VHDL projects

In Programmable logic devices class, each project cover some aspects of project, simulation and synthesis necessary to develop basic knowledge about VHDL.

VHDL reference

Lessons:

- Process
- Syncronous process and FSM
- Synthesis and Quartus
- Subprograms in VHDL
- Custom types in VHDL
- External files usage VHDL
- IP components
- SoftCore Nios
- SoftCore RISCV

The softwares used for those projects are:

- Sigasi - Text editor with intelisense for VHDL
- Modelsim - VHDL project simulator
- Quartus - Synthesis tool

In modelsim terminal

```
Choose the project diretory and execute
*do tb.do*

to terminate the simulation
*quit -sim*
```

Blue blocks:

- [x] Adder 16 bits (adder):
- [x] Mux 4 bits (mux4)
- [x] Shifter 8 bits to 16 bits shift left (shifter)
- [x] Seven segments display (seven_segment_cntrl)

- [x] Register 16 bits (reg16)
- [x] Counter 2 bits (Counter)
- [x] Mult 4 bits (mult44)
- [x] Implement and FSM
- [x] Implement 8x8 multiplier

***Procedures and Functions*:**

Do a package containing a `function` and a `procedure` described as follow:

- [x] Function

A function that implements an **arithmetic SHIFT left** operation for a **STD_LOGIC_VECTOR** signal.

Two arguments must be passed to your function:

- *data*
- *n*

The first is the vector that will be rotated and the second how many bits will be rotated. Make a generic function, that is, it should work for any vector size.

- Must be a **generic function** for any vector size
- Your function must be implemented in a package.
- You **cannot use** the numerical package change / rotation function.

Tip: use only concatenation. Also write a testbench to test the package and function.

- [ ] Procedure

A procedure that receives 8 signed signals and returns the **average**, the **highest** value and the **lowest** value of these 8 numbers.

As follow, respectively, names of those returns:

- *ave*
- *max*
- *min*

*Your procedure must be implemented in a package.

Also write a testbench to test the package and the procedure.

Circular queue

Based on the example of Register Bench:

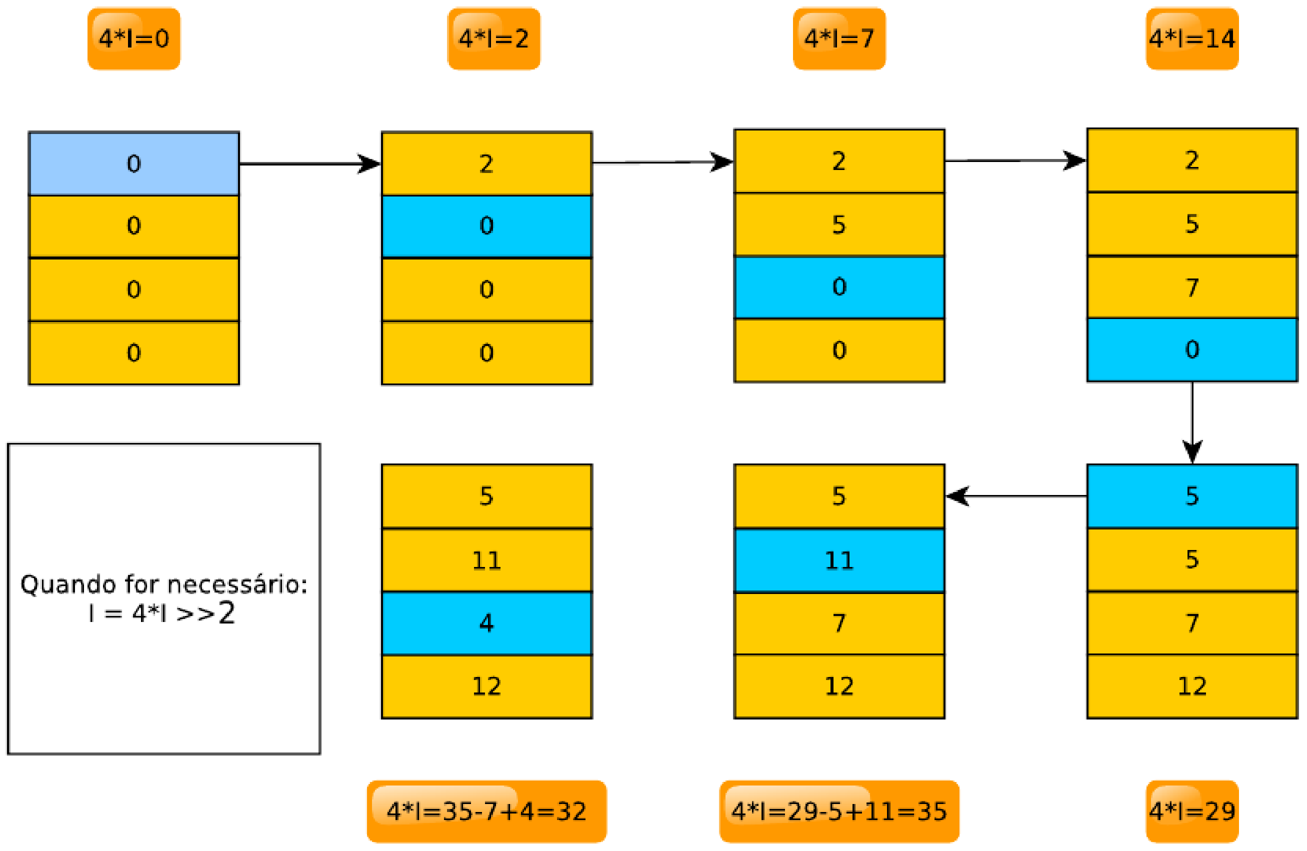
- [x] Create a FIFO `circular queue` based in 32 16-bit registers.

Entries, exits, entity name and behavior are up to you as long as they correspond to the logical behavior of a circular queue and FIFO order.

Digital filter

- [ ] Design an entity that implements a ***digital moving average filter*** according the equation and diagram bellow

$$\begin{aligned}
\overline{I_k} &= \frac{1}{8} * \sum_{k=0}^{7} i_k \\
8 * I_k &= i_{k-7} + i_{k-6} + i_{k-5} + i_{k-4} + i_{k-3} + i_{k-2} + i_{k-1} + i_k \\
8 * I_{k+1} &= 8 * I_k - i_{k-7} + i_{k-6} + i_{k-5} + i_{k-4} + i_{k-3} + i_{k-2} + i_{k-1} + i_k + +i_{k+1}
\end{aligned}$$



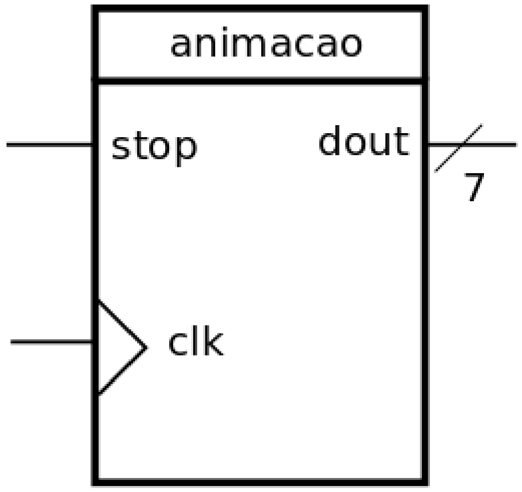
Requirements:

- 32 coeficients
- Use of 32 words of 16 bit from SRAM memory
- The code must infer SRAM IPs

Tip: The filter is syncronous (have clk and rst)

FSM animation

The component must be like shown bellow



- [ ] Design of a *FSM* that animates a seven segments display. The entity contains two entries:
- *clk*
- *stop*
  and an output
- *dout* (7 bits)

The *dout* output is the **diplay input**.
Use **clock frequency** as **1kHz**.

The animation should cause a clockwise movement, to create a smooth movement, you must activate 2 states at same time for a shorter period. The sequence of segments is:
a - ab - b - bc - c - cd - d - de - e - ef - f - fa

If the input *stop* is activated to High, the circuit must return 'a' state, keeping it until *stop* gets in the low state.

Softcore