Design Tools & Core Library

Or, *Brace yourself for deluge of facts.*

Outline for Today

- Pre-Made Peripherals Cores
- Online Demo
  - adding a core in XPS
  - command-line tools
- System Architectures

System Cores

- Platform FPGA assemble systems from cores such as
  - processors
  - buses
  - memory (controllers and on-chip memory)
  - peripherals/devices

Where Do Cores Come From?

- cores can be implemented in CMOS or CLBs
  - diffused (implemented in CMOS, license comes with chip)
  - pre-defined (provided by Xilinx)
  - third-party (buy license from vendor)
  - open source
  - custom

IBM CoreConnect

- Xilinx Development Kit for Platform FPGAs heavily uses a collection of cores from IBM
- designed for System-on-Chip (SoC)
  - many of the cores have been reimplemented for CLBs
  - use a subset of the features
- WARNING: acronym soup coming...

Processor Cores

- ***PPC405*** – PowerPC processor; diffused IP
- ***microblaze*** – Microblaze soft processor (implemented in CLBs)

Bus Cores

- ***PLB*** – Processor Local Bus (for PPC)
- ***LMB*** – Local Memory Bus (for Microblaze)
- ***OPB*** – On-Chip Peripheral Bus
- ***FSL*** – Fast Simplex Link Bus
- ***DCR*** – Device Control Register (a bus)
- ***OCM*** – On-Chip Memory Bus

Why So Many Bus Cores?

- different characteristics and trade-offs
  - bandwidth and latency
  - size (in CLBs)
  - number of masters and slaves
- different intended uses

PLB

- 64-bit data, 100 MHz full-featured bus
- + 1600 MB/s peak bandwidth (533 MB/s typical)
- + multiple masters/slaves
- - complex interface
- - requires a relatively large amount of CLBs

LMB

- 32-bit data, 125 MHz processor/memory bus
- + 500 MB/s peak bandwidth (333 MB/s typical)
- - one master, one slave
- + requires a relatively few CLBs

OPB

- 32-bit data, 100 MHz "compromise" bus
- + 500 MB/s peak bandwidth (167 MB/s typical)
- + simple interface
- + requires fewer CLBs than PLB

FSL

- 32-bit data, 100 MHz point-to-point bus
- + fast, FIFO interface
- + direct connection to Microblaze
- + low CLB usage

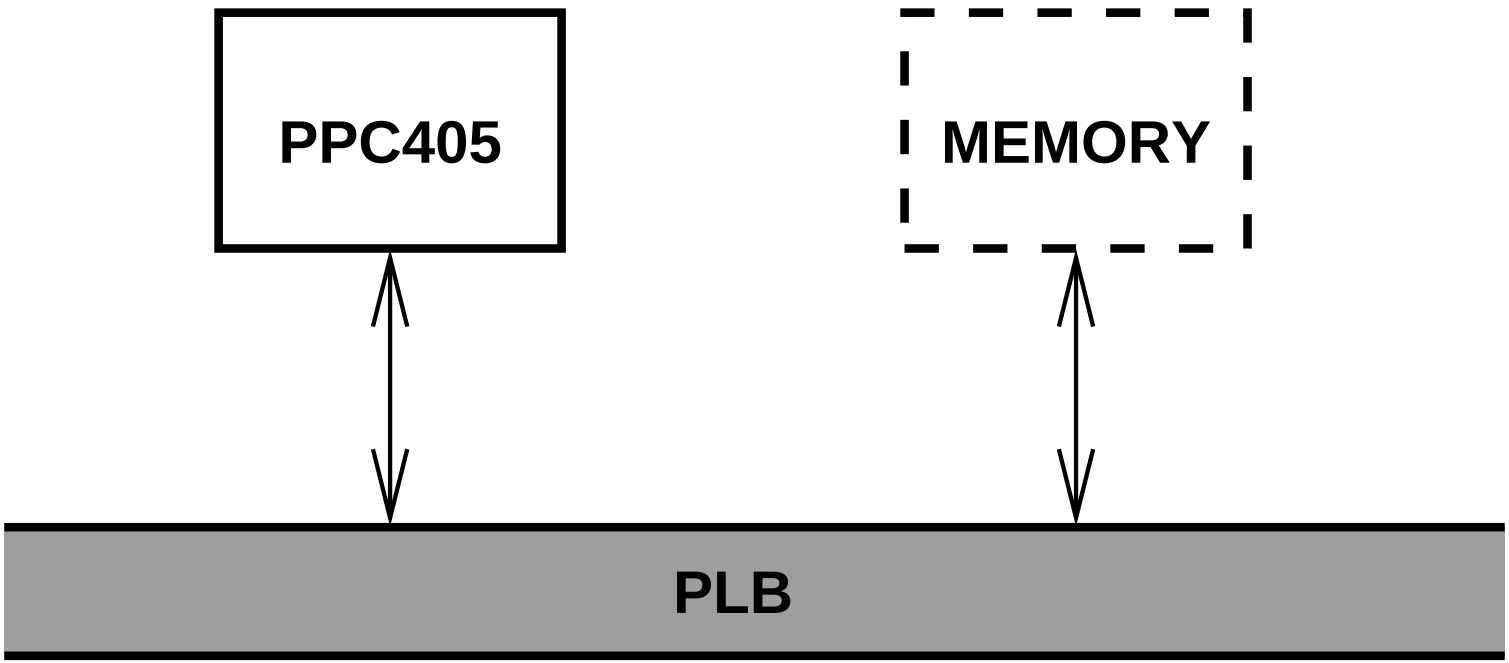
DCR

- 32-bit data but only 10-bit address special-purpose bus
- provides simple interface for low-bandwidth communication between processor and peripherals
- avoids taking cycles from high-speed busses like PLB
- very low CLBs usage

OCM

- 32-bit data, memory bus
- comes in two flavors (D and I) for data and instruction requests
- + 500 MB/s typical bandwidth
- + very low latency
- + directly connects PPC with Block RAMs
- provides predictable memory performance for a range of address spaces

Simple Process/Memory System



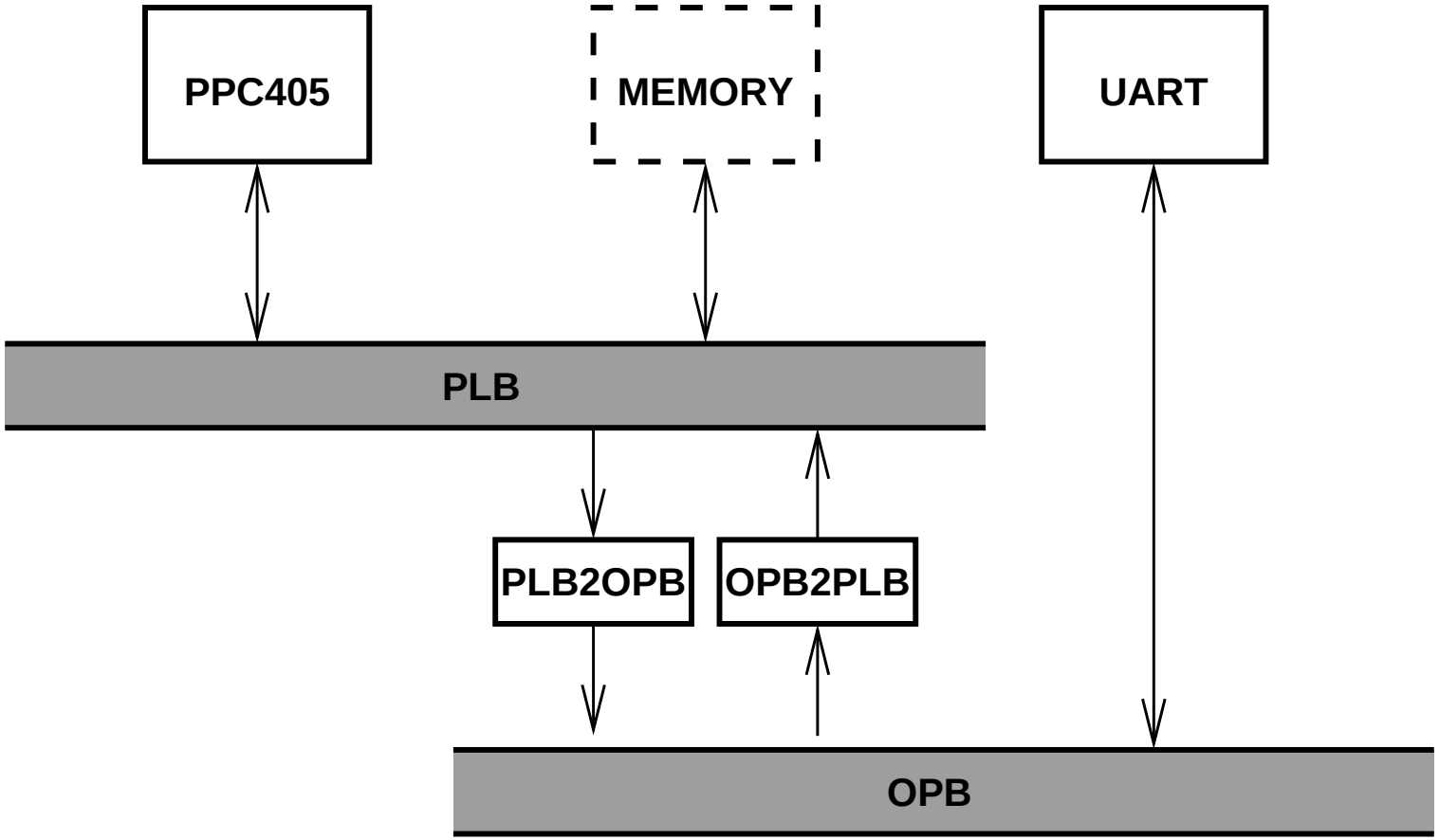
- let memory be abstract for the moment

Expanding Simple System

- if we were to add a simple peripheral (say a UART for a serial port)
  - peripheral does not need all the PLB features
  - PLB interface takes a relatively large number of CLBs
  - PLB is limited in the number of peripherals

  thus adding an OPB is a good solution

Adding OPB to Process/Memory System



Bridges

- notice that this design introduces two more cores
  - PLB2OPB bridge — slave on PLB to translate PLB requests to OPB requests
  - OPB2PLB bridge — master on PLB to translate OPB requests to PLB requests
- as the number of on-chip peripherals rise, this organization offers space savings and performance gains

Other Bridges

- *OPB2DCR*
- *OPB_OPB*
- *OPB_PCI*

Bus and Bus Arbiter

- some buses have an arbiter built-in to the core
- for others it comes in two cores; the bus itself and an arbiter (the latter is only needed if there is more than one master)

External (Off-Chip) Bus

- ML-310 has a PCI (Peripheral Component Interconnect) bus slots on the board
- a multi-function PCI core is available
  - simple OPB to PCI bridge
  - to the PCI slots, the FPGA appears as a PC's North bridge
- a PCI arbiter is also available

Memory and Memory Controllers

- ***plb_bram_if_cntlr*** — interface between PLB and a collection of BRAM resources
  (note: a second core, ***bram_block***, is a core that is the collection of BRAMs)
- ***opb_bram_if_cntlr*** — interface between OPB and bram_block
- ***plb_ddr*** (***opb_ddr***) — interface between PLB (OPB) and external DDR SDRAM

Memory Controller Options

Several more options related to memory...

- Multi-Ported Memory Controller — interfaces one external RAM resource to multiple on-chip entities for example, one port on PLB another port may be a custom core
- OCM has BRAM interface core
- newer technologies have cores as well (DDR2, QDR, RAMBUS)

Peripherals

- communication
  - low-speed, local
  - high-speed/remote networking
- Digital I/O (GPIO) and Analog I/O
- Clock and Clock control
- Timers
- Multimedia drivers (video, touchscreens, etc.)
- Function Units (Computation)

Low-Speed Communication Protocols

- ***UART*** – Universal Asynchronous Receiver/Transmitter
  a parallel-to-serial and serial-to-parallel converter
  - RS-232C (UART16550)
  - commonly known as a serial port
  - remote networking

- ***IIC or I2C*** – Inter-Integrated Circuit
  simple 2-wire bus for communication with multiple peers

Low-Speed Communication Protocols (cont'd)

- ***SPI*** – Serial Peripheral Interface
  a 4-wire solution developed by Motorola
- ***Dallas 1-wire*** – another simple serial interface
- ***I2O*** – Intelligent I/O
  designed to work with PCI to off-load some I/O processing (hardware RAID cards, for example)

Contrasting Low-Speed Communication Protocols

- UARTs, like RS-232C, were designed to interface a computer and communication equipment (i.e., 1-to-1)
- IIC – very small, lots of chipsets (used in TV InfraRed remotes and other applications), ideal for talking to slow EEPROMs
- SPI – alternative to IIC; large number of embedded devices (like LCD, sensors, audio chips, etc.)
- I2O – not `open' and not often used in embedded systems

High-Speed Communication Protocols

- ***CAN*** – Controller Area Network
  used in relatively large embedded systems (vehicles)
- ***Ethernet*** family
  - original – 10 Mb/s eight-wire packet based
  - FastEthernet – 100 Mb/s compatible with Ethernet
  - GigE – 1000 Mb/s compatible with FastEthernet

  switched or point-to-point

High-Speed Communication Protocols (parallel)

- ***SCSI*** – Small Computer System Interconnect parallel bus, originally designed for high-speed peripherals (like disk drives)
- ***ATA(IDE)*** – Advanced Technology Attachment (Integrated Drive Electronics) developed during PC era for inexpensive high-speed disk drives

High-Speed Communication Protocols (Serial)

- ***HDLC*** – standard version of SDLC; point-to-point high speed serial link
- ***Aurora*** – Xilinx-specific serial protocol
- ***InfiniBand*** – another high-speed serial protocol
- ***PCI-Express*** – (PCI-e) another high-speed serial protocol

EDK: System Components + Tools

- in addition to providing large catalog of IP cores to use in your system, EDK includes several tools
- software tools (applications) read in hardware and software specification files, then produce all of the "glue" code (VHDL, C libraries, board constraint files) needed to assemble the system

Hardware And Software Specification

- ***MHS*** – Microprocessor Hardware Specification file; a simple ASCII text file that lists the hardware components and their parameters
- ***MSS*** – Microprocessor Software Specification file; lists the device drivers needed for each of the hardware components

EDK Command-Line Tools

- in addition to providing large catalog of IP cores to use in your system, EDK includes several tools
- ***platgen*** – reads MHS file and generates VHDL code that connects up the IP of the system
- ***libgen*** – reads MSS file and creates C libraries for the device drivers associated with the IP cores in the system

Xilinx Platform Studio

- EDK is just handful of IP cores and command-line tools
- How do *build a system* from these cores?
  - manually create a netlist (schematic capture)
  - use structural VHDL and instantiate systems
  - EDK: `platgen`, `libgen`, ...
  - XPS: graphical user interface to EDK

EDK to XPS

- just like before (with the netlist to bitstream flow) all of this can be manually executed one step at a time
- however, it is much easier to set up a Makefile that does each step for you
- it is even *easier* to use a graphical user interface — Xilinx Platform Studio (XPS) does just that
- ***XMP*** – GUI settings for XPS