FPGA Programming

Class 5: Programming the Chip

September 15, 2017 Louis W. Giokas





This Week's Agenda

Monday Tuesday Wednesday Thursday Friday

FPGA Device Description Design Flow HDL Synthesis and Layout Programming the Chip



Presented by:



ROHDE&SCHWARZ

Course Description

We start with an introduction to the class of devices called Field Programmable Gate Arrays (FPGAs). The layout and design of several types and critical parameters will be described and discussed. It is important to understand the way the device is constructed to develop effective algorithms.

The device we will be using this week will be the Microsemi IGLOO2. We will also discuss other devices and their structure.

We will introduce two common Hardware Description Languages (HDL), but give examples in one (Verilog).

3





Today's Agenda

- Overview
- Configuration
- Generation
- Programming the Device
- Debugging
- Production
- Conclusion
- Contact/Resources





Overview

- We now come the topic of programming in the way that FPGA vendors consider it
 - Basically this consists of creating the bitmap and loading it onto the chip
- Seems simple, but there is a lot we need to do, and can do, at this stage
 - Debugging on the actual device
 - Monitor operation in real time





Overview

- Development tools also allow us to prepare everything necessary to program chips in a production environment
 - This includes more than the bitstream
 - Also includes the jobs and reports used to apply that bitstream and load data onto devices in bulk
 - We also generate data used for independent debugging





Overview

🖻 🕨 IO Analyzer	
💭 🛱 SSN Analyzer	
🖻 🕨 Program and Debug Design 🛛 🧹	We are
• Generate FPGA Array Data	the pro
Update eNVM Memory Content	the pro
🖻 🕨 Configure Hardware 🥿	
Programming Connectivity and Interface	
- 🔊 Configure Programmer	
🔚 Device I/O States During Programming - JTAG Mode O	There ar
Configure Programming Options	o otiviiti o o
🛛 🚱 Configure Security	activities
🖻 🕨 Program Design	at this st
- 🐻 Generate Bitstream	
Run PROGRAM Action	hardware
🖻 🕨 Debug Design 🧹	
🔍 Identify Debug Design	
🖙 🕸 SmartDebug Design	
Handoff Design for Production	
Export Bitstream	
Export FlashPro Express Job	
🛛 🛃 Export Job Manager Data	Tasks f
Export Pin Report	
Export BSDL	
Export IBIS Model	
Handoff Design for Debugging	
Export SmartDebug Data	



We are at the last part of the process.

There are a number of activities that are performed at this stage for/on actual hardware.

Tasks for production

Presented by:





7

Generation

- There are a number of steps in the generation process that allow us to control the environment and the image on the device
 - Configuration of the design (version, etc.)
 - Security
 - I/O states
- We can set parameters for these separate from the algorithms we are implementing







Generation

ad f	from file Save to file.			Show BSR Detai
	Port Name	Macro Cell	Pin Number	/O State (Output Only
1	SW1	ADLIB:INBUF	H12	Z
2	SW2	ADLIB:INBUF	H13	Z
3	clk	ADLIB:INBUF	H16	Z
1	green_led1	ADLIB:OUTBUF	J16	Z
5	green_led2	ADLIB:TRIBUFF	M16	Z
5	red_led1	ADLIB:TRIBUFF	K16	Z
7	red_led2	ADLIB:OUTBUF	N16	Z
3	UNUSED	UNUSED	T11	Z
)	UNUSED	UNUSED	N10	Z
0	UNUSED	UNUSED	P10	Z
1	UNUSED	UNUSED	M10	Z

One step is to configure the I/O. Here we are configuring pins in JTAG mode (for debugging)





Generation

Configure Programming Options	\times
Design name: LedBlinkingDSpeed Design version (number between 0 and 65535): Silicon signature (max length is 8 HEX chars): 0x Programming recovery settings: Enable Auto Update Enable Programming Recovery	
SPI clock frequency: MHz SPI data transfer mode SPS: SPO: SPH: SPH: SPH: SPH: SPH: SPH: SPH: SPH	
Help OK Cancel	

Allows us to have these different versions of the design and to determine a number of operation functions

Security Policy Manager	×
Security key mode	Security policies
Bitstream encryption with default key	Update Policy
C Enable custom security options	Debug Policy
	Key Mode Policy
User keys and Security policies protection	Microsemi factory test mode access level
C Write-protect using FlashLock/UPK1	C Allow factory test mode access
	C Protect factory test mode access using FlashLock/UPK1
C Permanently write-protect	C Permanently protect factory test mode access



10



Programming the Device

 At this point tools are provided that will finally put the bitstream onto the device

11

- For development we need to use hardware that will load the bitstream onto the chip

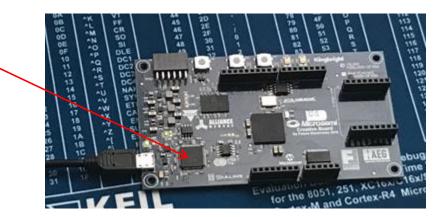
 This is not included in the chip itself
- We typically use a JTAG device which interfaces with the In-application programming functional block

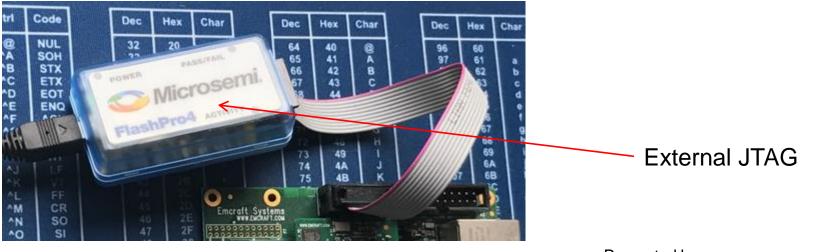




Programming the Device

Current style JTAG Is a chip on the board







12

Presented by:



SCHWARZ

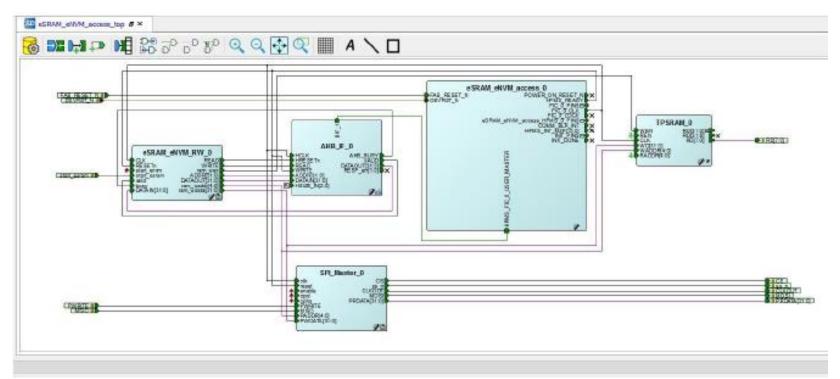
- We can debug the design right on the chip
- Much like regular software debugging we can set "breakpoints" called probes and inspect memory
 - This allows us to see the design in operation and to verify that we the design is operating correctly

13

If we find problems then we can make the necessary changes and regenerate







We will be looking at this design which manipulates memory blocks





Live Probes	Active Probes	Memory Blocks	Probe I	nsertion	1		
+ - +	•	Save		Load	.][Delete	Delete All
Name	·		Туре	1	Read V	alue	Write Value
MOSI 1:5	PI Master 0/MOSI	[_1:Q	DF	F	0		
parameter and a second second	er_0/PWDATA_buf[.	yrcarpote a do carpote a provincia de la provi	DF	F	40'h60/	182C31F	40'h

Here we have indicated specific elements we want to monitor. As we step the chip through its cycle we will see values changing on those elements.



15



emory Blocks Selection	e ×	FP	GA Arra	ny debu	ig data															
Aemory Blocks:	Select	1	Live Pr	obes	Acti	ive Pro	bes	Mer	погу В	locks	Pn	obe In	sertion	й Г.						
TPSRAM_0/eSRAM_eNV	/M_access_top_1		Current Data br			ck: T	(PSRA	M_0/e	SRAM,	_eNVM	l_acce	ss_top	TPSR	AM_0	TPSR	AM_R	C0/IN	ST_RA	M1K18	_P
			0000	000	0C3	0B2	0A1	001	0C3	082	0A1	002	0C3	082	0A1	003	0C3	082	0A1	*
			0010	004	0C3	082	0A1	005	0C3	082	0A1	006	0C3	082	0A1	007	0C3	082	0A1	
			0020	008	0C3	062	0A1	009	0C3	082	0A1	00A	0C3	082	0A1	008	0C3	082	0A1	
			0030	00C	0C3	082	0A1	00D	0C3	082	0A1	00E	0C3	082	0A1	00F	0C3	082	0A1	
			0040	010	0C3	062	0A1	011	0C3	082	0A1	01Z	0C3	082	0A1	013	0C3	OBZ	0A1	
			0050	014	0C3	0B2	0A1	015	0C3	082	0A1	016	0C3	082	0A1	017	0C3	082	0A1	
			0060	018	0C3	082	0A1	019	0C3	082	0A1	01A	0C3	082	0A1	018	0C3	082	0A1	
			0070	01C	0C3	0B2	0A1	01D	0C3	082	0A1	01E	0C3	082	0A1	01F	0C3	082	0A1	
			0080	068	011	180	128	0CD	09F	188	020	116	0FC	007	0AE	000	08E	0A0	086	
			anon	002	.004	107	040	074	-	-	-		000		NC5	01E	014	1.10	147	-
								1	Rea	d Bloc	k)	Write	e Block							
[]	Þ	1						(Rea	d Bloc	*)	Write	e Block							-

We can inspect memory blocks in detail with the operation halted



16



- Once a design has been developed, simulated, debugged and verified it is ready for production
- Various production setups are available, and we will not go into the hardware here
- Suffice it say that we will not be programming large quantities of these units using a JTAG device attached to our laptops

17





- The basic task is to package the bitstream for handoff to the production system
- This includes more that the bitstream
 - A job file for controlling the process of programming the device
 - FlashPro Express
 - Standard



18



		Existing bitstream files:	
ocation: 02\Lab2_ver\designer\LedBlinkingDS	peed\export	<no files="" found="" programming=""></no>	
Formats:			
STAPL Support for ISP			
Chain STAPL Support for ISP, Single Micro	osemi device in a JTAG chain		
DAT Support for Embedded ISP (J	TAG and SPI-Slave)		
SPI Support for Auto Programmir Programming Recovery and I			
SVF Support for ISP			
Limit SVF file size			
		1	
		Jabled	
elected Security options (modify via Configure Encrypt bitstream with default key. No User		nabled.	
Encrypt bitstream with default key. No User	r keys and Security Policies are e	nabled.	
Encrypt bitstream with default key. No User	r keys and Security Policies are en Bitstream components	nabled.	
elected Security options (modify via Configure : Encrypt bitstream with default key. No User itstream files to be exported File to program at trusted facility	r keys and Security Policies are e	nabled.	

DesignNews CEC CENTER

19

Presented by:



Options for

bitstream export

- Reports
 - Pin Report: assignment of all pins on the device
 - BSDL (Boundary Scan Description Language)
 - Used with VHDL
 - IBIS (Input Output Buffer Information Specification)
 - Allows one to hide the IP while providing information to users to understand the I/O required



Presented by:



ROHDE&SCHWARZ

Conclusion

- We have looked at the tools and processes available to create a bitstream and load it onto the chip
- This finishes the processes required to go from a design description to an operating FPGA
- We have touched on some of the debugging capabilities

21

• Finally we have discussed the process of getting the design out for production





Contacts/Resources

- I can be contacted several ways:
 - On Design News (naperlou, or search for Louis Giokas)
 - Email: <u>l.giokas@ieee.org</u>
 - Twitter: @naperlou for me or #DNCEC
 - Linkedin

Resources

- Vendor: websites have copious amounts of information available to prospective customers
- Verilog Courses:
 - https://www.altera.com/support/training/course/ohdl1120.html ٠
 - https://rtlacademy.com/verilog/overview
 - https://electronics.stackexchange.com/questions/26193/learning-verilog-online
 - http://www.multisoftvirtualacademy.com/embedded-systems/verilog-onlinetraining



22

