

# Teaching online electronics, microcontrollers and programming in Higher Education

### Sprzętowa implementacja algorytmów

9. Wykorzystanie pamięci blokowej RAM w układzie FPGA.

Lider projektu: Politechnika Warszawska

Autor: Łukasz Mik

Akademia Nauk Stosowanych w Tarnowie

# Declaration

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 Where other published and unpublished source materials have been used, these have been acknowledged.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. Wyświetlanie obrazów zapisanych w pamięci blokowej RAM układu Spartan-3A, na ekranie monitora VGA.

Układ Spartan-3A, w który jest wyposażona płyta Numato Elbert V2 ma w swoich zasobach pamięć blokową RAM o łącznej pojemności 54 kb oraz pamięć rozproszoną (złożoną z konfigurowalnych bloków logicznych) o pojemności 11 kb. W trakcie zajęć zajmiemy się wykorzystaniem pamięci blokowej (BRAM).

Przed przystąpieniem do realizacji ćwiczenia, należy pobrać pliki źródłowe do projektu z folderu "VHDL Sources" i zapisać je w folderze o odpowiedniej nazwie.

Podczas zajęć zostanie wykorzystany port VGA, którego schemat połączeń z układem FPGA został przedstawiony na poniższym rysunku.



Widać na nim, że zawiera 3 proste przetworniki DAC w postaci drabinek rezystorów. Dla kolorów: czerwonego i zielonego jest to przetwornik 3 – bitowy, natomiast dla koloru niebieskiego tylko 2 – bitowy. W związku z tym, konieczna jest konwersja 8 – bitowych wartości składowych kolorów RGB (odczytanych z pliku BMP) na docelowe wartości, odpowiadające długościom wektorów bitowych dla przetworników DAC. Do ćwiczenia został przygotowany w środowisku Matlab skrypt o nazwie *image\_converter.m*, którego zadaniem jest wczytanie pliku BMP, redukcja długości bitów

poszczególnych składowych RGB oraz zapisanie uzyskanego w ten sposób wektora danych do pliku o rozszerzeniu \*.*coe*.

Skrypt w Matlabie generuje wektor danych w postaci ciągów binarnych, dlatego w pierwszym wierszu pliku \*.*coe* jako podstawa systemu liczbowego podana jest liczba 2. W kolejnym wierszu rozpoczyna się wektor danych, każda wartość jest zapisywana w oddzielnym wierszu. Fragment pliku \*.*coe* został przedstawiony poniżej.

<pre>MEMORY_INITIALIZATION_RADIX=2;</pre>
MEMORY_INITIALIZATION_VECTOR=
01110010
01110010
01110010
01110010
01110010
01110010
01110010
01110010
01110010
01110010
01110010

**KROK 1:** Utworzenie nowego projektu, dodanie plików źródłowych oraz komponentu pamięci z wykorzystaniem generatora *IP Core*.

Uruchom program ISE Design Suite 14.7, z menu *File* wybierz opcję *New Project*. Wpisz odpowiednią nazwę projektu np. *block\_ram\_vga*, a w kolejnym oknie ustaw następujący układ docelowy dla tworzonego projektu:

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	$\sim$
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		

Po utworzeniu projektu i dodaniu pobranych wcześniej plików źródłowych w oknie *Design* powinny pojawić się wszystkie dodane pliki oraz typ wybranego układu:

Desig	n	↔□♂×
*	View: 💿 🄯 Implementation 🔿 🎆 Simulation	
<b>a</b>	Hierarchy	
	☐ block_ram_vga ☐ xc3s50a-4tq144	
00	UGA - behavior (block_ram_vga.vhd)	
e l		

Z menu *Project* wybierz opcję *New Source*. W oknie, które się pojawi wybierz moduł IP (*Coregen & Architecture Wizard*). Jako nazwę wpisz *rom\_inst*.

> 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
<u>M</u> ore Info	Next > Cancel

W kolejnym oknie rozwiń grupę *Memories & Storage Elements -> RAMs & ROMs ->* Block Memory Geneerator 7.3.

> New Source Wizard								×
←Select IP Create Coregen or	r Architecture Wizarc	l IP Core.						
View by Euroction	View by Name							
Nama	view by <u>in</u> ame		Vanian	AVIA	AVIA Character	AVIATO	Charlos A	
in in in in in in in in	ory Interface Gener ; & ROMs	ators	version	AA14	AVI4-2016911	AAI4 LILE	Status	
BI	ock Memory Gene	rator	7.3				Produ	
⊕ 🌽 Video & I	istributed Memory Bus Interfaces mage Processing	Generator	7.2				Produ	
<							>	
Search IP Catalog:							Clear	1
All IP versions					Only IP com	patible with (	chosen part	
More Info					< <u>B</u> ack	<u>N</u> ext >	Cance	el

Po zakończeniu wyboru opcji pojawi się okno generatora pamięci, gdzie przechodzimy do 2 strony konfiguracji, na której wybieramy typ pamięci ustawiając opcję *Single Port ROM*.

🏘 Block Memory Generator					-		×
Documents View							
IP Symbol	LogiC∛	RE Block	Memory	Generator	xilinx.com:ip:blk_	mem_ge	en:7.3
	Port A	Options	RGB = 8	8 bitów			
	DUTA[7:0] Writ	te Width 8	Range: 14608	Read Width: 8			
DINA[7:0]	Writ	te Depth 4096	Range: 29011200	Read Depth: 4096			
ENA> REGCEA> SB		rating Mode		Enable			
	BITERR	-		Always Enabled			
	DADDRECC[11:0]	Write First	$\sim$	C Lice ENA Pin			
CLKA				O USE ENA FIII			
		Read First	64x64 pi	iksele = 4096			
	C 1	lo Change					
				J [			
							•
🌾 IP Symbol 🧳 Power Estimation	Datashe	et	< <u>B</u> ack Page 3	of 6 <u>N</u> ext > <u>G</u> enera	ate <u>C</u> ancel	Hel	lp

Na kolejnej, trzeciej stronie wpisujemy rozmiar wektora danych (*Memory Size -> Width*) oraz liczbę tych wektorów (*Memory Size ->Depth*). Jest to definiowana przez nas szerokość i głębokość pamięci. Nazwy komponentu nie zmieniamy. Zaznaczamy opcję *Always Enabled*.

Na 4 stronie konfiguratora pamięci pozostawiamy okno z domyślnymi ustawieniami poza pozycją *Memory Initialization*, gdzie ładujemy plik o nazwie *lake.coe* z wektorem inicjalizującym pamięć ROM, tworzoną z komórek pamięci BRAM.

🂐 Block Memory Generator		- 🗆 X
Documents View		
IP Symbol 8	Block Memory Generator	xilinx.com:ip:blk_mem_gen:7.3
	Optional Output Registers	<b>^</b>
	Port A	
	Register Port A Output of Memory Primitives	
	C 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)	
RSTA> RDADDRECC[11:0]	Pipeline Stages within Mux 0 Mux Size: 1x1	
	- Momony Initialization	
	I⊻ Load Init File	
	Coe File C:\Xilinx_work\block_ram_vga\lake.coe	Browse Show
	Fill Remaining Memory Locations	
	Remaining Memory Locations (Hex) 0	
🍕 IP Symbol 📢 Power Estimation		

Na kolejnych stronach konfiguracji pamięci pozostawiamy ustawienia domyślne i klikamy przycisk *Generate*.

W zależności od wydajności komputera proces generacji pamięci ROM z komórek pamięci BRAM może zajęć od kilkudziesięciu sekund do kilku minut. Po prawidłowym wykonaniu tej operacji w oknie z plikami źródłowymi projektu pojawi się dodatkowy plik o nazwie *rom\_inst.xco*, który jest instancją pamięci ROM z wgranym już obrazkiem, który w dalszej części zajęć będzie wyświetlany na ekranie monitora VGA.



Po wygenerowaniu bloku pamięci można zobaczyć jego funkcjonalny opis w języku VHDL należy go zaznaczyć, w oknie *Processes* rozwinąć gałąź *CORE Generator* i dwukrotnie kliknąć na opcję *View HDL Functional Model*.

KROK 2: Dodanie do projektu cyfrowego menedżera sygnałów zegarowych (DCM), którego zadaniem będzie wygenerowanie sygnału zegarowego o częstotliwości 25 MHz.

Dodajemy nowe źródło do projektu i przy użyciu generatora *IP Core* dodajemy blok DCM wpisując *dcm\_inst* jako jego nazwę.

New Source Wizard					×
Select Source Type Select source type, file name and its location.					
<ul> <li>BMM File</li> <li>ChipScope Definition and Connection File</li> <li>Implementation Constraints File</li> <li>IP (CORE Generator &amp; Architecture Wizard)</li> <li>MEM File</li> <li>Schematic</li> <li>User Document</li> <li>Verilog Module</li> <li>Verilog Test Fixture</li> <li>VHDL Module</li> <li>VHDL Library</li> <li>VHDL Package</li> <li>VHDL Test Bench</li> <li>Embedded Processor</li> </ul>	Eile name: dcm_inst Logation: C:\Xilinx_work\bloc	k_ram_vga	\jpcore_c	dir	]
	Add to project				
More Info			<u>N</u> ext >	Can	cel
View by Function View by Name					
Name      Embedded Processing     FPGA Features and Design	<b>A</b>	Version	AXI4	AXI4-Strear	^
Clocking Wizard  Grow Spartan-3  Spartan-3		3.6			
Board Deskew with an Internal     Cascading in Series with Two I	Deskew (DCM_SP) DCM_SP	13.1 13.1			
Clock Forwarding / Board Des	A_SPs	13.1 13.1 13.1			
kiner in the second se				>	*
Search IP Catalog:				Clear	
All IP versions		Only IP com	patible v	with chosen p	art

W kolejnym dwóch krokach potwierdzamy ustawienia i wybieramy język VHDL jako źródłowy dla instancji bloku DCM. Następnie ustalamy parametry sygnału zegarowego na wejściu – 12 MHz i zaznaczamy wyjście CLKFX, pozostałe odznaczamy.

💸 Xilinx Clocking Wizard - General Setup	×
CLKIN CLKFB DCM RST PSINCDEC PSCLK	CLK90
Input Clock Frequency 12    MHz O ns Ty Va	ase Shift pe: NONE
CLKIN Source	Feedback Source
<ul> <li>Extemalinternal</li> <li>Single</li> <li>Differential</li> </ul> Divide By Value 2 V Use Duty Cycle Correction	<ul> <li>○ External</li> <li>○ Internal</li> <li>○ None</li> <li>○ Single</li> <li>○ Differential</li> </ul> Feedback Value ● 1X ○ 2X
More Info	< <u>Back</u> <u>N</u> ext > Cancel

Po naciśnięciu przycisku *Next* w tym oknie i następnym konfigurator bloku DCM przeniesie nas na stronę syntezera częstotliwości zegara. W oknie syntezera podajemy wyjściową częstotliwość sygnału zegarowego równą 25 MHz. Jest ona niezbędna do prawidłowej pracy generatora sygnału wideo dla monitora z portem VGA, pracującego z

rozdzielczością 640 x 480 pikseli. Takie parametry pracy monitora VGA są zdefiniowane przez standard VESA.

Xilinx Clocking Wizar	d - Clock Frequen	icy Synthesizer		
Valid Ranges for Speed G	rade -4			
DFS Mode	[	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
Inputs for Jitter Calculation	IS			
Input Clock Frequency:	12 MHz			
<u>U</u> se output frequency	/			
25	● M <u>H</u> z (	) <u>n</u> s		
O Use Multiply (M) and	Divide (D) values			
M 4	D 1 🛔	]		
Calculate				
Generated Output				
M	D	Output Freq (MHz)	Period Jitter (unit interval)	Period Jitter (pk-to-pk ns)
25	12	25	0.05	2.08

W projekcie powinien się pojawić plik dcm inst.xaw



UWAGA: Kod w języku VHDL z przykładem wykorzystania pamięci jest zawarty w plikach załączonych do ćwiczenia. W pliku *block\_ram\_vga.vhd* należy usunąć komentarze z istotnej części kodu:

```
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);
```

```
oraz
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);
```

Po skompilowaniu projektu i wygenerowaniu pliku konfiguracyjnego *block\_ram\_vga.bit* należy zaprogramować układ docelowy i sprawdzić efekt działania systemu na ekranie monitora VGA.

#### Zadania:

- Wygenerować inny wektor inicjalizujący dla pamięci ROM z obrazka przy użyciu skryptu *image\_converter.m.* Pliki BMP o rozmiarach 64x64 piksele zostały udostępnione do ćwiczenia w osobnym katalogu.
- 2. Na ekranie obok obrazka oryginalnego wygenerować jego negatyw.
- Przerobić projekt w ten sposób, aby wyświetlał obrazki monochromatyczne, w których piksele mają wartości binarne 0 lub 1. Redukcja szerokości magistrali danych do 1 bitu umożliwi wyświetlanie obrazków o rozdzielczości 128 x 128 pikseli.

## References

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