

Teaching online electronics, microcontrollers and programming in Higher Education

Hardware Implementation of Algorithms

9. Block RAM memory in FPGA - example of use.

Lead Partner: Warsaw University of Technology

Author: Lukasz Mik

University of Applied Sciences in Tarnow



This laboratory instruction has been prepared in the context of the ENGINE project. Where other published and unpublished source materials have been used, these have been acknowledged.

Copyright

© Copyright 2021 - 2023 the ENGINE Consortium

Warsaw University of Technology (Poland)

International Hellenic University (IHU) (Greece)

European Lab for Educational Technology- EDUMOTIVA (Greece)

University of Padova (Italy)

University of Applied Sciences in Tarnow (Poland)

All rights reserved.



This document is licensed to the public under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.

Funding Disclaimer

This project has been funded with support from the European Commission. This report reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

I. Displaying images stored in block RAM memory of Spartan-3A on a VGA monitor screen.

The Spartan-3A chip on the Numato Elbert V2 board has a block RAM memory with a total capacity of 54 kb and a distributed memory (consisting of configurable logical blocks) with a capacity of 11 kb. In the course we will deal with the use of block memory (BRAM).

Before starting the exercise, download the source files for the project from the "VHDL Sources" folder and save them in a folder with the appropriate name.

During the classes, the VGA port will be used, the connection diagram of which with the FPGA is shown in the figure below.



It shows that it contains 3 simple DACs in the form of resistor ladders. For the colors: red and green, it is a 3-bit converter, while for the blue color it is only 2-bit. Therefore, it is necessary to convert the 8-bit values of the RGB color components (read from the BMP file) to the target values corresponding to the lengths of the bit vectors for the DACs. For the exercise, a script called *image_converter.m* was prepared in the Matlab environment. Its task is to load a BMP file, reduce the bit length of individual RGB components and save the data vector obtained in this way to a file with the *.*coe* extension.

The Matlab script generates a data vector in the form of binary strings, so in the first line of the *.coe file the number 2 is given as the base of the number system. In the next line, the

data vector begins, each value is stored in a separate line. A fragment of the *.coe file is shown below.

- MEMORY_INITIALIZATION_RADIX=2; MEMORY_INITIALIZATION_VECTOR= 01110010 01110010 01110010 01110010 01110010 01110010 01110010 01110010 01110010 01110010 .
- **STEP 1:** Creating a new project, adding source files and a memory component using the IP Core generator.

Start *ISE Design Suite 14.7*, from the *File* menu, select *New Project*. Enter the appropriate name of the project, e.g. *block_ram_vga*, and in the next window set the following target chip for the project being created:

Property Name	Value	
Evaluation Development Board	None Specified	\sim
Product Category	All	\sim
Family	Spartan3A and Spartan3AN	\sim
Device	XC3S50A	\sim
Package	TQ144	\sim
Speed	-4	\sim
Top-Level Source Type	HDL	
Synthesis Tool	XST (VHDL/Verilog)	\sim
Simulator	ISim (VHDL/Verilog)	\sim
Preferred Language	VHDL	\sim
Property Specification in Project File	Store all values	\sim
Manual Compile Order		
VHDL Source Analysis Standard	VHDL-200X	\sim
Enable Message Filtering		

After creating the project and adding the previously downloaded source files, all added files and the type of the selected chip should appear in the *Design* window:



From the *Project* menu, select *New Source*. In the window that appears, select the *IP* (*Coregen & Architecture Wizard*) module. Enter *rom_inst* as the name.

New Source Wizard Select Source Type Select source type, file name and its location.	×
BMM File ChipScope Definition and Connection File Implementation Constraints File IP (CORE Generator & Architecture Wizard) MEM File Schematic User Document Verilog Module Verilog Test Fixture VHDL Module VHDL Library VHDL Package WHDL Test Bench Embedded Processor	Eile name: rom_inst Logation: C:\Xilinx_work\block_ram_vga\ipcore_dir
More Info	<u>N</u> ext > Cancel

In the next window, expand the group *Memories & Storage Elements* \rightarrow *RAMs & ROMs* \rightarrow *Block Memory Geneerator 7.3.*

> N	lew Source Wizard								×
←se	lect IP Create Coregen or /	Architecture Wizaro	d IP Core.						
	View by Function	View by Name							
	Name	New by <u>N</u> ame		Version	AXI4	AXI4-Stream	AXI4-Lite	Status ^	
	🖶 📂 Memor 🖻 📂 RAMs 8	ry Interface Gener & ROMs	ators						
	Blo	ck Memory Gene	rator	7.3				Produ	
	🖂 🖓 Dist	tributed Memory	Generator	7.2				Produ	
	🖶 💋 Standard B	age Processing							
								~	
	<							>	
	Search IP Catalog:							Clear	
	All IP versions					Only IP com	patible with o	chosen part	
							-		
M	ore Info					< <u>B</u> ack	<u>N</u> ext >	Cance	el

After selecting the option, the memory generator window will appear, where we go to the 2nd configuration page, where we select the memory type by setting the *Single Port ROM* option.

🂐 Block Memory Generator		- 🗆 X
Documents View		
IP Symbol ₽ ×	ANDE -	
	Block Memory Generat	vilinx.com:ip:blk_mem_gen:7.3
	Port A Options $RGB = 8 bitów$	
	Memory Size	
	Write Width 8 Range: 14608 Read Width: 8	-
DINA[7:0]	Write Depth 4096 Range: 2.,9011200 Read Depth: 409	
ENA		
REGCEA> SBITERR	Operating Mode Enable	
WEA[0:0]	 Always Enal 	led
RSTA -> RDADDRECC[11:0]	Write First Clise ENA Pir	
	\bigcirc Read First 64x64 niksele = 4)96
	C No Change	
		<u>•</u>
💜 IP Symbol 💜 Power Estimation	Datasheet < Back Page 3 of 6 Next >	<u>G</u> enerate <u>C</u> ancel <u>H</u> elp

On the next, third page, enter the size of the data vector (*Memory Size* \rightarrow *Width*) and the number of these vectors (*Memory Size* \rightarrow *Depth*). This is the memory width and depth that we define. We do not change the name of the component. Check the Always Enabled option..

On page 4 of the memory configurator, we leave the window with default settings except *Memory Initialization*, where we load the file named *lake.coe* with the ROM initialization vector, created from BRAM memory cells.

🂐 Block Memory Generator		– 🗆 X
Documents View		
Image: Strate in the strat	Optional Output Registers Port A Register Port A Output of Memory Primitives Register Port A Output of Memory Core Register Port A Input of SoftECC logic Use REGCEA Pin (separate enable pin for Port A output registers) Pipeline Stages within Mux Memory Initialization Image: Logic Coperation Image: Logic Coperation </th <th>- X</th>	- X
1P Symbol 1P Symbol	Fill Remaining Memory Locations Remaining Memory Locations (Hex) Datasheet Sector Page 4 of 6 Next > Gene	rate <u>C</u> ancel <u>Help</u>

On the following memory configuration pages, leave the default settings and click the Generate button.

Depending on the performance of the computer, the process of generating ROM memory from BRAM memory cells may take from several dozen seconds to several minutes. After correct execution of this operation, an additional file named rom_inst.xco will appear in the window with the project's source files, which is an instance of the ROM memory with the image already uploaded, which will be displayed on the VGA screen later.



After generating a memory block, you can see its functional description in VHDL language. Select it, expand the *CORE Generator* branch in the *Processes* window and double-click on the *View HDL Functional Model* option.

STEP 2: Adding a digital clock manager (DCM) to the project, whose task will be to generate a clock signal with a frequency of 25 MHz.

We add a new source to the project and using the *IP Core* generator we add a DCM block by typing *dcm_ins* as its name.

> New Source Wizard	×
Select Source Type Select source type, file name and its location.	
 BMM File ChipScope Definition and Connection File Implementation Constraints File IP (CORE Generator & Architecture Wizard) MEM File Schematic User Document Verilog Module Verilog Test Fixture VHDL Module VHDL Library VHDL Package VHDL Test Bench Embedded Processor 	File name: dcm_inst Logation: C:\Xilinx_work\block_ram_vga\jpcore_dir
	Add to project
More Info	<u>N</u> ext > Cancel
View by Function View by Name	
Name	Version AXI4 AXI4-Strear ^
Embedded Processing FPGA Features and Design Clocking Clocking Clocking Wizard Spartan-3 Souther 25 Sector 24	3.6
Board Deskew with an Internal I	Deskew (DCM SP) 13.1
👷 Cascading in Series with Two D	CM_SP 13.1
🔤 💸 Clock Forwarding / Board Desk	ew (DCM_SP) 13.1
Clock Switching with Two DCM	_SPs 13.1
Single DCM_SP	13.1
<	>
Search IP Catalog:	Clear
All IP versions	Only IP compatible with chosen part

In the next two steps, we confirm the settings and select the VHDL language as the source language for the DCM block instance. Then we set the parameters of the clock signal at the input - 12 MHz and select the *CLKFX* output, uncheck the rest.

💸 Xilinx Clocking Wizard - General Setup	×
CLKIN CLKFB DCM RST PSEN PSINCDEC PSCLK	CLK90
Input Clock Frequency 12 MHz ns Val	use Shift oe: NONE ✓ Iue: 0 ≑
CLKIN Source	Feedback Source
 Extemalinternal Single Differential Divide By Value 2 V Use Duty Cycle Correction	 ○ External ○ Internal ○ None ○ Single ○ Differential Feedback Value ● 1× ○ 2×
More Info	< <u>B</u> ack <u>N</u> ext > Cancel

After pressing the Next button in this window and in the next one, the DCM block configurator will take you to the clock frequency synthesizer page. In the synthesizer window, we specify the output clock frequency of 25 MHz. It is necessary for the proper operation of the video signal generator for a monitor with a VGA port, working with a resolution of 640 x 480 pixels. These performance parameters of a VGA monitor are defined by the VESA standard.

💸 Xilinx Clocking Wizard - Clock Frequency Synthesizer

DE2 MG	ode	Fin (MHz)	Fo	ut (MHz)
Low		0.200 - 333.000	5.00	0 - 333.000
High		0.200 - 333.000	5.00	0 - 333.000
puts for Jitter Calcul	lations			
put Clock Frequen	cy: 12 MHz			
) <u>U</u> se output frequ	Jency			
25	MH ₇	0.00		
2.3				
) Use Multiply (M)	and Divide (D) value	<u>о п</u> а		
Use Multiply (M)	and Divide (D) value	s		
Use Multiply (M)	and Divide (D) value	÷		
Use Multiply (M) M 4 4	and Divide (D) value	us ≑		
Use Multiply (M) M 4 ¢ Calculate	and Divide (D) value	us ≢		
Calculate	and Divide (D) value	© <u>us</u> s	Period Jitter (unit interval)	Period Jitter (pk-to-pk ns)

W projekcie powinien się pojawić plik dcm_inst.xaw



UWAGA: VHDL code with memory usage example is included in the files attached to the exercise. In the block_ram_vga.vhd file, remove comments from a significant part of the code:

```
component rom_inst
port (
   clka : IN std_logic;
   addra : IN std_logic_vector(11 DOWNTO 0);
   douta : OUT std_logic_vector(7 DOWNTO 0));
end component;
...
ROM1 : rom_inst port map(clock,address,data);
and
component dcm_inst
port(
        CLKIN_IN : IN std_logic;
        CLKFX_OUT : OUT std_logic
        );
```

```
end component;
...
DCM1: dcm inst PORT MAP(CLKIN_IN => clk , CLKFX_OUT => clk_25);
```

After compiling the project and generating the block_ram_vga.bit configuration file, program the target system and check the effect of the system operation on the VGA monitor screen.

Tasks:

- Generate another initialization vector for the ROM from the image using the image_converter.m script. BMP files with dimensions of 64x64 pixels have been made available for practice in a separate directory.
- 2. Generate its negative on the screen next to the original image.
- 3. Redo the project so that it displays monochrome images, in which pixels have binary values of 0 or 1. Reducing the width of the data bus to 1 bit will enable displaying images with a resolution of 128 x 128 pixels.

References

• J. Majewski, P. Zbysiński – Układy FPGA w przykładach. Wydawnictwo BTC, Legionowo 2007.