

EDK Training at University of Toronto

Processor IP Team November 2003

Table of Contents

Introduction and Overview	Page 3
PowerPC and MicroBlaze	Page 11
Processor IP	Page 19
Creating a Simple MicroBlaze System with	
XPS with Base System Builder	Page 49
Creating a Simple MicroBlaze System with	
XPS without Base System Builder	Page 79
Adding I/O Peripherals to a System	Page 96
Software Development with the EDK and XPS	Page 110
Device Drivers & Software Infrastructure	Page 119







Introduction and Overview



Programmable Logic Evolution



Implementing Processor Systems on Xilinx FPGAs

Xilinx Virtex-II Pro FPGA Setting the Standard in Programmable Logic



Virtex-II Pro Revolution





Up to 24 Serial Transceivers
 622 Mbps to 3.125 Gbps

PowerPC™

- Up to 4 PowerPC 405 Processors
 - Industry standard
 - 420 DMIPS at 300 MHz

Built on the Success of Virtex-II Fabric



RocketIO[™] SerDes Leading-Edge High-Speed Serial



- Multi-Rate
 - 3.125, 2.5, 2.0, 1.25, 1.0 Gbps
 - 2 24 transceivers
- Multi-Protocol
 - 1G, 10 G Ethernet (XAUI)
 - PCI Express
 - Serial ATA
 - InfiniBand
 - FibreChannel
 - Serial RapidIO
 - Serial backplanes...
- Key Features
 - Embedded 8B/10B Coding
 - 4-Level Pre-Emphasis
 - Programmable Output Swing
 - AC & DC Coupling
 - Channel bonding



Page 7

Virtex-II Pro Device Family Covers the Entire Design Spectrum

	2VP2	2VP4	2VP7	2VP20	2VP30	2VP40	2VP50	2VP70	2VP100	2VP125
EasyPath					XCE2VP30	XCE2VP40	XCE2VP50	XCE2VP70	XCE2VP100	XCE2VP125
Logic Cells	3,168	6,768	11,088	20,880	30,816	43,632	53,136	74,448	99,216	125,136
BRAM (Kbits)	216	504	792	1,584	2,448	3,456	4,176	5,904	7,992	10,008
DCMs	4	4	4	8	8	8	8	8	12	12
PowerPC" Processors	0	1	1	2	2	2	2	2	2	4
Rocket 10 m Multi-Gigabit Transceivers	4	4	8	8	8	12	16	20	20	24
X Multiplier Blocks	12	28	44	88	136	192	232	328	444	556
Package Maximum User I/O										
FG256 140	140	140								
FG456 248	156	248	248							
FF672 396	204	348	396							
FG676 416				404	416	416				
FF896 556			396	556	556					
FF1152 692				564	644	692	692			
FF1148* 812						804	812			
FF1517 964							852	964		
FF1704 1040								996	1040	1040
FF1696* 1200									1164	1200

* Note Special bond out option, NO MGT with Maximum Select I/O



Embedded Development Kit (EDK)

- All encompassing design environment for Virtex-II Pro *PowerPC[™]* and *MicroBlaze* based embedded systems in Xilinx FPGAs
- Integration of mature FPGA and embedded tools with innovative IP generation and customization tools
- Delivery vehicle for Processor IP





EDK System Design Comprehensive Tool Chain





PowerPC and MicroBlaze



PowerPC 405



D-Side On-Chip Memory (OCM)

- 5-stage data path pipeline
- 16KB D and I Caches
- Embedded Memory
 Management Unit
- Execution Unit
 - Multiply / divide unit
 - 32 x 32-bit GPR
- Dedicated on-chip memory interfaces
- Timers: PIT, FIT, Watchdog
- Debug and trace support
- Performance:
 - 450 DMIPS at 300 MHz
 - 0.9mW/MHz Typical Power



What is MicroBlaze?



- It's a soft processor, around 900 LUTs
- **RISC Architecture**
- 32-bit, 32 x 32 general purpose registers
- Supported in Virtex/E/II/IIPro, Spartan-III/II/E

More on MicroBlaze ...

- Harvard Architecture
- Configurable instruction cache, data cache
- Non-intrusive JTAG debug
- Support for 2 buses:
 - LMB (Local Memory Bus) 1 clock cycle latency, connects to BRAM
 - OPB (On-chip Peripheral Bus) part of the IBM
 CoreConnect[™] standard, connects to other peripheral
 "Portable" IP between PPC and MB
- Big endian, same as PowerPC PPC405

MicroBlaze Interrupts and Exceptions

- Interrupt handling
 - 1 Interrupt port
 - 32+ interrupts and masking supported through interrupt controller(s)
- Exception handling
 - No exceptions generated in Virtex-II versions
 - One in Virtex/E and Spartan-II versions for MUL instruction



Software Tools

- GNU tool chain
- GCC GNU Compiler Collection
- GDB The GNU debugger
 - Source code debugging
 - Debug either C or assembly code
- XMD Xilinx Microprocessor Debug utility
 - Separate Process
 - Provides cycle accurate program execution data
 - Supported targets: simulator, hardware board



Software - XMD

- Interfaces GDB to a "target"
- Allows hardware debug without a ROM monitor or reduces debug logic by using xmd-stub (ROM monitor)
- Offers a variety of simulation targets
 - Cycle accurate simulator
 - Real hardware board interface via UART or MDM
- Includes remote debugging capability



MicroBlaze Debug Module

- JTAG debug using BSCAN
- Supports multiple MicroBlazes
- Software non-intrusive debugging
- Read/Write access to internal registers
- Access to all addressable memory
- Hardware single-stepping
- Hardware breakpoints configurable (max 16)
- Hardware read/write address/data watchpoints
 - configurable (max 8)



Processor IP

Hardware and Software



Example MicroBlaze System



Example PPC405 System



Processor IP (HW/SW)

Infrastructure (includes Device Drivers)

- MicroBlaze CPU
- LMB2OPB Bridge
- PLB Arbiter & Bus Structure (PLB_V34)
- OPB Arbiter & Bus Structure (OPB_V20)
- DCR Bus Structure (DCR_V29)
- PLB2OPB Bridge
- OPB2PLB Bridge
- OPB2OPB Bridge Lite
- OPB2DCR Bridge
- System Reset Module
- BSP Generator (SW only)
- ML3 VxWorks BSP (SW only)
- Memory Test Utility (SW only)

OPB IPIF Modules (includes Device Drivers)

- PLB IPIF
 - OPB IPIF-Slave Attachment
 - OPB IPIF-Master Attachment
 - IPIF-Address Decode
 - IPIF-Interrupt Control
 - IPIF-Read Packet FIFOs
 - IPIF-Write Packet FIFOs
 - IPIF-DMA
 - IPIF-Scatter Gather
 - IPIF-FIFOLink
- PLB IPIF
 - PLB IPIF-Slave Attachment
 - PLB IPIF-Master Attachment
 - IPIF-Address Decode
 - IPIF-Interrupt Control
 - IPIF-Read Packet FIFOs
 - IPIF-Write Packet FIFOs
 - IPIF-DMA
 - IPIF-Scatter Gather
 - IPIF-FIFOLink

Processor IP (HW/SW)

Memory Interfaces (includes Device Drivers & Memory Tests)

- PLB EMC (Flash, SRAM, and ZBT)
- PLB BRAM Controller
- PLB DDR Controller
- PLB SDRAM Controller
- OPB EMC (Flash, SRAM, and ZBT)
- OPB BRAM Controller
- OPB DDR Controller
- OPB SDRAM Controller
- OPB SystemACE
- LMB BRAM Controller

Peripherals (includes Device Drivers & RTOS Adapt. Layers)

- OPB Single Channel HDLC Controller
- OPB<->PCI Full Bridge
- OPB 10/100M Ethernet
- OPB 10/100M Ethernet Lite
- OPB ATM Utopia Level 2 Slave
- OPB ATM Utopia Level 2 Master

Peripherals (continued)

- OPB IIC Master & Slave
- OPB SPI Master & Slave
- OPB UART-16550
- OPB UART-16450
- OPB UART Lite
- OPB JTAG UART
- OPB Interrupt Controller
- OPB TimeBase/Watch Dog Timer
- OPB Timer/Counter
- OPB GPIO
- PLB 1G Ethernet
- PLB RapidIO
- PLB UART-16550
- PLB UART-16450
- PLB ATM Utopia Level 2 Slave
- PLB ATM Utopia Level 2 Master
- PLB ATM Utopia Level 3 Slave
- PLB ATM Utopia Level 3 Master
- DCR Interrupt Controller



System Infrastructure

- Hardware IP
 - Common PowerPC and MicroBlaze peripherals
 - Peripherals are common across bus types
 - Parameterize for optimal functionality, optimal FPGA usage
 - IP Interface (IPIF) provides common hardware blocks
- Software IP (Device Drivers)
 - Common across processors and operating systems



The Benefits of Parameterization

Example: OPB Arbiter

Parameter Values					Resources	F _{MAX}	
NUM_MASTERS	PROC_INTERFACE	DYNAM_PRIORITY	PARK	REG_GRANTS	LUTs	MHz	
1	N	N	Ν	N	11	295	
2	N	N	Ν	N	18	223	
4	N	N	N	N	34	193	
4	Y	N	N	N	59	156	
4	N	Y	Ν	N	54	169	
4	N	N	Y	N	83	159	
4	N	N	Ν	Y	34	201	
4	Y	Y	Y	Y	146	145	
8	Y	Y	Y	Y	388	112	



- Significantly increases performance or saves area
- Only include what you need
- This can only be accomplished in a programmable system



Full IP Interface (IPIF)



- Consists of 8 modules
 - Each module is selectable and parameterizable
- Automatically configures a core to the processor bus
 - Xilinx IP Cores
 - 3rd Party IP Cores
 - Customer proprietary cores and external devices
- OPB & PLB supported
 - Bus independent IP Cores and associated Device Drivers
- IPIF will be added to other LogiCOREs



Buses and Arbiters

- PLB
 - Arbitration for up to 16 masters
 - 64-bit and 32-bit masters and slaves
 - IBM PLB compliant
- OPB
 - Includes arbiter with dynamic or fixed priorities and bus parking
 - Parameterized I/O for any number of masters or slaves
 - IBM OPB compliant
- DCR
 - Supports one master and multiple slaves
 - Daisy chain connections for the DCR data bus
 - Required OR function of the DCR slaves' acknowledge signal
- LMB
 - MicroBlaze single-master Local Memory Bus



Bridges

- PLB to OPB
 - Decode up to 4 different address ranges
 - 32-bit or 64-bit PLB slave, 32-bit OPB master
 - Burst and non-burst transfers, cache-line transactions
- OPB to PLB
 - 64-bit PLB master, 32-bit OPB slave
 - Burst and non-burst transfers, cache-line transactions
 - BESR and BEAR
- OPB (slave) to DCR (master)
 - Memory mapped DCR control
- OPB to OPB
 - Allows further OPB partitioning



More System Cores

- Processor System Reset
 - Asynchronous external reset input is synchronized with clock
 - Selectable active high or active low reset
 - DCM Locked input
 - Sequencing of reset signals coming out of reset:
 - First bus structures come out of reset
 - Second Peripheral(s) come out of reset 16 clocks later
 - Third the CPU(s) come out of reset 16 clocks after the peripherals
- JTAG Controller
 - Wrapper for the JTAGPPC primitive.
 - Enables the PowerPC's debug port to be connected to the FPGA JTAG chain
- IPIF User Core Templates
 - Convenient way to add user core to OPB or PLB

Timer / Counter

- Supports 32-bit OPB v2.0 bus interface
- Two programmable interval timers with interrupt, compare, and capture capabilities
- Programmable counter width
- One Pulse Width Modulation (PWM) output



Watchdog Timer / Timebase

- Supports 32-bit bus interfaces
- Watchdog timer (WDT) with selectable timeout period and interrupt
- Two-phase WDT expiration scheme
- Configurable WDT enable: enable-once or enabledisable
- WDT Reset Status (was the last reset caused by the WDT?)
- One 32-bit free-running timebase counter with rollover interrupt



Interrupt Controller

- Number of interrupt inputs is configurable up to the width of the data bus width
- Interrupt controllers can be easily cascaded to provide additional interrupt inputs
- Master Enable Register for disabling the interrupt request output
- Each input is configurable for edge or level sensitivity
 - rising or falling, active high or active low
- Output interrupt request pin is configurable for edge or level generation

UART 16550 / 16450 / Lite

- Register compatible with industry standard 16550/16450
- 5, 6, 7 or 8 bits per character
- Odd, even or no parity detection and generation
- 1, 1.5 or 2 stop bit detection and generation
- Internal baud rate generator and separate RX clock input
- Modem control functions
- Prioritized transmit, receive, line status & modem control interrupts
- Internal loop back diagnostic functionality
- Independent 16 word transmit and receive FIFOs



IIC

- 2-wire (SDA and SCL) serial interface
- Master/Slave protocol
- Multi-master operation with collision detection and arbitration
- Bus busy detection
- Fast Mode 400 KHz or Standard Mode 100 KHz operation
- 7 Bit, 10 Bit, and General Call addressing
- Transmit and Receive FIFOs 16 bytes deep
- Bus throttling

SPI

- 4-wire serial interface (MOSI, MISO, SCK, and SS)
- Master or slave modes supported
- Multi-master environment supported (requires tri-state drivers and software arbitration for possible conflict)
- Multi-slave environment supported (requires additional decoding and slave select signals)
- Programmable clock phase and polarity
- Optional transmit and receive FIFOs
- Local loopback capability for testing



Ethernet 10/100 MAC

- 32-bit OPB master and slave interfaces
- Media Independent Interface (MII) for connection to external 10/100 Mbps PHY Transceivers
- Full and half duplex modes of operation
- Supports unicast, multicast, broadcast, and promiscuous addressing
- Provides auto or manual source address, pad, and Frame Check Sequence


Ethernet 10/100 MAC (cont)

- Simple DMA and Scatter/Gather DMA architecture for low processor and bus utilization, as well as a simple memory-mapped direct I/O interface
- Independent 2K to 32K transmit and receive FIFOs
- Supports MII management control writes and reads with MII PHYs
- Supports VLAN and Pause frames
- Internal loopback mode



1 Gigabit MAC

- 64-bit PLB master and slave interfaces
- GMII for connection to external PHY Transceivers
- Optional PCS function with Ten Bit Interface (TBI) to external PHY devices
- Option PCS/PMA functions with SerDes interface to external transceiver devices for reduced signal count
- Full duplex only
- Provides auto or manual source address, pad, and Frame Check Sequence

1 Gigabit MAC (cont)

- Simple DMA and Scatter/Gather DMA architecture for low processor and bus utilization, as well as a simple memory-mapped direct I/O interface
- Independent, depth-configurable TX and RX FIFOs
- Supports MII management control writes and reads with MII PHYs
- Jumbo frame and VLAN frame support
- Internal loopback mode



Single Channel HDLC

- Support for a single full duplex HDLC channel
- Selectable 8/16 bit address receive address detection, receive frame address discard, and broadcast address detection
- Selectable 16 bit (CRC-CCITT) or 32 bit (CRC-32) frame check sequence
- Flag sharing between back to back frames
- Data rates up to OPB_Clk frequency



Single Channel HDLC (cont)

- Simple DMA and Scatter/Gather DMA architecture for low processor and bus utilization, as well as a simple memory-mapped direct I/O interface
- Independent, depth-configurable TX and RX FIFOs
- Selectable broadcast address detection and receive frame address discard
- Independent RX and TX data rates



ATM Utopia Level 2

- UTOPIA Level 2 master or slave interface
- UTOPIA interface data path of 8 or 16 bits
- Single channel VPI/VCI service and checking in received cells
- Header error check (HEC) generation and checking
- Parity generation and checking
- Selectively prepend headers to transmit cells, pass entire received cells or payloads only, and transfer 48 byte ATM payloads only

ATM Utopia Level 2 (cont)

- Simple DMA and Scatter/Gather DMA architecture for low processor and bus utilization, as well as a simple memory-mapped direct I/O interface
- Independent, depth-configurable TX and RX FIFOs
- Interface throughput up to 622 Mbps (OC12)
- Internal loopback mode

OPB-PCI Bridge

- 33/66 MHz, 32-bit PCI buses
- Full bridge functionality
 - OPB Master read/write of a remote PCI target (both single and burst)
 - PCI Initiator read/write of a remote OPB slave (both single and multiple)
- Supports up to 3 PCI devices with unique memory PCI memory space
- Supports up to 6 OPB devices with unique memory OPB memory space
- PCI and OPB clocks can be totally independent



System ACE Controller

- Used in conjunction with System ACE CompactFlash Solution to provide a System ACE memory solution.
- System ACE Microprocessor Interface (MPU)
 - Read/Write from or to a CompactFlash device
 - MPU provides a clock for proper synchronization
- ACE Flash (Xilinx-supplied Flash Cards)
 - Densities of 128 MBits and 256 Mbits
 - CompactFlash Type 1 form factor
 - Supports any standard CompactFlash module, or IBM microdrives up to 8 Gbits, all with the same form factor.
- Handles byte, half-word, and word transfers



GPIO

- OPB V2.0 bus interface with byte-enable support
- Supports 32-bit bus interface
- Each GPIO bit dynamically programmable as input or output
- Number of GPIO bits configurable up to size of data bus interface
- Can be configured as inputs-only to reduce resource utilization



Memory Controllers

- PLB and OPB interfaces
- External Memory Controller
 - Synchronous Memory (ZBT)
 - Asynchronous Memory (SRAM, Flash)
- Internal Block Memory (BRAM) Controllers
- DDR and SDRAM



Device Drivers and Board Support Packages (BSPs)

Board Support Package (BSP)

Ethernet 10/100 Ethernet 10/100 Bovice Driver Davice Driver Davice Driver Driver</

- Device drivers are provided for each hardware device
- Device drivers are written in C and are designed to be portable across processors
- Device drivers allow the user to select the desired functionality to minimize the required memory
- BSPs are automatically generated by EDK tools





Creating a Simple MicroBlaze System with XPS with Base System Builder Wizard

Design Flow

- Design Entry with Xilinx Platform Studio using the Base System Builder Wizard
- Generate system netlist with XPS
- Generate hardware bitstream with XPS
- Download and sanity check design with XPS and XMD



Simple MicroBlaze System Block Diagram





Start Xilinx Platform Studio





Create A New Project

Create New Project Using Base System Builder Wizard 🛛 🛛 🔀
New Project
The project file will be created in the current directory if a path is not specified.
Project File are_training\EDK_6_1\system.xmp Browse
Peripheral Repository Directory
User Peripheral Repository search path for IP, driver and library files. Can be a semicolon separated list of directories.
Browse
OK Cancel

- Select File from the Tools menu
- Select the New Project submenu and the Base System Builder submenu
- Browse to the location where the project is to be located
- Click OK to start creating the project



Selecting The Board



Welcome to the Base System Builder!

Please begin hu selecting your target development hoars

This tool will lead you through the steps necessary to create the hardware components of an embedded system.

	Xiins	•
Board Name	Virtex-II Multimedia FF896 Development Board	
Board Revision	1	•
Board Descript	ion	_
development pl Virtex-II). This I switches, 4 pus	lattom for designing and verifying applications using a Xilinx boards includes 32MB of SDRAM, 1 RS232 serial port, 8 DIP sh buttons, and 4 LEDs.	Y

- Select Xilinx as the Board Vendor
- Select Virtex-II Multimedia FF896 Development Board as the Board Name
- Select Board Revision 1
- Click Next to continue to the next step



Select the Processor

elect the processor for your embedd	ded system
MicroBlaze	MicroBlaze
C BowerPC	PRO
Not supported by this board.	PowerPt"
Processor Description	Merodiaze
The MicroBlaze(TM) 32-bit soft pro 32 register by 32 bit LUT RAM-bat instructions for data and memory a BlockRAM and/or external memor the FPGA labric and operate off th	ocessor is a RISC-based engine with a sed Register File, with separate access. It supports both on-chip ry. All peripherals are implemented on he on-chip peripheral bus (OPB).

- This board has a Virtex-II FPGA which does not contain a PowerPC processor
- Click Next to continue to the next step



Configuring The Processor

e System Builder - Configure Processor	
Select clock frequency, debug interface and other features of your embedded system. System Wide Setting Processor Clock Frequency: 27 MHz	
Processor Configuration Debug I/F C XMD with S/W debug stub Onrohip H/W debug module No Debug Micros Staze Local Data and Instruction Memory Uses BRAM Eache Cache Enabled	

- Select On-chip H/W debug module such that a ROM monitor is not required
- Select 64KB of Local Data and Instruction Memory (BRAM)
- There is no need to select caches when using internal BRAM
- Click Next to continue to the next step

Configuring I/O Interfaces

The following ID interfaces were found on your taxet by	oard	
Xiins Viter-II Multimedia FF896 Development Boar	d Revision 1	
Please select the ID interfaces or ports which you would	d like to use:	
ID Devices		
₩ RS232	Data Sheet	
Peripheral OPB UARTLITE -	0.010 011000	
Baudrate (Bits 9500 V		
Data Bits: 8		
Paily. NDNE V		
, ave merup		

- The board has a serial port and it is default behavior to build the hardware with it
- Since it is used as standard I/O it is not necessary to be interrupt driven
- Click Next to continue to the next step



Adding Internal Peripherals

se System Builder - Add Internal Per	ripherais	
Add other peripherals that do not interact of Peripheral" button to select from the list of	with off-chip components. Use the "Add I available peripherals.	
If you do not wish to add any non-IO perip	oherals, click the "Next" button.	
	Add Peripheral	
Peripherals		
Mana Inda - 1	(Park Contractor)	Carro

- Other peripherals can be added at this point, such as a timer counter.
- Click Next to continue to the next step



System Summary

Processor: Microbil System clock hequ Debug interface: D On Chip Memory :	sze ency: 27 MHz n-Chip HW Debug N 64 KB	lodule		 The syst and is re
The address maps b sing the editing fea OPB Bus : OPB	alow have been au itures of XPS. _V20 Inst. name	ionalically assigned	You can modily them	Review t
Core Name	Instance Name	Bate Addr	High Addr	
opb_natile	RS220	0x00030100	0x000301FF	cyctom c
inb_bram_il_critir	inb_criti	Dx0000000		necessa
Core Name	Vio inst. name	Base Add	High Adds	
inb_bram_if_cnth	dmb_cnth	0x00000000	0x0000FFFF	Click the
				_
				croato th
				create

- The system has been created and is ready to be generated
- Review the details of the system and backup if necessary to make changes
- Click the Generate button to create the data files for XPS

Base System Builder Finished



- The Base System Builder Wizard in XPS has completed
- The data files for XPS have been generated such that the system will be contained in an XPS project
- Click the Finish button to exit the wizard & return to XPS



Generating Hardware NetList



- Select the Tools menu
- Select the Generate Netlist submenu
- Wait for the generation to complete



Generating Hardware Bitstream



- Select the Tools menu
- Select the Generate Bitstream submenu
- Wait for the bitstream generation to complete

Adding Software Source Files

Image: Sector File Image: Sector File Image: Sector File Image: File Image: File Image: File	Xilinx XPS - Cr\mic	roblaze_system\ -[syste	m.pbd]					_ // X
Besyne Project Actign Defmalt Defmalter Defmalt Defmalter Defmalt Defmalt Defmalt Defmalt Defmalt Def		Add Program Sources Remove File	1 4 (H)	a a x x	* 🖬 📑	8 d E 8 e		-1812
Dispose Import NMS Import NMS Seve NMS Import NMS Flac system.mino Import NMS Flac system.mino		Besynic Project Accign Definult Drivers		1 150 TH 6n	× to to	K O J	9 701	
In open block Seve Mos is included: Seve is included: Seve i	plant Carponent	Inport MSS Import MSS Import MSS						
Debug Mode (Add/Eck Cores (dialog) STDOUT Linker Scipt: Sources Headen # opb_mdn_0 # leb_tren_R_creb_1 - leb_v10_0 - opb_v20_0 - leb_v10_1 Project Files - MHS File system.min MVS F	- bram_block III Imb_bram_it E microblaze_	Sove MSS Sove MVS Sove Makafás	Ť			-	-	
STOUT. Linker Scipt: Sources Headen Hopb_mdm_0 He Mobian_R_cent_1 Inth_v10_0 opb_v20_0 Inth_v10_1 Project Files MHS File system.min MVS File system.min PRD File syst	- Debug Mode E - STDIN_	Add/Edit Cores (dialog) Add Cores (text)				1000		
- Ind_v10_0 - opb_v20_0 - Ind_v10_1 Project Files - MHS File system miss - MVS File system miss	STDOUT - Linker So - Sources - Headers ⊞ opb_mdm_0 in Imb_bram_R_	nipt certin_1			-	-M	- 11 ¹	
MVS FRE system mvs	- Imb_v10_0 - opb_v20_0 - Imb_v10_1 Project Files - MHS File syst - MSS File syst	tern, miho ann. masa		×0 ×				
	MVS File syst PSD File such	en nha	푀	tystem.pbd	1			
// *** BATCH CMD : addDevice position 2 file implementation/download bit	H = BATCH CMD	addDevice position 2 file impl	ementation/downloa	dbł				14

- Select processor in System tab
- Select the Tools menu
- Select the Add Program Sources submenu
- Navigate to the source file and select it.



Setting Compile Options

Compiler Options - microblaze instance microblaze_0 🛛 🔀
Environment Optimization Directories Details Others
Optimization Parameters Optimization Level No Optimization Level 3 is the most optimized level.
 Use Global Pointer Optimization Use Hardware Multiplier (only on Virtex II)
Debug Options
C Do not generate debug symbols
Create symbols for debugging (-g option)
C Create symbols for assembly (-gstabs option)
NOTE: If an optimization level is set, and -g is also set, the debug information may not be correlated to source code.
OK Cancel Apply Help

- Select the Options menu
- Select the Compiler Options submenu
- Set the optimization to none
- Set the debug options to create symbols for debugging



Setting Up Standard I/O

S/W Settings - microblaze instance microblaze_0
Optimization Directories Details Others
Processor Property Environment
Driver Configuration
Device Driver Cpu
Driver Version 1.00.a
Interface Level
Default
Mode © Executable © XmdStub
Communication Peripherals
STDIN Peripheral opb_uartlite_0
STDOUT Peripheral opb_uartlite_0
Debug Peripheral
OK Cancel Apply Help

- Select the System tab
- Double click on the microblaze_0
- The dialog box illustrated is displayed
- Select the opb_uartlite_0 for the STDIN and STDOUT peripheral and click the OK button



Assigning Drivers To Devices

Peripheral Options		x
Peripheral Attributes Peripheral opb_e Base Address 0x800 High Address 0x800	themet 00000 03fff	
 Interrupt Handler Functi IP2INTC_Irpt 	on	
An Interrupt handler ca port of type INTERRUP	n be specified only if the peripheral has a PT and that port is connected in MHS.	
Driver Configuration		
Device Driver	emac 💌	
Driver Version	1.00.c	
Interface Level	1	
Connect to OS	none	
Please view the MDD file	for any other parameters to override.	
MDD Params De	efault OK Cancel	

- Assigning a driver to a device causes the driver to be compiled into the library & linked with the application
- Double click on the device in the System tab of the XPS project
- The latest version of the driver is displayed by default
- Choose the appropriate Interface Level of the driver



Setting Memory Options

S/W Settings - mi	croblaze instan	ce mblaze	×				
Processor	Property	Enviror	nment				
Optimization	Directories	Details	Others				
Memory Informa These options You can specif	Memory Information These options are not used for ''dcc'' compiler. You can specify them in the linker file for dcc.						
Program Start A	ddress <mark>0x80f</mark>	00000					
Stack Size	0x200	0					
Heap Size							
Pass Space Se	parated Options t	0					
Preprocessor (-	Wp)						
Assembler (-Wa	i) [
Linker (-WI)			_				
, For example, to pass a symbol definition to the linker, you would say -defsym SYMBOL=VALUE							
ОК	Cancel	Apply	Help				

- Double click on the processor instance in the System tab
- Select the Details tab
- Change the program start address or stack size
- Add other options to the preprocessor, assembler, or linker



Setting Library Options

S/W Settings - mi	croblaze inst	ance	mblaze		x
Processor Optimization	Property Directories	Ţ	Enviro Details	nment Others	4
Program Sourc Give those cor any other plac	es Compiler Op npiler options w e on this dialog	ions— hich c box.	ould not be s	pecified	
Library Genera Give compiler	tor	l by Lil	oGen if you v	want to	
Compiler Flags Extra compiler in addition to ti	flags will be use ne default optio	ed by L	ibrary Gener	ator	
Extra Compiler F	lags g -00				
OK	Cancel		Apply	Help	

- Double click on the processor instance in the System tab
- Select the Others tab
- Add or change options for library compilation
- Add other options for application compilation

Generating Libraries In XPS



- A Library containing the device drivers and the startup code is built for an application to be linked against
- The Library helps separate Board Support Package development from application development
- Libraries will automatically be built if they don't exist when the application is built

Compiling The Software



- Select the Tools menu
- Select the Compile
 Program Sources
 submenu
- Wait for the compile to complete



Updating the Bitstream



- Select the Tools menu
- Select the Update Bitstream submenu
- The hardware bitstream is updated to contain the contents of the software elf file



Downloading The Hardware

The sale many Project [100% Oppone Add window	Help					- 11	-18 X
	Generate Libraries Comple Program Sources Get Brogram Size Generate BSP for YoWorks	an ar	10××	× 1 1		10 -1 9	1 201	
otions Camponents System	Generate (Jetlet Generate Bistream	-			2 12	. V - 191	2	
System BSP	Update Bitstream	4	Ĩ.		-	1	L.	
(k) Imb_bram_il_criti_0 microblaze_0 Debug Peripheral	Sim (godel Generation Hardware Simulation					11.10 201		
- Mode: EXECUTA - STDIN: - STDOUT: - Linker Script	1940 Software Debugger Xygwin shel			3	27	-	27	
Sources	Export to ProMay Import from ProMay		9 0.000 0114					
⊕-opb_mdm_0 ⊛ imb_bram_ič_onti_1 — imb_v10_0 nnti_v20_0	Genetons Ward Clight Dieper Jepert (H15F2) Genetal Waard HDL Wither	1	sector se	ic w	u x		2	
INFD :MIRACT:1365 - Reading etclvc18v04.bod 21.Loading He Implementation done.	v/download.bit"							11

- Make sure that the board power is on and the parallel pod is connected
- Select the Tools menu
- Select the Download submenu
- Wait for the download to complete


Running XMD

C:\EDK_6_1\bin\nt\xmd.exe	
Xilinx Microprocessor Debug (XMD) Engine Xilinx EDK 6.1.1 Build EDK_G.13 Copyright (c) 1995-2002 Xilinx, Inc. All rights reserved. XMD% mbconnect mdm	
JTAG chain configuration	
Device ID Code IR Length Part Name 1 0a001093 8 System_ACE 2 01028093 6 XC2V1000 Assuming, Device No: 2 contains the MicroBlaze system Connected to the JTAG MicroBlaze Debug Module (MDM) No of processors = 1 MicroBlaze Configuration : Version - 2.00.a No of PC HW Breakpoints : 8 No of Read Addr/Data Watchpoints : 1 No of Write Addr/Data Watchpoints : 1 Instruction Cache Support : off Data Cache Support : off	
Connected to MicroBlaze "mdm" target. id = 0 Starting GDB server for "mdm" target (id = 0) at TCP port no 1234 XMD% _	

- Select the Tools menu
- Select the XMD submenu
- Type "mbconnect mdm" to connect XMD to the MicroBlaze processor.



Testing Memory

- Type "mrd 0x1000 2" to read 2 memory locations starting at address 0x1000
- Type "mwr 0x1000 0x12345678" to write to memory location 0x1000
- Perform writes to location 0x1004 also
- Type "mrd x1000 2" to read 2 memory locations and verify the values that were written

C:\EDK\bin\nt\xmd.exe	_ 🗆 🗙
XMD% mrd 0x1000 2 1000: 0000000 1004: 0000000 XMD% mwr 0x1000 0x12345678 XMD% mwr 0x1004 0xdeadbeef XMD% mrd 0x1000 2 1000: 12345678 1004: DEADBEEF	
	• •

Starting The GNU Debugger (GDB)

ð.	ystem	.c - 5	ource Wi	ndow										
Ble	Bun	New	⊆ontrol	Brefer	ences	Help								
4	'(†)	03	{} * ()	(1)	ŵ	*	A (- A	6	<u>+3</u> 🛛		Øxe Ø	5	eek
sy	sten	.C		_	Ŧ	mair	1	۳	\$	OURCE	¥			
-	1 3 4 5 6 7 8 9 10 11	int {	nain() st { } re	atic ile (turn	int (1) cou	coun	t;				2			
Frog	am no	: runni	ng, Clidko	n run io	on to s	tart.	_							_

- Select the Tools menu
- Select the Software Debugger submenu



Target Selection & Download

Target Selection	×
	🔽 Set breakpoint at 'main'
Target: Remote/TCP : XMD	Set breakpoint at 'exit'
Hostname: localhost	Set breakpoint at
Port: 1234	🔲 Display Download Dialog
	Use xterm as inferior's tty
More Options	<u>} </u>
	OK Cancel Help

- In GDB, select the Run menu and choose the Run menu item
- Wait for the Target Selection dialog box to be displayed
- Enter the data in the dialog box and click OK



Stepping in GDB

💏 system.c - Source Window
<u>File R</u> un <u>V</u> iew <u>C</u> ontrol <u>P</u> references <u>H</u> elp
🚯 🕐 🕐 🏷 🚯 🐨 👗 🙈 🚍 68 🐽 📲
system.c 💌 SOURCE 💌
1 int main()
- 3 static int count;
4 5 while (1)
6 { - 7 count++;
- 8 }
- 10 return 0;
Program stopped at line 5

- The debugger is ready, the program counter is at a breakpoint at line 5 of the source file
- Select the Control menu
- Select the Step submenu



Watching A Variable With GDB

Watch Expres	sions	4	
Name	Value	-	-
count	1		
 			- -
			Add Watch

- Double click on the variable count such that it is selected
- Right click and select in the submenu to Add count to Watch
- A dialog box is displayed which contains the variable count and it's contents



Creating a Simple MicroBlaze System with XPS

without Base System Builder Wizard

Design Flow

- Design Entry with Xilinx Platform Studio
- Generate system netlist with XPS
- Generate hardware bitstream with XPS
- Download and sanity check design with XPS and XMD



Start Xilinx Platform Studio





Create A New Project

Create New Project			×
- New Project-			
The project file will be c	reated in the curr	ent directory if a p	ath is not specified.
Project File	<pre>\microblaze_syste</pre>	em\system.xmp	Browse
Existing MHS to Import (Optional)			Browse
Target Device			
Architecture	Device Size	Package	Speed Grade
Virtex2	xc2v100(💌	fg456 💌	4
Peripheral Repository D	irectory		
Check this box if M installation area an	HS uses peripher d in 'myip' directo	als other than thos ry in the project dir	se in EDK 🗟 ectory.
			Browse
		OK	Cancel

- Select New Project from the Tools menu
- Enter all the project information
- Click OK on this dialog box.
- Click Yes on the next dialog box to start with an empty MHS file



Adding New Cores

🙀 Xilinx XPS - C:\microblaze_system\ - [system	n.pbd]	_ # ×
File Edit View Project Tools Options Add	Window Help	_ 8 ×
Add Program Sources Remove File		
Resync Project Assign Default Drivers	※ ※ ※ だ % が % × != != × != つ Ŧ ヽ O ヽ C) A E
Optione Component Import MHS Import MHS Import MMS		
Project Files Save MS5 MHS File:sy Save MVS		
MSS File:sy Save Makefile MVS File:sy PBD File:sy Add/Edit Cores (dialog) Add Cores (text) Device: xc2v1000ig456-4 Netist: Hier (Toplevel) Implementation: XPS HDL: VHDL Sim Modet BEHAVIDRAL	system.pbd	
Assigning default drivers to peripherals C:\microblaze_system\system.mpReading MHS file Done. Found 0 cores Reading MSS file system.mss Done Reading MVS file system.mvs Done 0utput Warnings Errors	C:\microblaze_system\system.mhs	*
Add cores, buses, ports to your system. Edit bus connect	tions, ports and parameters.	-78,1170]

- Select the Project menu
- Select the Add/Edit Cores submenu



Adding Bus Structures

enghesals Bus Connections Ports Parameters					
Click on squares to make macher, slave or master-slave (M. S. MS). Right click on any bus instance (column header) for a context mere	convections. A		Choose one or more (using shiftend Orl) buses and click Add		
			doi_v20_v1_ bil v20_v1_0	00_e	_
	1		ind vill vill och vill vill	00 a 10 a	
	1 11	P/03	pb_v34_v1	01_a	-
	Chocos the BRA	W part to the canin	connect to the	controller port.	
	Crife Port	BUA	el Part.	Connector	_

- Select the Bus Connections Tab
- Select the Imb_v10_v1_00_a bus & opb_v20_v1_10_b bus and click the Add button
- Select the Imb_v10_v1_00_a bus and click the Add button

Adding Basic Peripherals



- Select the Peripherals tab
- Select the bram_block, Imb_bram_if_cntlr, microblaze, and opb_mdm and click the Add button.
- Select the Imb_bram_if_cntlr and click the Add button.



Change The Memory Map



 Edit the Base Address and High Address for the Imb_bram_if_cntlr and opb_mdm peripherals



Setting Bus Masters & Slaves

dd/Edit Hardware Platfor	m Sj	peci	ifica	tions			×
Peripherals Bus Connection	ns	Ports	: 1	Parameters			
Click on squares to make m Right click on any bus insta	aste ince	r, sla (colu	me o umn l	r master-slave (M, S, MS) con header) for a context menu	nections.	Choose one or more buses and click Add	(using shiftand Ctrl)
	/3	2	3			dor_v29_v1_00_a tst_v20_v1_00_b	
Imb_bram_if_cntlr_0 slmb	\$				<< Add	opb_v20_v1_10_a	
microblaze_0 dimb	м					opb_v20_v1_10_b pib_v34_v1_01_a	
microblaze_0 ilmb			м		There a Mar BRAM	and the successful to the surplu	day and
microblaze_0 dopb		м			Give a name to the	connection.	oller port.
microblaze_0 iopb		м			Cotle Port	RRAM Dort	Connector
opb_mdm_0 sopb		5			Imb bram if c	bram block 0 PORTA *	conn 0
Imb_bram_if_cntlr_1 slmb			5		lmb_bram_if_c	bram_block_0 PORTB	conn_1
					ΟΚ	Cancel	Apply Help

- Select the Bus Connections tab
- Set the masters and slaves on the buses by clicking on the boxes with an 's' and 'M'



Setting MicroBlaze Parameters

or i a anexes assigned to it	P Instance in MHS		Choose IP Instance for a list of	d parameters.
hese parameter values will o	wenide default MPD values.		microblaze_0	 Open PDF D
Parameter	Value	-	Parameters with default value Choose one or more (using sh	s from MPD. ift and ctil) and click Add.
C_DEBUG_ENABLED	1	_	Parameter	Value
_NUMBER_OF_PC_BRK	5	- C	C_USE_BARREL	0
		N N	C_USE_DIV	0
			C_DEBUG_ENABLED	0
		<< Add	C_NUMBER_OF_PC_BRK	1
			C_NUMBER_OF_RD_AD	0
			C_NUMBER_OF_WR_AD	0
	R		C_INTERRUPT_IS_EDGE	0
		3	C_EDGE_IS_POSITIVE	1
		-	C_PSL_LINKS	0
		Del	C_FSL_DATA_SIZE	32
			C_USE_ICACHE	0
			C_ALLOW_ICACHE_WR	1
			C_ADDR_TAG_BITS	7
			C_CACHE_BVTE_SIZE	8192

- Select the Parameters tab
- Select microblaze_0 IP instance
- Select the
 - C_DEBUG_ENABLED and C_NUMBER_OF_PC_BRK parameters and click the Add button
- Edit the parameter values

Setting MDM Parameters

	a to re-instance in mina	Lho	ose in instance for a	list of parameters.
ese parameter values	will override default MPD valu	es. opt	o_mdm_0	 Open PDF Doc
arameter	Value	Para	meters with default v ose one or more (usin	values from MPD. Ig shift and ctrl) and click Add.
_USE_UART	0	Par	ameter	Value
		C 🖸	MB_DBG_PORTS	1
		X U	USE_UART	1
			JART_WIDTH	32
				v

- Select the opb_mdm_0 IP instance
- Select the C_USE_UART parameter and click the Add button
- Edit the parameter value



Setting LMB Parameters



- Select the Imb_v10_0 IP instance
- Select the C_EXT_RESET_HIGH parameter and click the Add button
- Edit the parameter value
- Repeat for Imb_v10_1
 IP instance



Setting OPB Parameters

dit Parameters assigned to I	PInstance in MHS	Choose IP Instance for	a list of parameters.
hese parameter values will o	venide default MPD values.	opb_v20_0	Dpen PDF D
Parameter	Value	Parameters with defaul Choose one or more (u	values from MPD. ing shift and ctil] and click Add.
C_EXT_RESET_HDGH	0	Parameter	Value
		C_USE_LUT_OR	1
		C_EXT_RESET_HIGH	1
		C_DYNAM_PRIORITY	0
		<< Add C_PARK	0
D		C_PROC_INTRFCE	0
98		C_REG_GRANTS	1
		C_DEV_BLK_ID	0
		C_DEV_MIR_ENABLE	0
		C_BASEADOR	0xFFFFFFFF
		Del C_HIGHAODR	0x00000000

- Select the opb_v20_0 IP instance
- Select the C_EXT_RESET_HIGH parameter and click the Add button
- Edit the parameter value



Connecting The Clock



Select the Ports tab

 Select the LBM_Clk and OPB_Clk ports on the lmb_v10_0, lmb_v10_0 and opb_v20_0 IP instances and click the Add button



Connecting The Reset

ports. The "	Use Add Port for Range'' column fr	external ports that ne or external ports is giv	ed to be en as "[]	GND or VCC. LB:UBJ" (for e.	g.,[D:31])		List of Ports. Select one or more ports and Click Add	-
instance	Port Name	Net Name	Туре	Kind	Range			_
nb_v10_0	LMB_Clk	sys_dk	Input	External 💌			Dbg_Update_5	-
xpb_v20_0	OPB_Clk	sys_dk	Input	External 💌			Dbg_TDI_6	
nb_v10_1	UMB_CIk	sys_ck	Input	External 💌		<< Add	Dbg_TDO_6 Dbg_Beg_Ep_6	
mb_v10_0	SYS_Rst	Inb_v10_0_5	Input	External 💌			Dbg_Capture_6	
pb_v20_0	SVS_Rst	opb_v20_0_5	Input	External 💌		Add Port	Dbg_Update_6 Dbg_Dk_7	
mb_v10_1	SYS_Rst	imb_v10_1_5	Input	External 💌	1		Dbg_TDI_7	
			ſ	જે		Del	bog_bpase_7 Imb_bram_if_onth_1 LMB_Cik Imb_v10_0 LMB_Cik SYS_Bat opb_v20_0 OPB_Cik SYS_Bat Imb_v10_1 LMB_Cik	

 Select the SYS_Rst ports of the lmb_v10_0, lmb_v10_1, and opb_v20_0 IP instances and click the Add button



Connecting The Reset (2)

Use c ports. The '1	til and shift form Use Add Port for Range'' column f	ultiple row selection external ports that or external ports is	is and click need to be given as "]	Connect to GND or VO LB:UBJ" (for	connect C. e.g.,[0:31])		List of Ports. Select one or	•
Instance	Port Name	Net Name	Туре	Kind	Range		nore ports and cites. Add	
mb_v10_0	LMB_Clk	sys_dk	Input	External	*		Dbg_Update_5	*
opb_v20_0	OPB_Clk	sys_dk	Input	External	•		Dbg_TDI_6	
mb_v10_1	LMB_Clk	sys_dk	Input	External	•	<< Add	Dbg_TDD_6	
mb_v10_0	SYS_Rst	sys_rst	Input	External	•		Dbg_Capture_6	
opb_v20_0	SYS_Ret	sys_rst	Input	External	•	Add Port	Dbg_Update_6	
mb_v10_1	5YS_Rst	sys_rst	Input	External	•		Dbg_LK_7 Dbg_TDI 7	
		ß				Connect	Inb_bran_it_onkt_1 LMB_CIk Imb_v10_0 LMB_CIk SYS_Rut opb_v20_0 0FB_CIk SYS_Rat Imb_v10_1 LMB_CIL	

- Select the Imb_v10_0, Imb_v10_1, and opb_v20_0 instances on the left side and click the Connect button
- Enter sys_rst for the net name in the dialog box and click the OK button



Completing the Add/Edit

• Click the OK button to set all the items changed in the Add/Edit Cores dialog box





Adding I/O Peripherals to a System

Adding A UartLite



- Select the Project menu
- Select the Add/Edit Cores submenu
- Select the opb_uartlite and click Add
- Edit the Base Address and High Address of the uartlite



Setting UartLite Parameters

here parameter values	will overvide default MFD values.		spb_uarMe_0	Open PDF Do
bewoodse	Vake	-	Paraveters with default Choose one or nore (an	inguishilf and citil and click Add
CON JUNE O	24,000,000	- Carel	Parameter	Value
Just Peetre		- c	C_DATA_BETS	8
			C.C.K. MREQ	125_000_000
			C_BAUDRATE	9600
		(r.Add	C_LISE_PARITY	1
	N		C_000_PARITY	
	-45			
		_		
		*		
		Det		
		1		

- Select the Parameters tab
- Select the C_CLK_FREQ and C_USE_PARITY parameters and click the Add button
- Edit the parameter values



Put UartLite On the OPB

ight click on any bus inst	ALC:N	icol.	mni	ave (M. S. MS) connections. La contest menu	buses and click A	dd. dd
	1	(S)	(a)		do: v29.v1.00	0
nih hann if anih ITalah	14	2.0	7.4	-	mb_v10_v1_00_b	
neroblaze 0 dmb	М			(¢ Ac	H 000_V20_V1_10 000_V20_V1_10	6
nicioblaza 0 ilvih		-	14		pb_v34_v1_01_	4.
nicrobiaze 0 doph		M	-	Choose the BRAM	port to connect to the con	vholler port.
victoblaze 0 inpb		м		Dive a name to the	CONNECTION.	
date 0 mbm date		1	-	Critit Port	BRAM Port	Connector
mb bram # colt 1 dmb			ŧ.	Into_brani_if_c	bram_block_0 PORTA	• conn_0
done 0 withit da		T		THE DUMP TO A	The track of the second	Tranci
				D2		

- Select the Bus Connections tab
- Click on the box to mark the UartLite as a slave on the OPB bus



Connecting UartLite I/O



- Select the Ports tab
 - Select the RX and TX ports of opb_uartlite_0 and click the Add button
- Edit the net names to be tx and rx for the opb_uartlite_0
- Click OK on the dialog box



Adding A Timer

To dele	te peripheral	grounds can be a c, choose one or n	nore rows and d	lick Delete.			Choose one or more IPs and shift for multiple select	use ctrl stions)
Peripheral	HW Ver	Instance	Base Address	High Address	Min		from the list below and cl	ick Add.
ram_block	1.00.a 💌	bram_block_0					opb2plb_bridge	2
nb_bran_i	1.00.b 💌	Imb_bram_if	0.<00000000	0x0000FFFF	0::80		opb_arbiter	
esaldoroin	2.00.a 💌	microblaze_0					opb_bran_it_cn/it	
pb_mdm	1.00.c 💌	opb_mdm_0	0.4ffff8000	0xffff80ff	0		opb_central_dma	
nb_bran_i	1.00.b 💌	lmb_bram_if	0.00000000	0x0000FFFF	0x80	Lucia I	opb_emc	
pb_uartike	1.00.b 💌	opb_uartite_0	0:dFFFF8100	0xFFFF81FF		<< Add	opb_ethemet	
pb_timer	1.00.b 💌	opb_timer_0	0xCD000000	0xC00001FF	0x10		opb_gpio	
						Delete >>	opb_memicon opb_opb_ite opb_pci opb_tran opb_spi opb_trans opb_trans opb_trans opb_trans opb_trans opb_trans opb_trans opb_trans opb_trans opb_trans	

- Select the Project menu
- Select the Add/Edit Cores submenu
- Select the opb_timer and click Add
- Edit the Base Address and High Address of the timer



Setting Timer Parameters

dit Parameters assigned to I	P Instance in MHS		Choose IP Instance for a k	st of parameters.
hese parameter values will o	venide default MPD values.		opb_timer_0	 Open PDF Do
Parameter	Value	-	Parameters with default va Choose one or more (using	lues from MPD. shift and ct() and click Add.
CONE_TIMER_ONLY	1		Parameter	Value
		c .	C_COUNT_WIDTH	32
		- X	C_ONE_TIMER_ONLY	0
			C_TRIGO_ASSERT	1
		<< Add	C_TRIG1_ASSERT	1
			C_GEN0_ASSERT	1
			C_GEN1_ASSERT	1
		Del		

- Select the Parameters tab
- Select the C_ONE_TIMER_ONLY parameters and click the Add button
- Edit the parameter value



Put Timer On the OPB

ld/Edit Hardware Platfor	m 5	peci	fica	tions					×
Peripherals Bus Connection	nı	Parts		Parameters					
Click on squares to make n Right click on any bus inste	noe	r, sla [cok	we o ann l	r master-slave (M. S. MS) con header) for a context menu.	nections.		Choose one or more buses and click Add	(using shiftend Ctrl)	
	/4	13/1		2) 2)			dor v29 v1 00 a fd_v20 v1 00 b		
Imb_bram_it_ontit_0 simb	9				ee Add	1	opb_v20_v1_10_a		
microblaze_0 dmb	м						opb_v20_v1_10_b		
microblaze_0 ilmb			м						
microblaze_0 dopb		м			Shoose the BRAM p Sive a name to the o	on to conne	connect to the contri-	oller port.	
microblaze_0 iopb		м			Cotik Doct	DD A	M Dout	Concertor	-
opb_mdm_0 copb		0			Inde brand if c	bran	block 0 PORTA	comp 0	
imb_bram_if_cntit_1 simb			I		lmb_bram_f_c	bran	block_0 PORTB	conn_1	
opb_uartite_0 sopb		0							
opb_timer_0 sopb		2							
			_		DK]	Cancel	Apply Hel	,

- Select the Bus Connections tab
- Click on the box to mark the Timer as a slave on the OPB bus



Adding an Interrupt Controller

To delet	le peripher	als, choose one or r	nore rows and d	lick Delete.			Choose one or more IPs (u and shift for multiple select	te ctrl ons)
eripheral	HW Ver	Instance	Base Address	High Address	Min		from the list below and clic	Add.
ram_block	1.00.a	bram_block_0			_		opb2plb_bridge	
nb_bran_i	1.00.b	http://www.if	0:000000000	0x0000FFFF	0::80		opb_arbiter	
sicroblaze	2.00.a	microblaze_0					opb_bram_il_cn/it	
pb_mdm	1.00.c	opb_mdm_0	0.dffff8000	0xffff80ff	0		opb_central_dma	
nb_bran_i	1.00.b	mb_bran_if	0x00000000	0x0000FFFF	0::80	Local I	apb_ema	
pb_uartike	1.00.b	opb_uartite_0	0xFFFF8100	0xFFFF81FF		<< Add	opb_ethemet	
pb_timer	1.00.b	opb_timer_0	0x00000000	0xC00001FF	0x10		opb_gpio	
pb_intc	1.00.c	opb_intc_0	0.480000000	0x800001FF	0x20		apb_hdla	
						Delete >> 1	opp_mom opb_remecon opb_opb_ke opb_pci opb_stram opb_stram opb_stram opb_trame opb_trame opb_trame opb_warfile opb_warfile opb_warfile opb_otridge plb_opb_bridge plb_otridge	

- Select the Project menu
- Select the Add/Edit Cores submenu
- Select the opb_intc and click Add
- Edit the Base Address and High Address of the intc



Put Intc On the OPB

Implementation Implementation b_bram_it_cmtb_0 simb M cooblaze_0 dmb M cooblaze_0 dopb M cooblaze_0 dopb M cooblaze_0 dopb M cooblaze_0 impb M b_bram_it_onth_1 simb % b_cuartite_0 sopb % b_cuartite_0 sopb % b_cimet_0 ropb % b_cimet_0 ropb %	ight click on any buz insta	/2	(cok	ann l	nieot menu.	buses and click Av dcr v29 v1 00	a a
cooblase_0 dimb M cooblase_0 dimb M cooblase_0 dimb M cooblase_0 dipb Potention contraction M cooblase_0 dipb Potention contraction M contraction M contraction Potention contraction Potention contrelion Potention	mb bram if critir 0 simb	2	10	7.9		Inb v10 v1 00	a
cooblaze_0 limb M cooblaze_0 dopb M cooblaze_0 lipb M cooblaze_0 lipb M cooblaze_0 lipb M cooblaze_0 lipb M b_mdm_0 sopb 0 b_bram_f_onth_1 simb 6 b_uartite_0 sopb 0 b_binmc_0 coopb 0 b_binmc_0 sopb 0	nicroblaze 0 dmb	м				dd opb_v20_v1_10_	b
cooblaze_0 dopb M cooblaze_0 lopb M cooblaze_0 lopb M do_mdm_0 sopb 0 b_bram_f_onth_1 simb 0 b_bram_f_onth_1 simb 0 b_bram_f_onth_1 simb 0 b_bram_f_onth_1 simb 0 b_inte_0 sopb 0 b_inte_0 sopb 0 b_inte_0 sopb 0	nicroblaze 0 ilmb			м		pb_v34_v1_01_4	5
cioblaze_0 lopb M bi_bram_f_on/t_1 simb 6 bi_scartite_0 sopb 6 bi_inte_0 sopb 6 bi_inte_0 sopb 7	nicroblaze 0 dopb		м		Choose the BRAM	port to connect to the con	itolier port.
bi_mdm_0 sopb 0 bi_mdm_0 sopb 0 bi_bram_it_ontir_1 simb 0 bi_warilie_0 sopb 0 bi_intc_0 sopb 0 bi_intc_0 sopb 0	sicroblaze 0 joob		м		Give a name to th	e connection.	
b_bram_if_critt_1 simb 6 b_uardite_0 sopb 6 b_inne_0 sopb 6 b_inne_0 sopb 7	ob mdm 0 sopb		0		Cntir Port	BRAM Port	Connector
b_userbite_0 topb	nh beam if centr 1 slmb			5	Imb_bram_P_c	bram_block_0 PORTA	conn_0
المراجعة الم المراجعة المراجعة الم	nh watite (Looph		8	-	mo_bram_r_c	_bran_bibb(_0 Pokils)	CourCt
b_intc_0 topb *	ob timer () conb		8				
and and a state of the state of	ob into 0 conb		5				

- Select the Bus Connections tab
- Click on the box to mark the Intc as a slave on the OPB bus



Connect Intc to MicroBlaze

/Edit Hardwa	re Platform S	pecifications Ports Parameters	1					
Port Sig Use ch ports. U The "R	nal Assignment and shift for mu loe Add Port for ange" column fr	s. Itiple row selections external ports that no or external ports is given	and click sed to be ven as "]	Connect to c GND or VCC. LB:UB)'' (for e	onnect g.(0:31))			Filter substring or instance List of Ports. Select one or
Instance	Port Name	Net Name	Pola	Scope	Range	Clas		more ports and Click Add
opb_uartik	RX	rx .	IN	External 💌		-		RX A
opb_uartlit	TX	bx	OUT	External 💌	1			
mb_v10_0	SYS_Rst	sys_rst	IN	External 💌			<< Add	opb_timer_0
lmb_v10_0	LMB_Clk	sys_dk	IN	External 💌		CLK		Interrupt
opb_v20_0	OPB_Clk	sys_dk	IN	External 💌		CLK	Add Port	Freeze
opb_v20_0	SVS_Rst	sys_rst	IN	External 💌				CaptureTrig1
lmb_v10_1	SVS_Rst	sys_rst	IN	External 💌				GenerateOut0
mb_v10_1	LMB_Ck	sys_ck	IN	External 💌	1	CLK		PwM0
microblaze_0	INTERRUPT	microblaze_0	. IN	External 💌		INTE		and inter 0
opb_intc_0	Irq	opb_intc_0_Irq	OUT	External 💌		INTE	Del	OP8_Cik
		Port Connec	tions	- Internal	an 💌	1		Intr Irq
		Net name to	ouse				Connect	Imb_v10_0
		intc_output	:					SYS_Rst
		Make Sel	ected F	orts				opb_v20_0 OPB_Clk SYCs.ext
				ancenide				Imb v10 1
•		Leave Net only the "In	name b iternal''	olank for cha or ''External	nging '' field.	Ľ		LMB_ČK
			ОК	Can	cel	E	Car	ncel Apply Help

- Select Ports tab
- Add Interrupt input of MicroBlaze and Irq output of Intc
- Highlight both of these and press
 Connect
- Name the net, select Internal, and press OK



Implementing Processor Systems on Xilinx FPGAs

Connect Intc to MicroBlaze



Connect Timer to Intc

- Port Sid	gnal Assignments	L					-		Filter substring or instance	
X Use ch	and shift for mu	tiple row selections -	and click	Connect to GND or VI		nnect			_	-
The "R	lange" column fo	or external ports is given	ren as "[l	LB:UB]" (fo	re.g	p.,[0:31])			List of Ports. Select one or	
Instance	Port Name	Net Name	Pola	Scope		Range	Clas		more ports and Click Add	
opb_uartit	RX	FX.	IN	External	•				RX	*
opb_uartiit	TX	tx	OUT	External	•					
mb_v10_0	SYS_Rst	sys_rst	IN	External	٠			<< Add	opb_timer_0	
mb_v10_0	LMB_Ck	sys_clk	IN	External	•		СLК		Interrupt	
opb_v20_0	OPB_Clk	sys_clk	IN	External	•		СK	Add Port	Freeze	
opb_v20_0	SYS_Rst	sys_rst	IN	External	-				CaptureTrig1	
imb_v10_1	SYS_Rst	sys_rst	IN	External	+				GenerateDut0	
mb_v10_1	LMB_Clk	sys_dk	IN	External	٠		СLК		PwM0	
microblaze_0	INTERRUPT	intc_output	IN	Internal	٠		INTE			
opb_intc_0	Irq	intc_output	OUT	Internal	٠		INTE	Del	opb_intc_0 OPB_Clk	
opb_intc_0	Intr	opb_intc_0_Intr	IN	External	•		INTE	Dei	Intr	
opb_timer_0	Interrupt	opb_timer_0	OUT	External	•		INTE		Irq	
		Port Con	nectio	ons - Int	eri	nal an.	. <u>×</u>	Connect	Imb_v10_0 LMB_CIk	
		Net nan	ne to us	e					512_Ha	- 11
		timer in	h		_				opb_v20_0	
		[unoi_ii							SYS_Rat	
		Make	Selecte	ed Ports-						
		O E	xternal	 Int 	ern	al			LMB Ck	-
•				.		?			•	2
		L esue	Notnor	ne blank i	for	chanci	0.0			
		only the	e "Interr	hal" or "E	xte	rnal'' fie	eld.	Car	cel Apply	He/p

- Select Ports tab
- Add Interrupt output of Timer and Intr input of Intc
- Highlight both of these and press
 Connect
- Name the net, select Internal,

and press OK
Connect Timer to Intc

Port Sig Use ctr ports. U	gnal Assignments I and shift for mu Jse Add Port for	s. Itiple row selection external ports that	s and click need to be	Connect to co GND or VCC.	nnect]		Filter substring or instance
Instance	Port Name	Net Name	Pola	Scope	Range	Clas		List of Ports. Select one or more ports and Click Add
opb_uartlit	RX	rx	IN	External 💌				BX A
opb_uartlit	тх	tx	OUT	External 💌				-10
lmb_v10_0	SYS_Rst	sys_rst	IN	External 💌			<< Add	opb_timer_0
lmb_v10_0	LMB_Clk	sys_clk	IN	External 💌		CLK		Interrupt
opb_v20_0	OPB_Clk	sys_clk	IN	External 💌		CLK	Add Port	Freeze
opb_v20_0	SYS_Rst	sys_rst	IN	External 💌				CaptureTrig1
lmb_v10_1	SYS_Rst	sys_rst	IN	External 💌				GenerateOut0
lmb_v10_1	LMB_Clk	sys_clk	IN	External 💌		CLK		EienerateUut1 PWM0
microblaze_0	INTERRUPT	intc_output	IN	Internal 💌		INTE		
opb_intc_0	Irq	intc_output	OUT	Internal 💌		INTE		OPB_CIk
opb_intc_0	Intr	timer_int	IN	Internal 💌		INTE	Dei	Intr
opb_timer_0	Interrupt	timer_int	ОЛТ	Internal 💌		INTE		Irq
							Connect	Imb_v10_0 LMB_Clk SYS_Rst opb_v20_0 OPB_Clk
•						Þ		Imb_v10_1 LMB_Clk





Software Development with the EDK and XPS

EDK System Design Comprehensive Tool Chain



Building Software in XPS

- XPS is an Integrated Development Environment (IDE) similar to other products with the primary difference being it allows the user to build hardware and software.
- The GNU tools (compiler, linker, etc.) including GDB are used by XPS for software development.
- The GNU tools are not native Windows tools such that they execute within a Xygwin (Xilinx Cygwin) environment.



XPS Project Directory Structure

- A project within XPS is a directory that contains multiple subdirectories.
- The *code* subdirectory is created by the user and contains application source code.
- The include files, drivers, and libraries are located in a directory based on the instance name of the microprocessor in the project.

XPS Example Project Directory

→ Back → → → → → → → → → → → → → → → → → → →	Back • → • ⊡ ② Search Polders ③ History ○ ▲ ▲ ▲ ▲ ▲ ▲ ▲ ▲ ▲ ▲ ▲ ▲ ▲ ▲ ▲ ▲ ▲ ▲ ▲
dress C:\EDK\d C:\EDK\d C Go	tress C:\EDK\d ers Insight_VII_MicroBlaze_System1e Image: Image
ders Insight_VII_MicroBlaze_System1e Image: Insight_How Image: Insight_How Image: Insight_H	ers
Insight_VII_MicroBlaze_System1e Insight_VII_MicroBlaze_System1e Insight_VII_MicroBlaze_System1e	 Insight_VII_MicroBlaze_System1e ⊥→□
	iast/s) (Disk from senses 1.07 CP)

Library Generation

- XPS compiles device drivers and C run-time CRT into a single library that is then linked with a user application program.
- Libgen is the tool executed from within XPS or from a command line, that copies the library source files and device driver source files to the project directory to build the library.



Command Line Builds

- XPS generates a single make file, system.make, that can be used to build the hardware or software from the command line of a Xygwin window.
- This make file could be used to create a build environment for software development groups that build from the command line.



Xilinx Microprocessor Debugger (XMD)

- Interfaces GDB to a "target"
- Supports script interface for built-in commands

XMD

- Allows debug with or without a ROM monitor
 - MDM target for true JTAG

GDB

- UART target for ROM monitor (xmd-stub)
- SIM target for instruction set simulator

UAR1

SIN

MDN

Microprocessor Debug Module (MDM)

- JTAG debug using BSCAN
- Software non-intrusive debugging
- Read/Write access to internal registers
- Access to all addressable memory
- Hardware single-stepping
- Hardware breakpoints configurable (max 16)
- Hardware read/write address/data watchpoints
 - configurable (max 8)





Device Drivers & Software Infrastructure

Device Drivers for FPGA Designs

- Hardware is parameterizable
 - Capabilities and features may change every build
- FPGA space is limited
 - User needs flexible driver architecture
 - Internal memory solution as well as external memory solution
- Processor may change
 - Portability of driver is key



Device Driver Goals

- Portability/Reusability
 - Drivers are to be portable across many different RTOSs, microprocessors, and toolsets
 - Minimize development effort
- Out-of-the-box solution for customers



Driver Design Considerations

Programming Languages

- Assembly Language
 - Minimized to allow maximum portability between microprocessors
 - Only boot code, which executes prior to the C/C++ runtime system, is typically necessary to be assembly language
 - Located in separate source files to help isolate it, as opposed to in-line assembly language in the C source code
- C Programming Language
 - The C programming language is the most utilized language for embedded systems
 - In order to support the largest number of customers, the first implementation utilizes the C programming language



Driver Design Considerations

- Object Oriented Design
 - Emphasize data abstraction, data hiding, and encapsulation in addition to greater potential for code reuse and ease of maintenance
 - Provides an easier transition from non-object oriented languages such as C to more object oriented languages such as C++ and Java
- Delivery Format
 - Delivered to customers in source code format, allowing it to be built and optimized for a wide range of microprocessors using customer selected tools



Device Driver Architecture

- Component based object oriented design implemented in ANSI C
- A device driver supports multiple instances of a device
- Layered device driver architecture to allow user selectable features and size

Layer 2 Drivers (RTOS Adapters)

Layer 1 (High Level) Drivers

Layer 0 (Low Level) Drivers



Device Driver Architecture (continued)

- Source code is provided
- Layer 0 and Layer 1 are OS-independent
- Device drivers in all layers have common characteristics
- Primitive data types for portability (Xuint8, Xuint16, Xuint32, etc.), in xbasic_types.h
- Isolation of I/O accesses for portability (XIo_In8(), XIo_Out8(), etc.), in xio.h
- Coding and documentation conventions

Layer 0, Low Level Drivers

- Interface contained in <driver>_I.h file
- Designed for a small system, typically for internal memory of an FPGA.
- Small memory footprint
- No error checking performed
- Supports primary device features only, not comprehensive
- Polled I/O, blocking functions



Layer 1, High Level Drivers

- Interface contained in <driver>.h file
- Designed to allow a developer to utilize all features of a device
- Larger memory footprint
- Robust error checking such as asserting input arguments
- Supports configuration parameters in <driver>_g.c
- Interrupt driven I/O, non-blocking functions
 - Interrupt service routines are provided



Layer 2 Drivers, RTOS Adapters

- Interface contained in <driver>_adapter.h file.
- Converts the Layer 1 device driver interface to an interface that matches the device driver scheme of the RTOS.
- Contains calls specific to the RTOS
- Can use RTOS features such as memory management, threading, inter-task communication, etc.



RTOS Independent Device Drivers

- Driver
 - Responsible for interfacing to the device (peripheral).
 It encapsulates communication to the device
 - Designed to be portable across processor architectures and operating systems
- Adapter
 - Integrates the driver into an operating system
 - Satisfies the "plug-in" requirements of the operating system
 - Needs to be rewritten for each OS

RTOS Support

- Xilinx supports VxWorks 5.4/5.5 for PowerPC in-house
- 3rd party support includes MontaVista Linux (PPC), ATI Nucleus, uCos, ucLinux
- VxWorks integration:
 - All device drivers can be used directly by the application
 - Some device drivers tightly integrated into VxWorks
 - UARTs to standard and file I/O
 - Ethernet to network stack (Enhanced Network Driver)
 - Interrupt controller
 - System ACE into VxWorks filesystem interface
- Automatic Tornado BSP generation using EDK



Naming Conventions

- A common name is used for all external identifiers of the device driver
 - <driver_name>_FunctionName();
 - <driver_name>_DataType;
- A common name is used for all source files of the device driver for ease of use
 - <driver_name>_l.h low level driver interface definition
 - <driver_name>.h high level driver interface definition
 - <driver_name>.c primary source file
 - <driver_name>_g.c configuration table source file
 - <driver_name>_intr.c interrupt processing source file



Multiple Instance Details

- Multiple instances of a single device (such as an Ethernet MAC) typically exist in a system
- A single device driver handles all instances of the device
- A layer 1 device driver uses a data type that is passed as the first argument to each function of the driver. The data type contains information about each device instance such as the base address



Example Layer 0 Device Driver API

- Each function of Layer 0 uses the base address of the device as the first argument
- No state information is kept by the driver and the user must manage multiple instances of the device
- void XEmac_mSetControlReg(Xuint32 BaseAddress, Xuint32 Mask)
- void XEmac_SendFrame(Xuint32 BaseAddress, Xuint8 *FramePtr, int Size)
- int XEmac_RecvFrame(Xuint32 BaseAddress, Xuint8 *FramePtr)



Example Layer 1 Device Driver API

- Each function of Layer 1 uses an instance pointer as the first argument
- XStatus XEmac_Initialize(XEmac *InstancePtr, Xuint16 DeviceId)
- XStatus XEmac_Start(XEmac *InstancePtr)
- XStatus XEmac_Stop(XEmac *InstancePtr)
- void XEmac_Reset(XEmac *InstancePtr)
- XStatus XEmac_SelfTest(XEmac *InstancePtr)



Example Layer 1 Device Driver API For FIFO Interrupt Support

- XStatus XEmac_FifoSend(XEmac *InstancePtr, Xuint8 *BufPtr, Xuint32 ByteCount);
- XStatus XEmac_FifoRecv(XEmac *InstancePtr, Xuint8 *BufPtr, Xuint32 *ByteCountPtr);
- void XEmac_SetFifoRecvHandler(XEmac *InstancePtr, void *CallBackRef, XEmac_FifoHandler FuncPtr);
- void XEmac_SetFifoSendHandler(XEmac *InstancePtr, void *CallBackRef, XEmac_FifoHandler FuncPtr);
- void XEmac_IntrHandlerFifo(void *InstancePtr); /* interrupt handler */



Device Driver & System Configuration

- xparameters.h contains important system parameters used by the drivers & the BSP
- Parameters for each device may include a device ID, a base address, an interrupt identifier, and any device unique parameters
- This file is the best place to start when trying to understand a system
- Libgen automatically generates xparameters.h



Example xparameters.h File

• The following example is for a system that has an Ethernet MAC device at address 0x60000000. The name of the hardware instance is opb_ethernet.

#define XPAR_XEMAC_NUM_INSTANCES 1
#define XPAR_OPB_ETHERNET_BASEADDR 0x6000000
#define XPAR_OPB_ETHERNET_HIGHADDR 0x60003FFF
#define XPAR_OPB_ETHERNET_DEVICE_ID 0
#define XPAR_OPB_ETHERNET_ERR_COUNT_EXIST 1
#define XPAR_OPB_ETHERNET_DMA_PRESENT 3
#define XPAR_OPB_ETHERNET_MII_EXIST 1



Device Driver Configuration Specifics

- Constants describing each device instance are contained in xparameters.h
- These constants are also inserted into a configuration table for each device driver contained in the <driver_name>_g.c
- The device driver looks up information for the specific instance of the device when it is initialized
- The data type definition for the configuration data is contained in the <driver_name>.h file
- Libgen generates the <driver_name>_g.c file for each device driver

Device Driver Configuration Example

• From xemac.h source file	typedef struct {
• From xemac_g.c source file	XEmac_Config XEmac_ConfigTable[] = { {
	}

Implementing Processor Systems on Xilinx FPGAs

Interrupt Processing

- Layer 1 device drivers provide interrupt driven I/O
- The device driver provides an interrupt handler that must be connected to the interrupt source by the application
- The device driver interrupt handler performs device specific details such as register reads & writes and calls a user specified handler (or callback) to process events and data
- The user application must setup the application callback to be called by the device driver interrupt handler
- The application callback can perform processing in an interrupt context or signal non-interrupt driven processing to perform the processing



Interrupt Processing Example

- When using MicroBlaze, XPS automatically connects the interrupt controller in the system to the exception vector of the processor so that the interrupt controller's interrupt handler gets called when an interrupt occurs
- The following code illustrates setting an application callback for the Ethernet MAC device driver, connecting the Ethernet MAC device driver interrupt handler to the interrupt controller, and enabling the MicroBlaze interrupts

XEmac_SetFifoSendHandler(InstancePtr, InstancePtr, SendHandler); XEmac_SetFifoRecvHandler(InstancePtr, InstancePtr, RecvHandler);

microblaze_enable_interrupts();

Error Processing

- Device driver functions which detect errors return a data type of XStatus to indicate the detailed error condition
- The error details are contained in xstatus.h
- Device driver functions use asserts to indicate errors during runtime.
- Errors detected during interrupt processing are returned to the application via an asynchronous callback function.



Assert Details

- Device drivers use Assert to validate input arguments
- The default is for asserts to be used by device drivers
- Asserts can be disabled when the libraries are generated by using the -DNDEBUG symbol
- The default behavior of Assert is to loop forever after calling a user defined function if defined
- The user can setup a function to be called when an assert is called.

void XAssertSetCallback(XAssertCallback Routine);

Device Drivers In The EDK Install Directory

This illustration shows the directories of a device driver (emac) in the EDK install area.

Note that there is an examples directory that contains example source files for using a device driver.


Writing An Application To Use A Device Driver

- Always start with an example provided in the device driver directory of the EDK install to save time
- Choose the example best fits your application, such as polled, interrupts, or DMA, and copy code snippets from the example

