SoC Architecture Design Approaches

Hung-Chih Chiang
Outline

- IP-Based System IC Design
  - Block-Based Architecture

- Platform-Based Architecture
  - Platform-based design concepts
  - Platform examples
  - Introduction to AMBA bus
A Canonical System IC Architecture

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

■ Soft IP (soft core/soft macro):
  ■ 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.
  ■ Hard to modify functionalities.
  ■ Easy to change processes (if well designed).

■ Hard IP (hard core/hard macro):
  ■ GDSII.
  ■ Very hard to modify functionalities.
  ■ Fixed processes.
IP Sources

- In company IP Libraries
- Cooperation Partners
- IP Providers
- Free IP (typically buggy, need some efforts to get them work)
IP Reuse Issues

- 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.
Characteristics of Good 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.
Outline

- IP-Based System IC Design
  - Block-Based Architecture

- Platform-Based Architecture
  - Platform-based design concepts
  - Platform examples
  - Introduction to AMBA bus
Block-Based System IC Architecture

IP0

G0

Core0

IP1

G1

IP1*

IP0*

G2

Core0

IP1/2*

IP1/2*
IP Reuse in Block-Based Design

IP0 → G0 → Core0 → G1 → IP1
IP0* → G3 → Core1 → G4 → IP1*
IP0** → G0 → Core0
IP1** → G1 → IP1
IP Reuse in Block-Based Design

- IP0
- Core0
- IP1
- G2

- IP1/2*

- IP0
- Core0
- IP2
- G5

- IP1/2**
Block Connection Example-1

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

IP0

IP1

IP2

IP3

Core

register files
Block Reuse-1

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

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

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

\[ a^{-1} = a^2 \quad \text{clock} \]
\[ (a^2)^2 \cdot a^2 \]
\[ ((a^2)^2)^2 \cdot (a^2)^2 \cdot a^2 \]
\[ (((a^2)^2)^2)^2 \cdot ((a^2)^2)^2 \cdot (a^2)^2 \cdot a^2 \]
\[ ((((a^2)^2)^2)^2)^2 \cdot (((a^2)^2)^2)^2 \cdot ((a^2)^2)^2 \cdot (a^2)^2 \cdot a^2 \]
\[ (((((a^2)^2)^2)^2)^2)^2 \cdot (((((a^2)^2)^2)^2)^2) \cdot ((a^2)^2)^2 \cdot (a^2)^2 \cdot a^2 \]
\[ ((((((a^2)^2)^2)^2)^2)^2)^2 \cdot (((((a^2)^2)^2)^2)^2) \cdot ((a^2)^2)^2 \cdot (a^2)^2 \cdot a^2 \]

1 square unit and 1 multiplier
+ 7 clocks
Outline

- IP-Based System IC Design
  - Block-Based Architecture

- Platform-Based Architecture
  - Platform-based design concepts
  - Platform examples
  - Introduction to AMBA bus
Platform-Based System IC Architecture

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

- Application specification and future roadmap.
- Well defined bus system, processor, memory, and other IPs.
- Complete verification methodology.
- Trade off among performance, cost programmability and configurability.
IP Reuse in Platform-Based Design

Plug-in and Play

• bus loading
• performance limited by the slowest element
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 to IP Interface (Point to Point)
IP to IP interface (SoC)
Point to Point with Analog Block

Diagram showing the integration of a Microprocessor, Memory, USB PHY, and PIE with a Point to Point connection.

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

- IP-Based System IC Design
  - Block-Based Architecture

- Platform-Based Architecture
  - Platform-based design concepts
  - Platform examples
  - Introduction to AMBA bus
AWM Wireless Multimedia Platform:
Xpert-GPS 3000 Platform
MIPS Platform
Outline

- IP-Based System IC Design
  - Block-Based Architecture

- Platform-Based Architecture
  - Platform-based design concepts
  - Platform examples
  - Introduction to AMBA bus
Basic AMBA-Based Architecture

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

- High-performance ARM processor
- High-bandwidth on-chip RAM

- UART
- Timer

- Keypad
- PIO

AMBA AHB

- High Performance
- Pipeline operation
- Multiple bus masters

AMBA ASB

- High Performance
- Pipeline operation
- Multiple bus masters

AMBA APB

- Low power
- Latched address and control
- Simple interface
- Suitable for many peripherals
Introduction to AMBA 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.
AHB 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: Burst Operations

- Burst information is provided using HBURST[2:0]:

<table>
<thead>
<tr>
<th>HBURST[2:0]</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>SINGLE</td>
<td>Single transfer</td>
</tr>
<tr>
<td>001</td>
<td>INCR</td>
<td>Incrementing burst of unspecified length</td>
</tr>
<tr>
<td>010</td>
<td>WRAP4</td>
<td>4-beat wrapping burst</td>
</tr>
<tr>
<td>011</td>
<td>INCR4</td>
<td>4-beat incrementing burst</td>
</tr>
<tr>
<td>100</td>
<td>WRAP8</td>
<td>8-beat wrapping burst</td>
</tr>
<tr>
<td>101</td>
<td>INCR8</td>
<td>8-beat incrementing burst</td>
</tr>
<tr>
<td>110</td>
<td>WRAP16</td>
<td>16-beat wrapping burst</td>
</tr>
<tr>
<td>111</td>
<td>INCR16</td>
<td>16-beat incrementing burst</td>
</tr>
</tbody>
</table>
AHB: Four-Bit Wrapping Burst
AHB: Four-Bit Incrementing Burst
AHB: Undefined-Length Burst
AHB: Control Signals

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

- **Transfer Size:**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8 bits</td>
<td>Byte</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>16 bits</td>
<td>Halfword</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>32 bits</td>
<td>Word</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>64 bits</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>128 bits</td>
<td>4-word line</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>256 bits</td>
<td>8-word line</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>512 bits</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1024 bits</td>
<td></td>
</tr>
</tbody>
</table>
AHB: Control Signals

- Protection control

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Opcode fetch</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>Data access</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>0</td>
<td>-</td>
<td>User access</td>
</tr>
<tr>
<td>-</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>Privileged access</td>
</tr>
<tr>
<td>-</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>Not bufferable</td>
</tr>
<tr>
<td>-</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>Bufferable</td>
</tr>
<tr>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Not cacheable</td>
</tr>
<tr>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Cacheable</td>
</tr>
</tbody>
</table>
AHB: Address Decoding
AHB: Transfer with Retry Response
AHB: Error Response
**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.

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

- **ENABLE**: The enable signal, PENABLE is asserted.
APB: Write Transfer

T1 T2 T3 T4 T5

PCLK

PADDR

PWRITE

PSEL

PENABLE

PWDATA

Addr 1

Data 1
APB: Read Transfer
Interfacing APB to AHB: Read
Interfacing APB to AHB: Write
Summaries

- System IC design principles - discipline, simplicity and locality.

- Properties of a good IP - configurability, standard interface, compliance to defensive design practices and complete set of deliverables.

- Register files - a commonly used interface to link IPs for block-based design.

- Platform-based design can significantly shorten time to market compared to block-based design.

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

- A good Soc platform bus example - AMBA 2.0.