

# VHDL

## Memórias

Rafael Corsi Ferrão - IMT



`rafael.corsi@maua.br`  
`http://www.maua.br`

5 de novembro de 2014

Conceitos

VHDL

Generalidades

**1** Conceitos

**2** VHDL

**3** Generalidades

Conceitos

VHDL

Generalidades

1 Conceitos

2 VHDL

3 Generalidades

Dois tipos distintos de memórias podem ser usados em FPGA:

- ▶ Memória interna
  - ▶ RAM
  - ▶ ROM
  - ▶ FIFO
- ▶ Memória externa
  - ▶ SRAM
  - ▶ DDR2, DDR3
  - ▶ Flash

No caso de memórias externas em alguns casos, faz-se necessário a utilização de um controlador de memória para gerenciar o acesso atualização, etc. Para o controle e acesso das memórias externas é aconselhável a utilização de IP Cores já validados e otimizados

Conceitos

**VHDL**

Generalidades

1 Conceitos

**2 VHDL**

3 Generalidades

Conceitos

VHDL

Generalidades

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity rams_04 is
    port (CLK : in std_logic;
          WE  : in std_logic;
          A   : in std_logic_vector(5 downto 0);
          DI  : in std_logic_vector(15 downto 0);
          DO  : out std_logic_vector(15 downto 0));
end rams_04;

architecture syn of rams_04 is

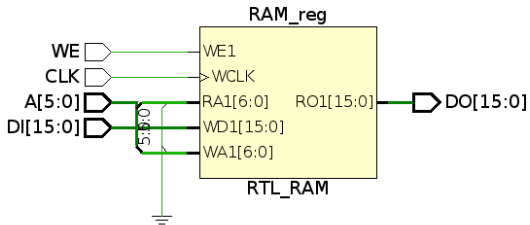
    type ram_type is array (63 downto 0) of std_logic_vector (15 downto 0);
    signal RAM : ram_type;

begin

    process (CLK)
    begin
        if (CLK'event and CLK = '1') then
            if (WE = '1') then
                RAM(conv_integer(A)) <= DI;
            end if;
        end if;
    end process;

    DO <= RAM(conv_integer(A));

end syn;
```



Conceitos

VHDL

Generalidades

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity single_port_rom is
port(
    addr : in natural range 0 to 255;
    clk   : in std_logic;
    q      : out std_logic_vector(7 downto 0)
);

end entity;

architecture rtl of single_port_rom is

    -- Build a 2-D array type for the RoM
    type memory_t is array(7 downto 0) of std_logic_vector(7 downto 0);

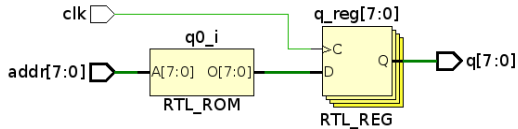
    -- Declare the ROM signal and specify a default value.
    signal rom : memory_t := (X"00", X"01", X"02", X"03", X"04", X"05", X"06", X"07");

begin

    process(clk)
    begin
        if(rising_edge(clk)) then
            q <= rom(addr);
        end if;
    end process;

end rtl;
```





single\_port\_rom

- Nets (25)
- Leaf Cells (9)
  - q0\_i (RTL\_ROM)
  - q\_reg[0] (RTL\_REG\_BREG\_)
  - q\_reg[1] (RTL\_REG\_BREG\_)

Cell Properties

q0\_i

INIT	
INIT_7	8'b00000000
INIT_6	8'b00000001
INIT_5	8'b00000010
INIT_4	8'b00000011
INIT_3	8'b00000100
INIT_2	8'b00000101
INIT_1	8'b00000110
INIT_0	8'b00000111

Conceitos

**VHDL**

Generalidades

## FIFO

Usar um IP Core para FIFO

Conceitos

VHDL

Generalidades

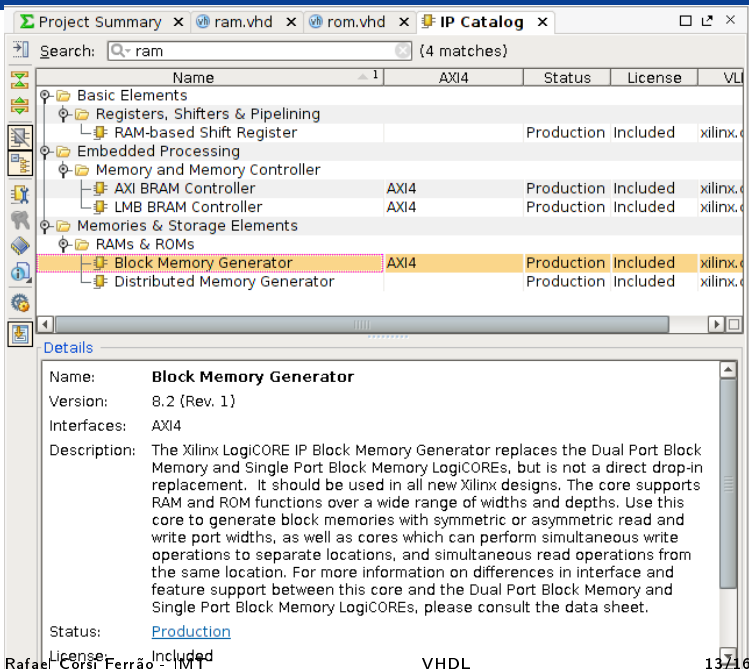
1 Conceitos

2 VHDL

3 Generalidades

Podemos usar memórias distribuídas (formadas por registradores) ou memórias em blocos (dedicadas) para sintetizar os tipos de memórias, as diferenças são:

- ▶ Distribuídas
  - ▶ Mais rápidas
  - ▶ Fazem uso de registradores para armazenar as informações
  - ▶ Maior consumo de área
- ▶ Bloco
  - ▶ Memórias dedicadas
  - ▶ Não fazem uso de registradores que poderiam ser usados em outras lógicas
  - ▶ Mais lentos que as distribuídas



The screenshot shows the Xilinx IP Catalog window with the search term "ram". The results list several IP cores, with the "Block Memory Generator" highlighted. The details for this core are shown below the list.

Name	AXI4	Status	License	VL
Basic Elements				
Registers, Shifters & Pipelining				
RAM-based Shift Register		Production	Included	xilinx.6
Embedded Processing				
Memory and Memory Controller				
AXI BRAM Controller	AXI4	Production	Included	xilinx.6
LMB BRAM Controller	AXI4	Production	Included	xilinx.6
Memories & Storage Elements				
RAMs & ROMs				
Block Memory Generator	AXI4	Production	Included	xilinx.6
Distributed Memory Generator		Production	Included	xilinx.6

**Details**

Name: **Block Memory Generator**

Version: 8.2 (Rev. 1)

Interfaces: AXI4

Description: The Xilinx LogiCORE IP Block Memory Generator replaces the Dual Port Block Memory and Single Port Block Memory LogiCOREs, but is not a direct drop-in replacement. It should be used in all new Xilinx designs. The core supports RAM and ROM functions over a wide range of widths and depths. Use this core to generate block memories with symmetric or asymmetric read and write port widths, as well as cores which can perform simultaneous write operations to separate locations, and simultaneous read operations from the same location. For more information on differences in interface and feature support between this core and the Dual Port Block Memory and Single Port Block Memory LogiCOREs, please consult the data sheet.

Status: [Production](#)

License: [Included](#)

Customize IP

## FIFO Generator (12.0)

Documentation IP Location Switch to Defaults

☒ Show disabled ports

Component Name

**Basic** Native Ports Status Flags Data Counts Summary

**Interface Type**

☒ Native ☐ AXI Memory Mapped ☐ AXI Stream

Fifo Implementation

**FIFO Implementation Options**

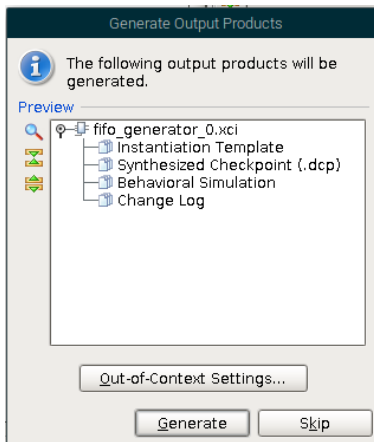
Supported Features

	Memory Type	(1)	(2)	(3)	(4)	(5)
<b>Common Clock (CLK)</b>	<b>Block RAM</b>		✓		✓	✓
Common Clock (CLK)	Distributed RAM		✓			
Common Clock (CLK)	Shift Register					
Common Clock (CLK)	Built-in FIFO		✓	✓	✓	✓
Independent Clocks (RD_CLK, WR_CLK)	Block RAM	✓			✓	✓
Independent Clocks (RD_CLK, WR_CLK)	Distributed RAM		✓			
Independent Clocks (RD_CLK, WR_CLK)	Built-in FIFO		✓	✓	✓	✓

(1) Non-symmetric aspect ratios (different read and write data widths)  
 (2) First-Word Fall-Through  
 (3) Uses Built-in FIFO primitives  
 (4) ECC support  
 (5) Dynamic Error Injection

OK Cancel

Quando gerado um IP Core pela interface do Vivado, os seguintes itens são adicionados ao projeto :



```
ENTITY fifo_generator_0 IS
  PORT (
    clk : IN STD_LOGIC;
    rst : IN STD_LOGIC;
    din : IN STD_LOGIC_VECTOR(17 DOWNT0 0);
    wr_en : IN STD_LOGIC;
    rd_en : IN STD_LOGIC;
    dout : OUT STD_LOGIC_VECTOR(17 DOWNT0 0);
    full : OUT STD_LOGIC;
    empty : OUT STD_LOGIC
  );
END fifo_generator_0;
```