Hands on with FPGA's: Module 6

Venkat Rangan

Questions on Module 5

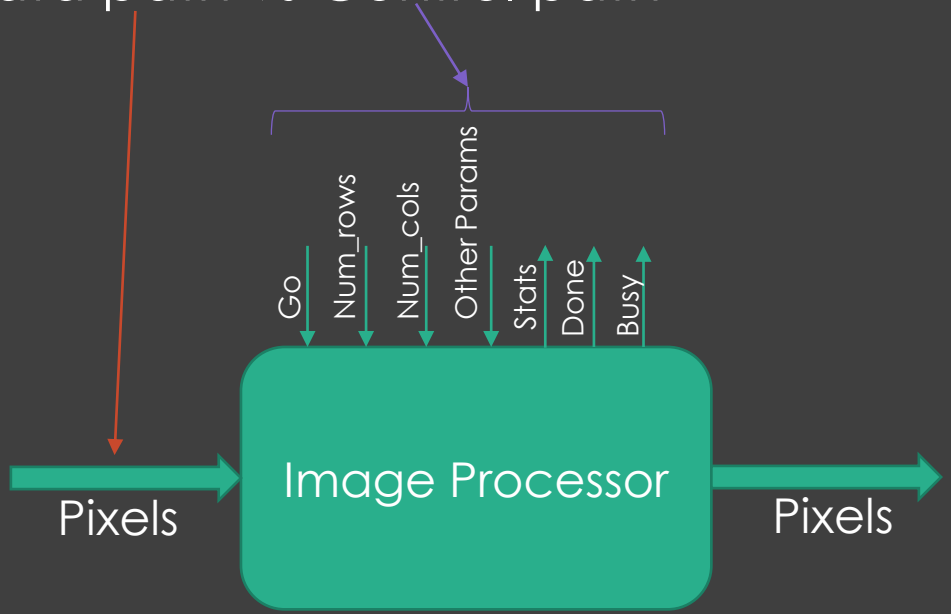
- Alarm clock

Topics

- Pre-class: Open floor for questions
  - Module 5
- Putting blocks together
  - Busses
  - NoC
  - Protocols
- FPGA Specifics
- Open discussion

Typical HW Design

- Code up small blocks, target reuse
- Interconnect blocks
- Data path vs Control path

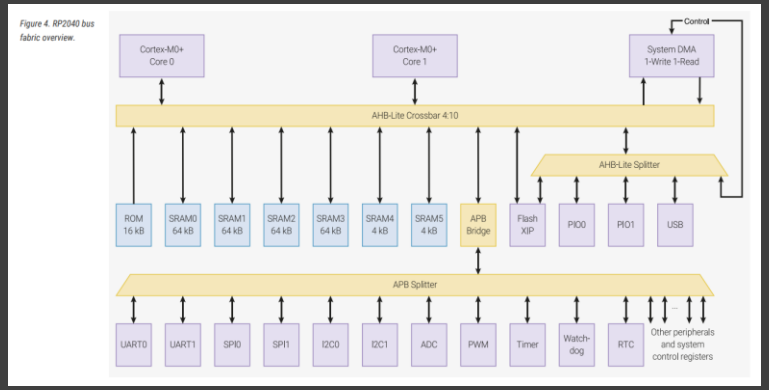


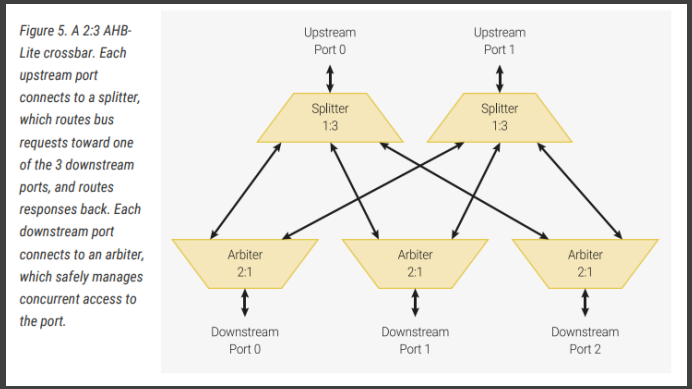
Busses

- Common interface specification
  - Fast design with fewer errors
- Unidirectional data flow
- Clearly defined accesses
  - Single transactions
  - Burst transactions
    - More efficient than single xactions
    - Well matched to external memories
- E.g. Wishbone Specification, AXI, Avalon…

The A in AXI, AHB, APB…

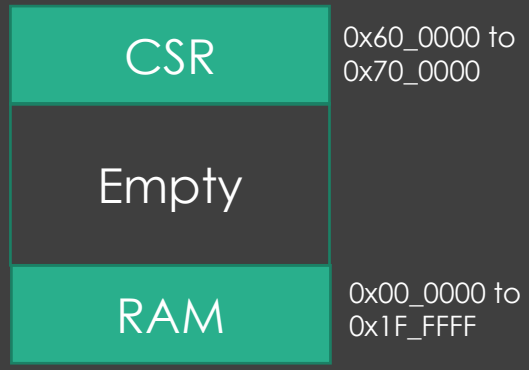
- AMBA: ARM specification for on-chip interconnect

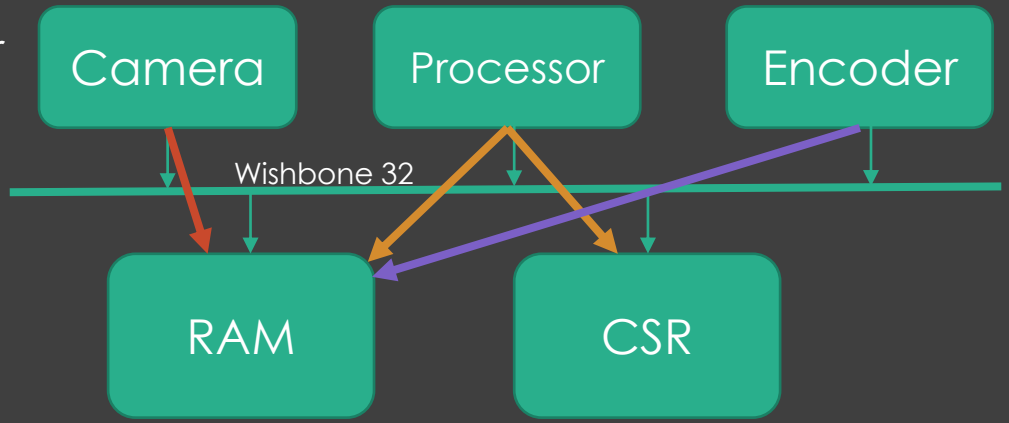




Lets build a bus!

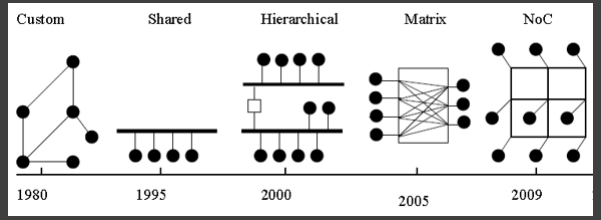
- Use wb_intercon script
- Actual blocks not implemented
- CSR: Config, Status Registers
- 32 bits wide, good enough for most configuration type access



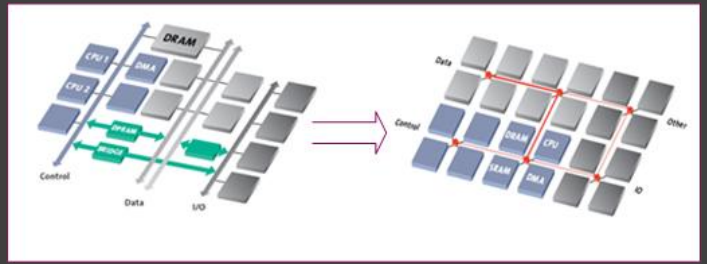


NoC vs. Busses

- Busses breakdown when there are many blocks
- Hierarchy added to alleviate but adds delay
- Network-on-Chip used for higher bandwidths
  - Highly scalable and easily reconfigurable
  - FLIT, PHIT
- https://www.design-reuse.com/articles/10496/a-comparison-of-network-on-chip-and-busses.html
- Open Source NoC using AXI



https://ignitarium.com/network-on-chip-an-overview/



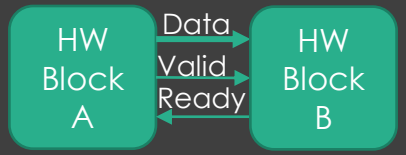
https://ignitarium.com/network-on-chip-an-overview/

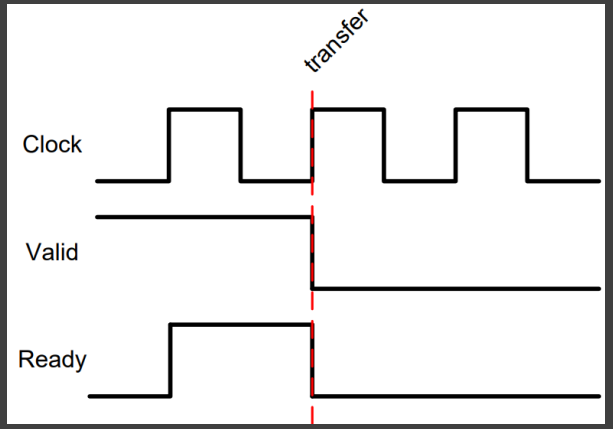
Data Path Interconnects

- Data only
  - Assumes data is valid at every clock cycle
- Data + Valid
  - Data bus is used only when valid is active
- Data + valid + ready
  - Can flow control back to source
  - Most flexible and easily extensible
- Can also use NoC
  - Higher overhead but good for large chips

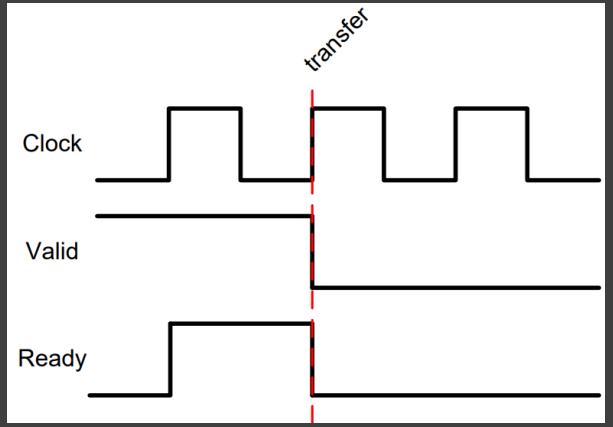
Valid-Ready Protocol

Adapted from https://inst.eecs.berkeley.edu/~cs150/Documents/Interfaces.pdf

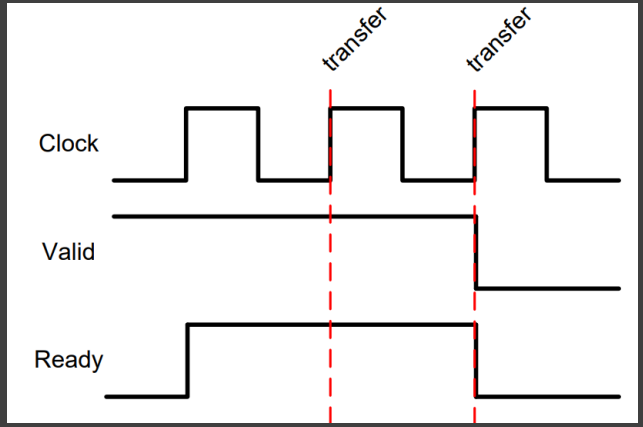




- Rule1: Data transfer happens when both Valid and Ready are active
- Rule 2: No valid teasing: Once valid, cannot take away valid till ready is high
- Rule 3: Ready can tease

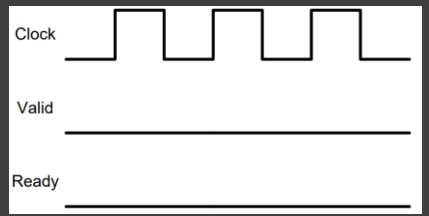


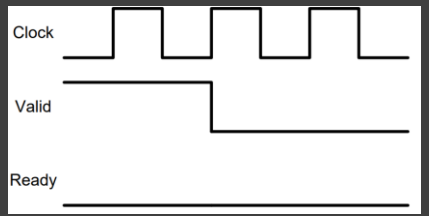
Single Data Transfer

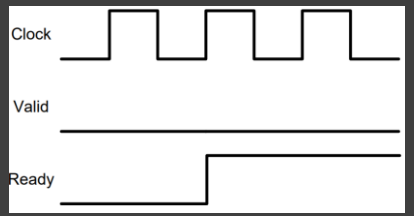


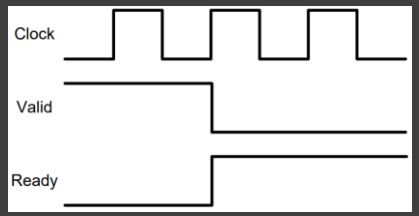
Multiple data transfer
(100% throughput)

No Data Transfer









FPGA Backend Process:

With great control comes a great amount of work

1. Select FPGA Part: Make sure of the right part number!
2. Allocate pins
3. Specify clock(s)
4. Specify detailed Input/Output constraints
5. Place:
   - Synthesized netlist is placed using complex algorithms
6. Route:
   - Wiring added between placed cells
7. Bitgen
   - Final binary generated that can be programmed into the FPGA

Pin Allocation: Not trivial!

- Choose IO Bank:
  - All pins in a bank share the same IO voltage
  - Some banks have faster IO than others
- Special purpose pins
  - Clock input (Low capacitance)
  - Special routing inside the FPGA (e.g. to PLL)
  - DDR requires a DQS signal, usually a special purpose pin
- Differential pair signals
- IO Specification
  - IO Standard: LVCMOS18, LVCMOS33…
  - Pull up/Down
  - Slew rate
  - Open Drain
- Go through FPGA datasheet in detail!

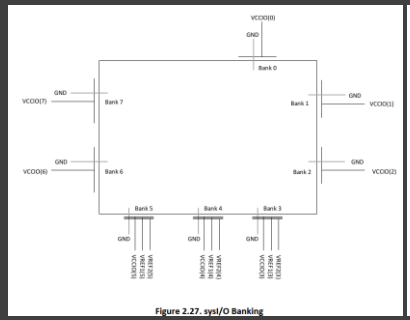


Figure 2.27. sysI/O Banking

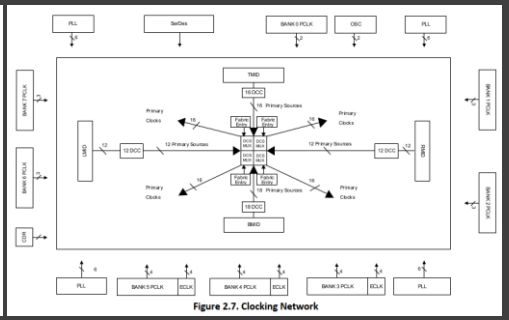
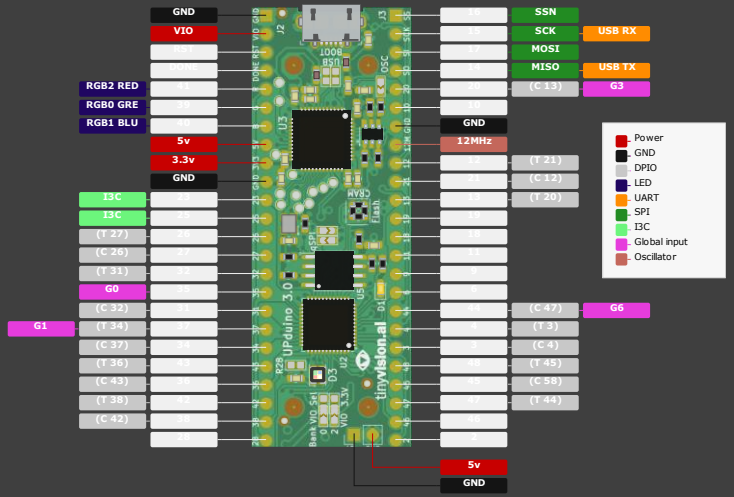


Figure 2.7. Clocking Network



https://github.com/nobodywasishere/upduino-pinout

Clock and related Specifications

- Whats the expected clock rate?
- How should IO be clocked?
  - How much delay does the board have?
- Special constraints on logic

- Can get very complex…

Module 6:

- Challenge: Go through a non-trivial FPGA design backend

Open Discussion