Laboratory 6

(Due date: **002/003**: April 4th, **004**: April 5th, **006**: April 6th)

OBJECTIVES

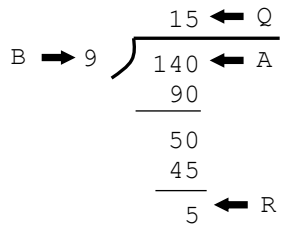
- ✓ Describe Finite State Machines (FSMs) in VHDL.
- ✓ Implement a Digital System: Control Unit and Datapath Unit.

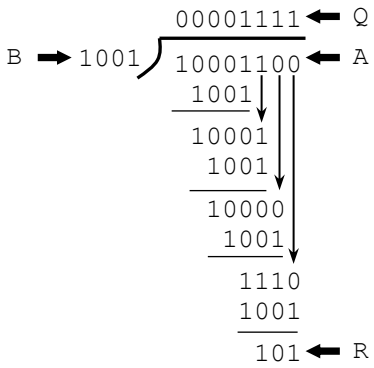
VHDL CODING

- ✓ Refer to the Tutorial: VHDL for FPGAs for parametric code for: register, shift register, counter, adder/subtractor.

ITERATIVE DIVIDER IMPLEMENTATION (100/100)

- Given two unsigned numbers A and B, we want to design a circuit that generates the quotient Q and a remainder R. $A = B \times Q + R$. The algorithm that implements the traditional long-hand division is as follows:

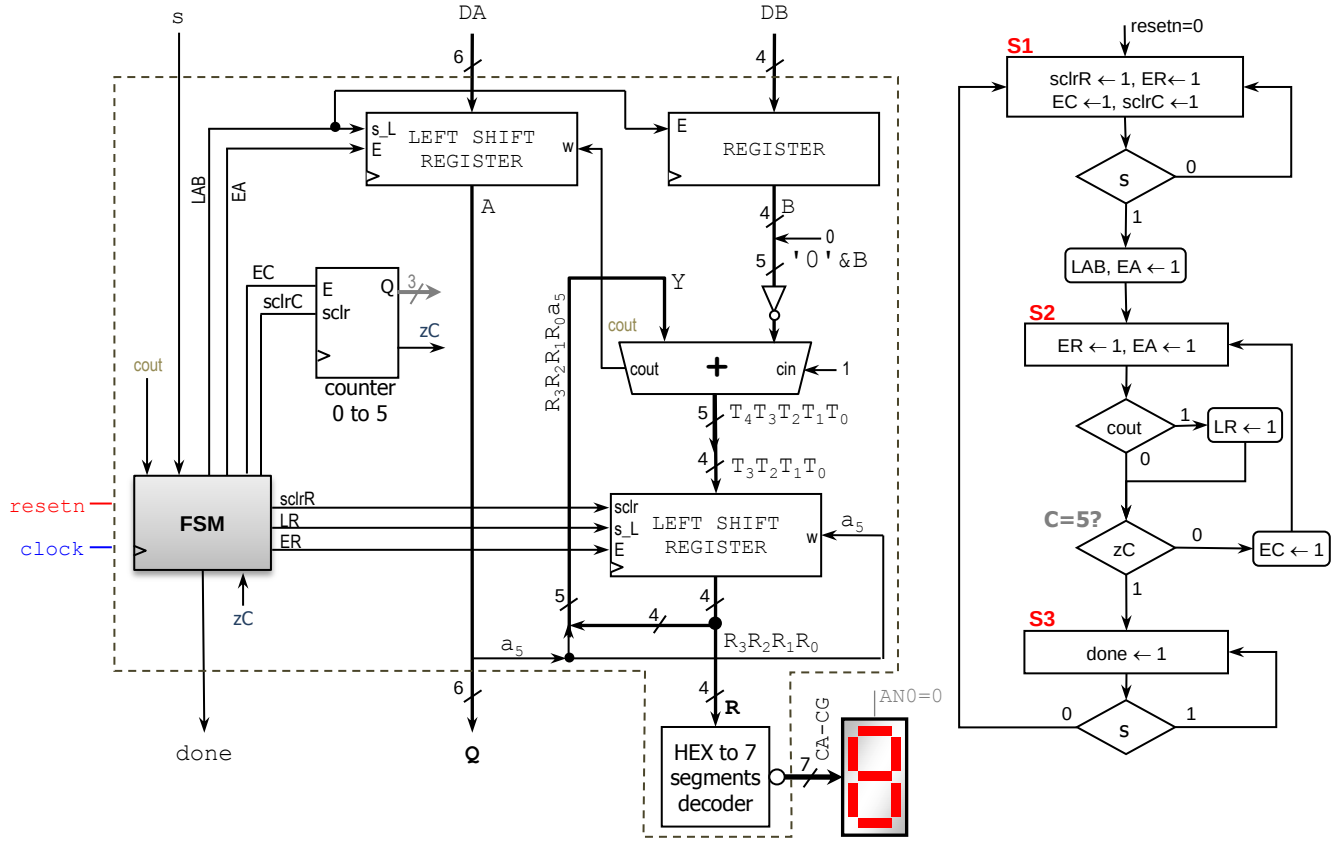





```
ALGORITHM

R = 0
for i = n-1 downto 0
    left shift R (input = a_i)
    if R ≥ B
        q_i = 1, R ← R-B
    else
        q_i = 0
    end
end
```

- Based on the algorithm, an iterative architecture is presented for **DA** with 6 bits and **DB** with 4 bits. The register **R** stores the remainder. A division operation is started when $s = 1$ (where **DA** and **DB** values are captured). The signal done is asserted to indicate that the operation has been completed and the result appears in **Q** and **R**.
  - ✓ At every iteration, we update **R** by either: i) shifting in the next bit of A, or ii) shifting in the next bit of A and subtracting B from **R**. Also, q_i (a bit of quotient Q) is computed and shifted into register **A**.



Instructor: Daniel Llamocca
TAs: Sumati Gupta, Fizza Shamim, Moath Algharibeh

- The circuit includes two parallel access left shift registers, a 4-bit register, a modulo-6 counter, a 5-bit adder/subtractor (addsub=0: add, addsub=1: subtract), and an FSM. Each sequential component has *resetn* and *clock* inputs.
  - ✓ Modulo-6 counter: It includes: i) a synchronous input sclr that clears the count when E=sclr=1, and ii) an output zC that is asserted when the count reaches 5. The counter increases its value when E=1, sclr=0. Note that *Q* is unused.
  - ✓ Parallel Access Left-shift registers: Note that one of the shift registers includes a synchronous input *sclr* that clears the register outputs when E=sclr=1. Refer to Lecture Notes – Unit 6 for a description of the circuit and its operation.

- The circuit is an example of a Digital System: It includes a Control Circuit (FSM) and a Datapath Circuit. The Datapath Circuit is made from combinational and sequential components. The circuit is also called a Special-Purpose Processor. In this case, the special purpose is the unsigned division.

PROCEDURE

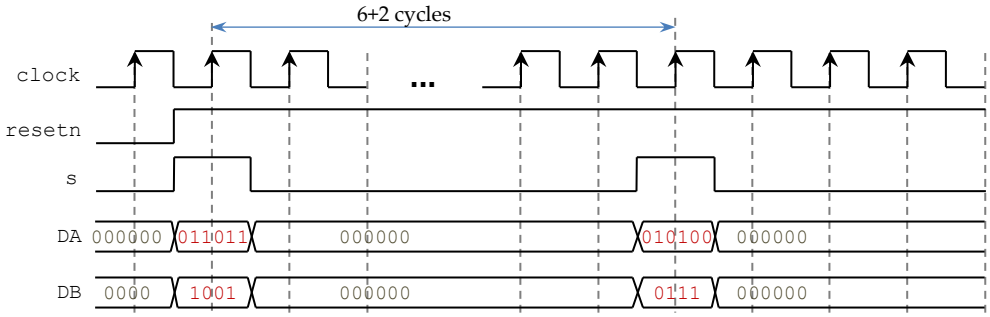
- **Vivado: Complete the following steps:**

  - ✓ Create a new Vivado Project. Select the corresponding Artix-7 FPGA device (e.g.: the XC7A50T-1CSG324 FPGA device for the Nexys A7-50T).
  - ✓ Write the VHDL code for the given circuit. Synthesize your circuit to clear syntax errors and critical warnings.
    - Use the Structural Description: Have a separate file for the modulo-6 counter, parallel access shift register with sclr, register, adder/subtractor, Hex to 7-segment decoder, FSM, and top file.
    - Suggestion: Use parametric code (set up the proper parameters with *generic map*) for these components:
      - Parallel access shift registers with sclr: my_pashiftreg_sclr
        * Note that one of these shift registers does not use the sclr input. In this case, that input should be tied to '0'.
      - Counter: my_genpulse_sclr (include in the top file: `use ieee.math_real.log2; use ieee.math_real.ceil;`)
      - Register with enable: my_rege
      - Adder/subtractor: my_addsub

  - ✓ Write the VHDL testbench (generate a 100 MHz input clock for your simulations) to test the following cases:

DA	**DB**	**Q**	**R**	CA–CG (complete!)
011011 (27)	1001 (9)	000011	0000	
010100 (20)	0111 (7)	000010	0110	
111110 (62)	1001 (9)	000110	1000	
111001 (57)	0110 (6)	001001	0011	
111011 (59)	1011 (11)	000101	0100	
111101 (61)	1101 (13)	000100	1001	

    - The testbench should be written according to the timing diagram shown in the figure, where the first two values for **DA/DB** are fed. Note that there are $6 + 2 = 8$ cycles between data values.

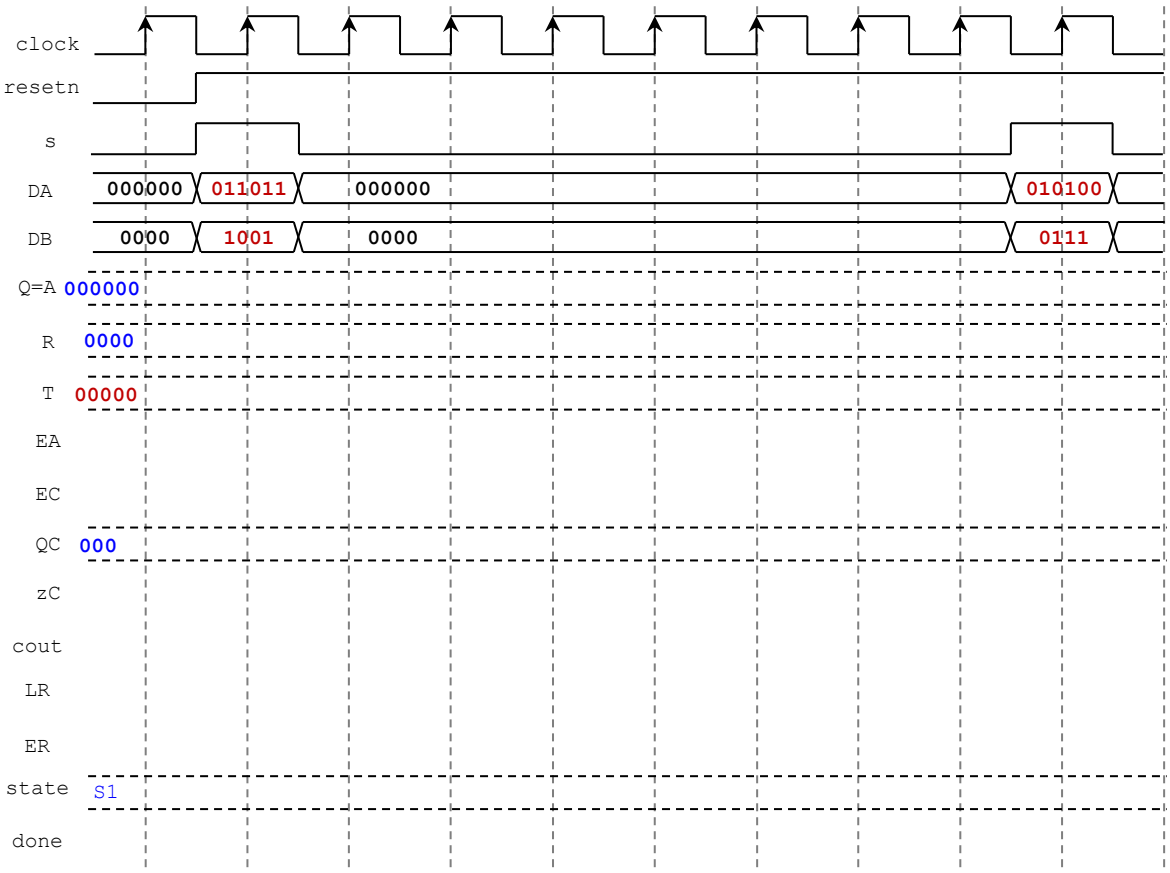


  - ✓ Perform Behavioral Simulation and Timing Simulation of your design. **Demonstrate this to your TA**.
    - Behavioral Simulation: For proper debugging, add internal signals (e.g.: state, **R**, **T**) to the waveform view. Go to: SCOPE window: testbench → UUT. Then go to Objects Window → Signal(s) → Add to Wave Window. Then, re-run the simulation.
      - Note that you can represent data as unsigned integers (use Radix → Unsigned Decimal).
      - Your simulation might need more time than Vivado Simulator's default (1us). For example, to add 5 us, you can go to the TCL console and type: `run 5 us`.

Instructor: Daniel Llamocca
TAs: Sumati Gupta, Fizza Shamim, Moath Alharibeh

- Debugging: If your circuit works as expected, the results appear on `Q` and `R` (when done=1) and they should match those listed in the previous table.
  - In the (likely) event that the results are incorrect, you need to perform cycle-accurate simulation for one operation: if one operation is correct, most likely the other would be also correct:
    - Manually complete the following timing diagram: it shows the detailed signal transitions for the first operation.
    - Re-run the simulation (include all the internal signals into your window). Then, cycle by cycle, compare the signal values until you find a mismatch. This will help you locate the source of the incorrect value(s).

  - If you get the correct results for all operations, complete the timing diagram based on your Vivado simulation.



✓ I/O Assignment: Create the XDC file associated with your board.
- Suggestion (for Basys 3, use `SW15` instead of `CPU_RESETN` for *resetn* input)

Board pin names	CLK100MHZ	CPU_RESETN	BTNC	SW9-SW4	SW3-SW0	LED15	LED5-LED0	CA-CG	AN7-AN0
Signal names in code	*clock*	*resetn*	s	DA_5-DA_0	DB_3-DB_0	done	Q_5-Q_0	CA-CG	AN_7-AN_0

- The board pin names (except `CPU_RESETN`) are used by all the listed boards (Nexys A7-50T/A7-100T, Nexys 4/DDR, Basys 3). I/Os: all switches and LEDs are active-high.
- **Basys 3**: There are only four 7-segment displays, hence you only have signals `AN3-AN0`.

- Note: synchronous circuits always require a clock and reset signal.
  - **Reset signal**: As a convention in this class, we use active-low reset (*resetn*). As a result, ensure that *resetn* is tied to the proper board resource:
    - Nexys A7-50T/A7-100T, Nexys 4/DDR: For *resetn*, use `CPU_RESETN` pin. This is an active-low push button.
    - Basys 3: There is no active low push button. Thus, for *resetn*, use `SW15`. Even though `SW15` is active high, we can still think of it as active-low *resetn*, where the circuit is reset when the switch position is OFF ('0').
  - **Clock signal**: Like other signals in the XDC file, you need to uncomment the lines associated with the clock signal and replace the signal label with name used in your code. In addition, there is parameter `-period` that is set by default to `10.00`. This is the period (in ns) that your circuit should support.
    - Nexys A7-50T: In these lines, replace the label `CLK100MHZ` with the signal name you use in your code (`clock`):

```
set_property -dict { PACKAGE_PIN E3    IOSTANDARD LVCMOS33 } [get_ports { CLK100MHZ }];
create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports {CLK100MHZ}];
```

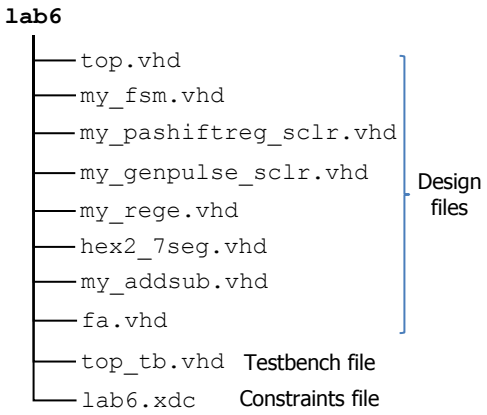

Instructor: Daniel Llamocca
TAs: Sumati Gupta, Fizza Shamim, Moath Algharibeh

- Basis 3: In these lines, replace the label `clk` with the signal name used in your code (`clock`):

```
set_property PACKAGE_PIN W5 [get_ports clk]
set_property IOSTANDARD LVCMOS33 [get_ports clk]
create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports clk]
```

✓ Generate and download the bitstream on the FPGA. Test the circuit (use the same input values as in the testbench). **Demonstrate this to your TA**.

- Submit (as a .zip file) all the generated files: VHDL code files, VHDL testbench, and XDC file to Moodle (an assignment will be created). DO NOT submit the whole Vivado Project.
  - ✓ Your .zip file should only include one folder (the figure shows an example). Do not include subdirectories.
    - It is strongly recommended that all your design files, testbench, and constraints file be located in a single directory. This will allow for a smooth experience with Vivado.



TA signature: _____ **Date:** _____