Reconfigurable Architecture (6)

osana@eee.u-ryukyu.ac.jp

Previously in this class...

- * Verilog-HDL syntax and design
 - Circuit description, Module hiearchy, Testbench, parameter... *
 - Design flow for simulation and implementation

Productivity in Software Design

- * Design reuse
 - * Once written, use again and again: write a clean code!
- Share codes if everyone want it
 - Libraries: printf(), STL, BLAS, FFTW, OpenCV, ... *



Examples of Libraries in Software

- * Mathematics: fast, optimized implementations of frequently-used algorithms
 - BLAS (Basic Linear Algebra Subprograms)
 - FFTW (Fastest Fourier Transform in the West)
- Image processing: read/write various image formats, or do transformations
 - * OpenJPEG, libPNG, ImageMagick, OpenCV, ...



Same in Hardware Design

- * Example of complicated stuff: Floating-point arithmetics
 - I bit sign x 52bit fraction x 2(11bit exponent)
 - * Fraction is always I.XXXX (must be adjusted with exponent)
 - Exponent biased by 1023 (exponent=1023 means x2⁰)
- Fraction and exponent must be calculated separately and together :(
 - * Never want to implement FP operators again and again...



Software vs Hardware

- Connecting software components are simple
 - * Argument, function call and return value
- * In hardware, procedure and timing is crucial
 - Protocol in time-space is important
 - Verification is difficult and time consuming



Design easier with IP cores

- * IP (Intellectual Property) cores: functional blocks
 - Blackbox modules (sometimes source code is provided, but usually not) *
 - Simulation with IP cores is possible
 - **Interface specification** is important and well documented *
 - * Many IP core comes with user-configurable parameters



Hard IP vs Soft IP

- * Soft IP: Constructed on FPGA's logic cells, as our HDL is mapped
 - * Flexible in number, location and functionality
 - * May be a pressure on logic utilization
- * Hard IP: "Ready-made" function block on FPGA chip
 - * Memory blocks, DSP arithmetics, Ethernet MAC, ARM processor, PCIe, ...
 - Less flexible and number is limited, but very fast



Where to IP cores

- In IP management tools of design suites
 - * Vivado has IP Integrator: supports both free and charged IP cores
- Downloadable RTLs: OpenCores.org and many other per-project sites
- Downloadable RTL generator: FloPoCo (Floating point tools) and similar tools *





IP cores: pros and cons

- Pros: Designs made easier
 - Usually faster and/or smaller than hand-written HDL designs
- Cons: Designs made harder
 - * IP core behavior must be considered in HDL-written modules
 - May be a problem when migrating to ASICs or other FPGAs *

IP cores vs Designers

- * Things to know
 - Interface, or module ports: How each port signals acts
 - * Many IP cores have some standard compliant interface ports
 - * Parameters: what is configurable in the IP core

Interface standards

- Connect IP cores directly, like drawing block diagram
 - * Requires common interface \rightarrow standards are defined
 - AXI4 (Xilinx / ARM), Avalon (Intel), Wishbone (OpenCores)
 - Block-diagram based design tools
 - IP Integrator (Xilinx), Qsys (Inte *



What is Interface standard?

- * How to connect sender and receiver
 - * "Are you ready?" → "Yes / Wait!"
 - * "Here data is" → "OK / Wait!"
 - * "Over"



Well-used interface types

- * Memory type
 - * Access with address to read/write
 - Random access order is possible for both reads and writes
- Stream type
 - Data transfer in single direction *
 - Read in written order only



Block RAM interface CLK WEN 2 3 **ADDR** 13 DIN 12 X DOUT

- Write data: same time with address
- * Read data: I clock later than address



Simple stream transfer

- Both synchronous to same clock
- Data is free-running
 - * with validity signal





Sender/receiver handshake

- Check status each other
 - * VALID + READY
 - Next data sent when both are ready to send/receive





Interface variations

- * Both memory type and stream type has many options
 - * AXI4 has both types and many options
 - * Can be connected together, though
 - * Even memory $\leftarrow \rightarrow$ stream connection is possible with adapter IPs



Typical IP cores for FPGAs

- * On-chip memory: BlockRAM or FIFOs
- Floating-Point operators
- PCI Express, Ethernet and many other standard interfaces *

* Clock managers: to generate clock signals with different frequency or phase

Today's exercise

- Using clock manager (CMT: Clock Management Tile)
 - LED flashing at 100MHz and 150MHz
- BlockRAM
 - Stopwatch version 2 with lap time feature



CMT (6 available on Artix-7 100)

- * PLL (Phase Locked Loop) + DLL (Delay Locked Loop) * PLL uses VCO (voltage controlled oscillator), DLL uses delay lines instead
- ***** Features:
 - Phase shift of 90°, 180° and 270°
 - Frequency generation by double, or with multipliers and dividers
- Multiply by n / divide by m of input frequency

Block RAM (135 available on Artix-7 100)

- * 36kb dual port memory blocks, distributed all over the FPGA chip
 - * Can be accessed individually or in parallel, provides a great bandwidth
 - * Configurable width x depth of Ix32k, 2x16k, 4x8k, 9x4k, 18x2k, 36x1k
- Also available as 2 independent 18kb RAM blocks
- * Or even as a large RAM by concatenating

Exercise I: LED flashing with CMT





Exercise I HOWTO

- Design sources, constraints, testbench are in ip-lab/lab l/src
 - But there's no CMT
 - Open IP Catalog after all sources are set in project





IP Catalog

- List of available IP cores
 - Launch Clocking Wizard to use CMT
 - ★ FPGA Features & Design → Clocking → Clocking Wizard

\Sigma Project Summary 🗙 🐓 IP Catalog 🗙 🖹 Search: 🔍 X **1** AXI4 Status | License | VLNV Name Alliance Partners 🔄 🗁 Automotive & Industrial 🔍 📴 🕸 🔍 🔍 <u>_</u> Basic Elements 🍽 🗁 🗁 🔍 🏟 🗁 Communication & Networking 💁 📴 Debug & Verification 🖗 🗁 Digital Signal Processing 🧇 🔄 Embedded Processing O, 🜵 🗁 FPGA Features and Design 🖗 🗁 Clocking 6 🔰 🖵 📴 Clocking Wizard AXI4 Production Included xilinx.com. 🏟 📴 IO Interfaces Ł 🛛 🏟 🗁 Soft Error Mitigation 🗣 🗁 System Management 🍳 🗁 XADC 💁 🗁 Math Functions 🐢 🗁 Memories & Storage Elements 💁 🗁 Standard Bus Interfaces 💁 🗁 Video & Image Processing Details **Clocking Wizard** Name: 5.1 (Rev. 4) Version: Interfaces: AXI4 Description: The Clocking Wizard creates an HDL file (Verilog or VHDL) that contains a clocking circuit custom Status: <u>Production</u> Liconco[.] Included



Clocking Wizard (1/2)

- * Set the module name
 - Default isn't good to avoid unexpected overwrite...!
- Set the input frequency (100MHz)



00

	X Customize	IP	
ard (5.1)			
📄 IP Location 🗔 Switch to	o Defaults		
Resource	 Component Name [cmt_100_150 	*****	
ports	Clocking Options Output Cloc	ks MMCM Settings Port Renaming	Summary
	Brimiting		
D			
D			
D	C MMCM OT LL		
	Clasking Factures	littor Opti	mization
	Clocking Features	jitter Opti	mization
	Frequency Synthesis	Minimize Power	Poloncod
CLKFB_OUT_D-			balanceu
psdane -	🗹 Phase Alignment 📃 :	Spread Spectrum 🛛 🔾 I	Minimize Output Jitter
clk_out1 -		0.1	
clk_aut3 =	🗌 Dynamic Reconfig 📃	Dynamic Phase Shift 🛛 🔍 🛛	Maximize input jitter filtering
cik_aut4 =			
clk_out6 -	Safe Clock Startup		
clk_out7 -			
input_clk_stapped =	Dynamic Reconfig Interface Options		
clk1b_stapped =			
lacked - cddcdane -	© AXI4Lite O DRP	Phase Duty Cycle Config	
	Input Clock Information		
	Input Clock Input Fr	reguency(MHz)	Jitter Options Input litter
	Primary 100.000	10000 - 800.000	UI v 0.010
		60.277 130.755	
	LI Secondary 100.000	60.377 - 120.755	0.010
►			

ОК

Cancel



Clocking Wizard (2/2)

- * Set output frequency
 - "Requested" and "Actual"
 are sometimes different
 - Because multiplier / divider
 has some restriction
 - * ex) Try 123.0MHz



X Customize IP

Clocking Wizard (5.1)



bol Re:	source	Component	Name 🛛	:mt_100_150					8
abled poi	ts	Clockin	g Option:	S Output Clocks	MMCM Se	ttings Port Renaming	Summa	iry	
ax i_lite		The phase	e is calcul	ated relative to the	e active input	t clock.			^
LK_IN1_D				Output Freq (MH	lz)	Phase (d	egrees)		Duty Cy
LK_IN2_D LKFB_IN_D		Output C	lock	Requested	Actual	Requeste	ed 🦾	Actual	Reques
jdip Ljacik		Clk_ou	ıt 1	150.000	8 150.00	0 0.000	8	0.000	50.000
ijaresetn in 1		Clk_ou	ıt2	100.000	N/A	0.000		N/A	50.000
in 2 in_sel	CLKF8_OUT_D4	CIK_OU	it3	100.000	N/A	0.000		N/A	50.000
o_in	psdane = clk_aut1 =	CIK_OU	it4	100.000	N/A	0.000		N/A	50.000
r I cdec	clk_aut2 = clk_aut3 =	CIK_OU	it5	100.000	N/A	0.000		N/A	50.000
nutl_ce nutl_ch	clk_aut4 = clk_aut5 =	CIK_OU	it6	100.000	N/A	0.000		N/A	50.000
out2_ce out2_ch	clk_out7 = clklo_out =	CIK_OU	it7	100.000	N/A	0.000		N/A	50.000
out3_ce out3_ch	input_clk_stapped = clk1b_stapped =					Clocking Feedback	;		
aut4_ch	lacked -		LUCK SE			Source			Signalin
out5_ce out5_ch		Out	nut Cloc	k Sequence N	umher	Dource			Jignam
out6_ce			put ciot			Auto	omatic Co	ntrol On-Chip	¢
sul7_ce			k_out1	1		O Auto	motic Co	ntrol Off, Chip	C
aut7_ch		cl	k_out2	1		U Auto	innanie eu	na or on-emp	
Ln		cl	k_out3	1		🔾 User	r-Control	led On-Chip	
er_dawn		c c	k_out4	1		0.000	Cantural	lad Off Chin	
per		c	k_out5	1		U User	-Control	ieu ori-chip	
	Þ	-							

ОК



Running simulation

- * DLL locks at 2.7us
 - * LOCKED goes high
 - This must be added to
 RST condition
 - Check source code for detail



est_behav.wcfg	×							×قا
					2,727.50) <mark>ns</mark>		
	Value		12 500 ps	12 600 ps	12 700 ns	12 800 ps	17 900 ns	13 000 ns
					2,~~ 13			
	0	IUUUUUU						
	0	<u> </u>	V	<u> </u>			V.W.W.V	
	1							
	0							
	о О							
	ů O							
	0							
	0							
	0							
	1							
	0							
	0							
	0							
	0				<u> </u>			
	0				<u> </u>			
	0				<u> </u>			
	0							
	10.0				10.0			
	1							
	0	UUUUUUUU	UUUUUUUUUUUUUUU		INNTN NUNNNNNNNN	սոհորորորորորորոր	JULUUUUUUUUUUUU	Thomananana
Þ	A D							•



To instantiate generate core...

- * The module's port definition is necessary
 - Opening the generator GUI is not very effective
 - Find the core in "IP sources", then there's "Implementation template"
 - .veo is the template for Verilog





Where's generated cores?

- * project_l/project_l.srcs/sources_l/ip/cmt_l00_l50/
 - Enough good for exercise or one-time test project
 - * Not good when migrating a test to large, complete project
 - * For large projects, managing IP cores separately is good



IP Location: manage cores outside

- * "Manage IP" in Vivado startup screen
 - Make a IP Location to manage generated IP cores outside an RTL project





Looks very similar to RTL project

	X New IP Location
Manage IP Settings Set options for cr	eating and generating IP.
Part: Target language: Target simulator: Simulator language:	xc7a100tcsg324-1 Verilog Vivado Simulator Mixed
IP location:	/home/osana/work/reconf-class/1110/core
	< <u>B</u> ack <u>N</u> ext > <u>Finish</u> Cancel

To use the core, add generated ".xci" or ".xcix" files to RTL project

	📉 Manage IP - [/home/osana/wo	ork/reconf-class/1110/core] - Vivado 2014.3.1	
<u>F</u> ile <u>E</u> dit <u>T</u> ools <u>W</u> indow	La <u>v</u> out ⊻iew <u>H</u> elp	Q- Search) commands
🧦 😁 🕼 🖛 🐂 📉 🗙	🇠 🐝 🧐 🔚 Default Layout		
Project Manager – xc7a100)tcsg324–1		×
Sources	_ 🗆 🖻 ×	🞐 IP Catalog 🗙	□ @ ×
🔍 🔀 😂 📾 🔂 🛃		∑earch: Q-	
IP Sources		Name Alliance Partners Automotive & Industrial AXI Infrastructure BaseIP Basic Elements Communication & Networking Debug & Verification Digital Signal Processing Details Select an IP to see details	▲ 1 AX 4 ▲ (4) ▲ (4)
Design Runs			_ 🗆 🖻 ×
🔍 🔄 Name	Constraints WNS	TNS WHS THS TPWS Failed Routes	LUT FF BR
Tcl Console	ages Pesign Runs		



Exercise 2: Using Block RAMs

* Make an IP location

- * lab2/core
- Generate a Block RAM (32bit x 1024 · True dual port)



Block Memory Generator

- * Basic Elements →
 Memory Elements
 - Block and distributed memory
 - * Hard IP (Block) or LUTs (distributed)
 - * Block Memory for today

🛑 😑 🛑 📉 Ma	nage IP - [/hom	e/osana/w	ork/red	conf-class	/1110/co	re] - Vivad	lo 2014.3.1			
<u>F</u> ile <u>E</u> dit <u>T</u> ools <u>W</u> indow La <u>v</u> out	<u>V</u> iew <u>H</u> elp						🔍 - Sear	rch command	ls .	
🧦 🖻 🕼 🖉 🐂 🐘 🗙 🍪 🐝	🇔 😬 Default L	.ayout	-	XXX	ي ا					
Project Manager – xc7a100tcsg324	-1									×
Sources	_		9	IP Catalog	×					Ŀ ×
🔍 🛣 🖨 📷 🔂 🛃			₽	Search:	2-					
●@ IP (1)						Name		<u>1</u>	AXI4	
			(🔍 🗁 🖉	ice Partne	ers notustrial				
				🗢 🗁 Autor 🔍 🍋 AXI Ir	nouve & i ifrastructi	ure				
				🗢 🗁 Basel	Р					Ξ
				🌳 🗁 Basic	Elements	; rc				_
			35		unters	15				
				— 🗜 DS	P48 Mac	ro				
				º- 🗁 Me	emory Ele Block Me	ments mory Gene	rator	۵۷۱4		
					Distribut	ed Memory	/ Generator	~~~		
				🖢 🗁 Re	gisters, S	hifters & Pi	pelining			
			E		_					▶
				Jetalis						
				Name:	Bloc	k Memory	Generator			
				Version:	8.2 (Rev. 2)				
				Interfaces	: AXI4					-
IP Sources						_				
Design Runs									_ 🗆	e ×
🔍 Name	Constraints	WNS	TINS	WHS	THS	TPWS	Failed Routes	LUT	FF	BRAN
P-COut-of-Context Module R	hik mem de							0.00	0.00	0
	bik_mem_ge							0.00	0.00	
•										
₩										
🔜 I CI Console 💭 Messages 🗐	Design Runs									





Component Name: ram36x1024

Memory Type:
 True Dual Port RAM

Block Memory Generator (8.2)	
鎻 Documentation 🛅 IP Location 🧔 Switch t	o Defaults
Documentation in Proceeding Switch to Switch to Power Estimation Show disabled ports	Component Name ram36x1024 Basic Port A Options Port B Options Other Options Summary Interface Type Native □ Generate address interface with 32 bits Memory Type True Dual Port RAM □ Common Clock ECC Options No ECC ▼ □ Error Injection Pins Single Bit Error Injection
Image: Comparise and the second and	Write Enable Byte Write Enable Byte Size (bits) Algorithm Options Defines the algorithm used to concatenate the block RAM primitives. Refer datasheet for more information. Algorithm Minimum Area Primitive 8kx2



ок

Cancel

Port A Options

- Read / Write Width: 32
- Write Depth: 1024

	X Customize IP
Block Memory Generator (8.2)	
<i>肖</i> Documentation 📄 IP Location 🧔 Switch	to Defaults
IP Symbol Power Estimation	Component Name ram36x1024
Show disabled ports	Basic Port A Options Port B Options Other Options Summary
	Memory Size
	Write Width32Range: 1 to 4608 (bits)Read Width32Range: 2 to 262144Write Depth1024Range: 2 to 262144Read Depth1024
	Operating Mode Write First ▼ Enable Port Type Use ENA Pin ▼ Port A Optional Output Registers Primitives Output Register □ Core Output Register
= s_acle = s_areseth = s_axLinectspherr	SoftECC Input Register REGCEA Pin
	Port A Output Reset Options
	RSTA Pin (set/reset pin) Output Reset Value (Hex)
	Reset Memory Latch Reset Priority CE (Latch or Register Enable)
	OK Cancel



Port B Options

* OK if Read/Write are 32bit





Assignment #2

- 12/3: deadline for stopwatch *
 - Start/Stop and Reset button
- 12/17: deadline for stopwatch 2.0 *
 - Start/Stop, Reset and Lap
 - * Up to 1000 laps, any usage / functions of buttons and switches are OK





書き込みは ADDR + WEN + DIN が同時、 読み出しは DOUT が1サイクル遅れます。 DOUTは実際にはずっと何かしら出ています。







* "EN (enable)" must kept high while Block RAM is active

* Actually, DOUT is also active while the RAM is being written

