Review of Embedded System Design using Zynq

Objectives

- **After completing this module, you will be able to:**
  - Identify some of the features of the ARM Cortex™-A9 processor
  - Review embedded system fundamentals from a hardware point of view
  - Review embedded system fundamentals from a software point of view
  - List embedded processor components common to every processor design
  - Describe some of the features of an AXI port peripheral

Outline

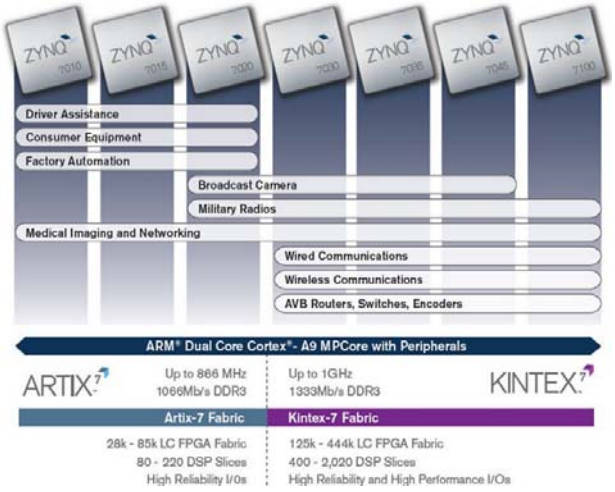
- *Introduction*
- Hardware Design Flow using Vivado
- Software Design Flow using XSDK
- AXI Interfaces
- Summary

Embedded System Design Architecture in Zynq

- **Embedded design in Zynq is based on:**
  - Processor and peripherals
    - Dual ARM® Cortex™ -A9 processors of Zynq-7000 SoC
    - AXI interconnect
    - AXI component peripherals
    - Reset, clocking, debug ports
  - Software platform for processing system
    - Standalone or other (e.g. Linux) OS
    - C language support
    - Processor services
    - C drivers for hardware
  - User application
    - Interrupt service routines (optional)

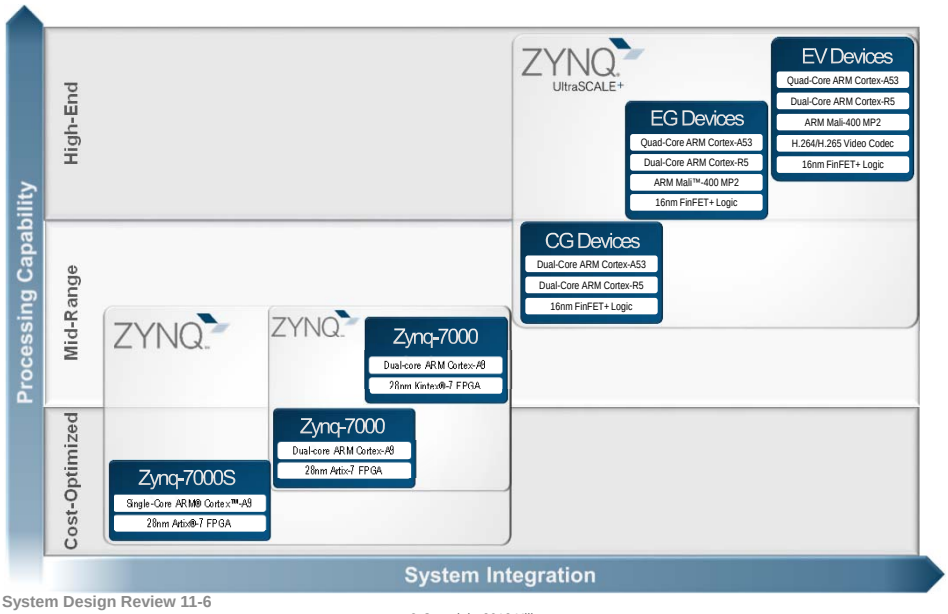
The PS and the PL

- **The Zynq-7000 SoC architecture consists of two major sections**
  - PS: Processing system
    - Dual ARM Cortex-A9 processor based
    - Multiple peripherals
    - Hard silicon core
  - PL: Programmable logic
    - Uses the same 7 series programmable logic
      - Artix™-based devices: Z-7010, Z-7015 and Z-7020 (high-range I/O banks only)
        - Single core versions: Z-7007S, Z-7012S, and Z-7014S
      - Kintex™-based devices: Z-7030, Z-7035, Z-7045, and Z-7100 (mix of high-range and high-performance I/O banks)



XILINX.

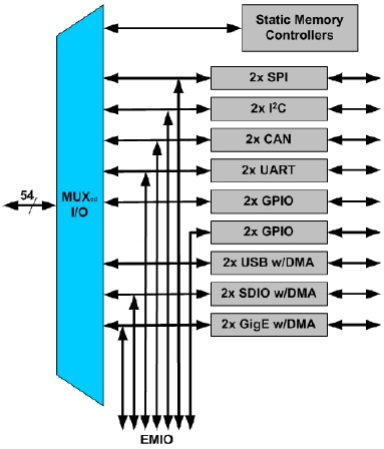
Extending Scalability Across the Zynq Portfolio



XILINX.

Zynq Architecture Built-in Peripherals

- **Two USB 2.0 OTG/Device/Host**
- **Two Tri- Mode GigE (10/100/1000)**
- **Two SD/SDIO interfaces**
  - Memory, I/O and combo cards
- **Two CAN 2.0Bs, SPIs , I2Cs, UARTs**
- **Four GPIO 32bit Blocks**
  - 54 available through MIO; other available through EMIO
- **Multiplexed Input/Output (MIO)**
  - Multiplexed pinout of peripheral and static memories
- **Extended MIO**
  - Maps PS peripheral ports to the PL



XILINX.

Hardware Design Flow using Vivado

XILINX.

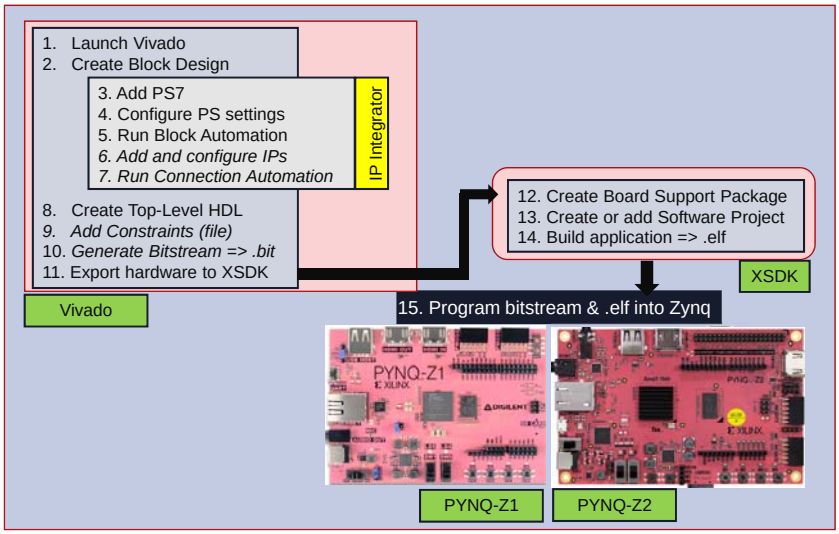
Embedded System Design Flow using Vivado and XSDK (1)

- Create a new project, or open an existing project
- Add/Create a new embedded source in Vivado
- Use IP integrator, Block automation, and connection automation features of Vivado IPI to construct(modify) the hardware portion of the embedded design
- Create(Update) top level HDL model
- Add additional logic at the top-level
- Synthesize, implement, and generate the design in Vivado
- Export the bitstream, processor hardware description, and launch XSDK

Embedded System Design Flow using Vivado and XSDK (2)

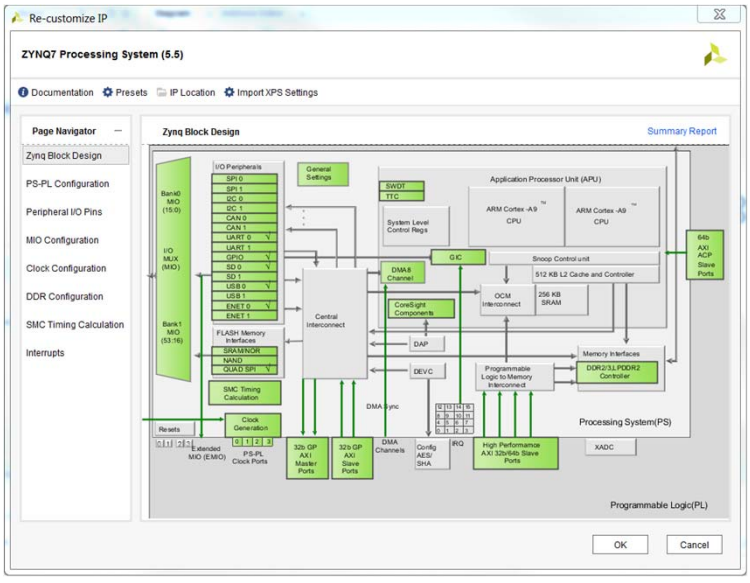
- Create a new software board support package and application projects in the XSDK
- Compile the software with the GNU cross-compiler in XSDK
- [optional] Download the programmable logic's completed bitstream using XSDK or through a hardware manager in Vivado
- Use XSDK to download the program (the ELF file)

Embedded System Design using Vivado and XSDK



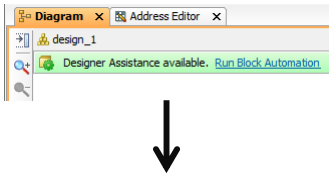
XILINX.

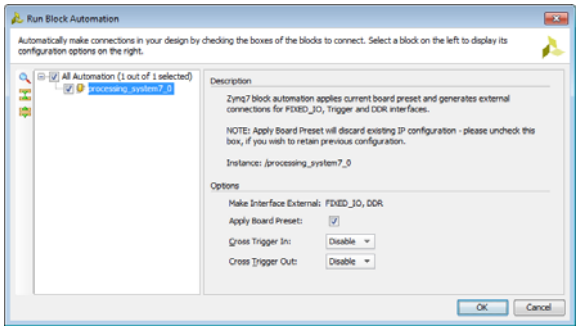
Zynq Device Processing System Configuration

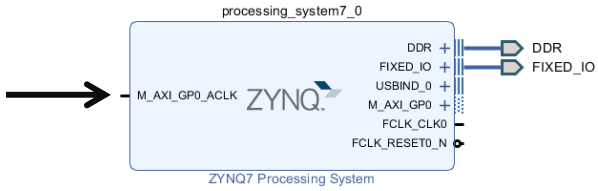


XILINX.

Run Block Automation



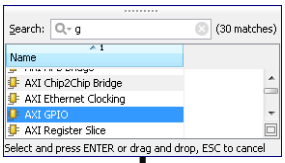


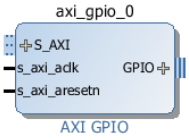


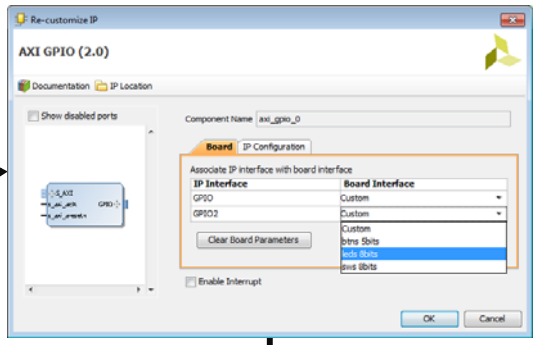
- **Create default configuration for the platform based on the board specified in project settings**
  - e.g. PYNQ-Z2: DDR, GPIO, Uart, USB, QSPI...

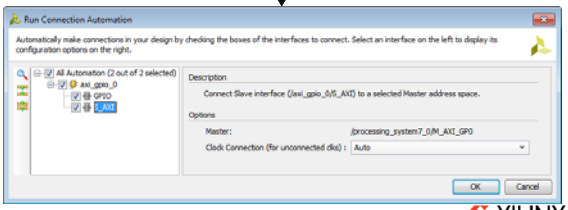
Extending Hardware in IP Integrator

- **Add IPs**
- **Configure IPs**
- **Run Connection Automation**



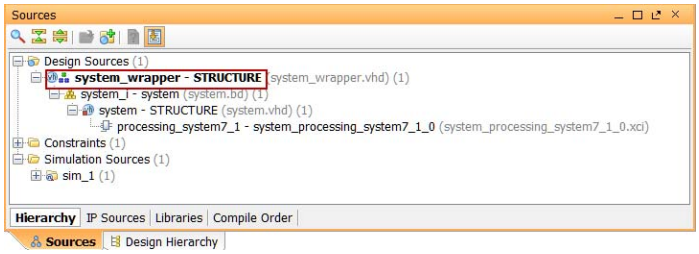






Extending Hardware in Vivado

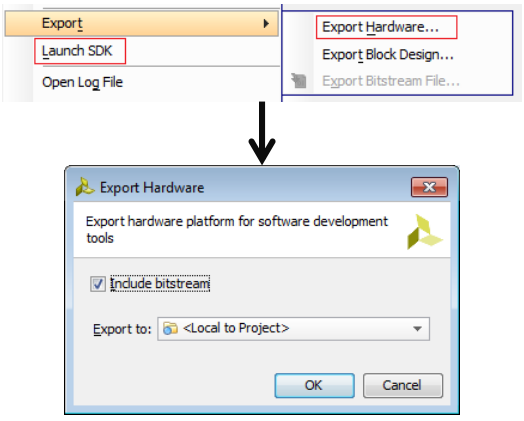
- **Create a top level HDL model**
- **Optionally, add other hdl files to the design**
- **Add user constraint files to connect PL pins**
  - PS/MIO handled automatically
  - If you miss any pin constraints (IO standard must be explicitly specified), the tools will error out during the bit generation process
- **Generate bitstream for PL**



XILINX.

Export Hardware Design to XSDK

- **Software development is performed with the Xilinx Software Development Kit tool (XSDK)**
- **An Hardware Description file (.hdf) of the hardware is imported in the XSDK tool**
  - The hardware platform is built on this description
- **XSDK will then associate user software projects to hardware**



XILINX.

Software Design Flow using XSDK

XILINX.

Embedded System Tools: Software

- **Eclipse IDE-based Software Development Kit (XSDK)**
  - Board support package creation : hsi::generate_bsp
  - GNU software development tools
  - C/C++ compiler for the ARM Cortex-A9 processor (gcc)
  - Debugger for the ARM Cortex-A9 processor (gdb)
- **Board support packages (BSPs)**
  - Stand-alone BSP
    - Free basic device drivers and utilities from Xilinx
    - NOT an RTOS
  - FreeRTOS

XILINX.

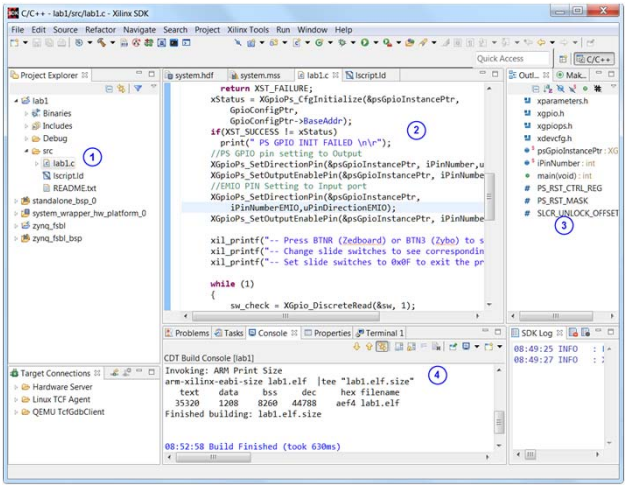
XSDK Workbench Views

① **C/C++ project outline displays the elements of a project with file decorators (icons) for easy identification**

② **C/C++ editor for integrated software creation**

③ **Code outline displays elements of the software file under development with file decorators (icons) for easy identification**

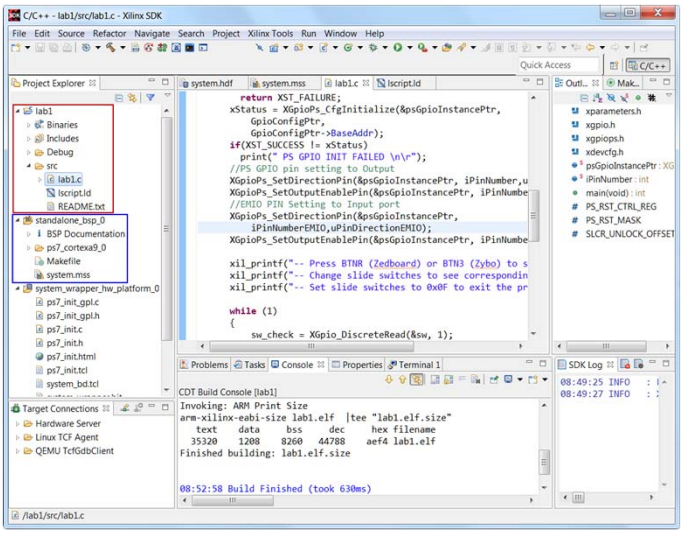
④ **Problems, Console, Properties views list output information associated with the software development flow**



XILINX.

Build Software Application in XSDK

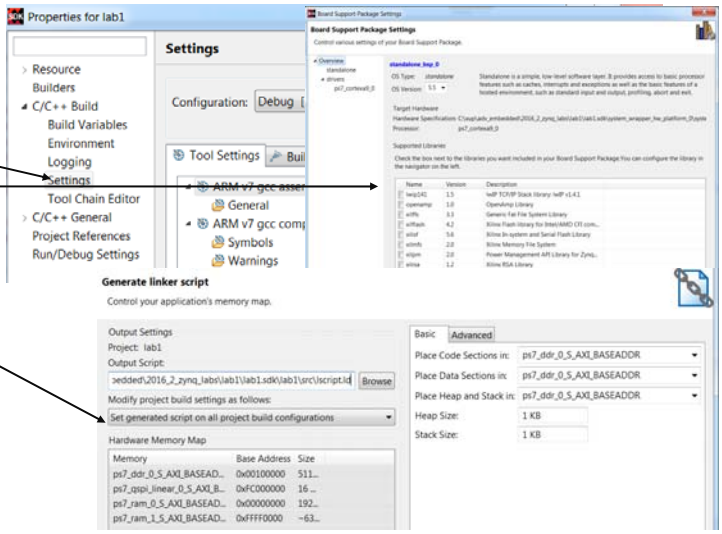
- **Create software platform**
  - System software, board support package
  - hsi::generate_bsp program
- **Create software application**
- **Optionally, create linker script**
- **Build project**
  - compile, assemble, link output file *<app_project>.elf*



XILINX.

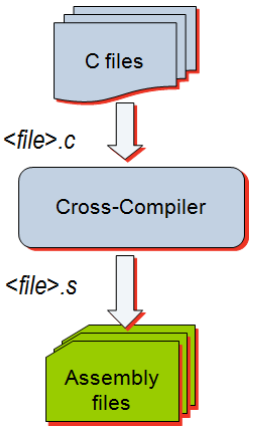
Software Management Settings

- **Software is managed in three major areas**
  - Compiler/Linker Options
    - Application program
  - Software Platform Settings
    - Board support package
  - Linker Script Generation
    - Assigning different components of software to various memory resources



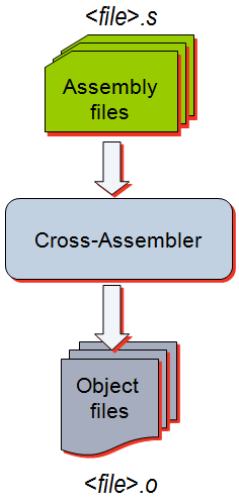
GNU Tools: GCC

- **GCC translates C source code into assembly language**
- **GCC also functions as the user interface, passing options to the GNU assembler and to the GNU linker, calling the assembler and the linker with the appropriate parameters**
- **Supported cross-compilers**
- **ARM processor compiler**
  - GNU GCC (arm-none-eabi-gcc)
  - GNU Linux GCC (arm-none-linux-eabi-gcc)



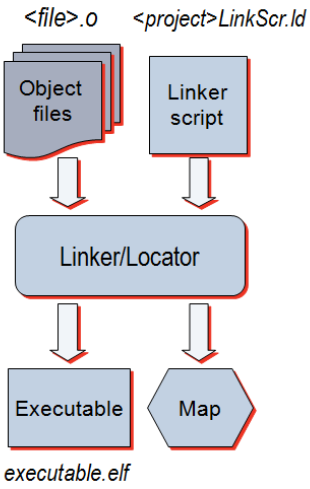
GNU Tools: AS

- **Input: assembly language files**
  - File extension: *.s*
- **Output: object code**
  - File extension: *.o*
- **Contains**
  - Assembled piece of code
  - Constant data
  - External references
  - Debugging information
- **Typically, the compiler automatically calls the assembler**
- **Use the -Wa switch if the source files are assembly only and use gcc**



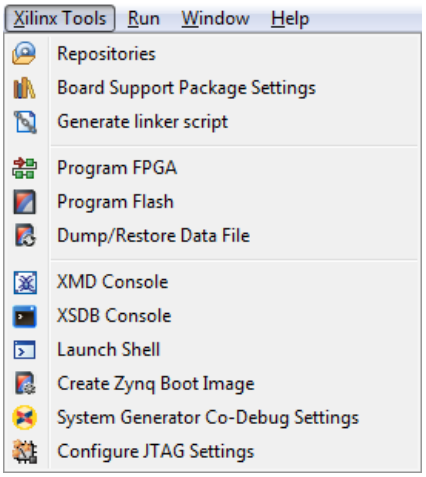
GNU Tools: Linker (LD)

- **Inputs**
  - Several object files
  - Archived object files (library)
  - Linker script (**.ld*)
- **Outputs**
  - Executable image (ELF)
  - Map file



Integrated Xilinx Tools

- **Xilinx additions to the Eclipse IDE**
  - Software Repositories
  - BSP Settings
  - Generate Linker Script
  - Program the programmable logic
    - Bitstream must be available
  - Program Flash Memory
  - Launch XMD Console
  - Launch Shell
  - Create Zynq Boot Image
  - SysGen Co-Debug Settings
  - Configure JTAG Settings



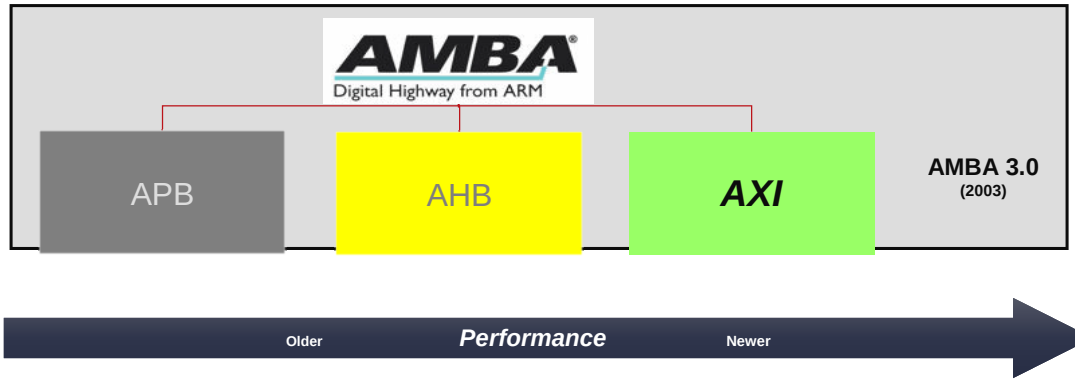
Configuring FPGA and Downloading Application

- **Download the bitstream**
  - Only required if PL is used
  - Input file *<top_name>.bit*
- **The Xilinx hardware session allows downloading the bitstream in to the target**
- **The hardware session can be created from**
  - XSDK
  - Vivado
- **Requires that the download cable is connected**

AXI Interfaces

XILINX.

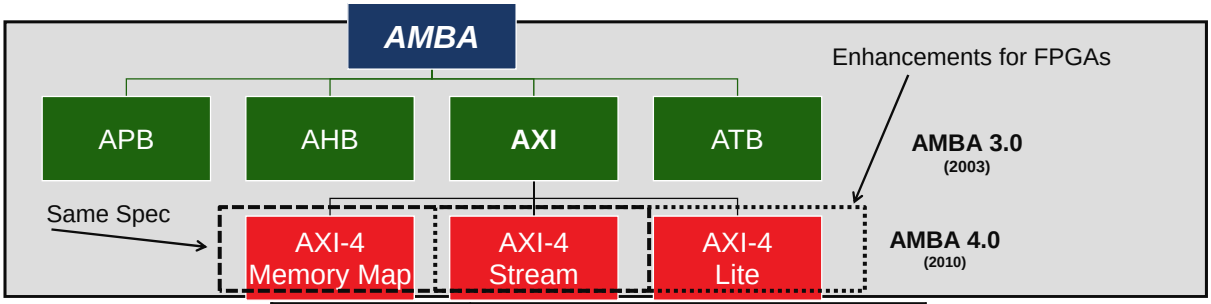
AXI is Part of ARM's AMBA



AMBA: Advanced Microcontroller Bus Architecture
AXI: Advanced Extensible Interface

XILINX.

AXI is Part of AMBA



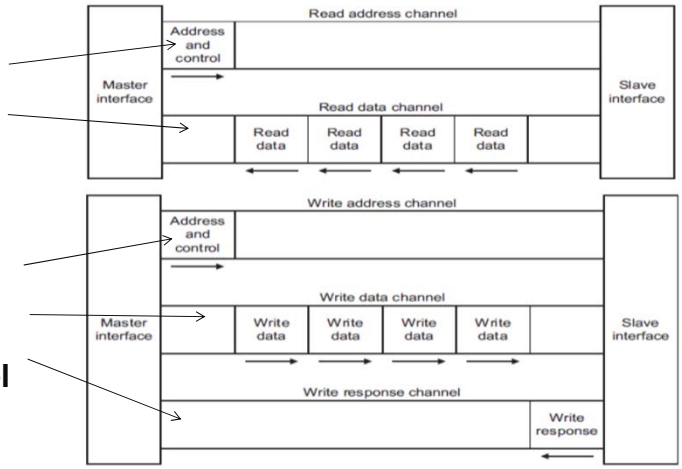
Interface	Features
Memory Map / Full (AXI4)	**Traditional Address/Data Burst (single address, multiple data)**
Streaming (AXI4-Stream)	**Data-Only, Burst**
Lite (AXI4-Lite)	**Traditional Address/Data—No Burst (single address, single data)**

AXI Interconnect

- **AXI is an interconnect system used to tie processors to peripherals**
  - AXI Full memory map: Full performance bursting interconnect
  - AXI Lite: Lower performance non bursting interconnect (saves programmable logic resources)
  - AXI Streaming: Non-addressed packet based or raw interface

Basic AXI Signaling – 5 Channels

1. **Read Address Channel**
2. **Read Data Channel**
3. **Write Address Channel**
4. **Write Data Channel**
5. **Write Response Channel**



XILINX.

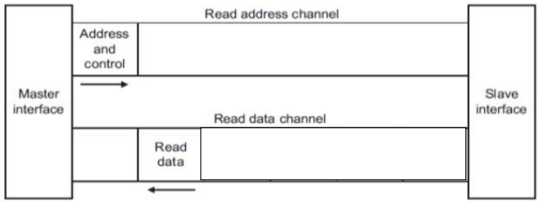
All AXI Channels Use A Basic "VALID/READY" Handshake

- ***SOURCE*** **asserts and holds VALID when DATA is available**
- ***DESTINATION*** **asserts READY if able to accept DATA**
- **DATA transferred when VALID and READY = 1**
- ***SOURCE*** **sends next DATA (if an actual data channel) or deasserts VALID**
- ***DESTINATION*** **deasserts READY if no longer able to accept DATA**

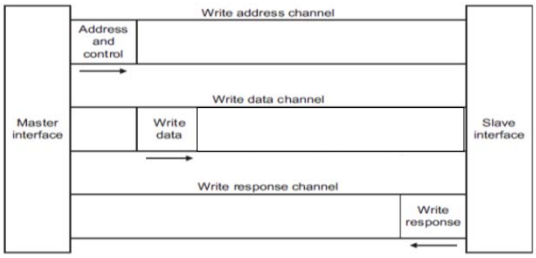
XILINX.

The AXI Interface—AXI4Lite

- **No burst**
- **Data width 32 or 64 only**
  - Xilinx IP only supports 32-bits
- **Very small footprint**
- **Bridging to AXI4 handled automatically by AXI_Interconnect (if needed)**



AXI4-Lite Read

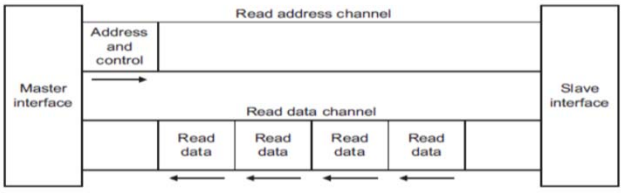


AXI4-Lite Write

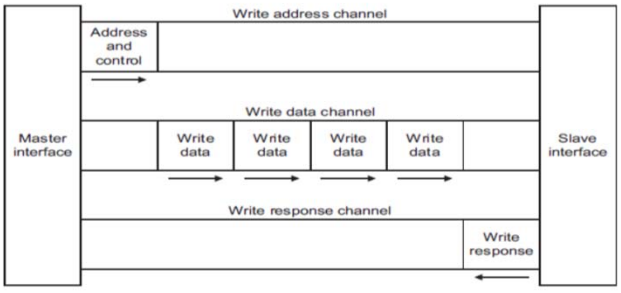
XILINX.

The AXI Interface—AXI4

- **Sometimes called "Full AXI" or Memory Mapped"**
  - Not ARM-sanctioned names
- **Single address multiple data**
  - Burst up to 256 data beats
- **Data Width parameterizable**
  - 1024 bits



AXI4 Read



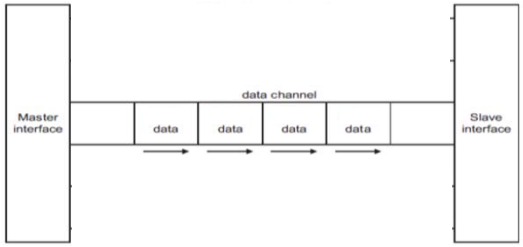
AXI4 Write

XILINX.

The AXI Interface—AXI4Stream

- **No address channel, no read and write, always just master to slave**
  - Effectively an AXI4 “write data” channel
- **Unlimited burst length**
  - AXI4 max 256
  - AXI4-Lite does not burst
- **Virtually same signaling as AXI Data Channels**
  - Protocol allows merging, packing, width conversion
  - Supports sparse, continuous, aligned, unaligned streams

AXI4-Stream Transfer



XILINX.

Summary

XILINX.

Summary

- Cortex-A9 processor is a hard processor that interfaces AXI Full
- Hardware is developed graphically in Vivado using the IP Integrator
- IP Integrator's block design, block automation, and connection automation features can be used for quickly building a hardware system
- PS Configuration Wizard is used to configure the processor block
- Software is developed in an XSDK workspace
- AXI interface provides higher performance using point-to-point connection
- AXI has separate, independent read and write interfaces implemented with channels
- The AXI4 interface offers improvements over AXI3 and defines
  - Full AXI memory mapped
  - AXI Lite
  - AXI Stream

XILINX.

Adaptable.
Intelligent.

XILINX.