – NEXYS A7-50T**
  - ✓ Create a new Vivado Project. Select the **XC7A50T-1CSG324 Artix-7 FPGA** device.
  - ✓ Write the VHDL code for the 32-bit floating point adder subtractor. Utilize the Structural Description: have a separate file for the components (2-to-1 bus MUX, SM to 2C, U_ABS, LZD, Barrel shifter, adder/subtractor, U_ABS_SIGN), and the top file (where you will interconnect all the components).
  - ✓ Write the VHDL testbench to test the following cases:

```
10DAD000 – 90FAD000 = 116AD000
60A10000 + C2F97000 = 60A10000
40B00000 + C2FA8000 = C2EF8000
42FA8000 + C0E00000 = 42EC8000
3DE38866 – B300D959 = 3DE3886A
60A10000 – 60A1F000 = DCF00000
```

  - ✓ Perform Functional Simulation and Timing Simulation of your design. **Demonstrate this to your TA**.
    Note that when testing, it might be very useful to represent the inputs and output in single floating point precision. Or we might also want to represent the intermediate signals not only as integer numbers but also as fixed point numbers. You can use the `Radix → Real Settings` in Vivado simulator window to do so.

SUBMISSION

- Submit to Moodle (an assignment will be created):
  - ✓ This lab sheet (as a .pdf) completed and signed off by the TA (or instructor).
  - ✓ (As a .zip file) All the generated files: VHDL code files, and VHDL testbench. DO NOT submit the whole Vivado Project.
    - Your .zip file should only include one folder. Do not include subdirectories.
    - It is strongly recommended that all your design files, testbench, and constraints file (if applicable) be located in a single directory. This will allow for a smooth experience with Vivado.
    - You should only submit your source files AFTER you have demoed your work. Submission of work files without demoing will be assigned NO CREDIT.

TA signature: _____ **Date:** _____

Instructor: Daniel Llamocca
TAs: David Stern, Luke Nuculaj