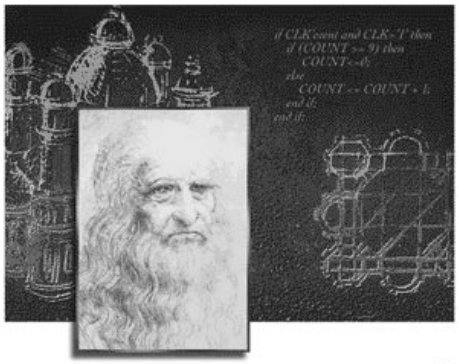
RAM Inference Using Exemplar Logic's Leonardo

By TOM HILL ◆ Manager of Vendor Relations ◆ Exemplar Logic

When using synthesis, component instantiation has been the preferred method for inserting RAM into a design. Although instantiation works, it is cumbersome and adds an unnecessary level of complexity to the HDL coding and simulation steps. Exemplar Logic's Leonardo supports RAM inference from your RTL code. This article describes the technical aspects of inferring RAM from VHDL and Verilog.

Leonardo recognizes RAM elements from a basic two-dimensional array memory model, coded in VHDL or Verilog. When a RAM is detected, Leonardo inserts a generic RAM cell into the netlist along with EDIF properties that tell the Xilinx Alliance Series software how to implement the RAM. Currently, the following two Xilinx RAM types are supported:



LEONARDO™

Advancing the World of Logic Design

RAM_DQ: *Synchronous or asynchronous single-port RAM. Leonardo determines if the RAM is synchronous or asynchronous based on the existence of clock lines.*

RAM_IO: *Synchronous or asynchronous single-port RAM with bi-directional data line.*

Asynchronous, Single-port RAM Example

VHDL

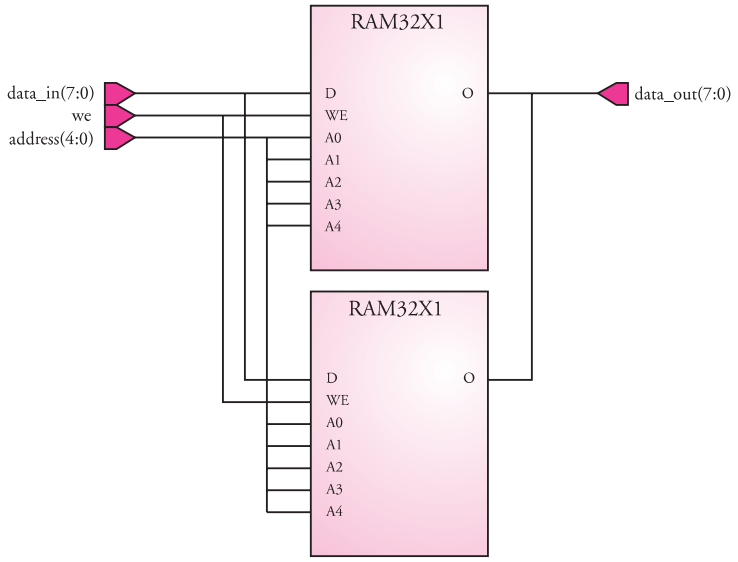
```
     architecture rtl of ram is
type mem_type is array
 (2**address_width downto 0)
UNSIGNED(data_width - 1 downto 0) ;
signal mem : mem_type ;
  begin
    I0 : process
 (we,address,mem,data_in)
     begin
       if (we = '1') then
         mem(conv_integer(address))
 <= data_in ;
       end if ;
       data_out <=
 mem(conv_integer(address)) ;
    end process ;
end RTL ;
```

Verilog

```
      module ram (data_in, address,
we, data_out);
 parameter data_width=8,
address_width=8, mem_elements=256;

 input  [data_width-1:0]    data_in;
 input  [address_width-1:0] address;
 input                      we;
 output [data_width-1:0]
data_out;

 reg    [data_width-1:0]    mem
[mem_elements:0];

 always @(we or address or data_in)
begin
   if (we) mem[address] = data_in;
 end

 assign data_out = mem[address];
endmodule
```

There is no limit to the size of the RAM that can be inferred; Leonardo will build a RAM array out of available elements for a particular technology. In the **Figure 1** example, two 32x4 RAMs are required.

Figure 1. In this example, two 32x4 RAMs are required.



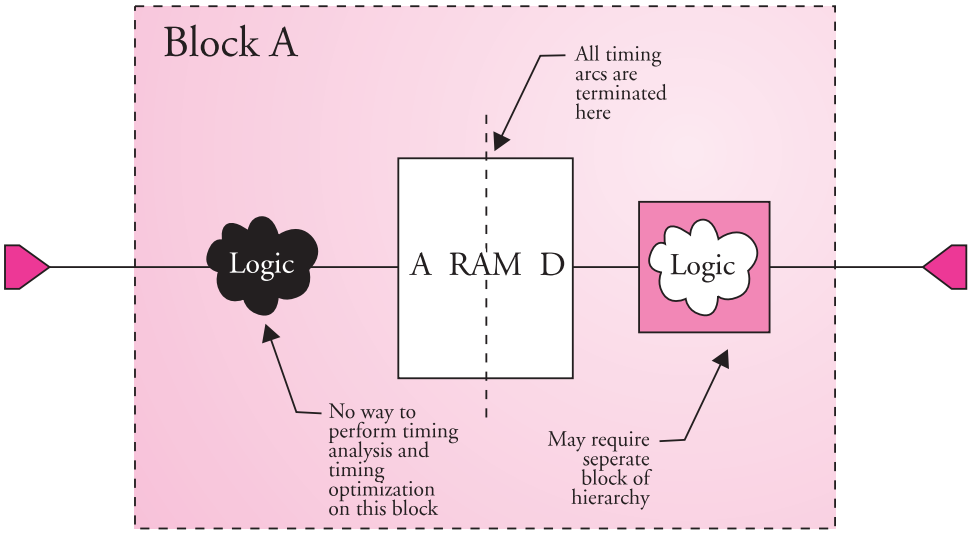
Timing Analysis and Optimization of RAM Control Logic

Because Xilinx uses a black-box approach to RAM instantiation, no timing information is available for the inferred RAM cell. This prevents Leonardo from performing timing analysis and timing optimization on logic directly connected to the RAM, which often includes the address-to-data-out path of the RAM. Therefore, you should perform a timing analysis in the Alliance Series environment to detect any timing problems through paths that contain RAM. *(See Figure2.)*

If further optimization is required on a path, use the following procedure:

1. Re-optimize the circuit with the *"-delay"* and the "*-effort standard"* options.
2. Place critical logic into a separate hierarchical block. This could be done with the *group* command or may require a code re-write. Leonardo supports the VHDL block statements for hierarchy which may provide a simple method of achieving this.
3. Set timing constraints and perform optimizations.

Figure 2. Leonardo's black-box approach.



Conclusion

If you're using RAM, Exemplar Logic's Leonardo can decrease your development time and make your life a little easier. For more information on Leonardo, contact Exemplar at 1-800-632-3742, or visit www.exemplar.com. ◆