SoC BUSSES
What Is a Bus?

• A Bus is:
  • Shared communication link.
  • Single set of wires used to connect multiple subsystems.

• A bus is also a fundamental tool for composing large, complex systems.
  • Systematic means of abstraction.
Busses
Advantages of Buses

- Versatility:
  - New devices can be added easily.
  - Peripherals can be moved between computer systems that use the same bus standard.

- Low cost:
  - A single set of wires is shared in multiple ways.
Disadvantages of Buses

- It creates a communication bottleneck.
  - Bandwidth of that bus can limit the maximum I/O throughput.
- The maximum bus speed is largely limited by:
  - The length of the bus.
  - The number of devices on the bus (and, hence, bus loading).
  - The need to support a range of devices with:
    - Widely varying latencies.
    - Widely varying data transfer rates.
General Organization of a Bus

- Control lines:
  - Signal requests and acknowledgements
  - Indicate what type of information is on the data lines
- Data lines carry information between the source and the destination:
  - Data and addresses
  - Complex commands
Bussing Strategies

• Register-to-Register Communications:
  • Point-to-point,
  • Single shared bus.
  • Multiple special purpose busses.
• Tradeoffs between datapath/control complexity and amount of parallelism supported by the hardware.
Master vs. Slave

- A bus transaction includes two parts:
  - Issuing the command (and address) - request
  - Transferring the data - action.
- Master is one who start the bus transaction by:
  - Issuing the command (and address).
- Slave is the one who responds to the address by:
  - Sending data to the master ask for data.
  - Receiving data from the master if the master wants to send data.
What Defines a Bus?

- Transaction Protocol
- Timing and Signaling Specification
- Bunch of Wires
- Electrical Specification
- Physical / Mechanical Characteristics – the connectors
Bus Protocols

• Protocols determine:
  • The transactions that are supported.
  • The timing of their cycles.
  • How modules are addressed.
  • Allocation of resources.

• Arbitration determines what module gets to use the bus.
  • Modules make request.
  • Arbiter grants the bus to one requester.
    • Fixed priority.
    • Round robin.
    • Random.

• Preemption.
  • Interrupt long transaction to run more critical operation.

• Arbitration is often pipelined with bus use.
Simplest Bus Paradigm

• All agents operate synchronously.
• All can source/sink data at same rate
• Simple protocol.
  • Just manage the source and target
Simple Synchronous Protocol

- Even memory busses are more complex than this.
  - Memory (slave) may take time to respond.
  - It may need to control data rate.
Communication Architectures

- **SHARED BUS** (broadcast);
  - Low area
  - Poor scalability
  - High energy consumption

- **NETWORK on CHIP** (point-to-point)
  - Scalability;
  - Low energy consumption;
  - High area.

---

**Shared bus is state-of-the-art**

**Network** ⇒ **future**
Standard “Bus” Architecture

- Large semiconductor firms
  - CoreConnect (IBM)
  - STBUS (STMicroelectronics)
- Core vendors
  - AMBA (ARM Ltd.)
  - AIX (Xilinx)
- Interconnect IP vendors
  - CoreFrame (Palmchip)
  - WishBone (Silicore)
  - SiliconBackPlane (Sonics)
- Many others!
Synchronous vs. Asynchronous

- In a *synchronous* bus: operations are synchronized to a global *bus clock*
- In an *asynchronous* bus: control signal edges signal bus events
- On-chip buses are generally synchronous

⇒ *But asynchronous busses are emerging*
Bus Components: terminology

- Initiator
  - The FU that initiates transactions
- Target
  - Responds to incoming transaction
- Master/Slave
  - The initiator/target side of the bus interface
- Arbiter
  - Controls the access to the bus
- Bridge
  - Connects two buses
  - It acts as an initiator on one side and a target on the other
Bus Arbitration

- Buses can support multiple initiators
- Need a protocol for allocating bus resources and resolving conflicts
  - *Bus arbitration*
- Need a decision procedure to choose
  - *Arbitration policy*
Bus Arbitration

- Example:
  - Each bus master has pair of dedicated REQ and GNT signals
  - Needs at least 1 p2p control signal (REQi)
Bus Attributes

- **Latency**
  - Time required to initiate a transaction on the bus
- **Maximum Bandwidth**
  - Maximum capacity for data transfer in bit/s
- **Effective Bandwidth**
  - Accounts for contention
- **Energy per bit**
  - Average energy for transferring one bit of information
- **Pipelining**
  - Splitting of a bus transaction over multiple clock cycles
- **Endianness**
  - Determines how bytes are ordered in a word
Pipelining Bus Transactions

- Example of a 5-stage pipeline
- 5 outstanding transactions
- Increase throughput but increase complexity
WISHBONE
WISHBONE System-on-Chip Interconnect Architecture

- Simple architecture
- Truly open specification
  - Silicore placed specifications in the public domain
  - In September 2002 Silicore handed stewardship over to OpenCores
- Patent & Royalty free
WISHBONE
A Primer

• Multi-Master, Multi-Slave bus
• Interconnect independent architecture
• Synchronous bus
  • All signals are triggered on the rising edge of the clock
• Flexible bus
  • Masters and Slaves interfaces can be as complex as the designer wants/requires
WISHBONE Interfaces

- SYSCON
  - Generates clock and reset
- INTERCON
  - IP core that connects one or more masters to one or more slaves
- MASTER
  - Interface that is capable of generating bus cycles
- SLAVE
  - Interface that is capable of responding to bus cycles
WISHBONE
Peer-to-Peer Example
Summary

• Busses are an important technique for building large-scale systems.
  • Their speed is critically dependent on factors such as length, number of devices, etc.
  • Critically limited by capacitance.
  • Tricks: esoteric drive technology such as GTL.
• Important terminology:
  • Master: the device that can initiate new transactions.
  • Slaves: devices that respond to the master.
• Two types of bus timing:
  • Synchronous: bus includes clock.
  • Asynchronous: no clock, just REQ/ACK strobing.
• System-on-a-chip approach invites new solutions.
  • Well-defined and clear communication protocols.
  • Physical layer hidden to designer.
Shared Bus or NoC?

Shared bus throughput decline