Advanced Zynq Architecture

XILINX.

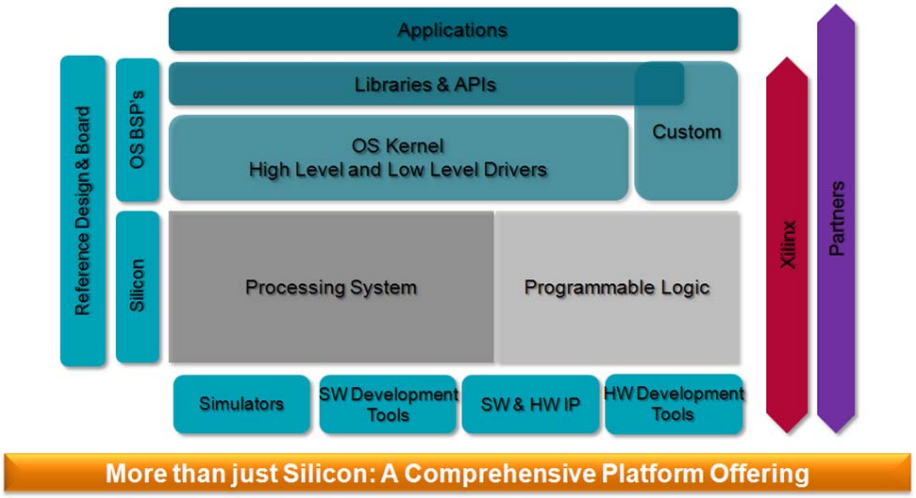
Objectives

- **After completing this module, you will be able to:**
  - Describe the features of the Cortex™-A9 processor
  - List the operations that are accelerated by the NEON co-processor
  - Explain how the snoop control unit (SCU) integrates memory operation between dual processors
  - Identify the cache resources in the APU
  - Explain the purpose of the system-level control registers (SLCR)
  - List the peripherals contained in the Zynq™ SoC IOP and describe some of their features

Outline

- *Review of Zynq SoC*
- Application Processing Unit (APU)
- Input/Output Peripherals (IOP)
- Summary

Zynq-7000 SoC



Zynq-7000 Family Highlights

- **Complete ARM®-based processing system**
  - Application Processor Unit (APU)
    - Dual ARM Cortex™-A9 processors
    - Caches and support blocks
  - Fully integrated memory controllers
  - I/O peripherals
- **Tightly integrated programmable logic**
  - Used to extend the processing system
  - Scalable density and performance
- **Flexible array of I/O**
  - Wide range of external multi-standard I/O
  - High-performance integrated serial transceivers
  - Analog-to-digital converter inputs

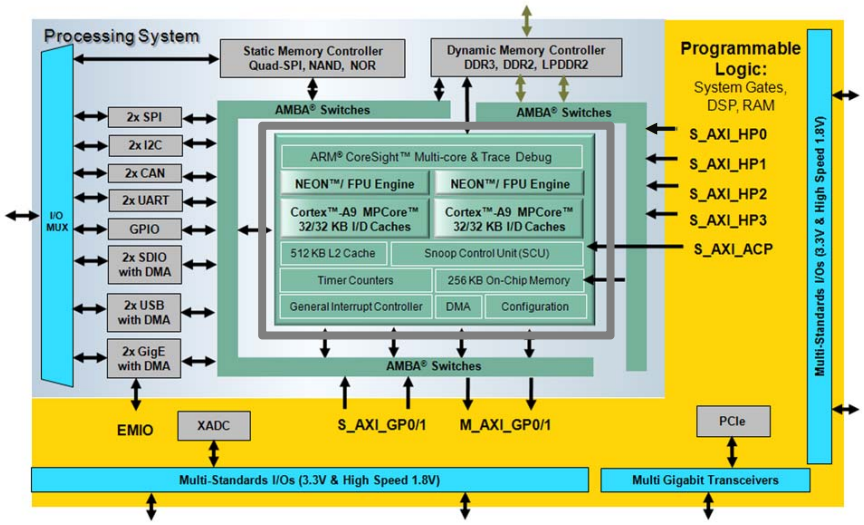
ARM Processor Product Families

- **Legacy ARM processors**
  - ARM7, ARM9 (not the Cortex-A9 processor), ARM11
- **Cortex family of processors**
  - Cortex-A#: "A" application
    - The products support a memory management unit (MMU)
    - Excellent for operating systems
  - Cortex-R#: "R" real time
    - The products support a memory protection Unit (MPU)
    - Better determinism than an MMU
  - Cortex-M#: "M" Embedded microcontroller
- **There are some products that are implemented differently but use the same ARM Architecture**
  - Cortex-A8 and Cortex-A9 processors

ARM Processor Architecture

- **ARM Cortex-A9 processor implements the ARMv7-A architecture**
  - ARMv7 is the ARM Instruction Set Architecture (ISA)
    - Thumb instructions: 16 bits; Thumb-2 instructions: 32 bits
    - NEON: ARM's Single Instruction Multiple Data (SIMD) instructions
  - ARMv7-A: Application set that includes support for a Memory Management Unit (MMU)
  - ARMv7-R: Real-time set that includes support for a Memory Protection Unit (MPU)
  - ARMv7-M: Microcontroller set that is the smallest set
- **ARM Advanced Microcontroller Bus Architecture (AMBA®) protocol**
  - AXI3: Third-generation ARM interface
  - AXI4: Adding to the existing AXI definition (extended bursts, subsets)
- **Cortex is the new family of processors**
  - ARM family is older generation; Cortex is current; MMUs in Cortex processors and MPUs in ARM

Zynq-7000 SoC Block Diagram



PS Components

- **Application processing unit (APU)**
- **I/O peripherals (IOP)**
  - Multiplexed I/O (MIO), extended multiplexed I/O (EMIO)
- **Memory interfaces**
- **PS interconnect**
- **DMA**
- **Timers**
  - Public and private
- **General interrupt controller (GIC)**
- **On-chip memory (OCM): RAM**
- **Debug controller: CoreSight**

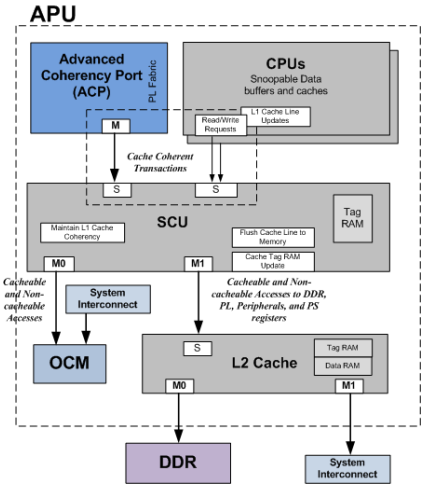
XILINX.

Application Processing Unit (APU)

XILINX.

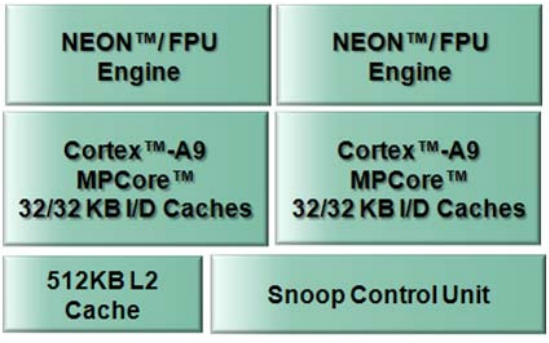
Introduction to the APU

- **Heart of the PS**
- **Tightly coupled processors and sub-components for maximum performance**
- **Tied to other PS components and PL via the PS interconnect**



Inside the APU

- **Dual ARM® Cortex™-A9 MPCore with NEON extensions**
  - Up to 1GHz operation
  - 2.5 DMIPS/MHz per core
  - Separate 32KB instruction and data caches
- **Snoop control unit**
  - L1 cache snoop control
    - Accelerator coherency port
- **Level 2 cache and controller**
  - Shared 512 KB cache with parity



APU Sub-Components

- General interrupt controller (GIC)
- On-chip memory (OCM): RAM and boot ROM
- Central DMA (eight channels)
- Device configuration (DEVCFG)
- Private watchdog timer and timer for each CPU
- System watchdog and triple timer counters shared between CPUs
- ARM CoreSight debug technology

APU Internal Address Map

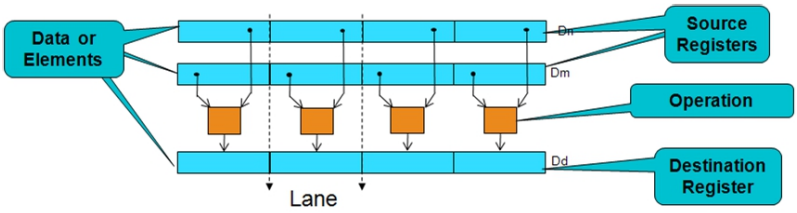
- All registers for both CPUs are grouped into two contiguous 4KB pages
  - Accessed through a dedicated internal bus
- Fixed at 0xF8F0_0000 with a register block size of 8 KB
  - Each CPU uses an offset into this base address

0x0000-0x00FC	SCU registers
0x0100-0x01FF	Interrupt controller interface
0x0200-0x02FF	Global timer
0x0600-0x06FF	Private timers and watchdog timers
0x1000-0x1FFF	Interrupt distributor

Vector Processing using NEON

- **NEON is the ARM codename for the vector processing unit**
  - Provides multimedia and signal processing support
- **FPU is the floating-point unit extension to NEON**
  - Both NEON and FPU share a single set of registers
- **NEON technology is a wide single instruction, multiple data (SIMD) parallel and co-processing architecture**
  - 32 registers, 64-bits wide (dual view as 16 registers, 128-bits wide)
  - Data types can be: signed/unsigned 8-bit, 16-bit, 32-bit, 64-bit, or 32-bit float

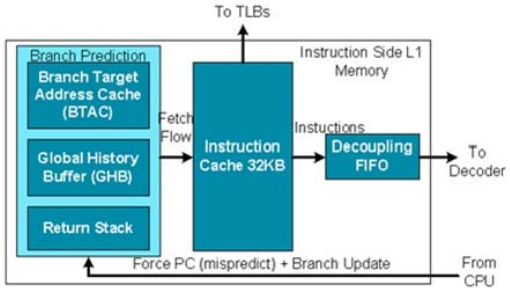
NEON™/FPU Engine



XILINX.

L1 Cache Features

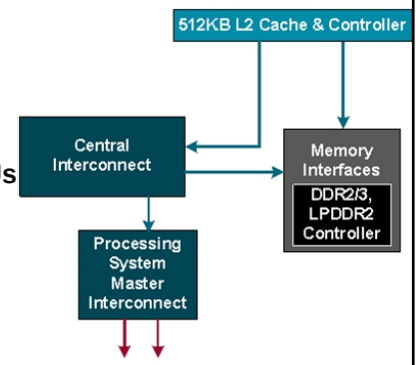
- **Separate instruction and data caches for each processor**
- **Caches are four-way, set associative and are write-back**
- **Non-lockable**
- **Eight words cache length**
- **On a cache miss, critical word first filling of the cache is performed followed by the next word in sequence**



XILINX.

L2 Cache Features

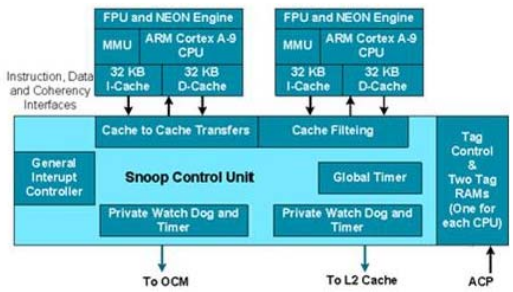
- **512K bytes of RAM built into the SCU**
  - Latency of 25 CPU cycles
  - Unified instruction and data cache
- **Fixed, 256-bit (32 words) cache line size**
- **Support for per-master way lockdown between multiple CPUs**
- **Eight-way, set associative**
- **Two AXI interfaces**
  - One to DDR controller
  - One to programmable logic master (to peripherals)



XILINX.

Snoop Control Unit (SCU)

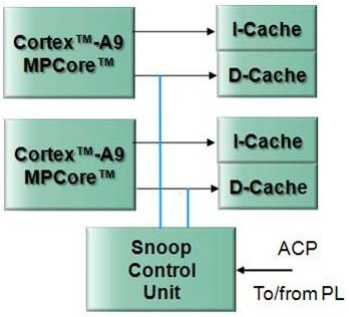
- **Shares and arbitrates functions between the two processor cores**
  - Data cache coherency between the processors
  - Initiates L2 AXI memory access
  - Arbitrates between the processors requesting L2 accesses
  - Manages ACP accesses
  - A second master port with programmable address filtering between OCM and L2 memory support



XILINX.

Cache Coherency using SCU

- **High-performance, cache-to-cache transfers**
- **Snoop each CPU and cache each interface independently**
- **Coherency protocol is MESI**
  - M: Cache line has been modified
  - E: Cache line is held exclusively
  - S: Cache line is shared with another CPU
  - I: Cache line is invalidated
- **Uses Accelerator Coherence Port (ACP) to allow coherency to be extended to PL**



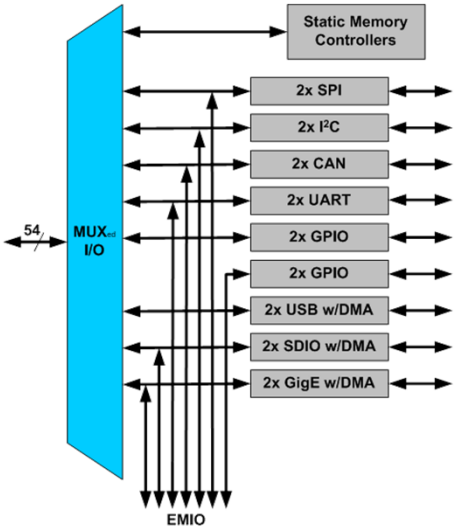
System-Level Control Registers (SLCR)

- **A set of special registers in the APU used to configure the PS**
  - Power and clock management
  - Reset control
  - MIO/EMIO management
- **Accessible through software**
  - Standalone BSP support
- **SLCR categories**
  - System clock and reset control/status registers
  - TrustZone control register
  - APU control registers
  - SoC debug control registers
  - DMA initialization registers
  - MIO/IOP control/status registers
  - DDR control registers
  - Miscellaneous control registers
  - PL reset registers
  - RAM and ROM control registers

Input/Output Peripherals (IOP)

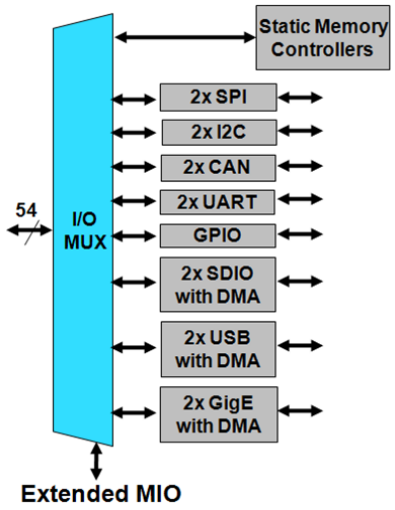
Zynq Built-in Peripherals

- **Two USB 2.0 OTG/device/host**
- **Two tri-mode gigabit Ethernet (10/100/1000)**
- **Two SD/SDIO interfaces**
  - Memory, I/O, and combo cards
- **Two CAN 2.0Bs, SPIs, I2Cs, UARTs**
- **Four GPIO 32-bit blocks**
  - 54 available through MIO; other 64 available through EMIO



Zynq Built-in Peripherals

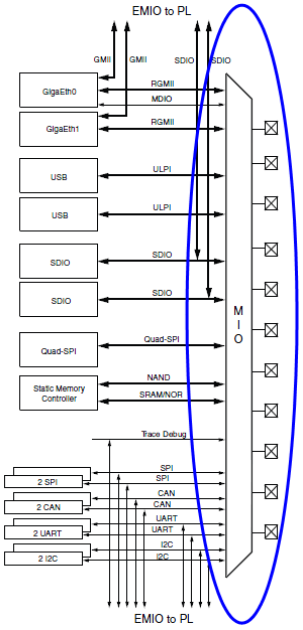
- **Multiplexed input/output (MIO)**
  - Multiplexed output of peripheral and static memories
  - Two I/O banks; each selectable: 1.8V, 2.5V, or 3.3V
  - Configured using configuration
  - Dedicated pins are used
    - User constraints (LOC) should not be present
      - The BitGen process will throw errors if LOC constraints are present
- **Extended MIO**
  - Enables use of the SelectIO™ interface with PS peripherals
  - User constraints must be present for the signals brought out to the SelectIO pins
    - The BitGen process will throw errors if LOC constraints are not present



XILINX.

Multiplexed I/O (MIO)

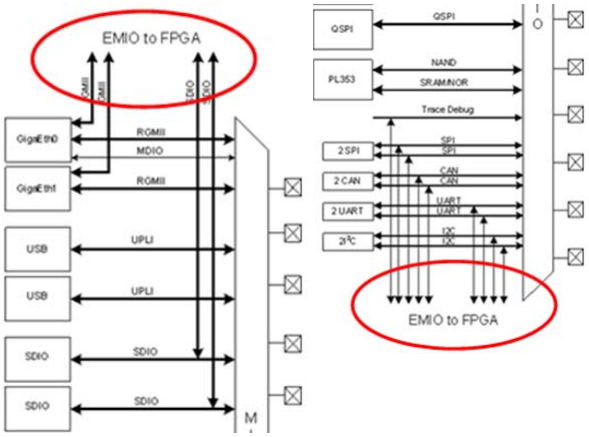
- **External interface to PS I/O peripheral ports**
  - 54 dedicated package pins available
  - Software configurable
    - Automatically added to bootloader by tools
  - Not available for all peripheral ports
    - Some ports can only use EMIO



XILINX.

Extended Multiplexed I/O (EMIO)

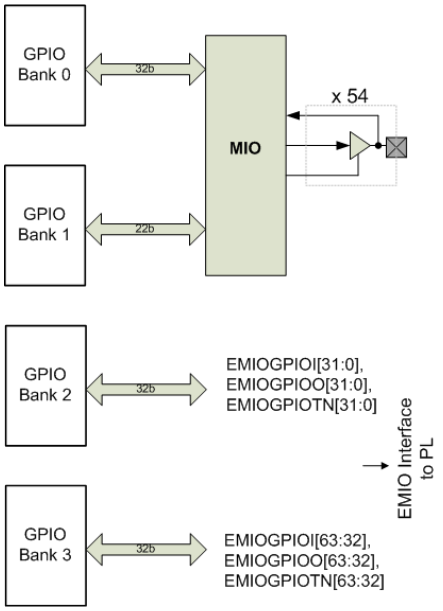
- **Extended interface to PS I/O peripheral ports**
  - EMIO: Peripheral port to programmable logic
  - Alternative to using MIO
  - Mandatory for some peripheral ports
  - Facilitates
    - Connection to peripheral in programmable logic
    - Use of general I/O pins to supplement MIO pin usage
    - Allows additional signals for many of the peripherals
    - Alleviates competition for MIO pin usage



XILINX.

General-Purpose I/O

- **GPIO blocks**
  - Four separate banks of 32 GPIO bits each
    - Two banks connect to the 54 MIO pins
      - 32 bits and 22 bits, respectively
    - Two banks connect to EMIO (64 bits)
  - Each GPIO bit can be dynamically programmed as input or output
  - Reset values independently configurable for each bit
  - Programmable interrupt generation for each bit
    - One interrupt generated per GPIO bank



XILINX.

Driver Support for GPIO

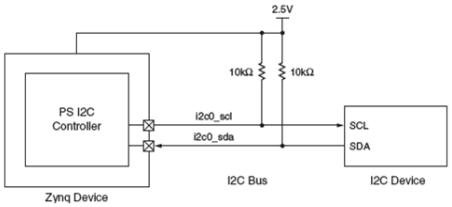
- **Include files needed**
  - xgpiops.h, xgpiops_hw.h (for low-level functions)
- **Initialize GPIO device driver**
  - ConfigPtr = XGpioPs_LookupConfig(GPIO_DEVICE_ID);
  - Status = XGpioPs_CfgInitialize(&Gpio, ConfigPtr, ConfigPtr->BaseAddr);
- **Configure and use GPIO device for output**
  - XGpioPs_SetDirectionPin(&Gpio, OUTPUT_PIN, 1);
  - XGpioPs_SetOutputEnablePin(&Gpio, OUTPUT_PIN, 1);
  - XGpioPs_WritePin(&Gpio, OUTPUT_PIN, 0x0);
- **Configure and use GPIO device for input**
  - XGpioPs_SetDirectionPin(&Gpio, INPUT_PIN, 0x0);
  - *DataRead = XGpioPs_ReadPin(&Gpio, INPUT_PIN);

Driver Support for GPIO

- **Interrupts setup**
  - XGpioPs_SetIntrType (InstancePtr, Bank, IntrType, IntrPolarity, IntrOnAny);
  - XGpioPs_SetCallbackHandler (InstancePtr, CallBackRef, FuncPtr);
  - XGpioPs_IntrEnable (InstancePtr, Bank, Mask);
- **Interrupts processing**
  - XGpioPs_IntrGetStatus (InstancePtr, Bank);
  - XGpioPs_IntrGetStatusPin (InstancePtr, Pin);
  - XGpioPs_IntrClear (InstancePtr, Bank, Mask);
- **There are more functions than listed here**

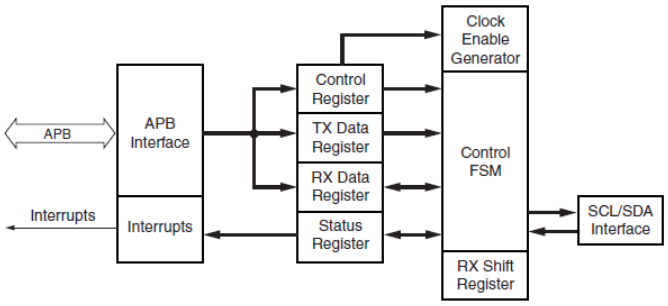
I²C Controller

- I²C bus specification version 2
- Programmable to use normal (7-bit) or extended (10-bit) addressing
- Programmable rates: fast mode (400 kbit/s) , standard (100 kbits/s), and low (10 kbits/s)
  - Rates higher than 400 kbits/sec are not supported
- Programmable as either a master or slave interface
- Capable of clock synchronization and bus arbitration
- Fully programmable slave response address



I²C Controller

- Reversible FIFO operation supported
- 16-byte buffer size
- Slave monitor mode when set up as master
- I²C bus hold for slow host service
- Slave timeout detection with programmable period
- Transfer status interrupts and flags



I²C Controller Registers

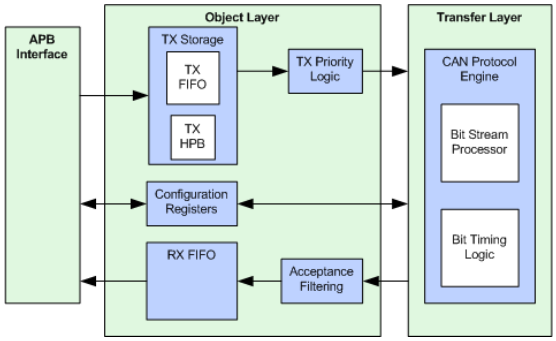
Register Name	Address Offset	Type	Reset Value	Description
Control_reg0	0x00000000	Read/write	0x00000000	Control register
Status_reg0	0x00000004	Read only	0x00000000	Status register
I2C_address_reg0	0x00000008	Read/write	0x00000000	IIC address register
I2C_data_reg0	0x0000000c	Read/write	0x00000000	IIC data register
Interrupt_status_reg0	0x00000010	Read/write	0x00000000	IIC interrupt status register
Transfer_size_reg0	0x00000014	Read/write	0x00000000	Transfer Size Register
Slave_mon_pause_reg0	0x00000018	Read/write	0x00000000	Slave Monitor Pause Register
Time_out_reg0	0x0000001c	Read/write	0x0000001F	Time out register
Intrpt_mask_reg0	0x00000020	Read only	0x000002FF	Interrupt mask register
Intrpt_enable_reg0	0x00000024	Read/write	0x00000000	Interrupt Enable Register
Intrpt_disable_reg0	0x00000028	Read/write	0x00000000	Interrupt Disable Register

Driver Support for the I²C Controller

- **Include files needed**
  - xiicps.h, xiicps_hw.h (for low-level functions)
- **Initialize I²C device driver**
  - ConfigPtr = XIicPs_LookupConfig(IIC_DEVICE_ID);
  - Status = XIicPs_CfgInitialize(&Iicps, ConfigPtr, ConfigPtr->BaseAddr);
- **Setp up I²C with slave address and clock rate**
  - XIicPs_SetupSlave(&Iicps, IIC_SLAVE_ADDR); // 7-bits or 10-bits address
  - XIicPs_SetSClk(&Iicps, IIC_SCLK_RATE);
- **Use I²C device in slave mode**
  - XIicPs_SlaveSendPolled(&Iicps, SendBuffer, SEND_BUFFER_SIZE);
  - XIicPs_SlaveRecvPolled(&Iicps, RecvBuffer, SEND_BUFFER_SIZE);
  - XIicPs_BusIsBusy(&Iicps);

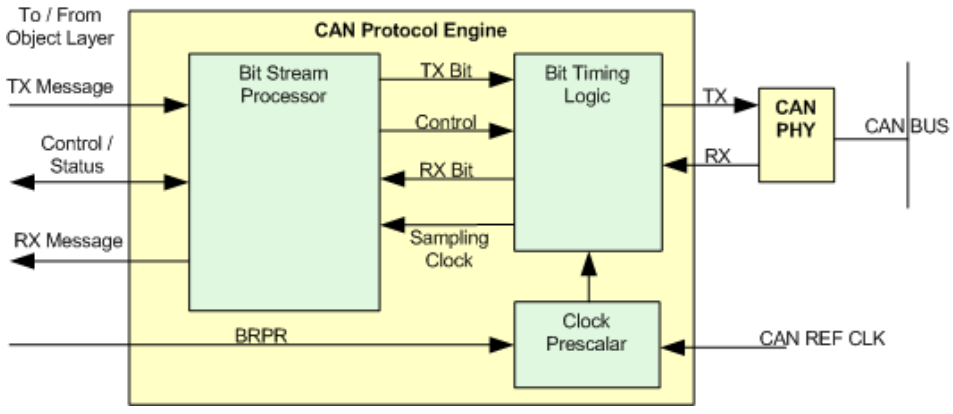
CAN Controller

- Up to 24-MHz CAN_REF clock as system clock
- 64 message-deep receiver and transmitter buffer
- Full CAN 2.0B compliant; conforms to ISO 11898-1
- Maximum baud rate of 1 Mb/s
- Four message filters required for buffer mode
- Listen-only mode for test and debug
- External PHY I/O
- “Wake-on-message”
- Time-stamping for receive messages
- TX and RX FIFO watermarking
- Exception: no power-down mode



XILINX.

CAN Controller Usage Example



XILINX.

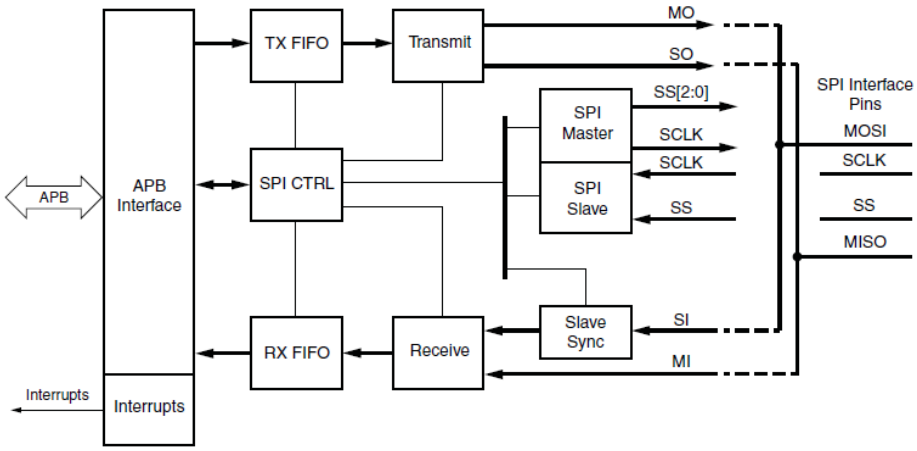
SD-SDIO Controller

- Support for version 2.0 of SD Specification
- Full-speed (4 MB/s) and low-speed (2 MB/s) support
  - Low-speed clock (400 KHz) used until bandwidth negotiated
- 1-bit and 4-bit data interface support
- Host mode support only
- Built-in DMA controller
- Full-speed clock (0-50 MHz) with maximum throughput at 25 MB/s
- 1 KB data FIFO interface
- Support for MMC 3.31 card at 52 MHz
- Support for memory, I/O, and combo cards
- Support for power control modes and interrupts

SPI Controller

- Master or slave SPI mode
- Four wire bus: MOSI, MISO, SCK, nSS
- Supports up to three slave select lines
- Supports multi-master environment
- Identifies an error condition if more than one master detected
- Software can poll for status or function as interrupt-driven device
- Programmable interrupt generation
- 50-MHz maximum external SPI clock rate
- Selectable master clock reference
- Integrated 128-byte deep read and write FIFOs
- Full-duplex operation offers simultaneous receive and transmit

SPI Controller Block Diagram



XILINX.

UART

- Two UARTs with programmable baud rate generator
- 64-byte receive and transmit FIFOs
- 6, 7, or 8 data bits and 1, 1.5, or 2 stop bits
- Odd, even, space, mark, or no parity with parity, framing, and overrun error detection
- "Line break" generation and detection
- Normal, automatic echo, local loopback, and remote loopback channel modes
- Interrupts generation
- Support 8 Mb/s maximum baud rate with additional reference clock or up to 1.5 Mb/s with a 100-MHz peripheral bus clock
- Modem control signals: CTS, RTS, DSR, DTR, RI, and DCD (through EMIO)
- Simple UART: only two pins (TX and RX through MIO)

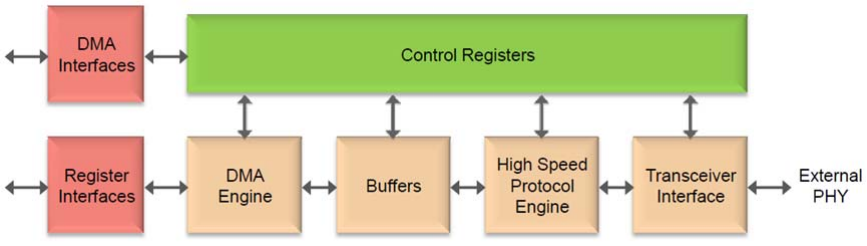
XILINX.

USB Peripherals

- **Two USB 2.0 hardened IP peripherals per Zynq device**
  - Each independently controlled and configured
- **Supported interfaces**
  - High-speed USB 2.0: 480 Mbit/s
  - Full-speed USB 1.1: 12 Mbit/s
  - Low-speed USB 1.0: 1.5 Mbit/s
  - Communication starts at USB 2.0 speed and drops until sync is achieved
- **Each block can be configured as host, device, or on-the-go (OTG)**
- **8-bit ULPI interface**
- **All four transfer types supported: isochronous, interrupt, bulk, and control**
- **Supports up to 12 endpoints per USB block in the Zynq device**
  - Running in host mode
- **Source-code drivers**

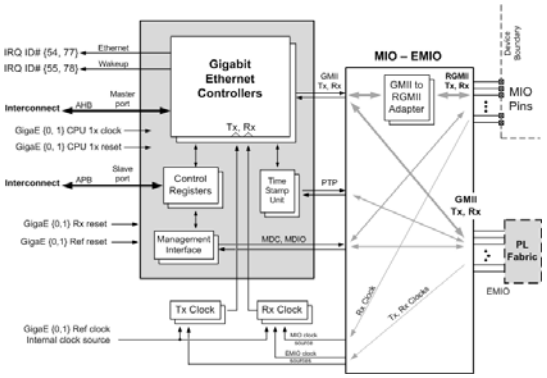
USB 2.0 OTG Controller

- **52 control and configuration registers for each USB block**
- **Software-ready with standalone and OS linux source-code delivered drivers**
- **EHCI compliant host registers**
- **USB host controller registers and data structures compliant to Intel EHCI specifications**
- **Internal DMA**
- **Must use the MIO pins**



Gigabit Ethernet Controller

- **Tri-mode Ethernet MAC (10/100/1G) with native GMII interface**
- **IEEE1588 rev 2.0**
  - Time stamp support
  - 1 us resolution
- **IEEE802.3**
- **RGMII v2.0 (HSTL) interface to MIO pins**
  - Need MIO set at 1.8V to support RGMII speed
  - Need to use large bank of MIO pins for two Ethernets
- **MII/GMII/SGMII/RGMII ver1.3 (LVCMOS) and ver2.0 (HSTL) interface available through EMIO (programmable logic I/O)**
- **TX/RX checksum offload for TCP and UDP**
- **Internal DMA and wake on LAN**



Summary

Summary

- **The APU is a tightly coupled processing system that is the heart of the Zynq SoC platform**
- **The major components of the APU consist of**
  - Dual-core Cortex-A9 processors
  - Snoop control unit
  - Eight-channel DMA engine
  - Multiple timers
  - General interrupt controller (GIC)
- **A NEON co-processor with built-in FPU is integrated with each processor**
- **L1 and L2 caches enhance DDRx performance by reducing memory latency**

Summary

- **The APU is configured via system-level control registers (SLCR)**
- **Many high-speed and low-speed peripherals comprise the Zynq SoC IOP**
- **MIO signals have dedicated pins on the package and do not require LOC constraints**
- **EMIO signals, when brought to SelectIO, require LOC constraints**


Adaptable.
Intelligent.
XILINX.
© Copyright 2018 Xilinx