Arquiteturas de Alto Desempenho 2024/2025

Second practical assignment — VHDL description and simulation of an indexed accumulator

Tomás Oliveira e Silva

1 Work to be done

The main purpose of this assignment is to write a VHDL description of a sequential logic circuit that implements the following C function:

```
unsigned int acc(unsigned int w_addr,unsigned int inc,unsigned int r_addr)
{
  static unsigned int a[16]:
  unsigned int r;

  r = a[r_addr % 16];
  a[w_addr % 16] += inc;
  return r;
}
```

The implementation has to receive one value of each of its arguments in each clock cycle. Use the following entity declaration:

```
entity accumulator is
  generic
  (
    ADDR_BITS : integer range 2 to 8  := 4;
    DATA_BITS : integer range 4 to 32 := 8
  );
  port
  (
    clock        : in std_logic;
    -- write port
    write_addr : in  std_logic_vector(ADDR_BITS-1 downto 0);
    write_inc  : in  std_logic_vector(DATA_BITS-1 downto 0);
    -- read port
    read_addr  : in  std_logic_vector(ADDR_BITS-1 downto 0);
    read_data  : out std_logic_vector(DATA_BITS-1 downto 0)
  );
end accumulator;
```

A write enable signal is not needed: when no accumulation is desired just use any write address and a zero increment. This simplifies the implementation.

The work has four parts. For each of them provide an architecture and a testbench.

1. **[Mandatory]** Single-cycle implementation: do the read, addition and write operations in the same clock cycle.
2. **[Highly recommended]** Multi-cycle implementation: do the read, addition and write operations in three clock cycles (one for each). Be aware: you must deal with the situtation when the same write address is used in consecutive clock cycles.
3. **[Optional]** Do at least one of the two previous items for the following modified accumulator.

   ```
   unsigned int acc(unsigned int w_addr,unsigned int inc,unsigned int shift,unsigned int r_addr)
   {
   ```

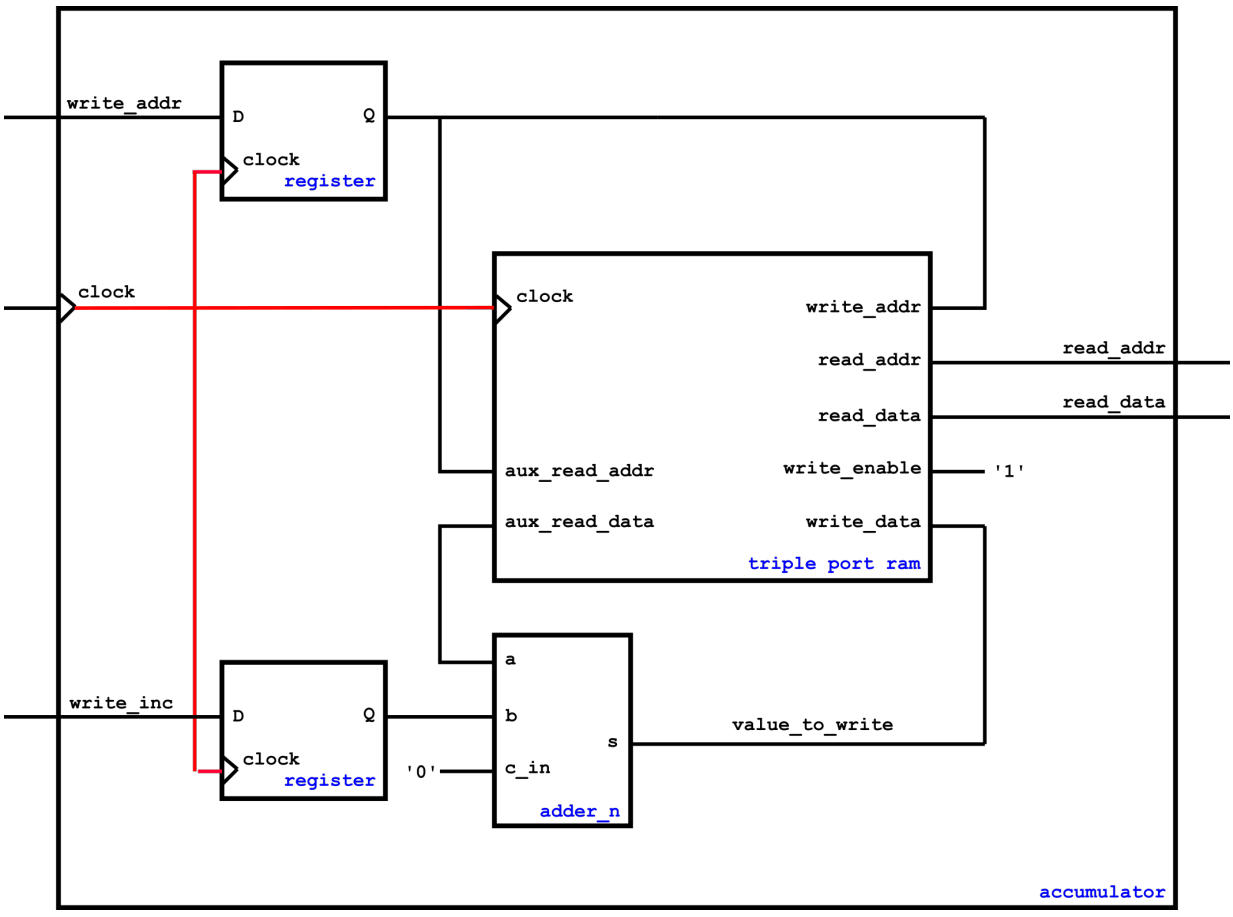
```
  static unsigned int a[16]:
  unsigned int r;

  r = a[r_addr % 16];
  a[w_addr % 16] += inc << (shift % 32);
  return r;
}
```

For that you must implement an efficient barrel-shifter entity. (Just create an entity that shifts or not an input vector by a fixed number of bits, and intantiate it several times in tandem for shifts of 1, 2, 4, etc., and use appropriate control signals.)

In your accumulator use the `adder_n` entity of the P10 class. Also, modify and use the `dual_port_ram` entity of the P11 class.

2 Possible block diagram for the first work item

The following figure presents a possible implementation of the first part of the work. The aux read port is synchronous or asynchronous?



To do part 3 in single-cycle mode, place the barrel shifter either between `write_inc` and b or between s and `write_data`.