Outline

- Primary Concerns of SoC Design
- IP Reuse
  - Definition, IP Integration, IP Reusability
- Platform-Based SoC Design
  - Platform-based design concepts
  - Application Specific Platform & Platform Examples
  - Introduction to AMBA Bus Architecture
A Canonical System IC Architecture

- Microprocessor
- Data Cache
- Instr. Cache
- Memory
- High Speed I/O Ctrl
- Memory Ctrl
- HS IP
- Bus Bridge
- Peripheral Bus
- Timer
- Intr Ctrl
- GPIO
- UART
- LS IP
Primary Concerns of Soc Design

- Time to Market
  - Profit & Territory

- Cost Reduction
  - Profit & Competition

- Application Specification
  - Satisfy current market application & Rapid move to next generation

- IP Reuse
  - IP Verification, IP Reusability, IP Integration
Market/Product Life Cycle

Product Acceptance/Sales volume

- Introduction
- Grow up
- Maturity
- Decline
- Renew
Fundamental System Design Premises

- **Discipline:**
  The design domain need to be restricted to consistently produce scalable, supportable, and easy to integrate designs.

- **Simplicity:**
  All designs have problems. The simpler the design, the easier it is to find and fix them.

- **Locality:**
  Making timing and verification problems local rather than global has a huge pay-off in reducing design time.
Outline

- Primary Concerns of SoC Design
- IP Reuse
  - Definition, IP Integration, IP Reusability
- Platform-Based SoC Design
  - Platform-based design concepts
  - Application Specific Platform & Platform Examples
  - Introduction to AMBA Bus Architecture
Silicon IP

- Silicon Intellectual Properties (Silicon IP)
  - an expression used for the design or intellectual know-how used to make a chip (or IC) function in a given way.
Classification of Silicon IP

- **Software IP**
  - Reusable software/firmware for embedded processors
  - OS, driver, application program, development tools, debugging tools, ...

- **Hardware IP**
  - **Soft IP (soft core/soft macro):**
    - High level description language (i.e. VHDL or Verilog HDL), Synthesizable RTL.
    - Easy to modify functionalities (if well documented).
    - Easy to change processes (if well designed).
  - **Firm IP (firm core/firm macro):**
    - Gate level netlist synthesized for specific library.
    - Hard to modify functionalities.
    - Easy to change processes (if well designed).
  - **Hard IP (hard core/hard macro):**
    - Layout level, GDSII.
    - Very hard to modify functionalities.
    - Fixed processes.
Characteristics of Good Hardware IP

- Configurability:
  - Meet the needs of many different designs.

- Standard Interface:
  - Easy to integrate cores.

- Compliance to Defensive Design Practices:
  - Facilitate timing closure and functional correctness.

- Complete Set of Deliverables:
  - Synthesizable RTL.
  - Complete test benches.
  - Synthesis scripts.
  - Documentation.
Hardware IP Integration

- **Block-based architecture**
  - No standard interfaces
  - Not good for SoC design due to extra efforts to design proprietary interface logic as well as increased complexity to verify system integration.

- **Platform-based architecture**
  - Standard interface depending on platform bus system
  - Plug-n-Play IP
  - Abundant silicon approved IP resources for popular platforms
Block-Based System IC Architecture
IP Reuse in Block-Based Design

IP0

G2

IP1

Core0

IP1/2*

IP0

G5

IP2

Core0

IP1/2**
IP Reuse in Block-Based Design
Block Connection Example-1

- Simple but slow, limited by interface complexity
Block Connection Example-2

IP0

IP1

IP2

IP3

register files

Core
IP Reuse

- It takes much more effort to design an IP for reuse than for casual reuse.

- Many existing IPs were not developed for reuse.

- Need to develop a process of designing reusable IPs.

- IP sources and loyalty.

- IP quality and reliability.
Block Reuse-1

\[ a \in GF(2^8) \], to compute \( a^{-1} \) using square units and multipliers

\[
a^{-1} = a^{128} \cdot a^{64} \cdot a^{32} \cdot a^{16} \cdot a^{8} \cdot a^{4} \cdot a^{2}
\]

\[
= ((((((a^2)^2)^2)^2)^2)^2) \cdot ((((((a^2)^2)^2)^2)^2)^2) \cdot ((((((a^2)^2)^2)^2)^2)^2) \cdot ((((((a^2)^2)^2)^2)^2)^2) \cdot ((((((a^2)^2)^2)^2)^2)^2) \cdot ((((((a^2)^2)^2)^2)^2)^2) \cdot ((((((a^2)^2)^2)^2)^2)^2)
\]

\[
= (a^2)^2 \cdot (a^2)^2 \cdot a^2
\]

7 square units and 6 multipliers
+ a huge combinational logic delay!
\[ a \in GF(2^8), \text{ to compute } a^{-1} \text{ using square units and multipliers} \]

\[ a^{-1} = a^2 \]

\[ \text{clock} \]

\[ (a^2)^2 \cdot a^2 \]

\[ ((a^2)^2)^2 \cdot (a^2)^2 \cdot a^2 \]

\[ (((a^2)^2)^2)^2 \cdot (((a^2)^2)^2)^2 \cdot (a^2)^2 \cdot a^2 \]

\[ (((((a^2)^2)^2)^2)^2)^2 \cdot (((((a^2)^2)^2)^2)^2)^2 \cdot (((a^2)^2)^2)^2 \cdot (a^2)^2 \cdot a^2 \]

\[ (((((a^2)^2)^2)^2)^2)^2 \cdot (((((a^2)^2)^2)^2)^2)^2 \cdot (((a^2)^2)^2)^2 \cdot (((a^2)^2)^2)^2 \cdot (a^2)^2 \cdot a^2 \]

1 square unit1 and 1 multiplier

+ 7 clocks
Outline

- Primary Concerns of SoC Design
- IP Reuse
  - Definition, IP Integration, IP Reusability
- Platform-Based SoC Design
  - Platform-based design concepts
  - Application Specific Platform & Platform Examples
  - Introduction to AMBA Bus Architecture
Definition of a Platform

- A generic platform contains the following components:
  - Bus system (AMBA, PalmBus, …)
  - CPU/MCU, DSP
  - SRAM, DRAM, non-volatile memory, …
  - Basic I/O functions (UART, SPI, USB, PCI, …)

- It’s probably not a good idea to develop an all-purpose (overkilled) platform for IP integration, since different applications requires different CPU/DSP powers, memories, I/Os, etc.
Platform-Based System IC Architecture

- Microprocessor
- Data Cache
- Instr. Cache
- Memory
- High Speed I/O Ctrl
- Bus Bridge
- High Speed Bus
- Peripheral Bus
- Memory Ctrl
- HS IP
- Timer
- Intr Ctrl
- GPIO
- UART
- LS IP
Pros & Cons of Platform-Based Design

- Props:
  - Shorten time to Market.
  - Easy to locate problems in a complicated system where major components are already pre-verified.
  - Easy to extend functions from a basic design.
  - Several popular platforms are already there for use.

- Cons:
  - Predefined platform restricts design flexibility.
  - Hardware design converges. Main differentiation is provided in software.
IP Reuse in Platform-Based Design

- Microprocessor
- Data Cache
- Instr. Cache
- Memory
- High Speed I/O Ctrl
- Memory Ctrl
- HS IP
- HS IP
- Bus Bridge
- Timer
- Intr Ctrl
- GPIO
- SPI
- LS IP

Plug-in and Play

- bus loading
- performance limited by the slowest element

bus loading
performance limited by the slowest element
Point to Point with Analog Block

- Microprocessor
- Memory
- USB Phy
- USB PIE
- HS IP

Point to Point
Need to guarantee only one master is active at the same time during operation
Special cares need to be taken in DFT program.
Bus floating/keeping issues
Mux Bus

- Need additional routing area
On-Chip Bus Suggestions

- Multiplexer-based bus architecture.
- A single-clock-edge, flip-flop based architecture.
- Separate data and address buses.
- Separate control buses.
- Support multiple masters.
Outline

- Primary Concerns of SoC Design
- IP Reuse
  - Definition, IP Integration, IP Reusability
- Platform-Based SoC Design
  - Platform-based design concepts
  - Application Specific Platform & Platform Examples
  - Introduction to AMBA Bus Architecture
Application Specific Platform

The specification of an ASIC (Application-Specific Integrated Circuit) chip often includes application adaptability due to the following factors:
- Significant market shift,
- Rapid evolved standards,
- Product differentiation,
- Hardware reuse.

An application specific platform contains:
- Application specification and future roadmap.
- A generic platform (bus system, processor, memory) + pre-integrated IP for the specified application.
- Complete verification methodology.
- Trade off among performance, cost programmability and configurability.
Platform Design Objectives:
Design the platform to support multiple applications

Platform Design Objectives:
Design the platform to support multiple applications
Platform Selection:
For a given application, select the best platform in terms of performance, cost, etc.
ARM PrimeXsys Platform

Sys Ctrl. Timer Watch dog

VIC

ARM Caches TCMs

MPMC

Multi-Layer AHB/ APB

GPIO RTC

UART SCI SSP

CLCD

DMAC

SMC
AWM Wireless Multimedia Platform:
Xpert-GPS 3000 Platform
MIPS Platform
PowerPC CoreConnect Platform
Outline

- Primary Concerns of SoC Design
- IP Reuse
  - Definition, IP Integration, IP Reusability
- Platform-Based SoC Design
  - Platform-based design concepts
  - Application Specific Platform & Platform Examples
  - Introduction to AMBA Bus Architecture
AMBA history

- **AMBA 1.0 - ASB+APB in 1995**
  - First generation, tri-state bus

- **AMBA 2.0 - AHB+APB in 1999**
  - Second generation, mux bus

- **AMBA 3.0 - AXI +AHB+APB in 2003**
  - Third generation, more than 30 participants

<table>
<thead>
<tr>
<th>Agere Systems</th>
<th>Fujitsu</th>
<th>NEC Electronics (Europe)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agilent</td>
<td>Hewlett-Packard Company</td>
<td>OKI Electric Industry</td>
</tr>
<tr>
<td>ARM</td>
<td>Infineon</td>
<td>Philips Semiconductors</td>
</tr>
<tr>
<td>Atmel</td>
<td>LSI Logic</td>
<td>QUALCOMM</td>
</tr>
<tr>
<td>Cadence</td>
<td>Mentor Graphics</td>
<td>Samsung</td>
</tr>
<tr>
<td>Conexant Systems</td>
<td>Matsushita</td>
<td>STMicroelectronics</td>
</tr>
<tr>
<td>CoWare Inc.</td>
<td>Micronas</td>
<td>Synopsys</td>
</tr>
<tr>
<td>Epson</td>
<td>Motorola</td>
<td>Toshiba Corporation</td>
</tr>
<tr>
<td>Ericsson Mobile Platforms</td>
<td>NEC Electronics Corporation</td>
<td>Verisity</td>
</tr>
</tbody>
</table>
Basic AMBA(1.0,2.0)-Based Architecture

AMBA AHB
- High Performance
- Pipeline operation
- Multiple bus masters
- Burst transfers
- Split transactions

AMBA ASB
- High Performance
- Pipeline operation
- Multiple bus masters
- Burst transfers

AMBA APB
- Low power
- Latched address and control
- Simple interface
- Suitable for many peripherals
AMBA2.0 AHB

- **Features**
  - burst transfers
  - split transaction
  - single-cycle bus master handover
  - single-clock edge operation
  - non-tristate implementation
  - wide data bus configuration s(8/16/32/64/128 bits)

- **Bus Components**
  - AHB master: A bus master is able to initiate read and write operations.
  - AHB slave: A bus slave responds to a read or write operation within a given address range.
  - AHB arbiter: The bus arbiter ensures that only one bus master at a time is allowed to initiate data transfers.
  - AHB decoder: The AHB decoder is used to decode the address of each transfer and provide a select signal for the involved slave.
AHB2.0 Mux Bus Interconnection
Overview of AMBA AHB Operation

1. A bus master gets granted access to the bus by the bus arbiter.
2. The granted bus master starts an AHB transfer by driving the address and control signals.
   - Address
   - Direction
   - Data width
   - Burst Indication
3. Every transfer consists of:
   - An address and control cycle
   - One or more data cycles
4. During a transfer the slave shows the status using the response signals, HRESP[1:0]:
   - OKAY
   - ERROR
   - RETRY
   - SPLIT
AHB: Simple Transfer

2-level pipeline:
The address phase of any transfer occurs during the data phase of the previous transfer.
AHB: Transfer with Wait States
AHB: Multiple Transfers
AHB: Transfer Type

- Four transfer types indicated by HTRANS[2:0]:
  - **IDLE**
    No data transfer is required.
  - **BUSY**
    The BUSY transfer type allows bus masters to insert IDLE cycles in the middle of bursts of transfers.
  - **NONSEQ**
    Indicates the first transfer of a burst or a single transfer.
  - **SEQ**
    The remaining transfers in a burst are SEQUENTIAL and the address is related to the previous transfer.
AHB: Transfer Type Example
### AHB: Control Signals

- **Transfer direction:**
  
  HWRITE = high for write; HWRITE = low for read

- **Transfer Size:**

<table>
<thead>
<tr>
<th>HSIZE[2:0]</th>
<th>Size</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>8bits</td>
<td>Byte</td>
</tr>
<tr>
<td>001</td>
<td>16bits</td>
<td>Halfword</td>
</tr>
<tr>
<td>010</td>
<td>32bits</td>
<td>Word</td>
</tr>
<tr>
<td>011</td>
<td>64bits</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>128bits</td>
<td>4-word line</td>
</tr>
<tr>
<td>101</td>
<td>256bits</td>
<td>8-world line</td>
</tr>
<tr>
<td>110</td>
<td>512bits</td>
<td></td>
</tr>
<tr>
<td>111</td>
<td>1024bits</td>
<td></td>
</tr>
</tbody>
</table>
## AHB: Control Signals

- **Protection control**

<table>
<thead>
<tr>
<th>HPROT[3:0]</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>- - - 0</td>
<td>Opcode fetch</td>
</tr>
<tr>
<td>- - - 1</td>
<td>Data access</td>
</tr>
<tr>
<td>- - 0 -</td>
<td>User assess</td>
</tr>
<tr>
<td>- - 1 -</td>
<td>Privileged access</td>
</tr>
<tr>
<td>- 0 - -</td>
<td>Nit bufferable</td>
</tr>
<tr>
<td>- 1 - -</td>
<td>Bufferable</td>
</tr>
<tr>
<td>0 - - -</td>
<td>Not cacheable</td>
</tr>
<tr>
<td>1 - - -</td>
<td>Cacheable</td>
</tr>
</tbody>
</table>
AHB: Address Decoding

M1 -> Address
M2 -> Address
M3 -> Address

Decoder

SEL

S1
S2
S3
S4
AHB: Data Buses

- HWDATA[31:0]: Write data bus; HRDATA[31:0]: Read data bus.
- Active byte lanes for a 32-bit little-endian data bus

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Word</td>
<td>0</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Halfword</td>
<td>0</td>
<td>●</td>
<td>●</td>
<td></td>
<td>●</td>
</tr>
<tr>
<td>Halfword</td>
<td>2</td>
<td>●</td>
<td></td>
<td>●</td>
<td></td>
</tr>
<tr>
<td>Byte</td>
<td>0</td>
<td>●</td>
<td></td>
<td></td>
<td>●</td>
</tr>
<tr>
<td>Byte</td>
<td>1</td>
<td></td>
<td></td>
<td>●</td>
<td></td>
</tr>
<tr>
<td>Byte</td>
<td>2</td>
<td></td>
<td>●</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Byte</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td>●</td>
</tr>
</tbody>
</table>
AHB: Data Buses

- Active byte lanes for a 32-bit big-endian data bus

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Word</td>
<td>0</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
</tr>
<tr>
<td>Halfword</td>
<td>0</td>
<td>■</td>
<td>■</td>
<td></td>
<td>■</td>
</tr>
<tr>
<td>Halfword</td>
<td>2</td>
<td></td>
<td>■</td>
<td>■</td>
<td></td>
</tr>
<tr>
<td>Byte</td>
<td>0</td>
<td>■</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Byte</td>
<td>1</td>
<td></td>
<td>■</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Byte</td>
<td>2</td>
<td></td>
<td></td>
<td>■</td>
<td></td>
</tr>
<tr>
<td>Byte</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td>■</td>
</tr>
</tbody>
</table>
AHB: Arbitration

- **HBUSREQx:**
  Bus request from a bus master.

- **HLOCKx:**
  Bus lock from a bus master.

- **HGRANTx:**
  Grant signal generated by the bus arbiter.

- **HMASTER[3:0]:**
  Current bus master indicator.

- **HMASTERLOCK:**
  Lock indicator for the current bus master.

- **HSPLIT[15:0]:**
  Indicator to show which bus master can complete a SPLIT transaction.
AHB: Granting with No Wait States
AHB: Granting with Wait States
AHB: a Narrow Slave on a Wide Bus
AHB: a Wide Slave on a Narrow Bus
AMBA APB

- Low power
- Latched address and control
- Simple interface
- Suitable for many peripherals
APB: State Diagram

- **IDLE**: The default state.
  - PSELx=0
  - PENABLE=0

- **SETUP**: When a transfer is required the bus moves into setup state, where the appropriate select signal, PSELX, is asserted.
  - PSELx=1
  - PENABLE=0

- **ENABLE**: The enable signal, PENABLE is asserted.
  - PSELx=1
  - PENABLE=1

No transfer → Transfer

Transfer → No transfer

[Diagram]
APB: Write Transfer
APB: Read Transfer
Interfacing APB to AHB: Read
Interfacing APB to AHB: Write
Why AMBA 3.0 AXI Protocol

- Develop for future SoC designs demand:
  - High-bandwidth and low-latency
  - High-frequency without complex bridge
  - Flexibility in interconnect architecture
  - Reduction of system power consumption

- Retaining existing AMBA strengths:
  - Backward compatible with existing AHB and APB
  - Modular design, component re-use
  - Simple to understand interface
AXI Protocol interface and interconnect

- Single interface definition for describing interface between:
  - A master and the interconnect
  - A slave and the interconnect
  - A master and a slave

- Interconnect approaches
  - Shared address and data buses
  - Shared address buses and multiple data buses
  - Multilayer, with multiple address and data bus
AMBA 3.0 AXI Protocol

- Features
  - Channel architecture
  - Registers Slices
  - Burst addressing
  - Multiple outstanding bursts
  - Out of order completion
  - Shared bus, multi-layer and mixed

- More than just a bus protocol
  - RTL, Modelling and Verification environments
Read Channel Architecture

Master

Address/Control
Read address channel

Slave

Read data channel
Read data  Read data  Read data  Read data
Write Channel Architecture

Master

<table>
<thead>
<tr>
<th>Address/Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write address channel</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Write data channel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read data</td>
</tr>
</tbody>
</table>

Slave

Write response channel

Write response
Register Slices

M1

M2

M3

Arbiter

Address/ control

Write data

Read data

Decoder

S1

S2

S3

S4
Trade-off between Latency and Frequency

- Register slices can be added at almost any point within a given connection

- Use a direct connection between a processor and high-speed memory. Use simple register slices to isolate a long path to less performance-critical peripherals.

- Allow maximum frequency of operation by matching channel latency to channel delay.
**AMBA 2.0 AHB Burst**

- **AHB Burst**
  - Address and Data are locked together
  - Single pipeline stage
  - HREADY controls intervals of address and data
AMBA 3.0 AXI Burst

- AXI Burst
  - One Address for entire burst
AXI - Outstanding Transactions

- **AXI Burst**
  - One Address for entire burst
  - Allows multiple outstanding addresses
Out of Order Interface

- Each transaction has an ID attached
  - Channels have ID signals - AID, RID, etc.

- Transactions with the same ID must be ordered

- Requires bus-level monitoring to ensure correct ordering on each ID
  - Masters can issue multiple ordered addresses
AMBA 2.0 AHB Burst - Slow slave

- With AHB
  - If one slave is very slow, all data is held up.
- Out of order completion allowed
- Fast slaves may return data ahead of slow slaves
- Complex slaves may return data out of order
### AXI - Data Interleaving

<table>
<thead>
<tr>
<th>ADDRESS</th>
<th>A11</th>
<th>A21</th>
<th>A31</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>DATA</th>
<th>D21</th>
<th>D22</th>
<th>D11</th>
<th>D23</th>
<th>D12</th>
<th>D31</th>
<th>D13</th>
<th>D14</th>
</tr>
</thead>
</table>

- Returned data can even be interleaved
- Gives maximum use of data bus
- Note - Data within a burst is always in order
AXI Multi-layer

- Parallel paths between masters and slaves
- Key Advantages
  - Increased bandwidth
  - Design flexibility
- Uses the same interface protocol
AMBA Class Library

- Defined method for modelling AXI components
  - Data structure used to fully describe a transaction
  - Standard method calls for generating and receiving transactions
- Modelling abstraction levels
  - Programmer view level
  - Programmer’s view + timing
  - Cycle callable
Summaries

- Key factors for a successful system IC design: time to market, cost reduction, application specification and IP reuse.

- A good IP must have the following properties- configurability, standard interface, compliance to defensive design practices and complete set of deliverables.

- Mux bus is recommended for on-chip bus, while tri-state bus is recommended for on-board bus.

- Platform-based architecture is regarded as the only way for complex system chip design.

- Soc platform bus example- from AMBA 2.0 to AMBA3.0.