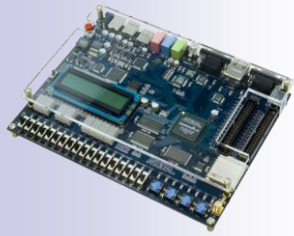
Struktury počítačových systémů

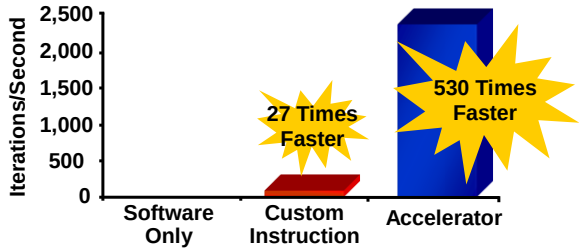


Přednáší: Richard Šusta

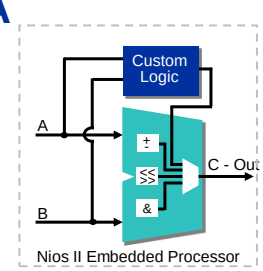
8. prosince 2011 verze 1.0

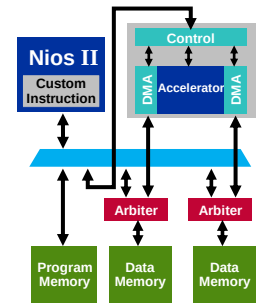
ČVUT-FEL v Praze – kód předmětu A0B35SPS

Accelerating Software in FPGA



- **Accelerator running 64Kb CRC at 100 MHz (**CRC can be easily calculated on a Linear Feedback Shift Register, see the next lecture**)**



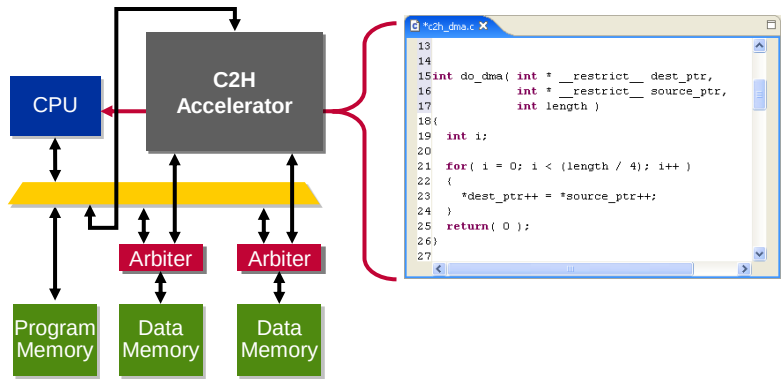


ALTERA

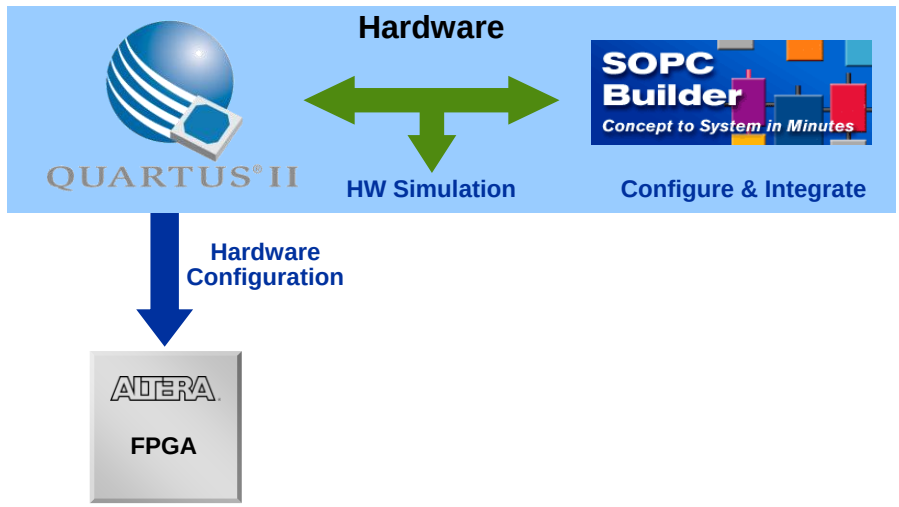
[»]

Altera C2H – the SW Accelerator Solution

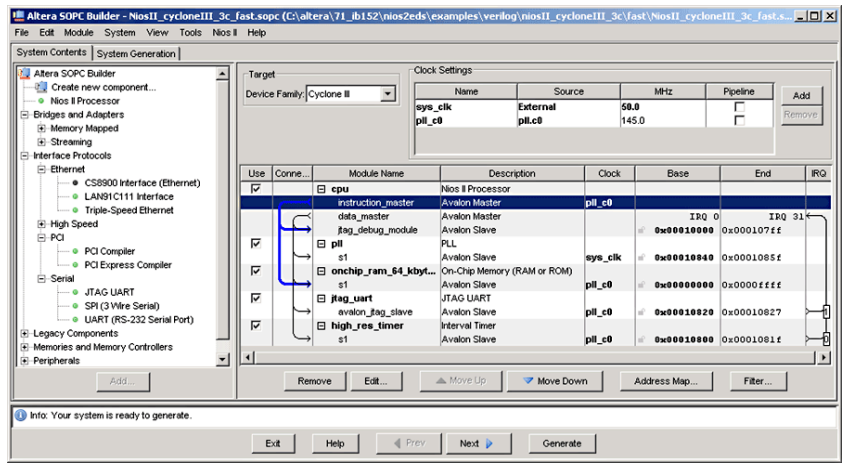
- Generates and integrates a custom hardware accelerator from an ANSI C function.



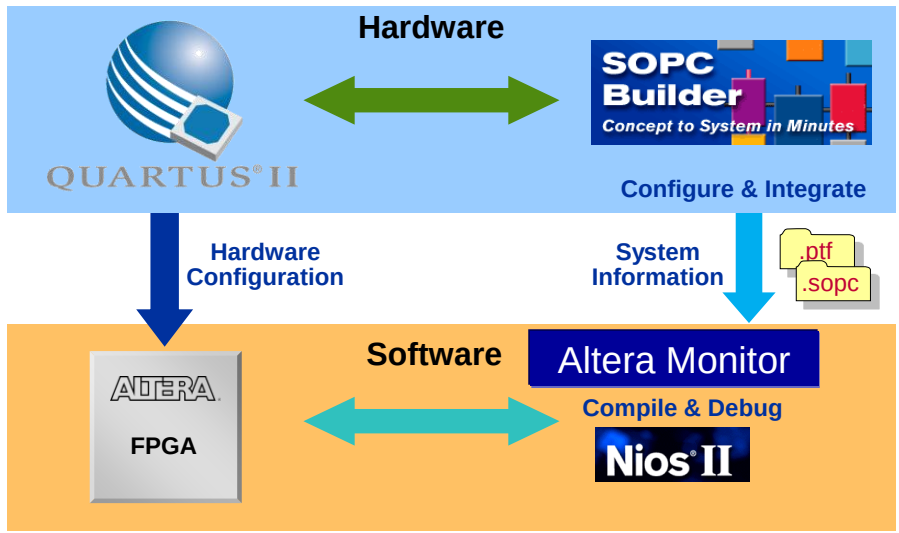
Hardware Development Flow

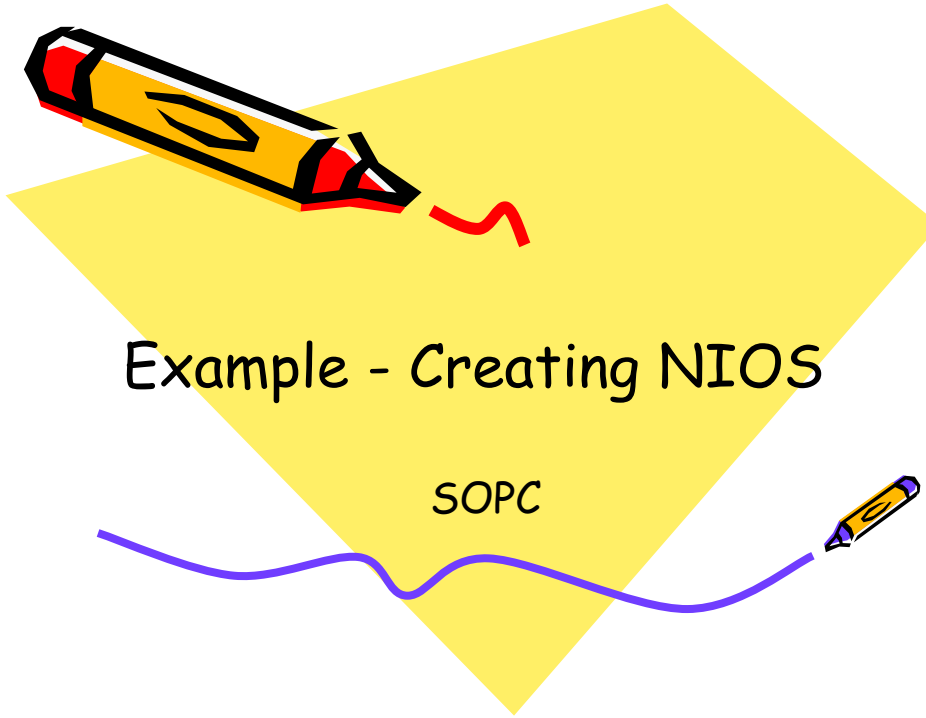


Creating Systems With SOPC Builder
- we will show in the next lecture



Embedded Development Flow





DE2 Media Computer is also logic circuits 1/2

1. Copy DE2 Media Computer to new location, i.e. copy directory

   C:\altera\91\University_Program\NiosII_Computer_Systems\DE2\DE2_Media_Computer

   to new location, e.g. **C:\Workdir\DE2_My_MediaComp**
   Never change original NiosII_Computer_System(s) in C:\altera

2. *Open Quartus project C:\workdir**DE2_My_MediaComp**\DE2_Media_Computer.qpf*
3. *In Quartus Project Navigator, open file **DE2_Media_Computer.vhd***
4. *Create symbol file of your **DE2_Media_Computer.vhd** by "File->Create/Update->Create symbol file for current file"*

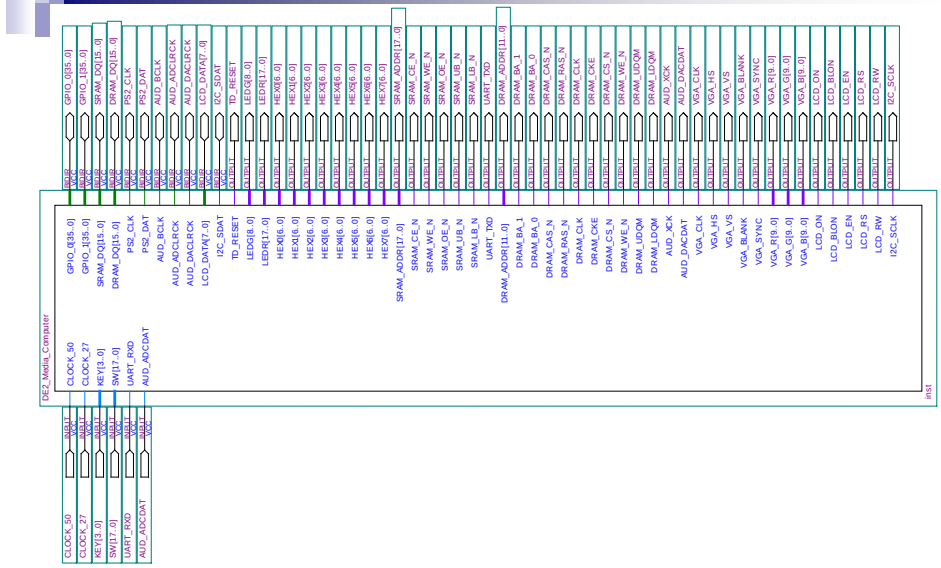
8

DE2 Media Computer is also logic circuits 2/2

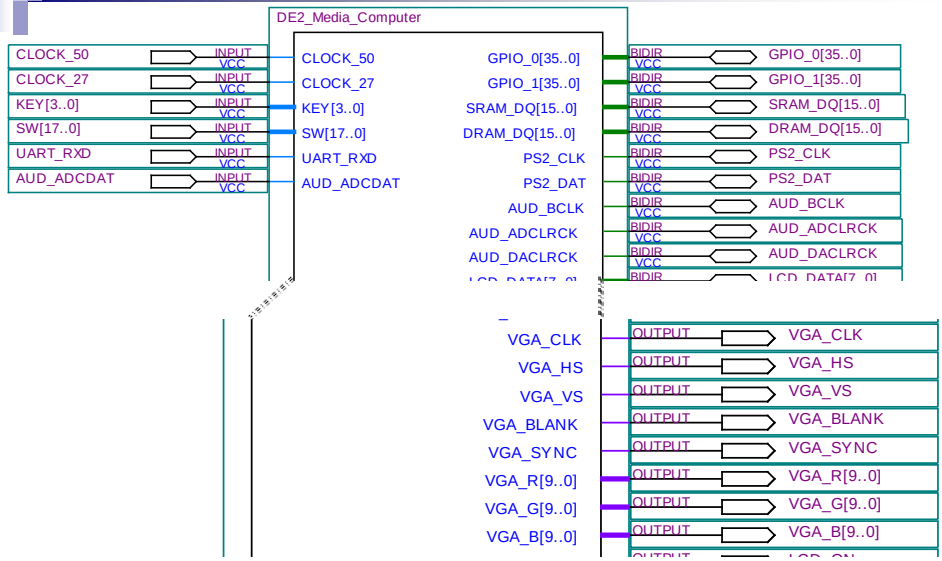
5. *Crete new Block Diagram/Schematic File*
   File->New->Block Diagram/Schematic File…
6. *Insert **DE2_Media_Computer** in schema*
7. *Open Symbol Tool and insert **DE2_Media_Computer** into schema*
8. *By right mouse click on **DE2_Media_Computer** symbol, open its context menu*
9. *Select "Generate pins for symbol ports"*
10. *Save your Block Diagram/Schematic File, e.g. as MyMedia.bdf*
11. *Select MyMedia.bdf as Top Level Entity*
12. *Compile project and take a rest - the compilation will last several minutes.*

9

DE2 Media Computer schema



DE2 Media Computer Schema Detail



Copy of Media Computer compiled in 6 minutes

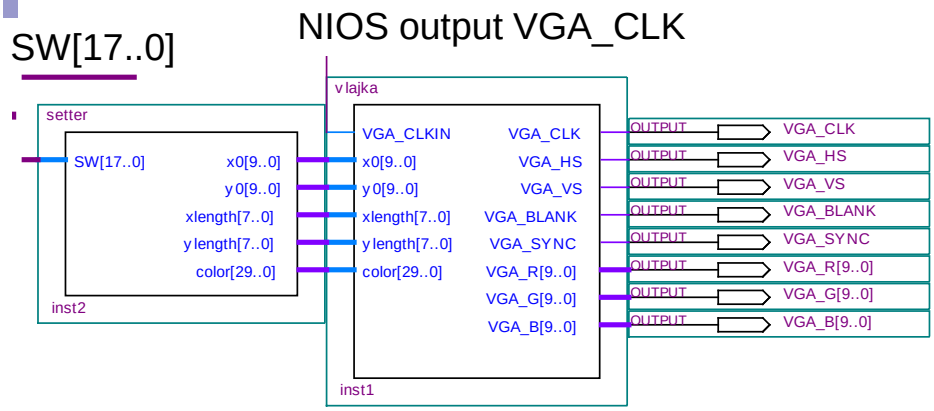
Flow Status	Successful - Thu Dec 08 08:23:37 2011
Quartus II Version	9.1 Build 350 03/24/2010 SP 2 SJ Full Version
Revision Name	DE2_Media_Computer
Top-level Entity Name	DE2_Media_Computer
Family	Cyclone II
Device	EP2C35F672C6
Timing Models	Final
Met timing requirements	Yes
Total logic elements	13,391 / 33,216 (40 %)
Total combinational functions	11,984 / 33,216 (36 %)
Dedicated logic registers	8,311 / 33,216 (25 %)
Total registers	8689
Total pins	317 / 475 (67 %)
Total virtual pins	0
Total memory bits	157,116 / 483,840 (32 %)
Embedded Multiplier 9-bit elements	11 / 70 (16 %)
Total PLLs	2 / 4 (50 %)

Adding Accelerator to DE2 Media Computer

13. *Open created Block Diagram/Schematic File of your Media Computer*
14. *Insert your perfect debugged accelerator – tested your accelerator carefully as a separate circuit because using in Media Computer with long compilation*
15. *Disconnect VGA outputs from Media Computer a connect them to your accelerator*
16. *Disconnect other outputs from Media Computer, e.g. unused GPIO_0[31..0] and GPIO_0[31..0] (page 5 and 6 of DE2_Media_Computer.pdf). and utilize them as inputs to your accelerator.*
17. *Compile the project and downloaded into DE2 board.*

13

Testing version

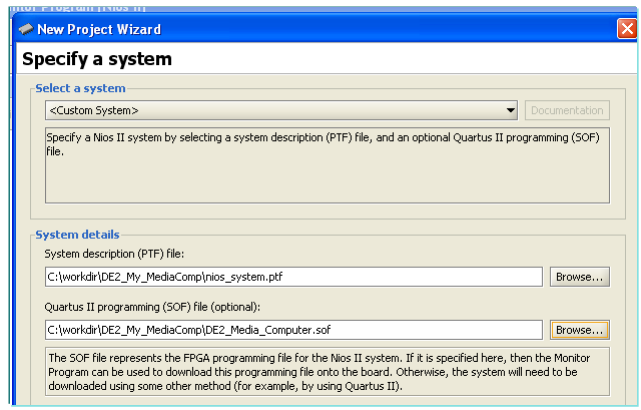


After compile time ~ 7 minutes

Flow Status	Successful - Thu Dec 08 09:31:32 2011
Quartus II Version	9.1 Build 350 03/24/2010 SP 2 SJ Full Version
Revision Name	DE2_Media_Computer
Top-level Entity Name	DE2_Media_Computer_Modul
Family	Cyclone II
Device	EP2C35F672C6
Timing Models	Final
Met timing requirements	No
Total logic elements	13,904 / 33,216 (42 %)
Total combinational functions	11,921 / 33,216 (36 %)
Dedicated logic registers	8,174 / 33,216 (25 %)
Total registers	8552
Total pins	317 / 475 (67 %)
Total virtual pins	0
Total memory bits	127,676 / 483,840 (26 %)
Embedded Multiplier 9-bit elements	11 / 70 (16 %)
Total PLLs	2 / 4 (50 %)

Creating NIOS project 1/2

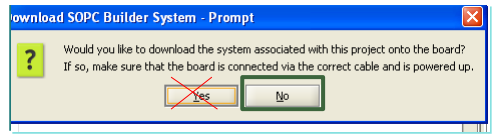
18. *In Altera monitor program create new NIOS project in Altera monitor*
19. *Select your custom system file*



16

Creating NIOS project 12/2

20. *Do no let download SOPC system – you must now use Quartus programmer*



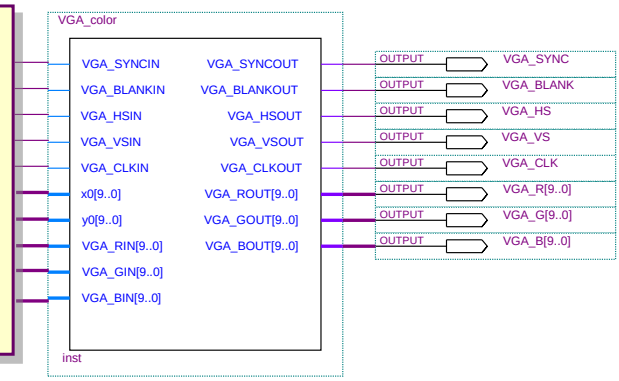
21. *Create your assembler program, now much simple, compile it and load into NIOS. If you use GPIO ports, do not forget set their direction bits as outputs from assembler before writing to them (see page 3, 5 and 6 of DE2_Media_Computer.pdf)*

22. *Run program.*

17

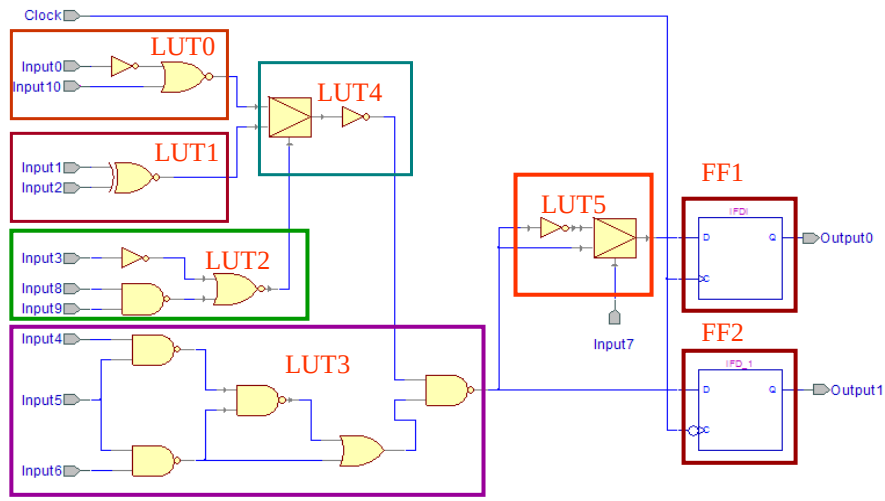
Example of Advanced DE2 Media Extended

DE2 Extended Media Computer

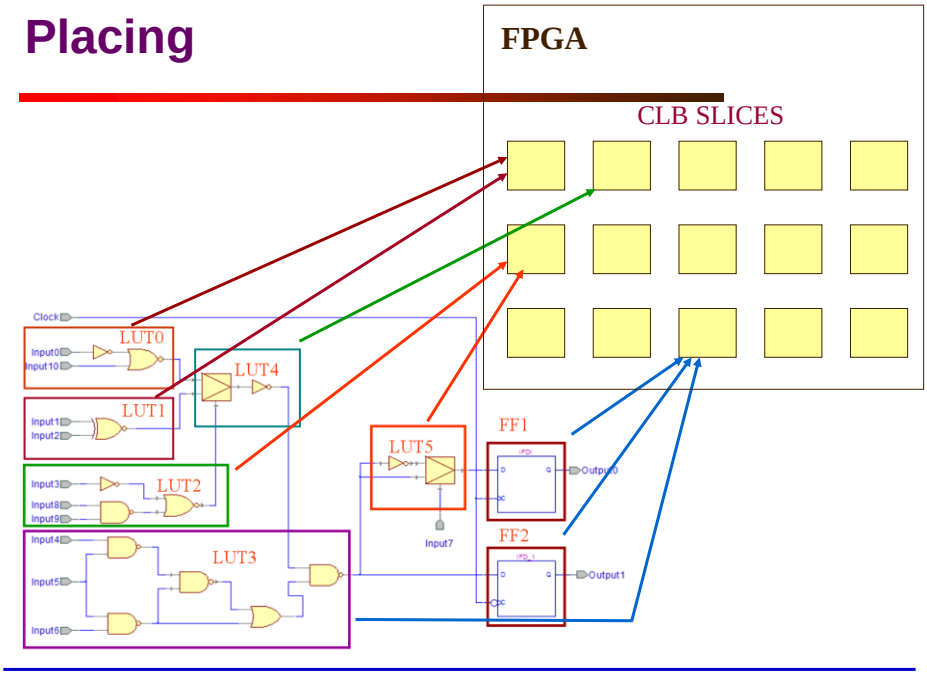


Routing in Systems on Chips

Mapping

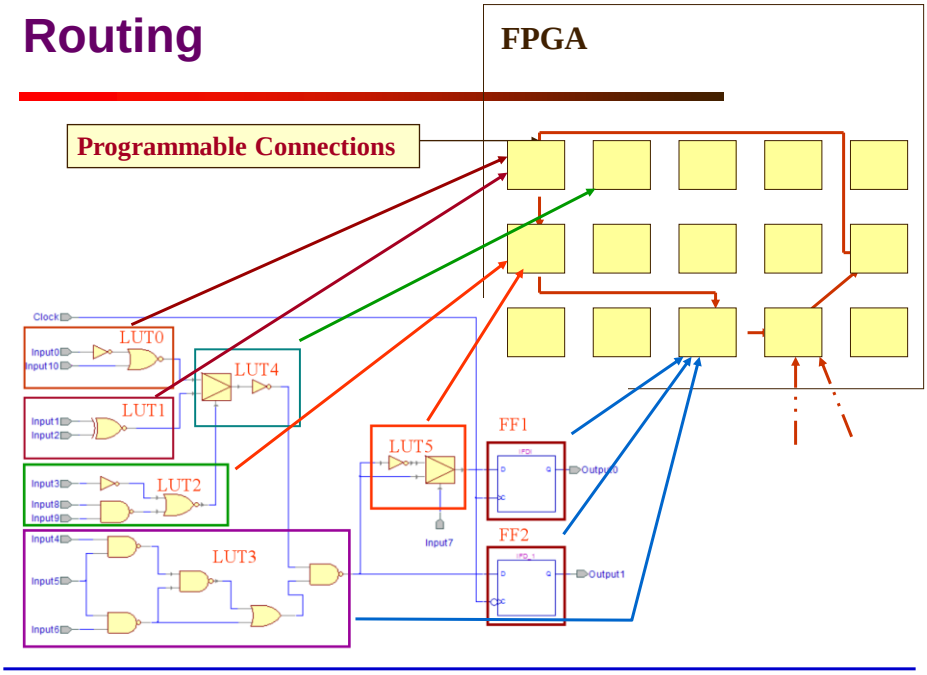


Placing



21

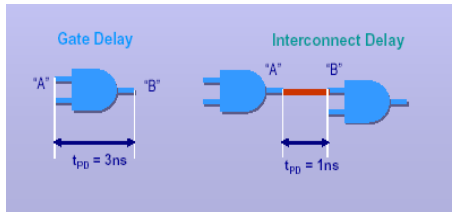
Routing

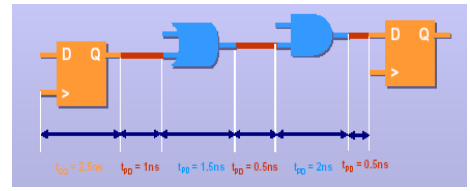


22

Interconnect Delays and Maximal Frequency

- Gate delay – Delay of logic element
- DFF delay tco (tsu - Very small)
- Interconnect delay



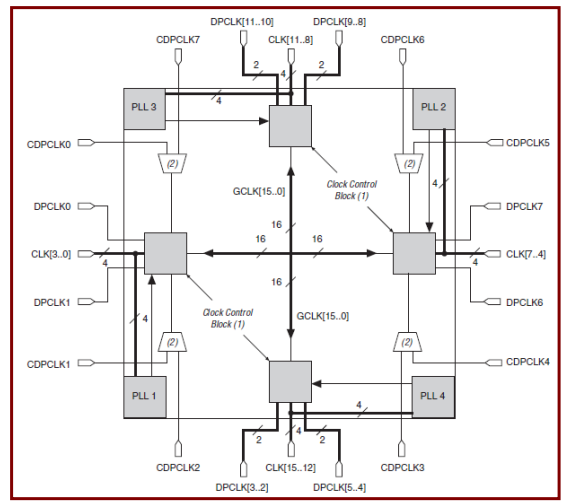


$$1/F_{max} = Tco + Tpd_{logic} + Tpd_{interconnect}$$

Maximum Frequency is the fastest speed a circuit containing flip-flops can operate.

Routing versus Clocking

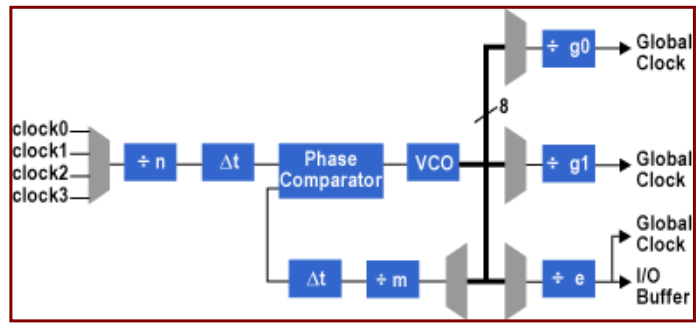
- 16 Global Clocks
- 4 PLLs



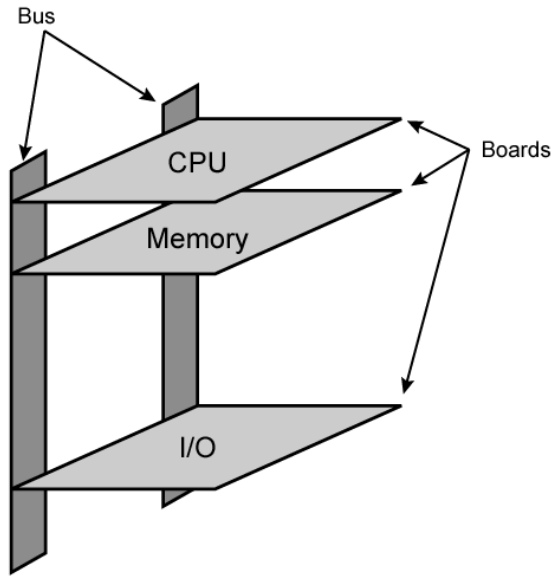
Cyclone II Clocking

Cyclone II PLL

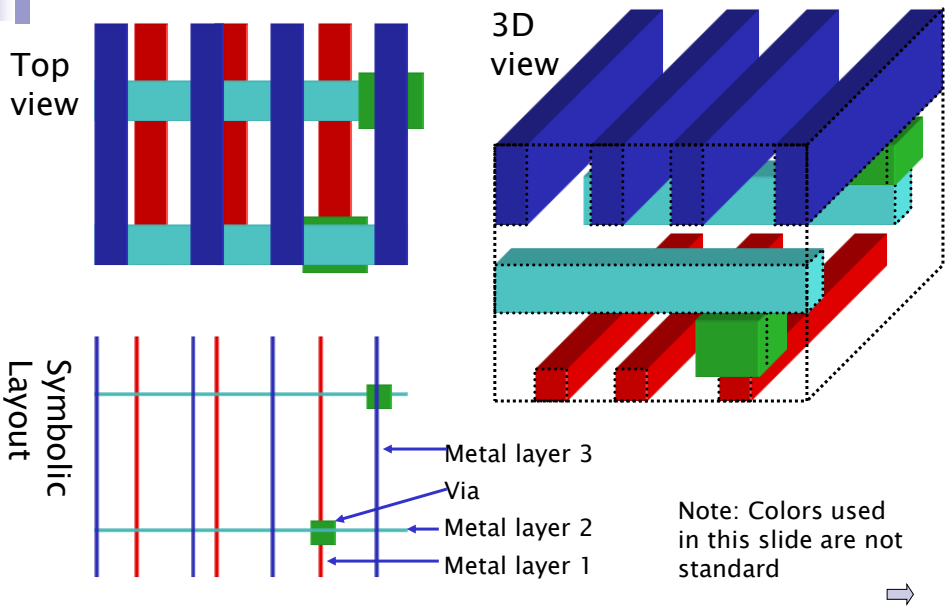
- 3 Outputs
- Clock Division
- Clock Multiplication
- Phase shift



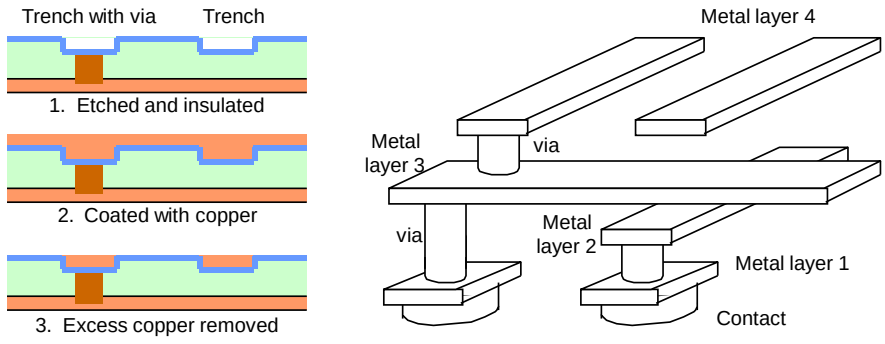
Physical Realization of Routing in Motherboards



Routing Anatomy in SoC



Intra- and Intersystem Links



(a) Cross section of layers

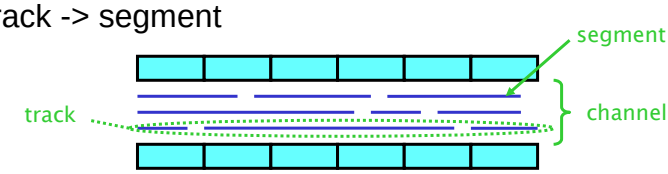
(b) 3D view of wires on multiple metal layers

Multiple metal layers provide intrasystem connectivity on microchips or printed-circuit boards.

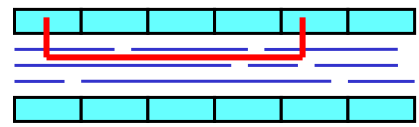
[Source Loomis J.: Buses]

FPGA Routing Channels Architecture

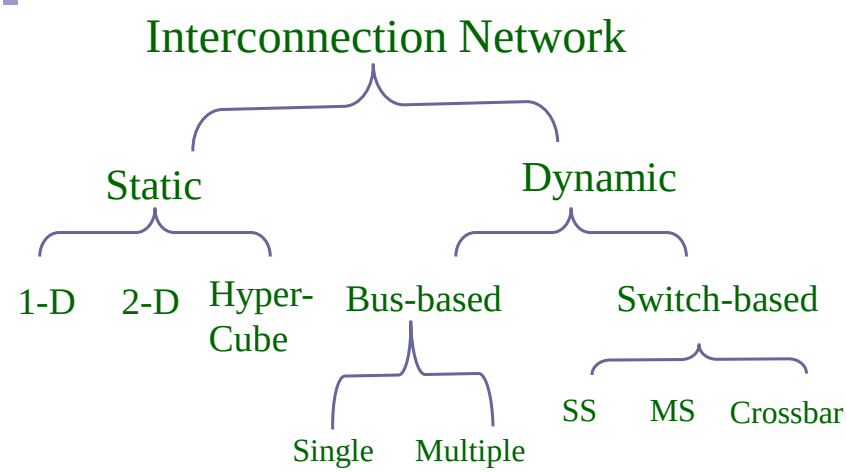
- Note: fixed channel widths (tracks)
- Should “predict” all possible connectivity requirements when designing the FPGA chip
- Channel -> track -> segment



- Segment length?
  - Long: carry the signal longer, less “concatenation” switches, but might waste track
  - Short: local connections, slow for longer connections



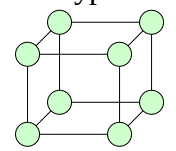
Interconnection Network Taxonomy



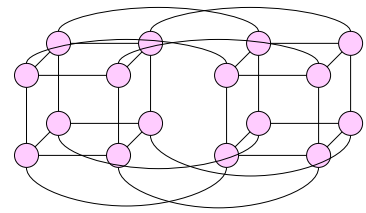
Interconnection Network - Hypercube

- Hypercube
  - Each element directly connected to `d` other elements
  - Shortest path between a pair of elements

3-D Hypercube

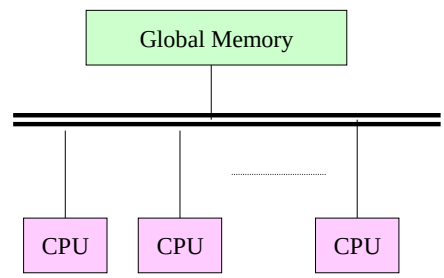


4-D Hypercube

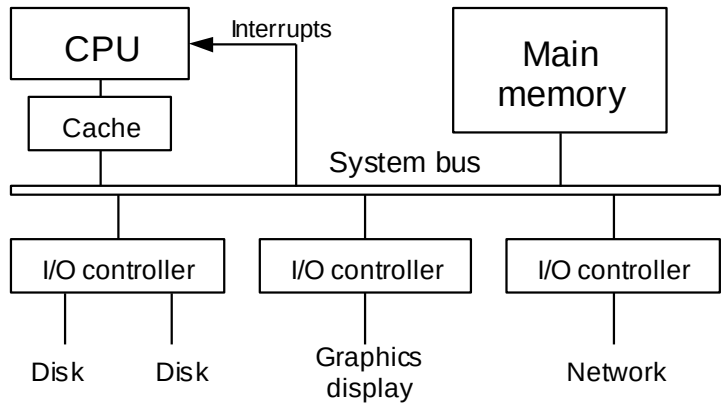


Interconnection Network - Bus

- Bus
  - A single shared data path
  - Pros
    - **Simplicity**
      - cache coherence
      - synchronization
  - Cons
    - fixed bandwidth
      - Does not scale well

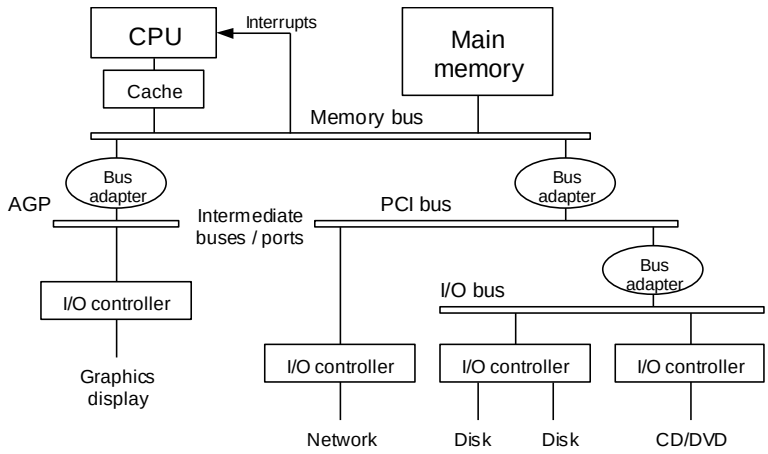


Simple Organization for Input/Output



Input/output via a single common bus.

I/O Organization for Greater Performance



Input/output via intermediate and dedicated I/O buses

Historical and Current Buses

- Unibus (PDP-11)
- Multibus (8086)
- VME bus (physical lab)
- ISA bus (PC/AT)
- EISA bus (80386)
- Nubus (macintosh)
- PCI bus (PCs)
- SCSI bus (workstations)
- ***SOC bus*** ***(system on-chip)*** : AMBA (AMD) and **Avalon (Altera Nios)**

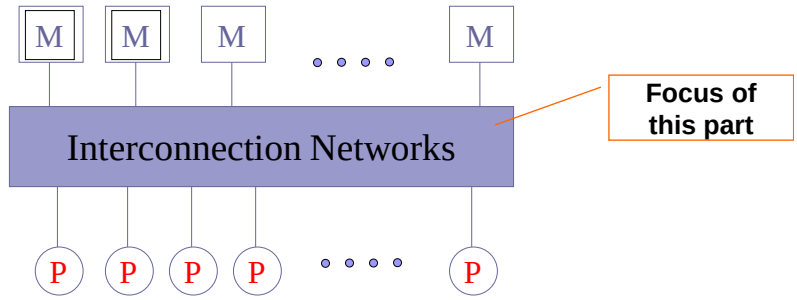
Serial Transmission

- USB bus,
- Fire wire
- PCIe
- Serial Attached SCSI (SAS)

Routing by Buses

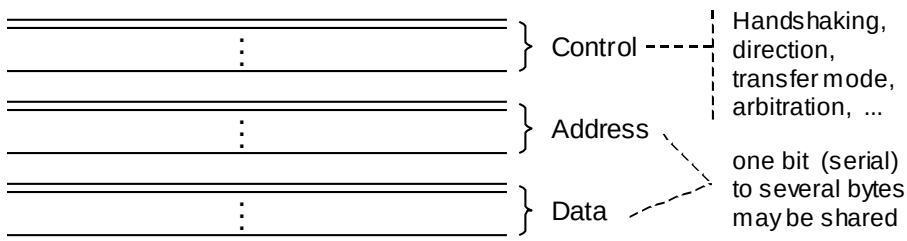
Communication

Big Picture



37

23.2 Buses and Their Appeal

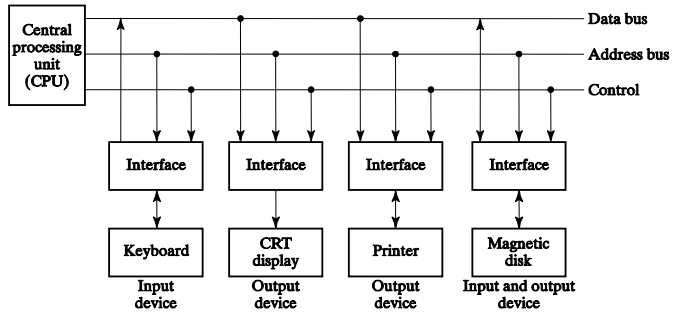


The three sets of lines found in a bus.

A typical computer may use a dozen or so different buses:

1. Legacy Buses: PC bus, ISA, RS-232, parallel port
2. Standard buses: PCI, SCSI, USB, Ethernet
3. Proprietary buses: for specific devices and max performance

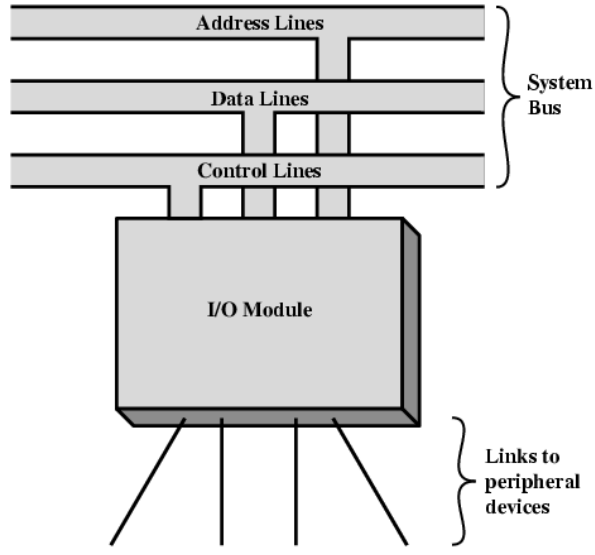
Connection of I/O Devices to CPU



Each peripheral has an I/O interface unit associated with it. A common bus from the CPU is attached to all peripheral interfaces.

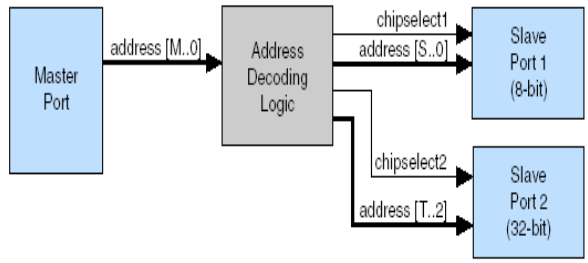
To communicate with a particular device, the CPU places a device address on the address bus. Each interface has a decoder that monitors the address lines. When an interface detects its address it assumes control of the bus and transfers data to and from the CPU on the data lines.

Generic Model of I/O Module

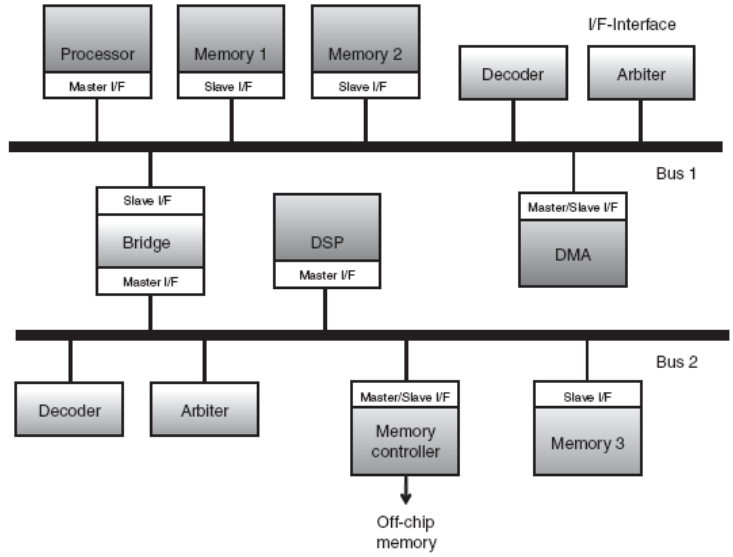


Address Decoding

Block Diagram of Address Decoding Logic



Bus Terminology



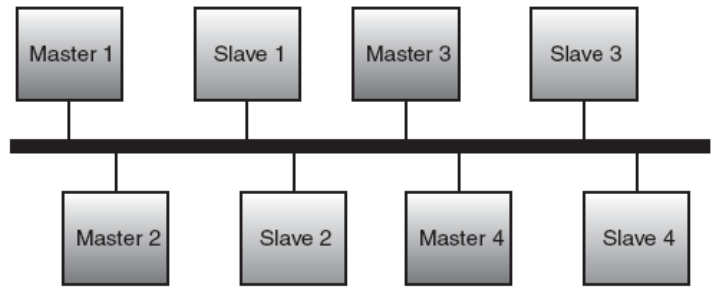
Bus Terminology

- Master (or Initiator)
  - IP component that initiates a read or write data transfer
- Slave (or Target)
  - IP component that does not initiate transfers and only responds to incoming transfer requests
- Arbiter
  - Controls access to the shared bus
  - Uses arbitration scheme to select master to grant access to bus
- Decoder
  - Determines which component a transfer is intended for
- Bridge
  - Connects two busses
  - Acts as *slave* on one side and *master* on the other

Source: Sudeep Pasricha, On Chip Communication, Colorado 2011 43

Types of Bus Topologies

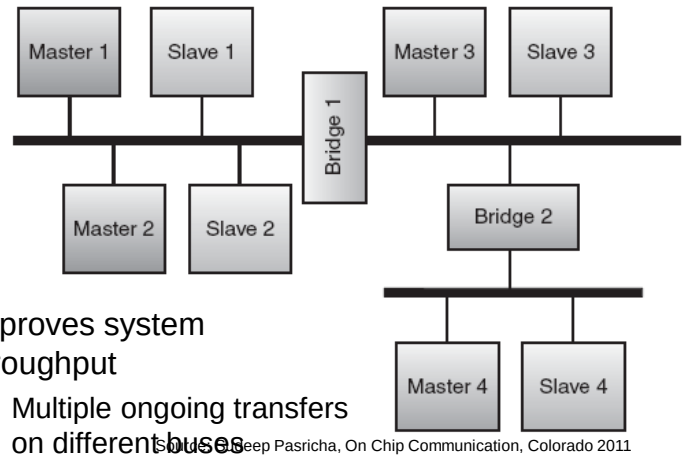
- Shared bus



Source: Sudeep Pasricha, On Chip Communication, Colorado 2011 44

Types of Bus Topologies

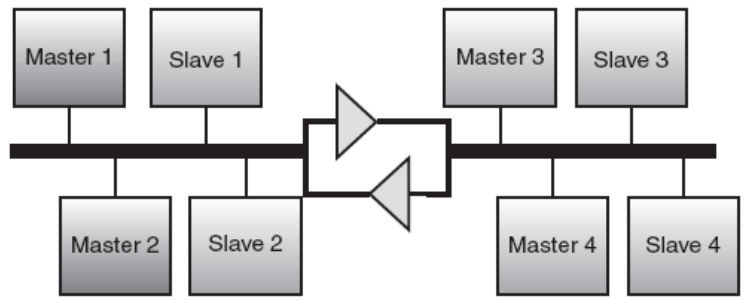
- Hierarchical shared bus



- Improves system throughput
  - Multiple ongoing transfers on different buses

Types of Bus Topologies

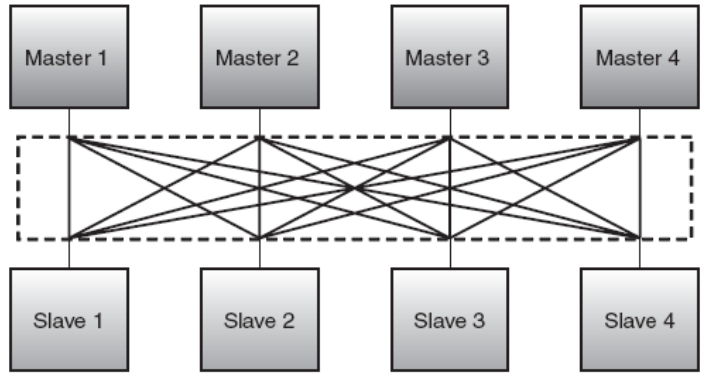
- Split bus



- Reduces impact of capacitance across two segments
- Reduces contention and energ

Types of Bus Topologies

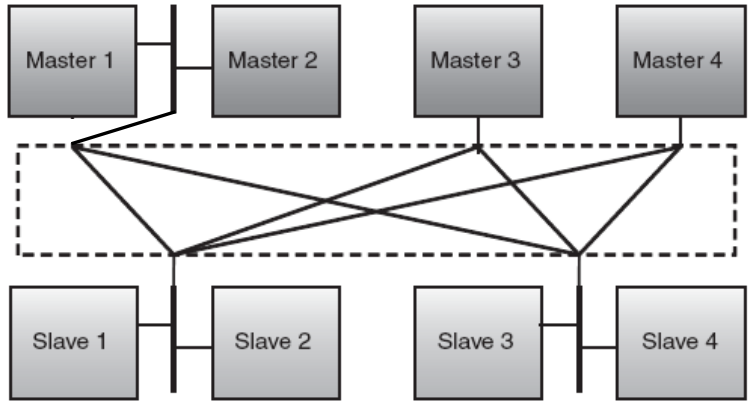
- Full crossbar/matrix bus (point to point)



Source: Sudeep Pasricha, On Chip Communication, Colorado 2011 47

Types of Bus Topologies

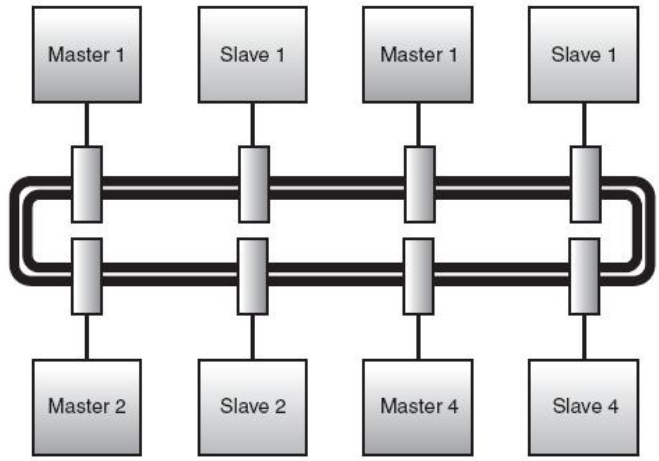
- Partial crossbar/matrix bus



Source: Sudeep Pasricha, On Chip Communication, Colorado 2011 48

Types of Bus Topologies

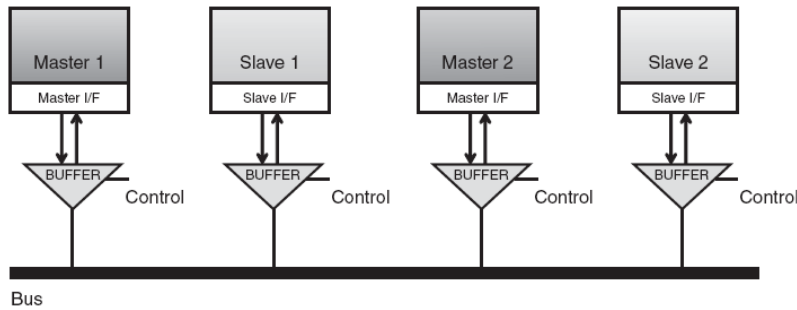
- Ring bus



Source: Sudeep Pasricha, On Chip Communication, Colorado 2011 49

Bus Physical Structure

- tri-state buffer based bidirectional signals

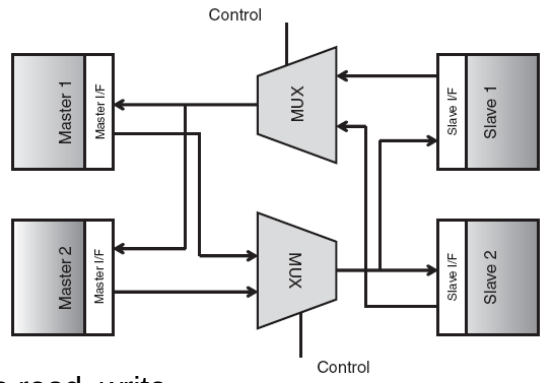


- Commonly used in off-chip/backplane buses
  - + take up fewer wires, smaller area footprint
  - - higher power consumption, higher delay, hard to debug

Source: Sudeep Pasricha, On Chip Communication, Colorado 2011 50

Bus Physical Structure

- MUX based signals

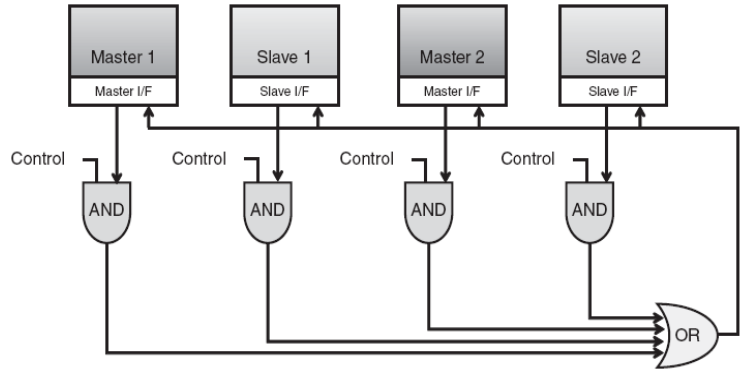


- Separate read, write channels

Source: Sudeep Pasricha, On Chip Communication, Colorado 2011 51

Bus Physical Structure

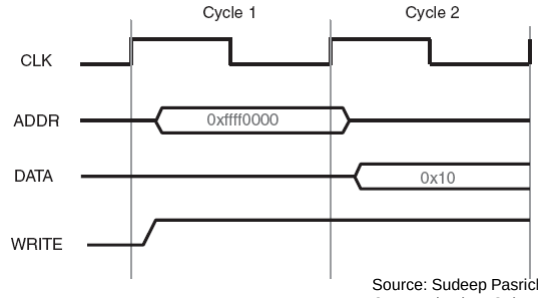
- AND-OR based signals



 52

Bus Clocking

- Synchronous Bus
  - Includes a clock in control lines
  - Fixed protocol for communication that is relative to clock
  - Involves very little logic and can run very fast
  - Require frequency converters across frequency domains

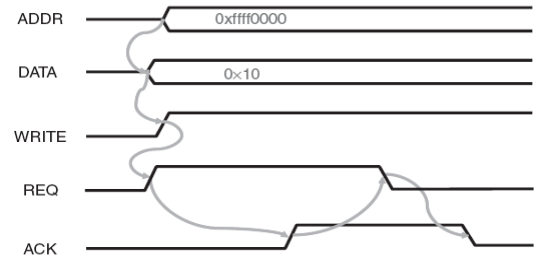


Source: Sudeep Pasricha, On Chip Communication, Colorado 2011

53

Bus Clocking

- Asynchronous Bus
  - Not clocked
  - Requires a handshaking protocol
    - performance not as good as that of synchronous bus
    - No need for frequency converters, but does need extra lines
  - Does not suffer from clock skew like the synchronous bus

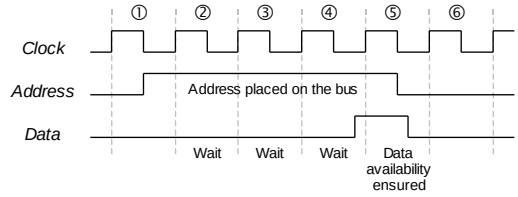


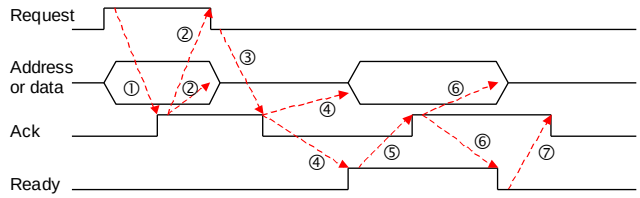
Source: Sudeep Pasricha, On Chip Communication, Colorado 2011

54

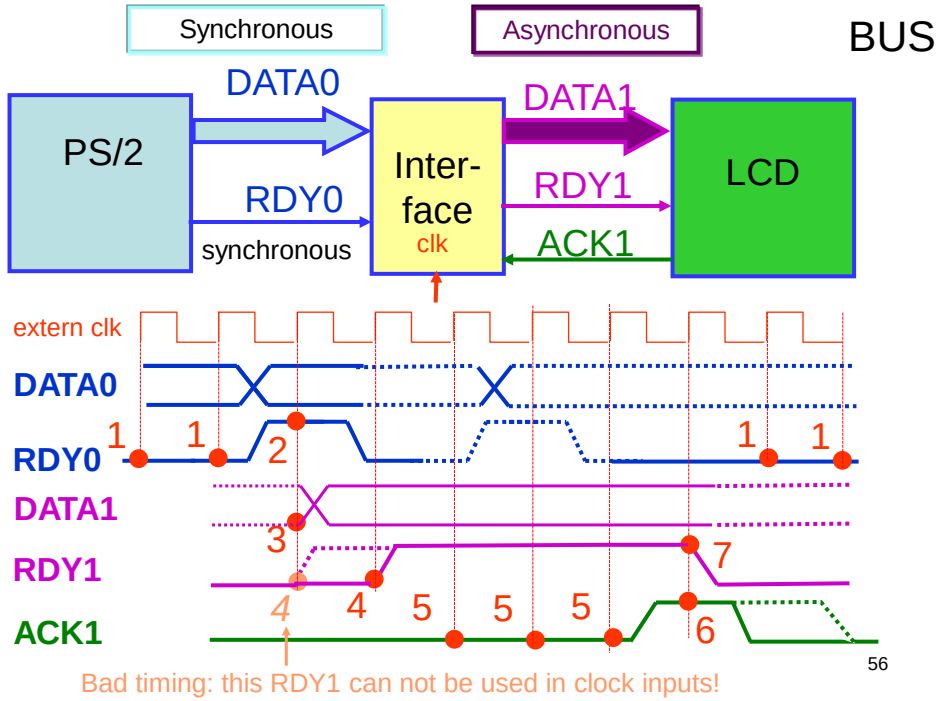
Bus Communication Protocols

Synchronous bus with fixed-latency devices.



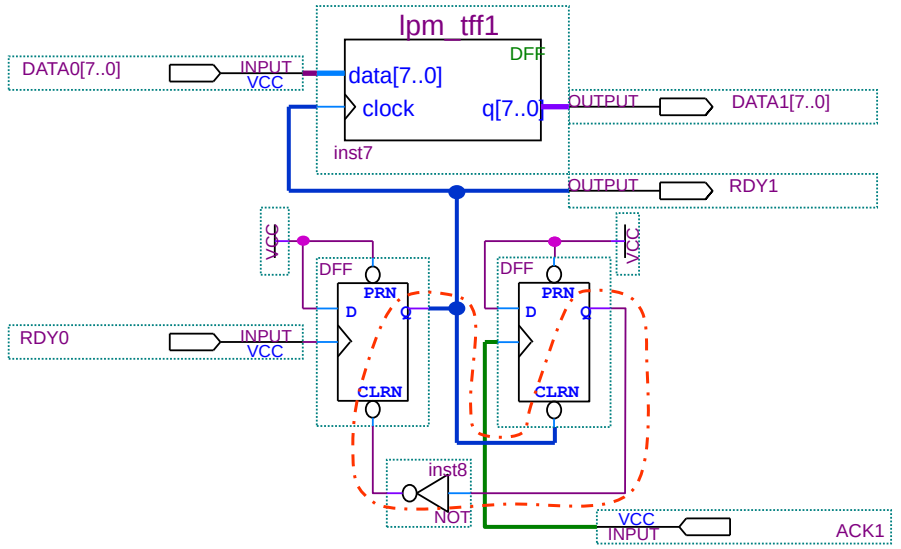


Handshaking on an asynchronous bus for an input operation (e.g., reading from memory).



56

2 DFF Handshake



RDY0 and ACK1 must be designed for driving clock inputs.

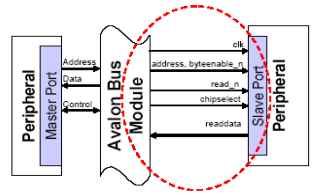
VHDL handshaking

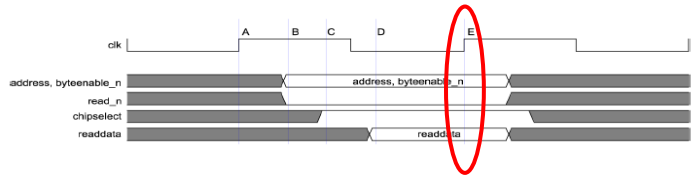
```
LIBRARY ieee; USE ieee.std_logic_1164.all;
ENTITY handshake is PORT(RDY0,  ACK1 :  IN  STD_LOGIC;
                         DATA0 :  IN  STD_LOGIC_VECTOR(7 downto 0);
                         RDY1 :   OUT  STD_LOGIC;
                         DATA1 :  OUT  STD_LOGIC_VECTOR(7 downto 0));
end handshake;

ARCHITECTURE bdf_type of handshake is
signal qRDY1, qA  :  STD_LOGIC:='0';
begin
        PROCESS(RDY0, qA)                          -- set qRDY1 - reset by qA
         begin  if (qA='1') then  qRDY1<='0';
                elsif (RISING_EDGE(RDY0))
                then  qRDY1 <= '1'; DATA1<=DATA0;
                end if;
        end PROCESS;

        PROCESS(ACK1, qRDY1)                       -- set qA  - reset by qRDY1
        begin   if (qRDY1 = '0') then qA<='0';
                elsif (RISING_EDGE(ACK1)) then qA<='1';
                end if;
        end PROCESS;

        RDY1<=qRDY1;
end bdf_type;
```

58

Avalon slave
read, 0 wait, asynchronous peripheral

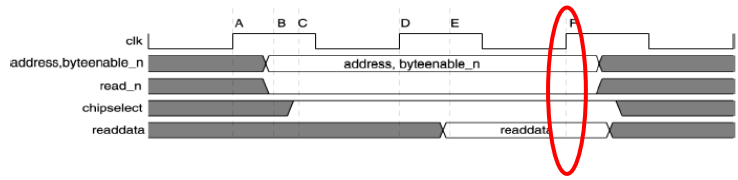
This Example Demonstrates	*Relevant PTF Parameters*
Read transfer from an asynchronous peripheral	
Zero wait states	Read_Wait_States = "0"
Zero setup	Setup_Time = "0"



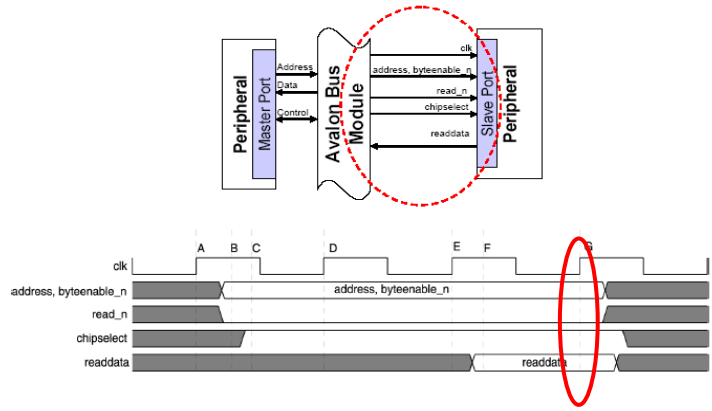


Avalon slave
read, 1 wait

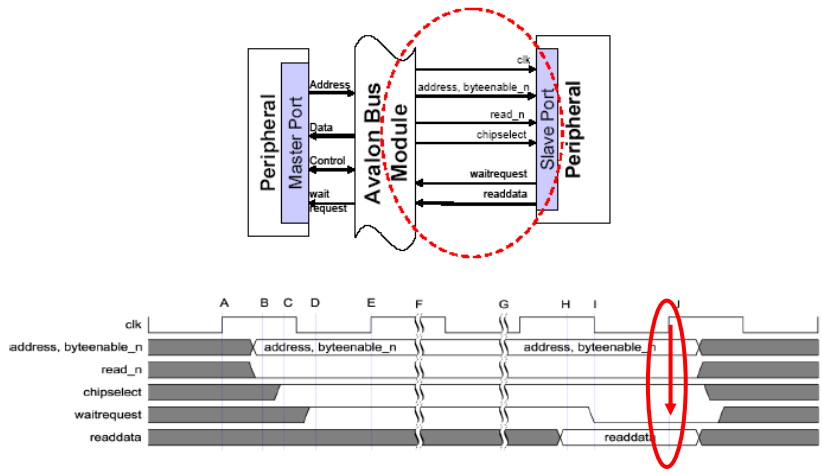
Wait cycle specified by design



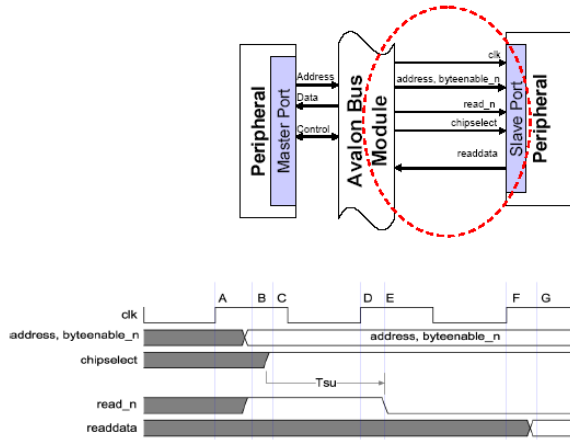
Avalon read slave, 2 wait



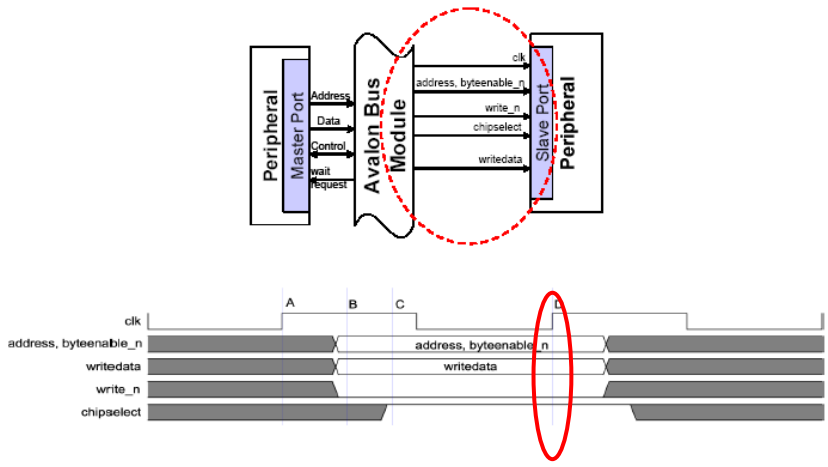
Avalon read
slave, wait request generated by slave device



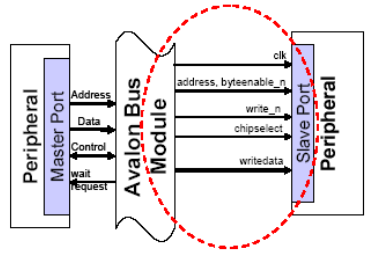
Avalon read
slave, 1 set up and 1 wait

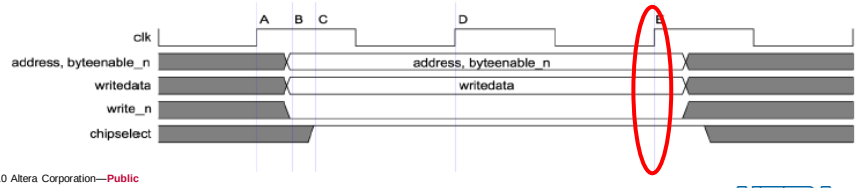


Avalon write
slave, 0 wait

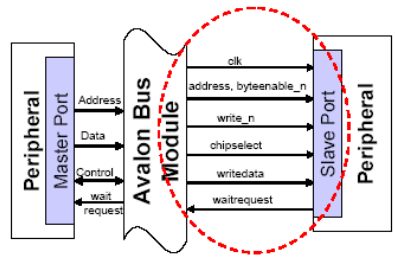


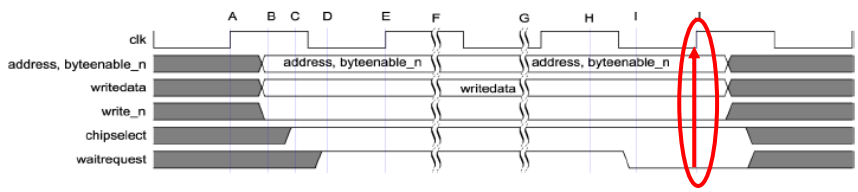
Avalon write slave, 1 wait



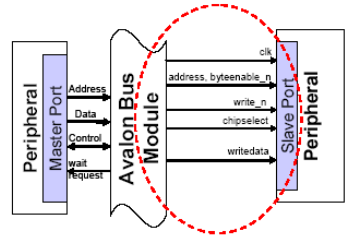


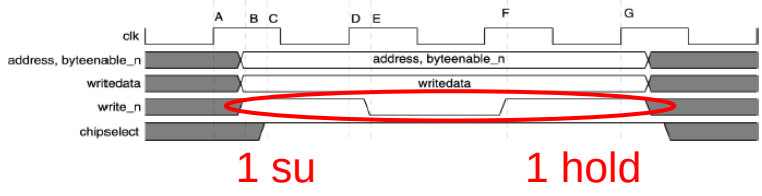
Avalon write slave, wait request generated by slave



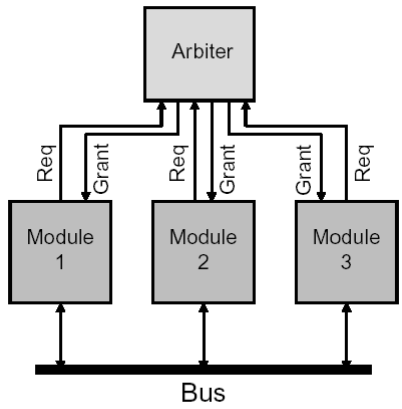


Avalon write
slave, 1 set up, 1 hold, 0 wait



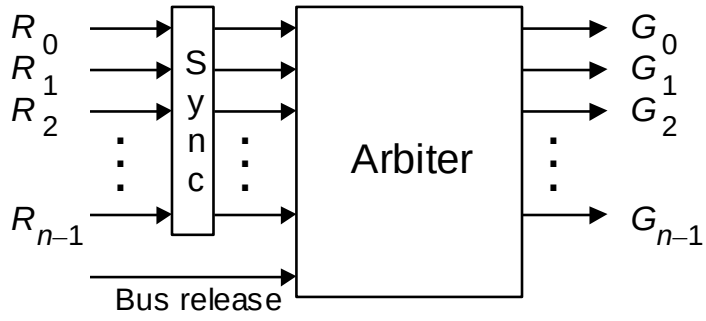


Independent Request and Grant



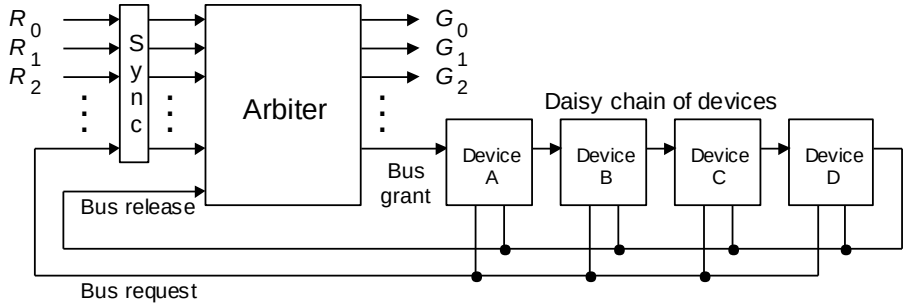
- Multiple *bus-request* and *bus-grant* signal lines are provided for each master
- Any priority-based or fairness based bus allocation can be used.
- Advantages
  - flexibility
  - faster arbitration time
- Disadvantages:
  - large number of arbitration lines

Bus Arbitration and Performance



General structure of a centralized bus arbiter.

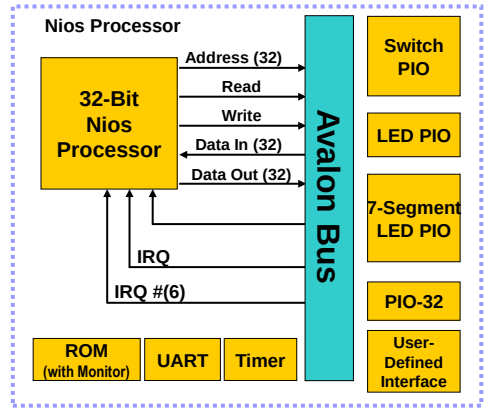
Daisy Chaining



Daisy chaining allows a small centralized arbiter to service a large number of devices that use a shared resource.

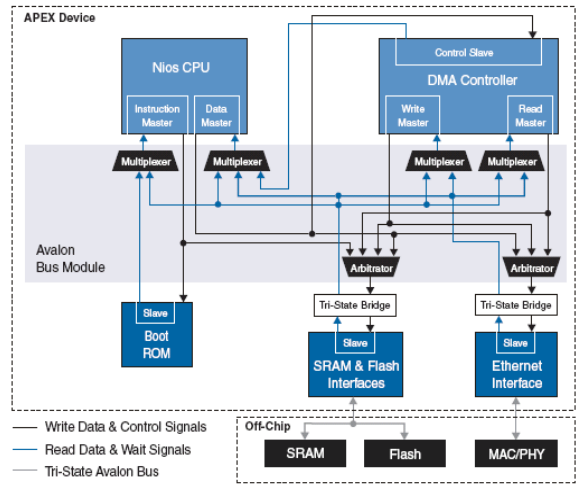
Avalon Bus

- Proprietary bus specification used with Nios II
- Principal design goals of the Avalon Bus
  - Address Decoding
  - Data-Path Multiplexing
  - Wait-State Insertion
  - Arbitration for Multi-Master Systems
- Transfer Types
  - Slave Transfers
  - Master Transfers
  - Pipelined Transfers
  - Burst transfers



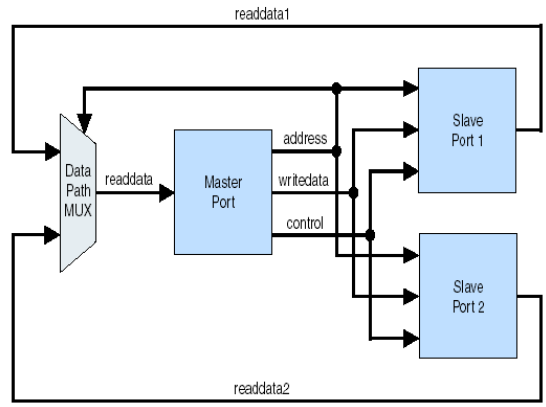
AVALON

- Designed for connecting on-chip processors and peripherals together into a System–On–a–Programmable Chip (SOPC)



Data-Path Multiplexing

Block Diagram of Data-Path Multiplexing Logic



AVALON versus Traditional Bus

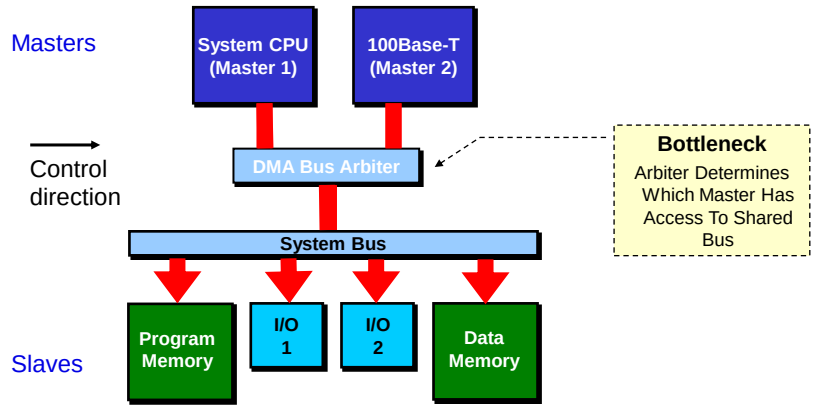
- Traditional
  - a single arbitrator controls communication between multiple bus masters and bus slaves. The arbitrator will grant a single master access to the bus after each potential master giving the control request. If more than one masters attempt to access the bus, the arbitrator allocates bus resources to a single master based on a fixed set of arbitration rules
  - Traditional systems have a bandwidth bottleneck because only one master can access the system bus at a time.

AVALON versus Traditional Bus

- Avalon
  - The Avalon is simultaneous multi-master bus architecture which increases system bandwidth by eliminating this bottleneck because bus masters contend for individual slaves, not for the bus itself.
  - In Avalon, multiple masters can be active at the same time and can simultaneously transfer data to their slaves. Masters do not have to wait to access a target slave, as long as another master does not access the same slave at the same time.

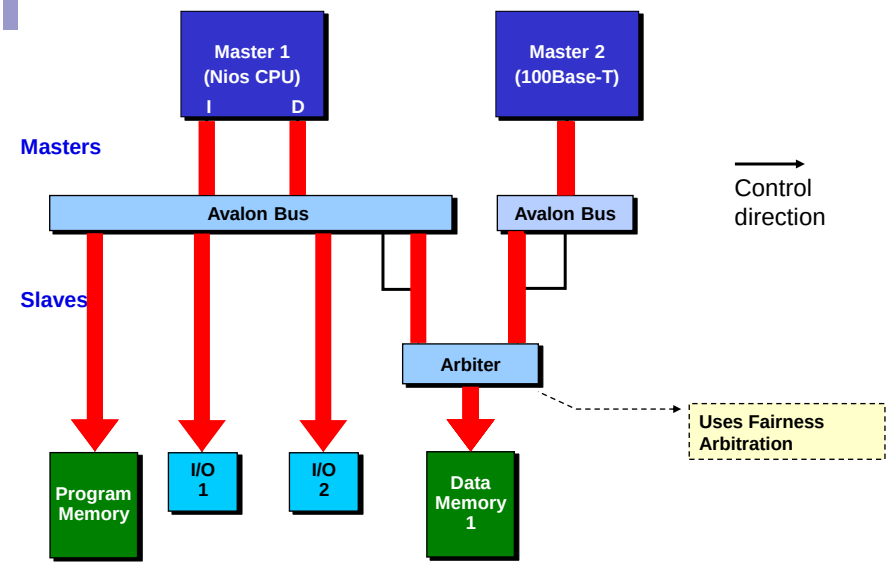
Traditional Multi-Masters

- Direct Memory Access (DMA)
  - Processor Waits For Bus During DMA



77

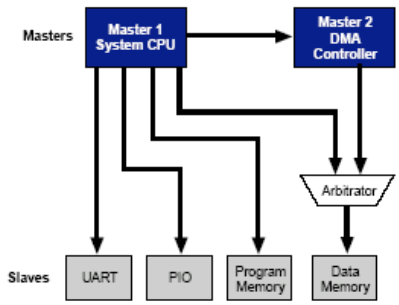
Simultaneous Multi-Master Bus



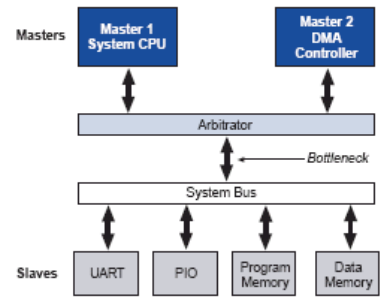
78

Example Buses (cont')

Avalon



Traditional



Avalon master signals (1)

Signal Type	Width	Direction	Required	Description
`clk`	1	in	yes	Global clock signal for the system module and Avalon bus module. All bus transactions are synchronous to `clk`.
`reset`	1	in	no	Global reset signal. Implementation is peripheral-specific.
`address`	1 - 32	out	yes	`Address` lines from the Avalon bus module. All Avalon masters are required to drive a byte address on their `address` output port.
`byteenable`	0, 2, 4	out	no	Byte-enable signals to enable specific byte lane(s) during transfers to memories of width greater than 8 bits. Implementation is peripheral-specific.
`read`	1	out	no	Read request signal from master port. Not required if master never performs read transfers. If used, `readdata` must also be used.
`readdata`	8, 16, 32	in	no	Data lines from the Avalon bus module for read transfers. Not required if the master never performs read transfers. If used, `read` must also be used.
`write`	1	out	no	Write request signal from master port. Not required if the master never performs write transfers. If used, `writedata` must also be used.

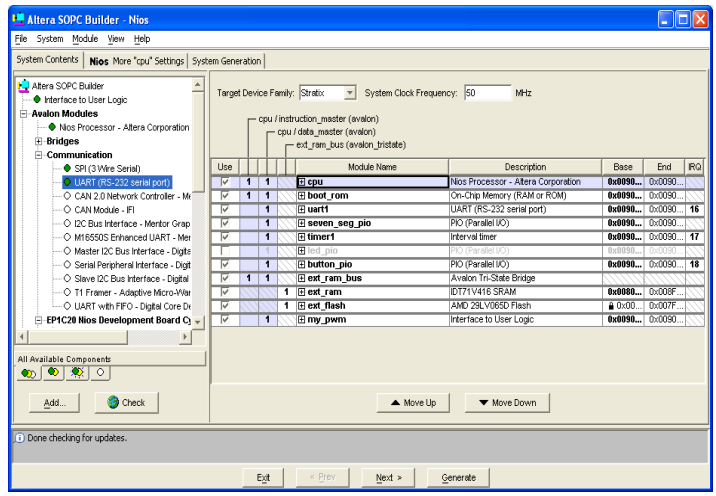
Master Arbitration Scheme

- Nios Multi-Master Avalon Bus utilizes Fairness arbitration scheme
  - Each Master/Slave pair is assign an integer “shares”
  - Upon conflict Master with most shares takes bus until all shares are used
  - Master with least shares then takes bus until all shares are used
  - Assuming all Masters continuously request the bus, they will each be granted the bus for a percentage of time equal to the percentage of total master shares that they own

81

Set Arbitration Priority

- View => Show Arbitration Priorities



82