Real Time Embedded Systems

"System On Programmable Chip"

NIOS II – Avalon Bus

René Beuchat

Laboratoire d'Architecture des Processeurs

rene.beuchat@epfl.ch

Embedded system on intelFPGA(Altera) FPGA

Goal :

- To understand the architecture of an embedded system on FPGA
- To be able to design a specific interface
- To be able to construct a full system based on a standard softcore bus in a FPGA and using blocs modules
- To understand, use and program a softcore processor

Contents

- NIOS II a softcore processor
- System On FPGA
- Avalon Bus
- Design of a specific slave programmable interface on Avalon

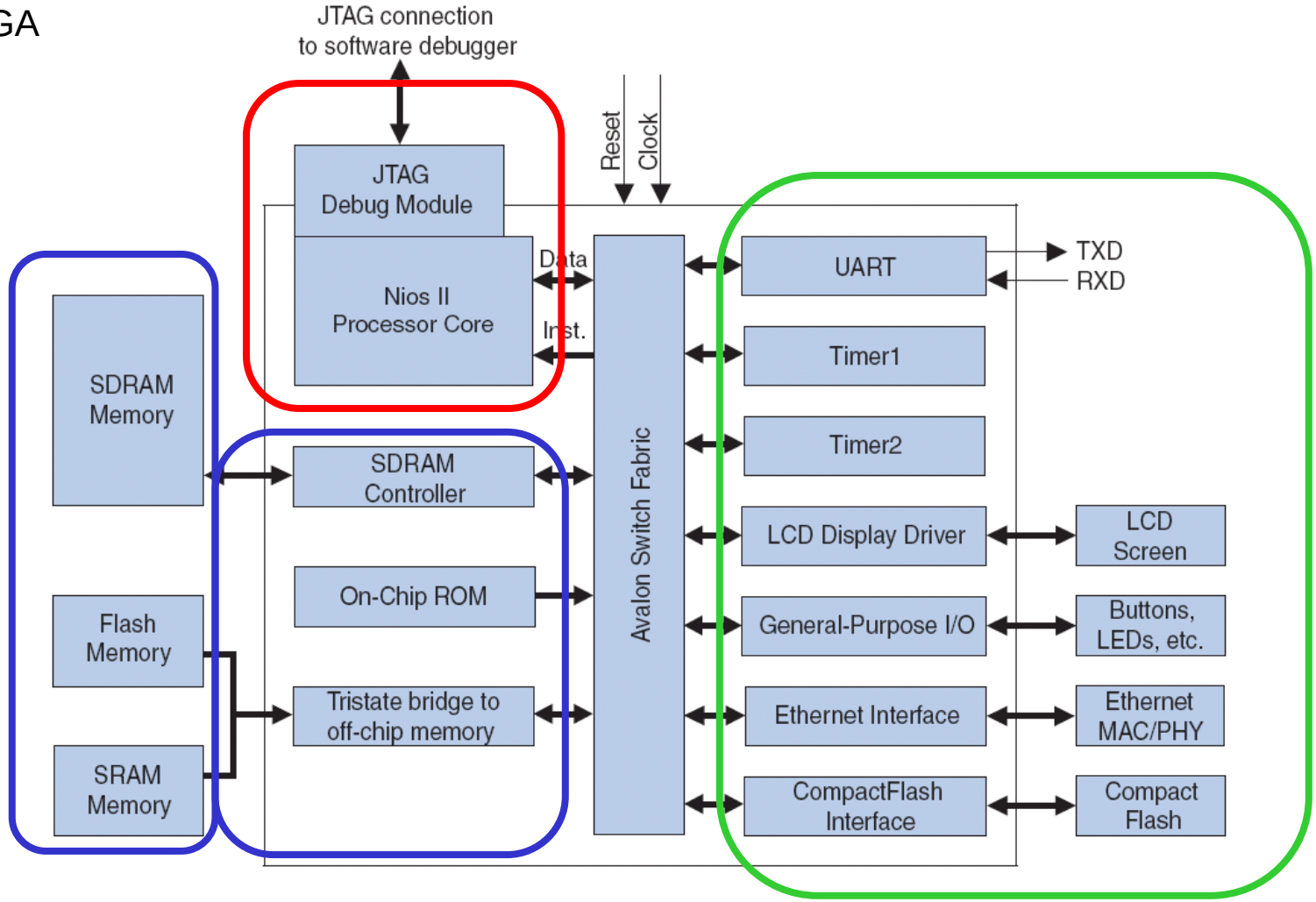
- Reference: https://www.intel.com/content/www/us/en/products/programmable.html

NIOS II

- Softcore Processor from intelFPGA
  - A processor implemented with Logic Elements (LUT+DFF) in a FPGA
  - A processor synthesized by a compiler and placed & routed on the FPGA
  - A processor described by a HDL langage(VHDL/Verilog/…)
- 32 bits Architecture
- 3 versions
- 256 instructions available for user implementation

NIOS II –
Embedded system NIOSII/Avalon Architecture

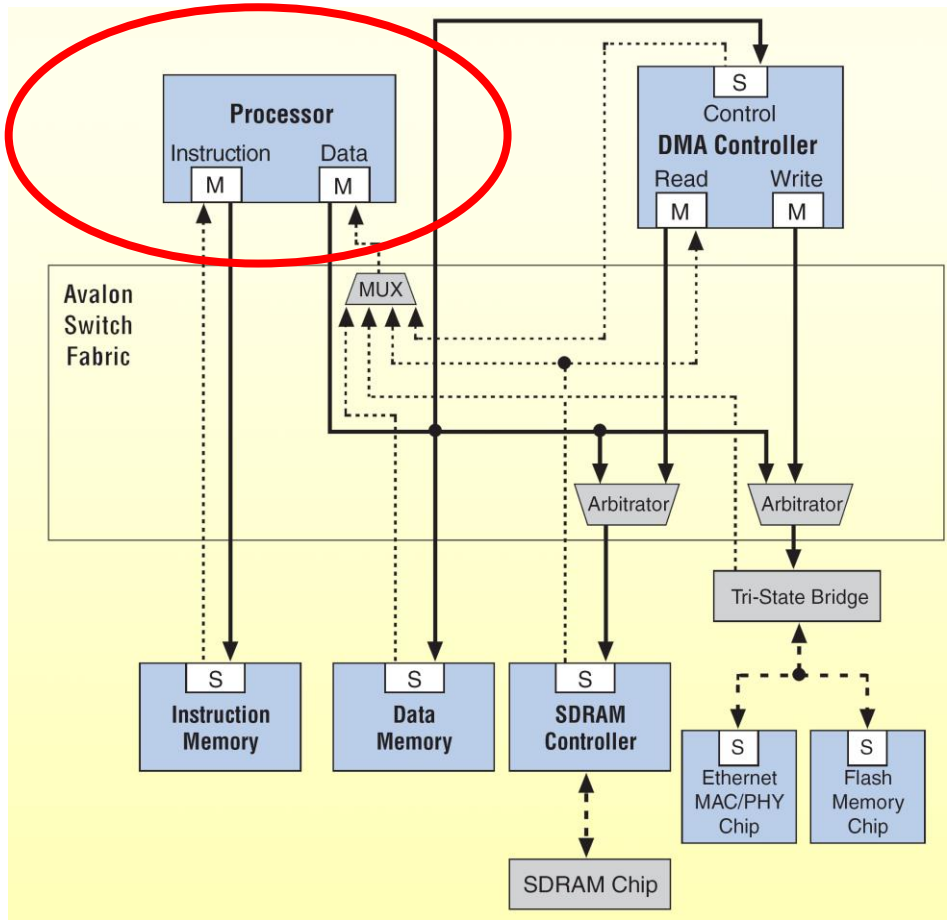
Note: The same principles are available for intelFPGA (Altera), Xilinx, Actel or others FPGA



AVALON Switch Fabric

Some Avalon specifications :

- ❑ Multi-Master
- ❑ Arbitrage « slave-side »
- ❑ **Concurrent Master-Slave Access**
- ❑ Synchronous transfers



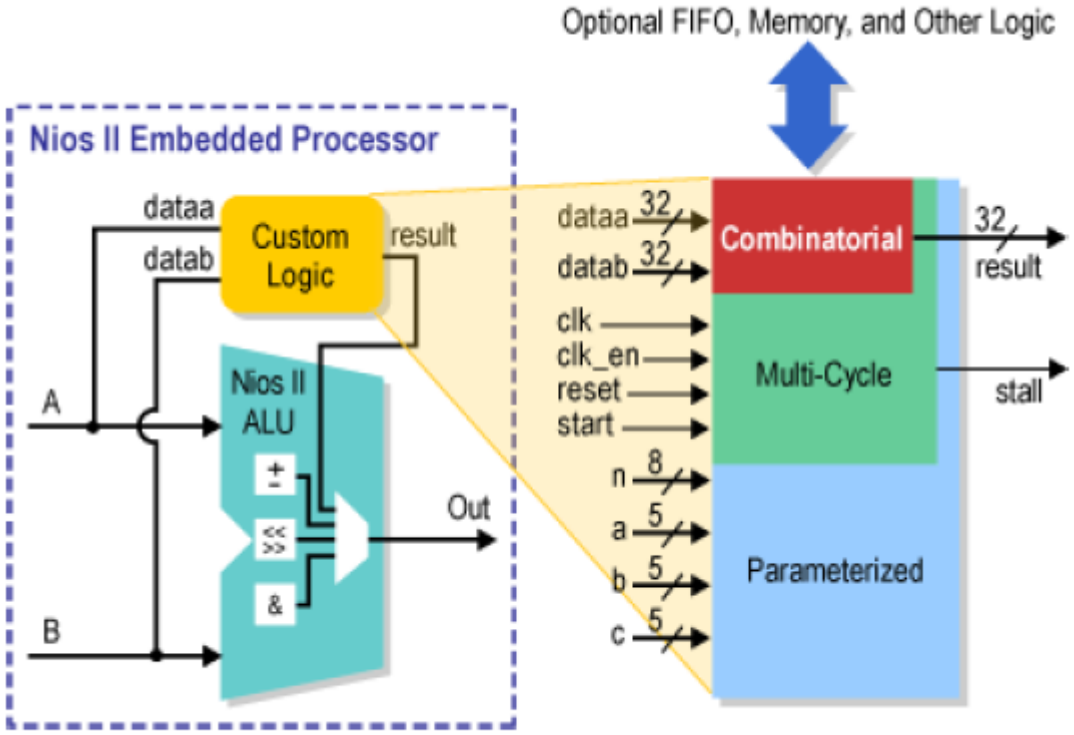
NIOS II Processor

3 processor architectures

	Nios II /f Fast	Nios II /s Standard	Nios II /e Economy
Pipeline	6 Stage	5 Stage	None
Multiplier *	1 Cycle	3 Cycle	None
Branch Prediction	Dynamic	Static	None
Instruction Cache	Configurable	Configurable	None
Data Cache	Configurable	None	None

NIOS II Processor, user instructions

- The ALU can be extended by user own instructions, until 256.

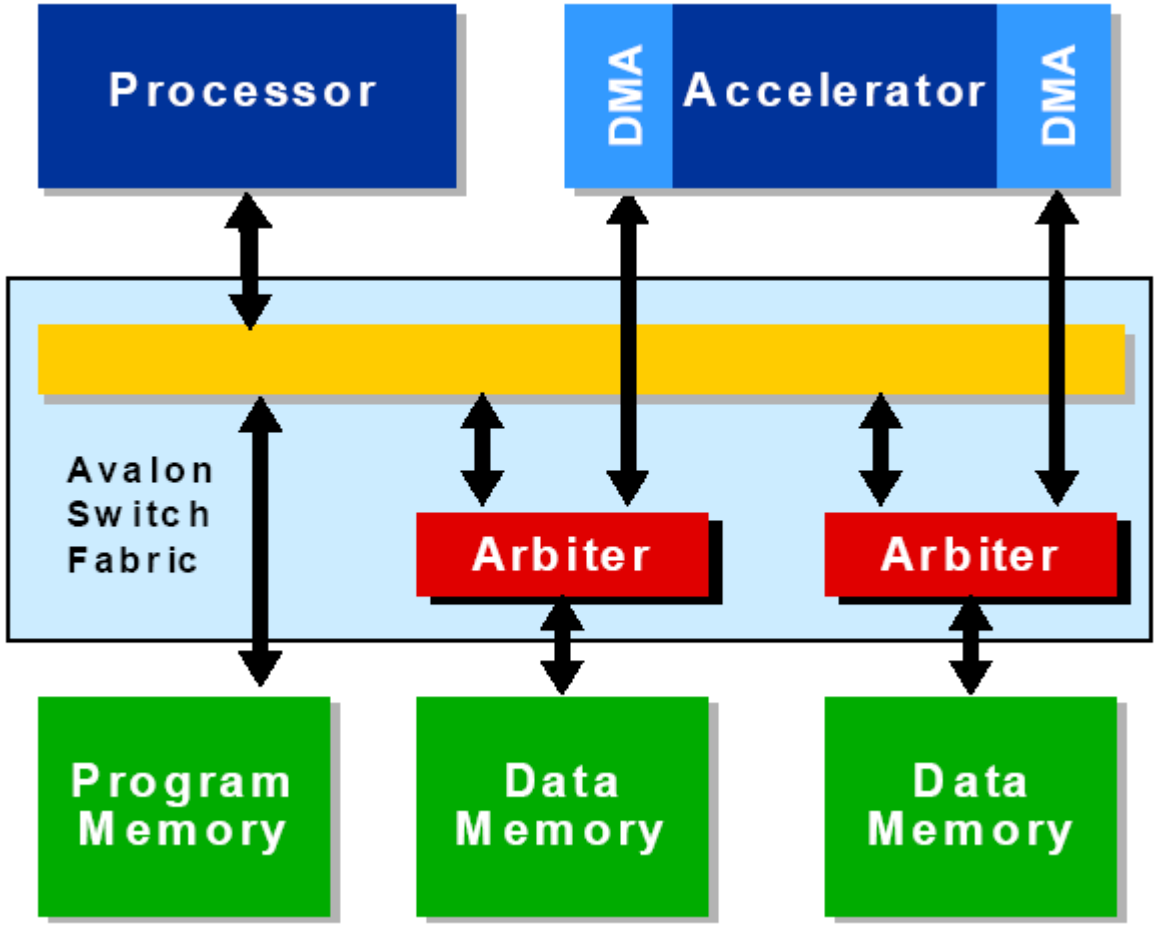


NIOS II Processor, user instructions

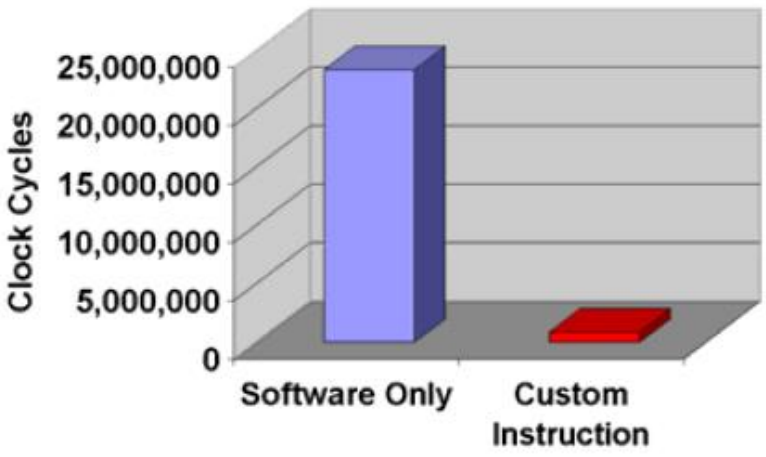
- The instructions can be:
  - Combinatorial, single clock cycle
  - Multi-cycles, synchronized by clk and stall
  - Parameterized
- They can have access to all the FPGA resources
- They can use their own internal registers

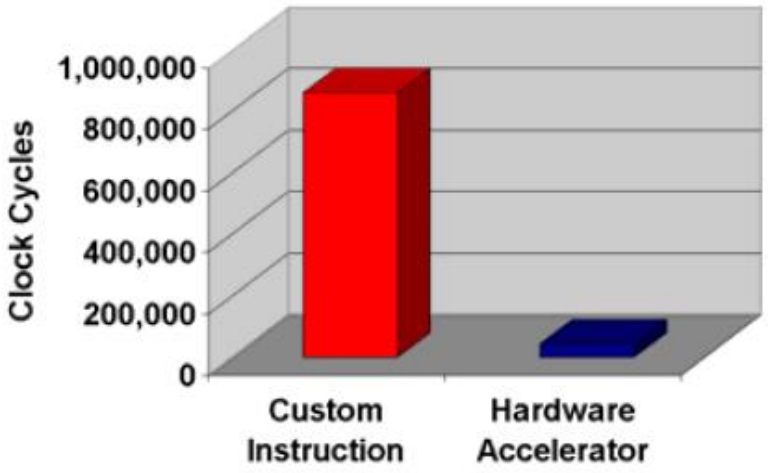
- For cycles consuming operations, a **hardware accelerator** can be included/developed
- A **Master unit** which has access to Memory and Programmable Interfaces for accelerated operations or with hard real time constrains

NIOS II Processor, hardware accelerator



NIOS II Processor, performances gain (commercial view)





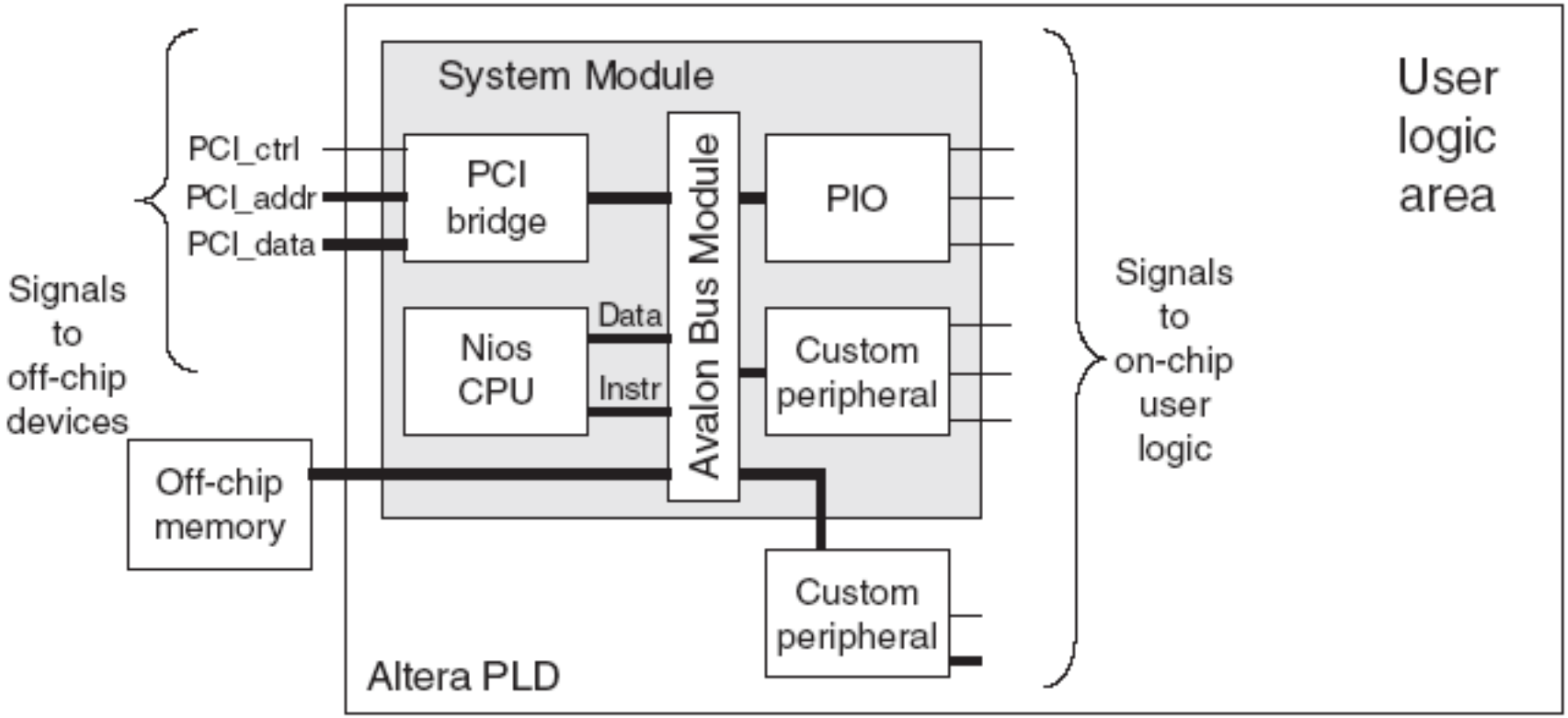
Computer architecture

- Classical architecture,
  with common tri-state bus
  - Processor
  - Memories
  - Input/Output (programmable) interface
  - Address bus
  - Data Bus (tri-state)
  - General decoder

Computer architecture on FPGA (intelFPGA)

- SOPC architecture (intelFPGA)
  - Processor
  - Memories
  - Input/Output (programmable) interface
  - Address bus
  - **Separated Data Bus Read/Write → multiplexers**
  - Local decoder on the Avalon bus
  - Bus transfers size adaptation is done at Avalon bus level

System on FPGA
example



Avalon Bus

To interconnect all the masters and slaves inside the FPGA, an generated internal bus :

- Master/Slave modules
- Synchronous bus on clock rising edge
- Separate data Read and data Write
- Wait state by configuration or dynamic
- Hold / Set up available
- Actual version (>1.0) allows data path until **1024 bits** (8, 16, 32, 64, 128, 256, 512, 1024)

Avalon
« slave » main signals

Signal Type	Width	Direction	Required	Description
clk	1	In	(No)	Global clk for system module and Avalon bus modules. **All transactions synchronous to clk rising edge**
nReset	1	In	No	Global Reset of the system
address	1..32	In	No	Address for Avalon bus modules
ChipSelect	*1*	*In*	*Old signal*	*Selection of the Avalon bus module*
read/ read_n	1	In	No	Read request to the slave
ReadData	8, 16, 32, .. (1024)	Out	No	Read data from the slave module
write/ write_n	1	In	No	Write request to the slave
WriteData	8, 16, 32, .. (1024)	In	No	Data from Master to Slave module
Irq	1	Out	No	Interrupt request to the master

Avalon
« slave » signals

- The **Address**[n .. 0] is used to access a specific register/memory position in the selected module.
- An address is a **word address** view from the slaves. A word has the width of the slave interface: 8, 16, 32, 64, 128, 256, 512 or 1024 bits
- Only the minimum number of addresses is necessary. *Ex: a module with 6 internal registers needs 3 bits of addresses (6< 2**3)*

Avalon
« slave » signals

- The **ChipSelect** is generated by the Avalon bus and selects the module, is included in read/write signals. *Thus, it is deprecated*
- The **Read** and **Write** signals specifies the direction of the transfers and validate the cycle.
  They are provided by a Master and received by the slave modules
- The direction is the view of the Master unit
- **ReadData**(..) and **WriteData**(..) bus transfers the data from (read)/ to (write) the Slaves

Avalon
« slave » signals

- BE (**Byte Enable**) signals specify the bytes to transfers.
  - The number of BE activated are a power of 2
  - They start at a multiple of the size to transfer
- A **master address** is a **byte** address
- A **slave address** is a **word** address
- The Avalon make the addresses translation and the multiple accesses if necessary

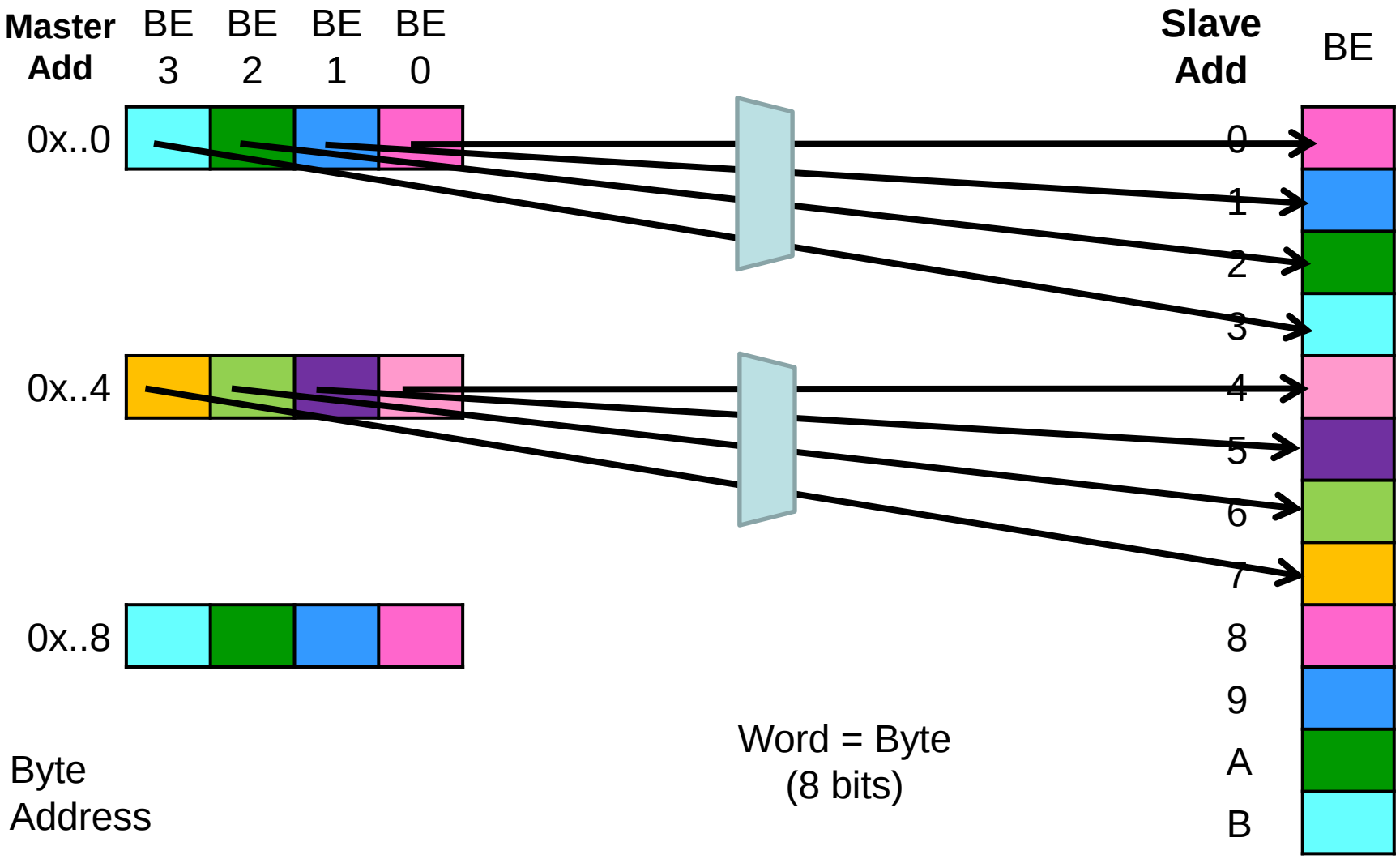
Avalon
Byte Enable (BE)

ByteEnable_n[3..0]	Transfer action
0 0 0 0	Full 32 bits access
1 1 0 0	Lower 2 Bytes access
0 0 1 1	Upper 2 Bytes access
1 1 1 0	Lower Byte (0) access
1 1 0 1	Mid Low Byte (1) access
1 0 1 1	Mid Upper Byte (2) access
0 1 1 1	Upper Byte (3) access

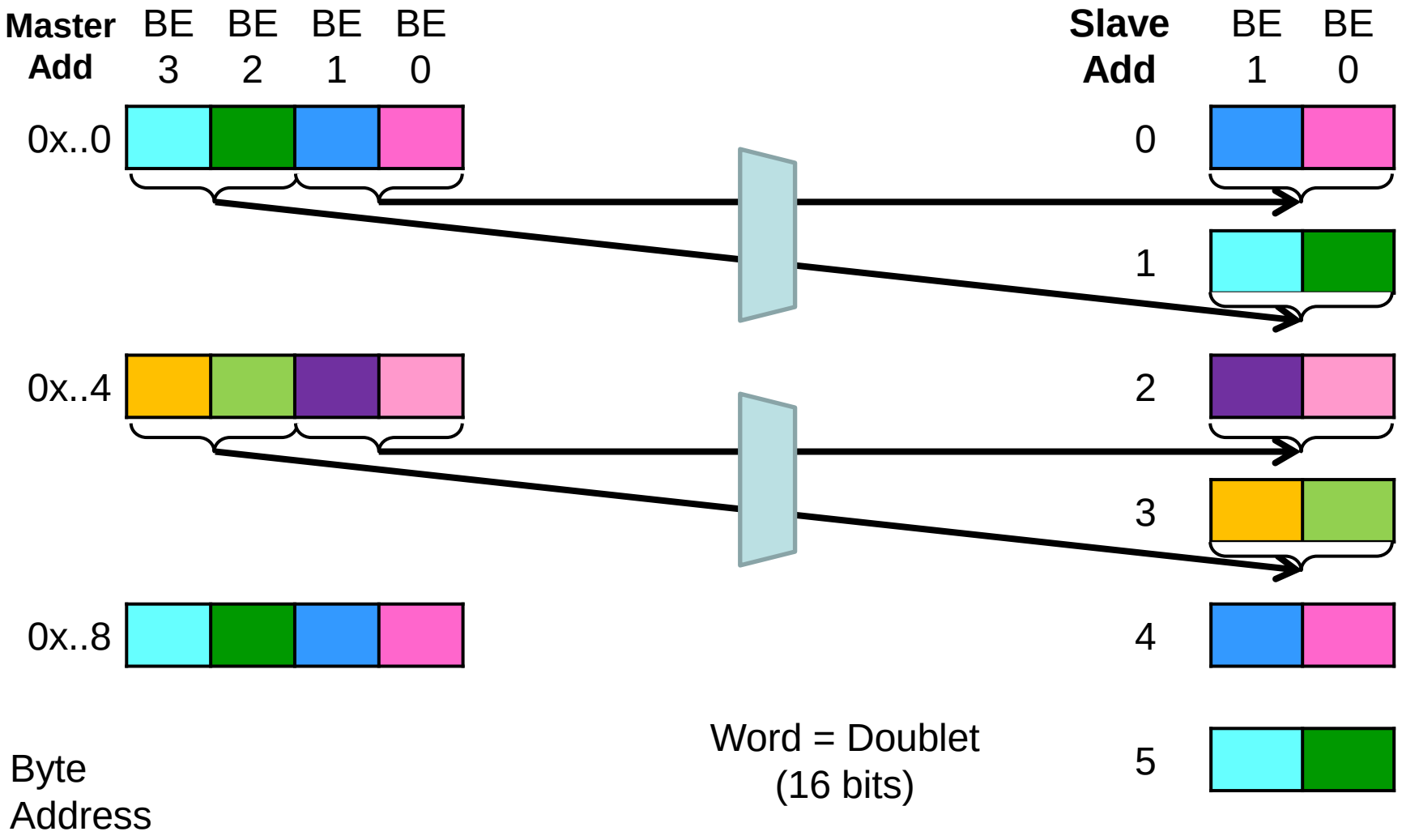
Specify bytes to be transferred
Active low signals in this representation:

- *byteenable_**n***

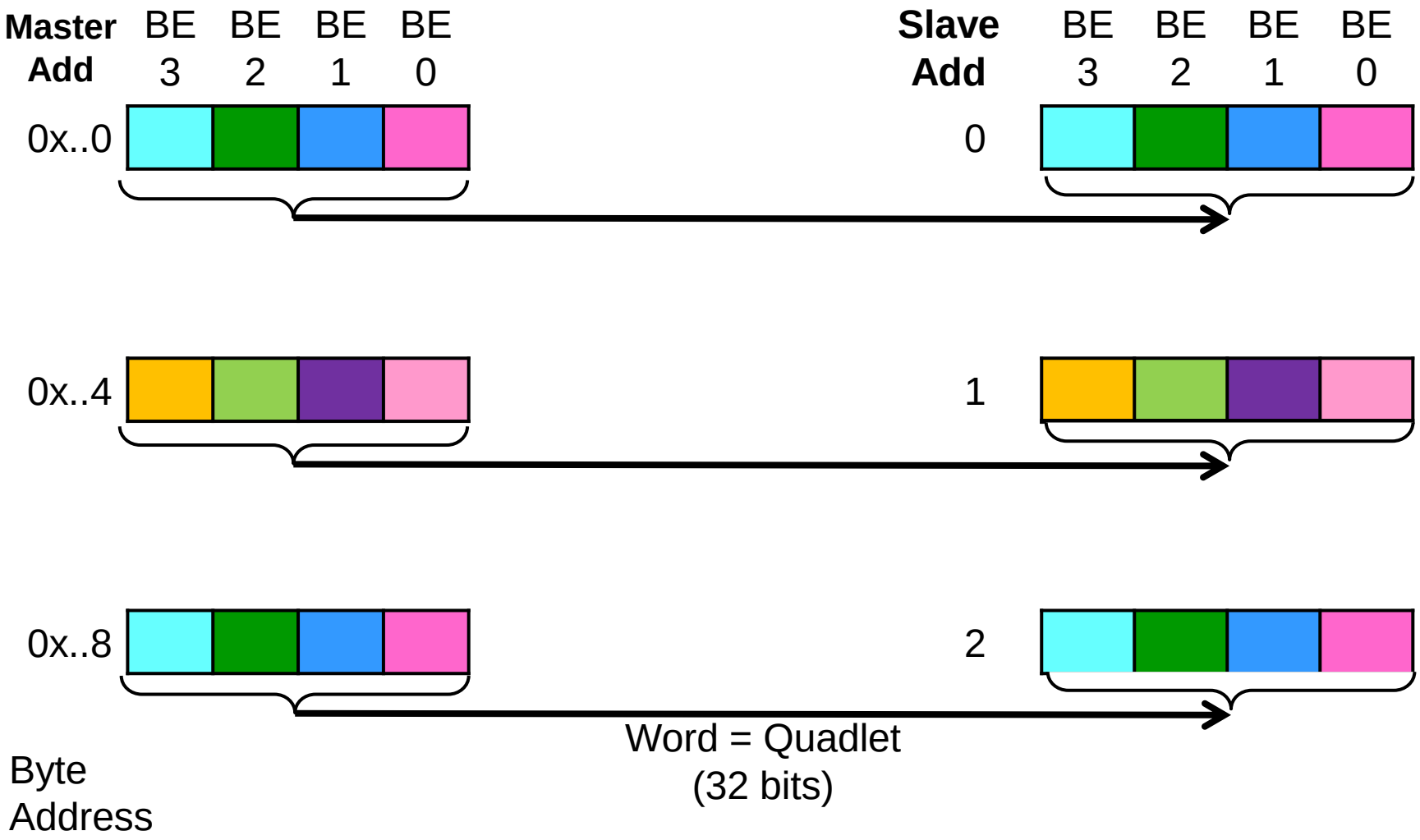
Avalon
Master to slave addresses : Master 32 bits, Slave 8 bits



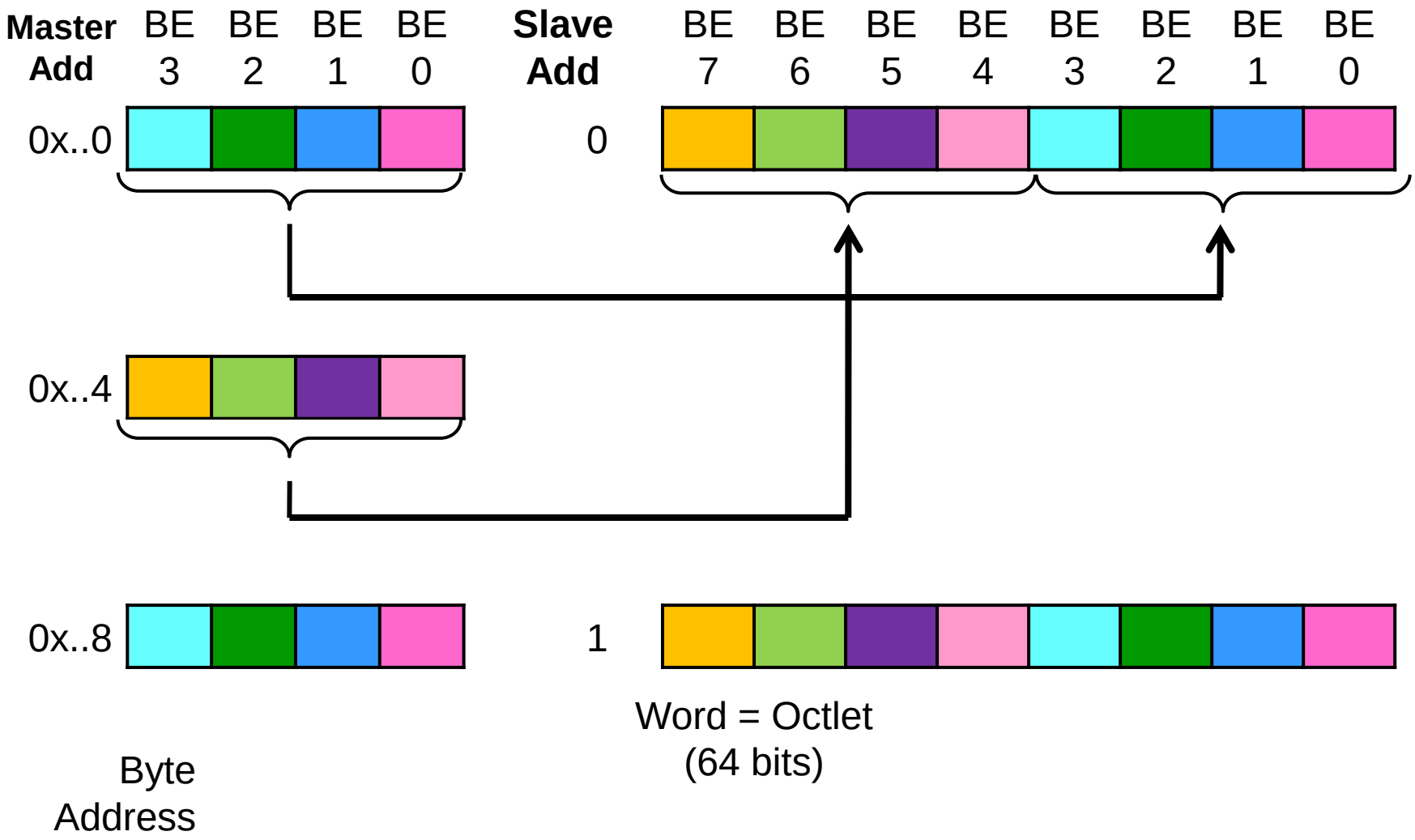
Avalon
Master to slave addresses : Master 32 bits, Slave 16 bits



Avalon
Master to slave addresses : Master 32 bits, Slave 32 bits



Avalon
Master to slave addresses : Master 32 bits, Slave 64 bits



Avalon « slave » signals

Signal Type	Width	Direction	Required	Description
WaitRequest/ WaitRequest_n	1	Out	No	Assert by the slave when it is not able to answer in this clock cycle to read or write access
ByteEnable/ ByteEnable_n	1, 2, 4, 8, .., 128	In	No	The bytes to transfer
BeginTransfer (deprecated)	1	In	No	Inserted by Avalon fabric at and only at first clock of each transfer
ReadDataValid/ ReadDataValid_n	1	Out	No	For read transfer with **variable latency or burst read**, means data are valid to master
BurstCount	1..11	In	No	Number of burst transfers
BeginBurstTransfer (deprecated)	1	In	No	First cycle of a burst transfer, valid for 1 clock cycle

Avalon « slave » signals

Signal Type	Width	Direction	Required	Description
ReadyForData	1	Out	No	
DataAvailable	1	Out	No	
ResetRequest/ ResetRequest_n	1	Out	No	
ArbiterLock/ ArbiterLock_n	1	In	No	

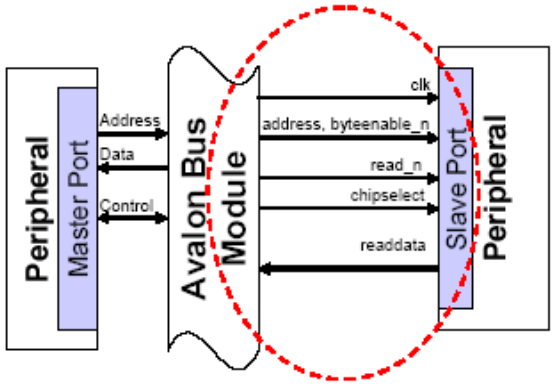
Avalon Bus

Slave view of transfers

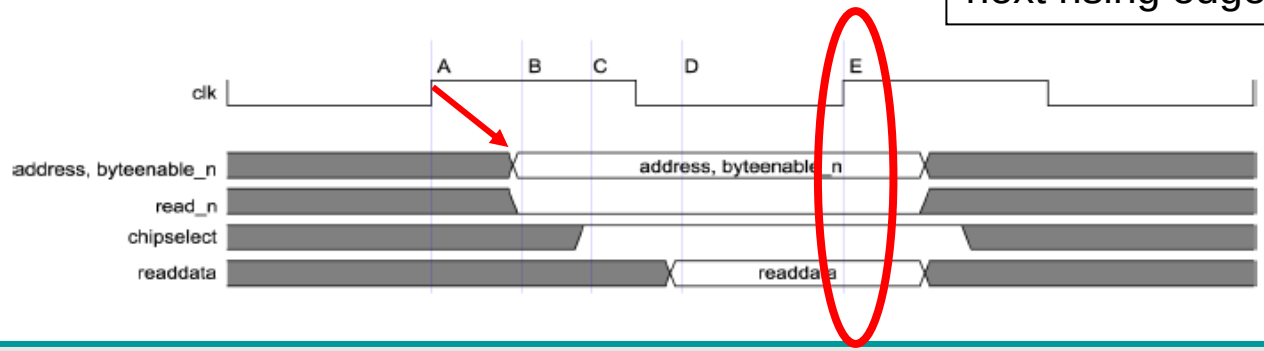
- Transfers are synchronous on the **rising** edge of the Clk
- Between Clk, the timing relation between signals are NOT relevant

Avalon (slave view)
Read transfer, 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"



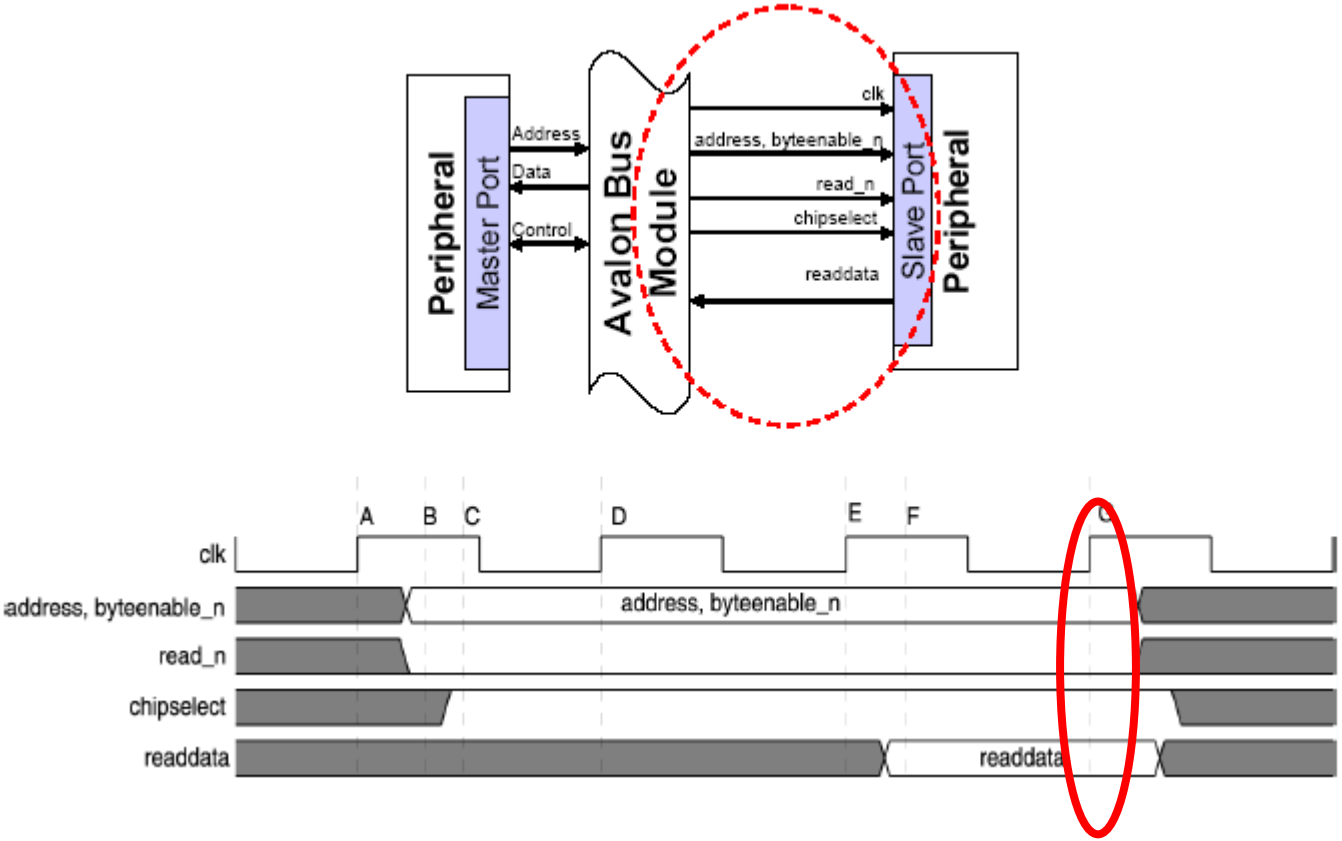
ReadData available at next rising edge of clk (E)



Avalon (slave view)
Read transfer, 1 wait

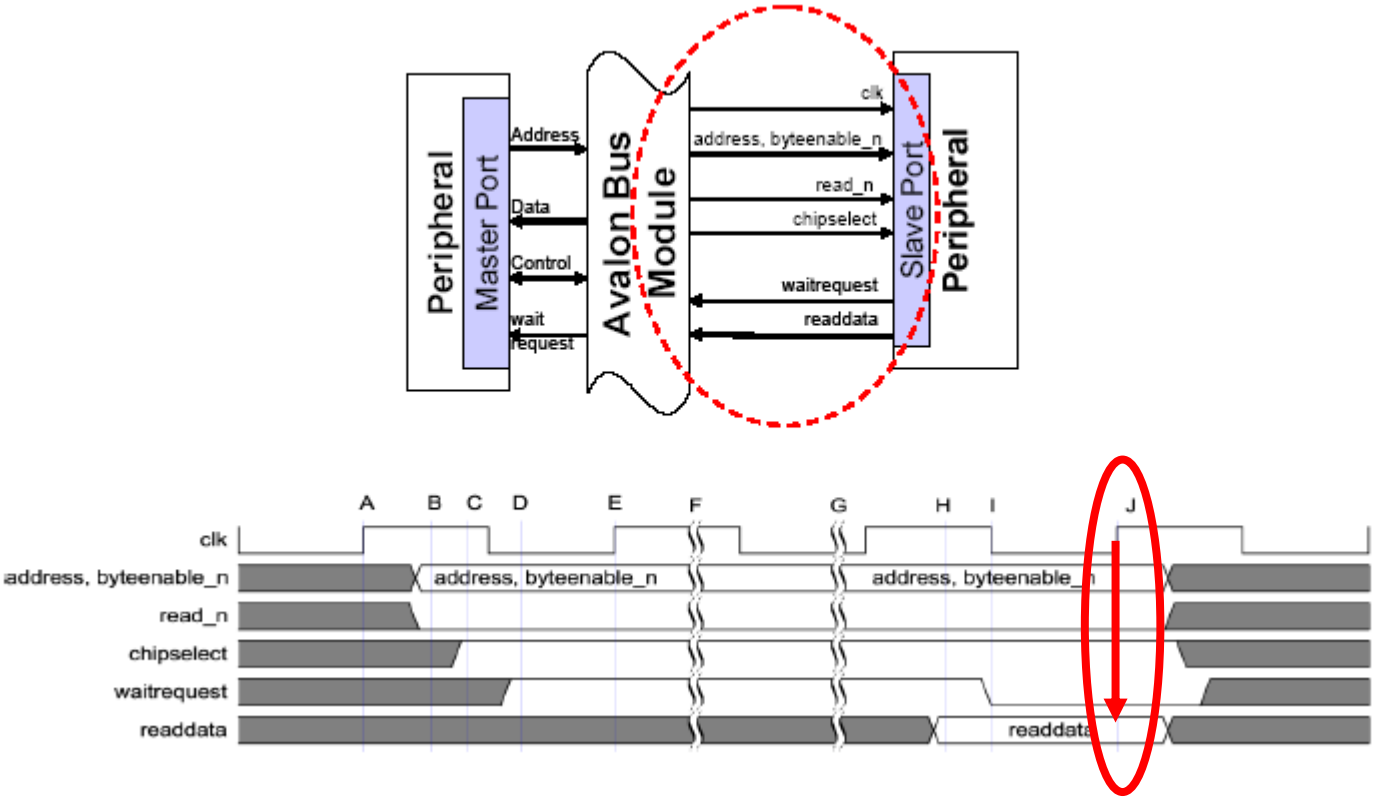
Wait cycle specified by design

Avalon (slave view)
Read transfer, 2 wait

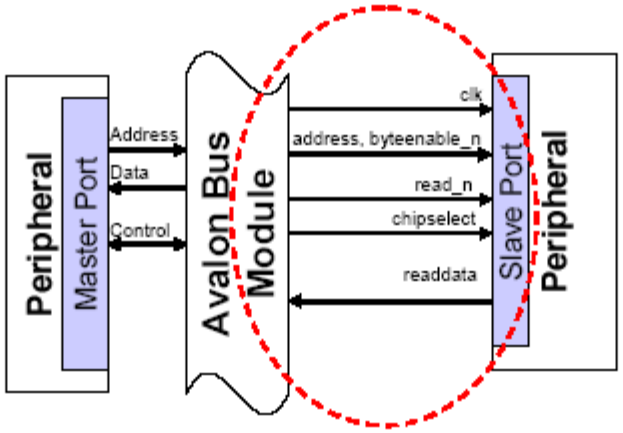


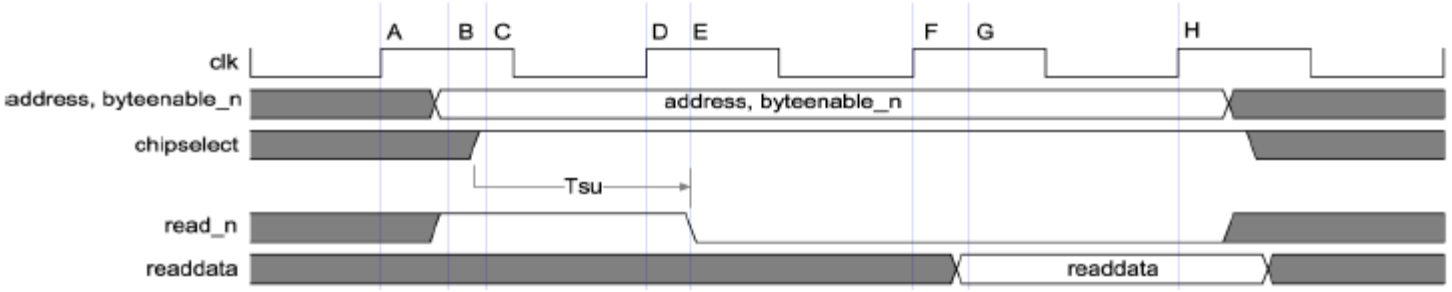
Avalon (slave view)

Read transfer, wait request generated by slave device

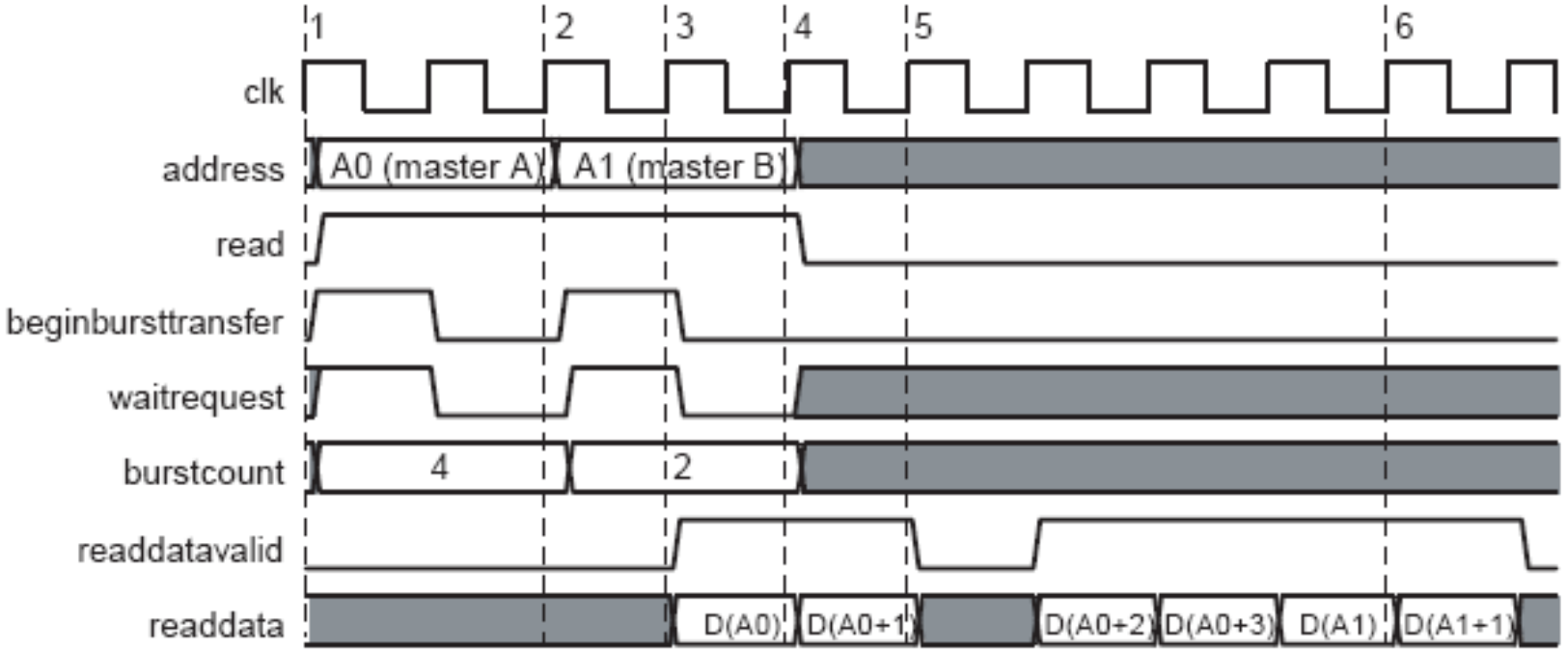


Avalon (slave view)
Read transfer, 1 set up and 1 wait





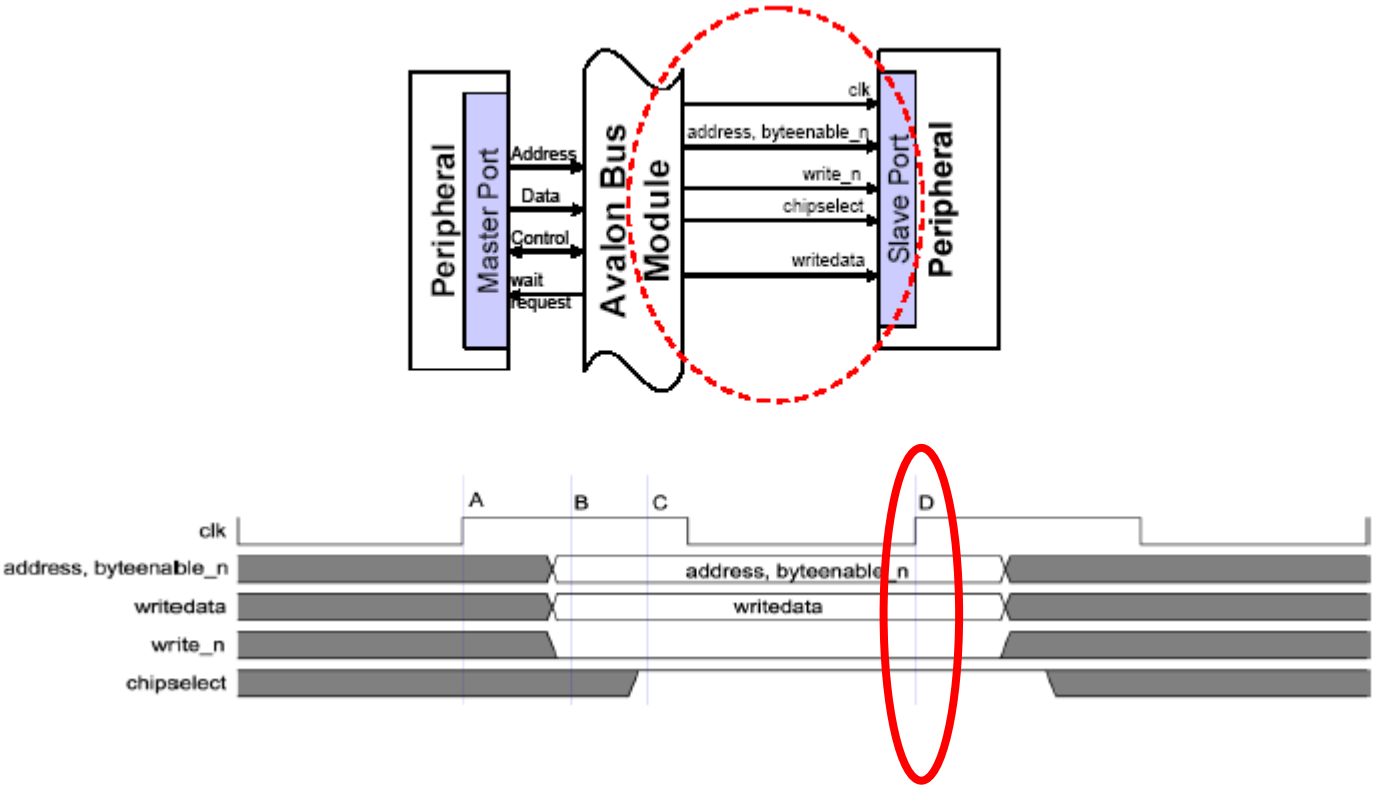
Avalon (slave view)
Read transfer, burst of 4 from Master A, 2 from master B



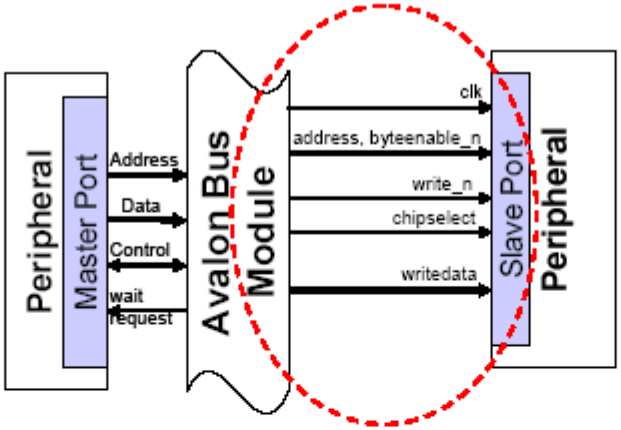
Pipeline of master access

ReadDataValid activated by slave for each data

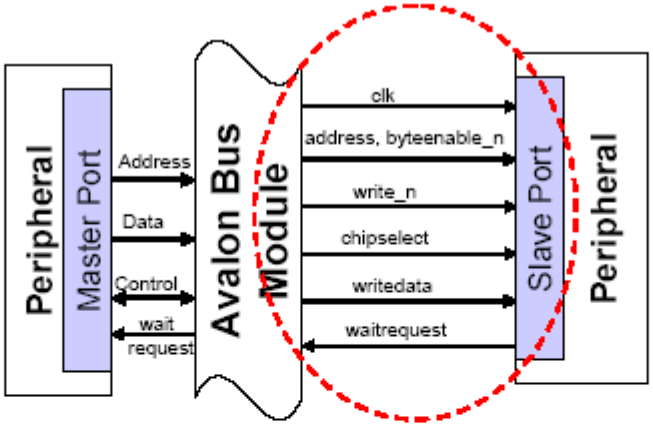
Avalon (slave view)
Write transfer, 0 wait

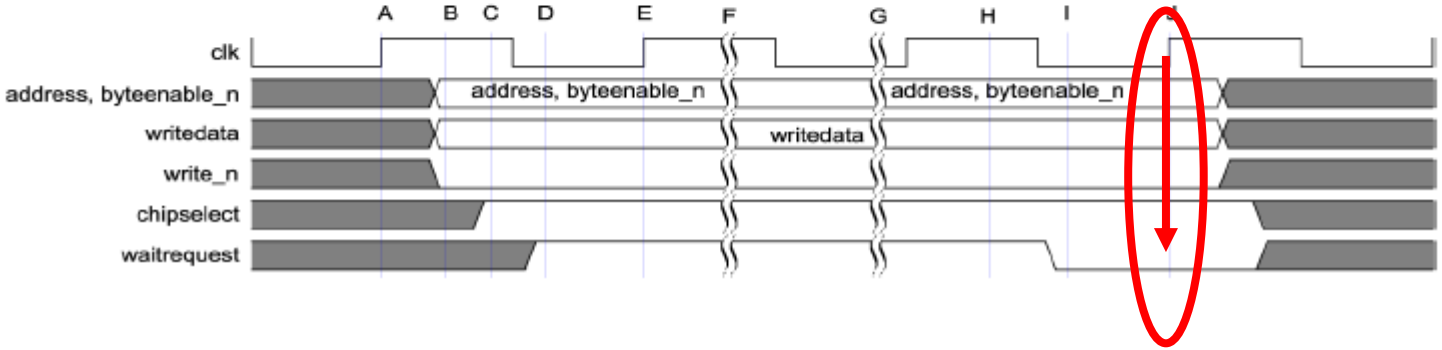


Avalon (slave view)
Write transfer, 1 wait

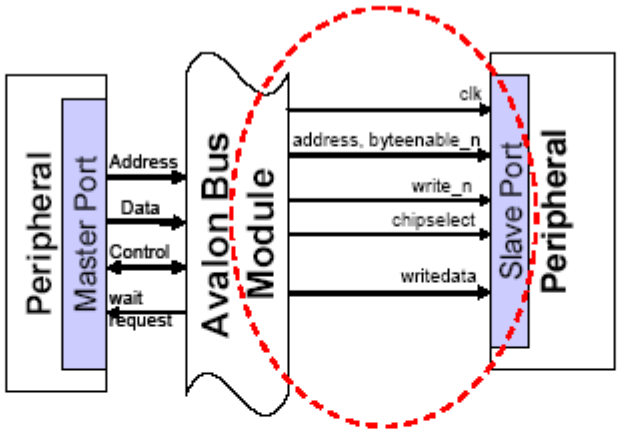


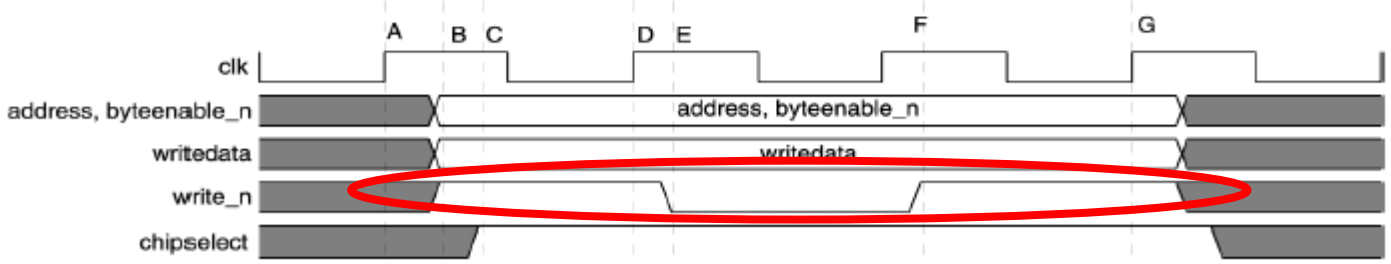
Avalon (slave view)
Write transfer, wait request generated by slave





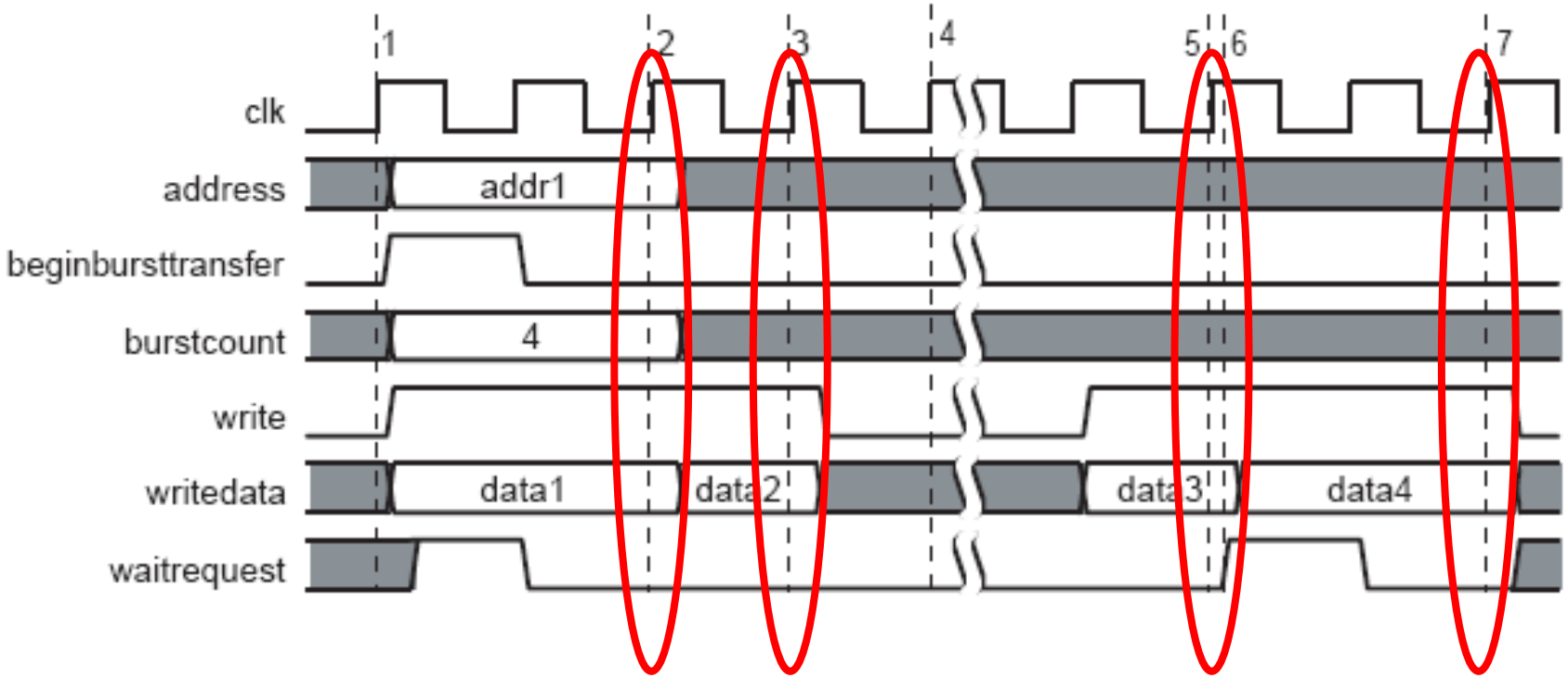
Avalon (slave view)
Write transfer, 1 set up, 1 hold, 0 wait



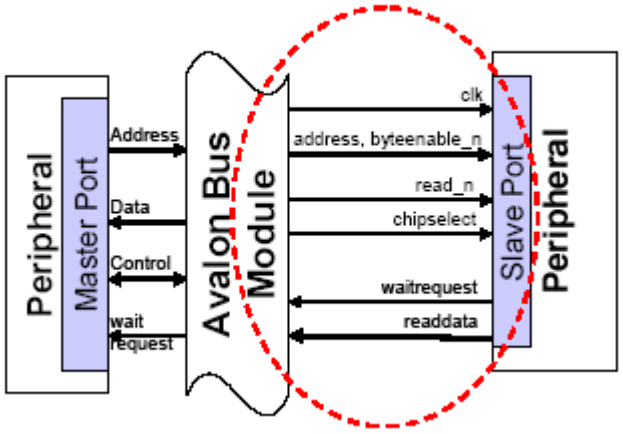


1 su 1 hold

Avalon (slave view)
Write transfer, burst transfer of 4, wait request generated by slave

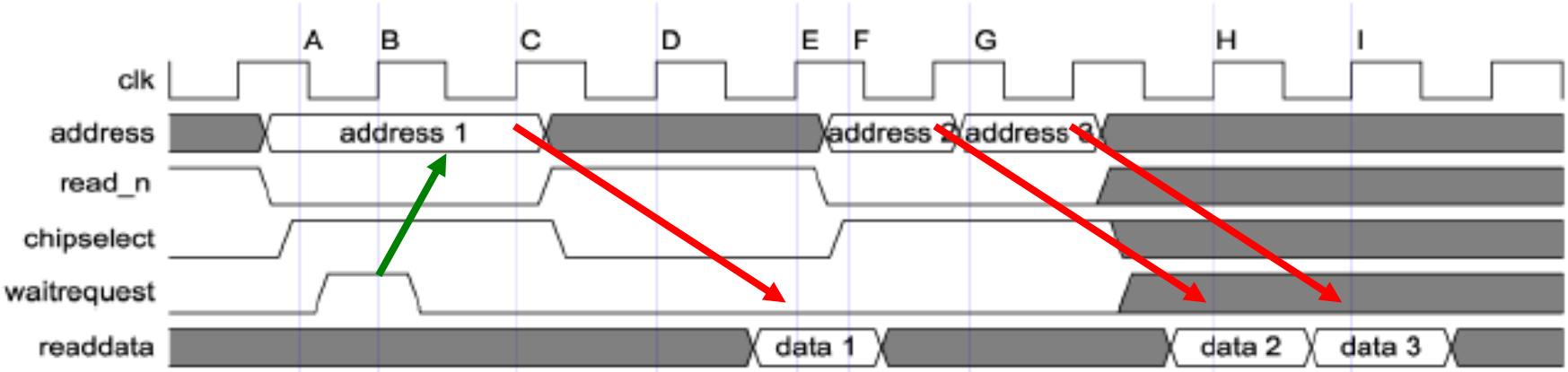


Avalon (slave view)
Read transfers with latency (ex. 2 cycles)



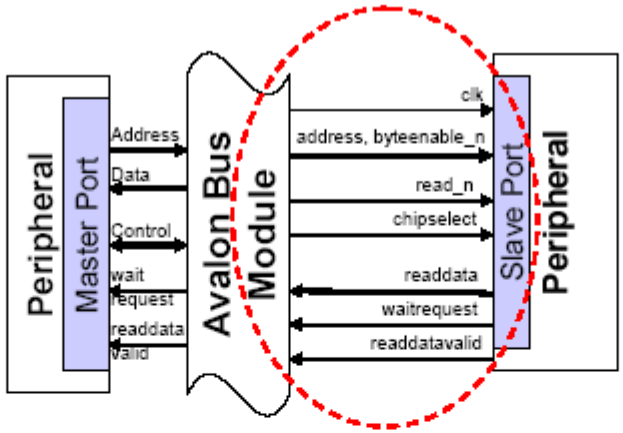
Wait request here means :

delay address cycle
Fixed latency (here 2)

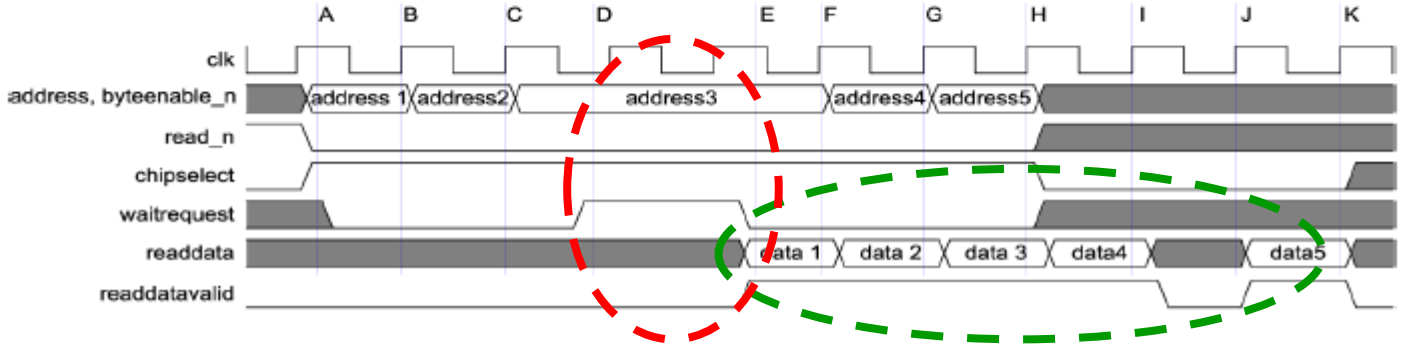


Avalon (slave view)

Read transfers with latency, and ***readdatavalid*** generated by slave



Readdatavalid specify when data are ready



Bus avalon

Master view

- The master start a transfer (read or write)
- It provide the Addresses (32 bits on NIOSII)
- It waits on **WaitRequest** signal to resume the transfer

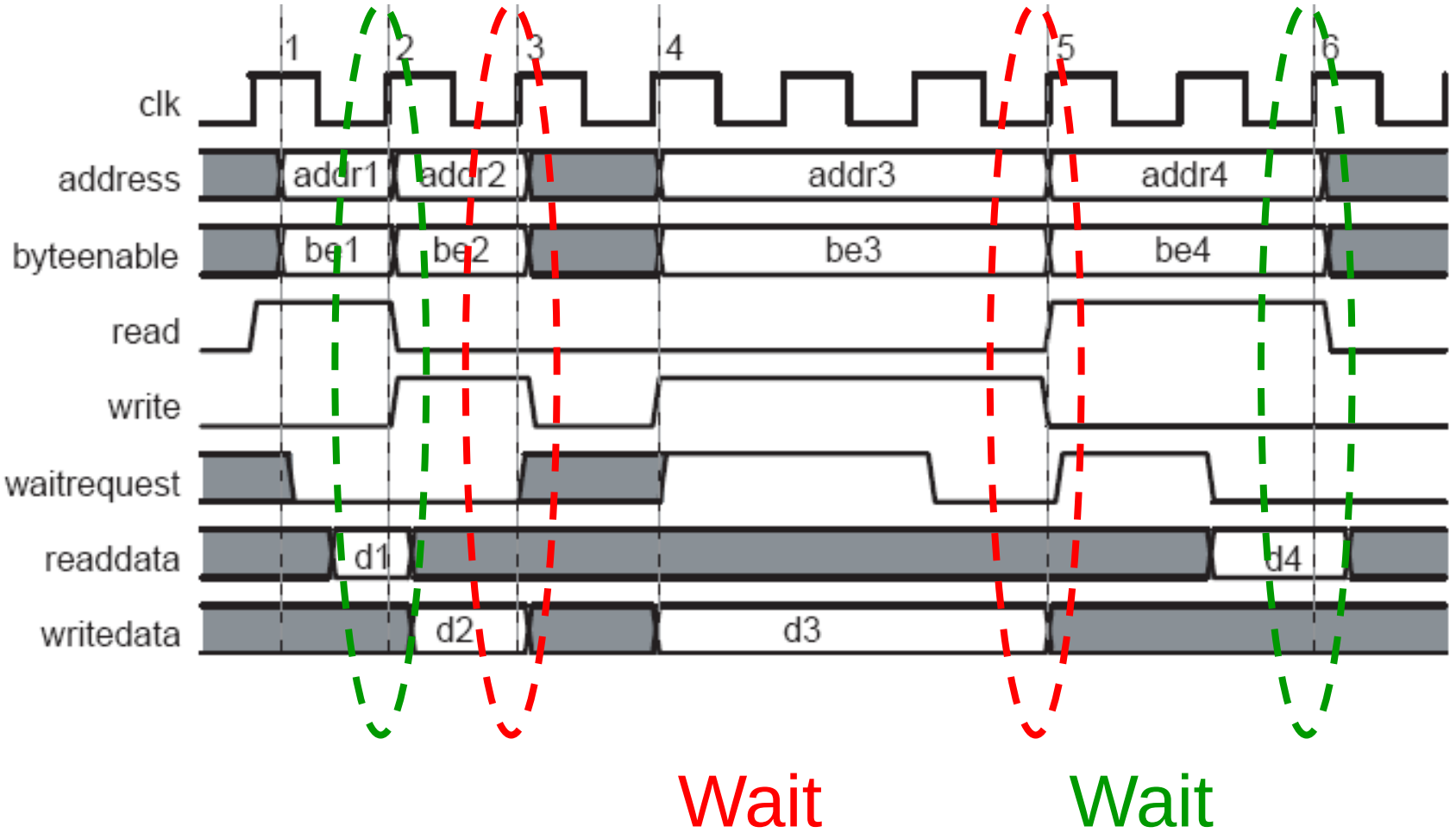
Avalon
master signals (1)

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.

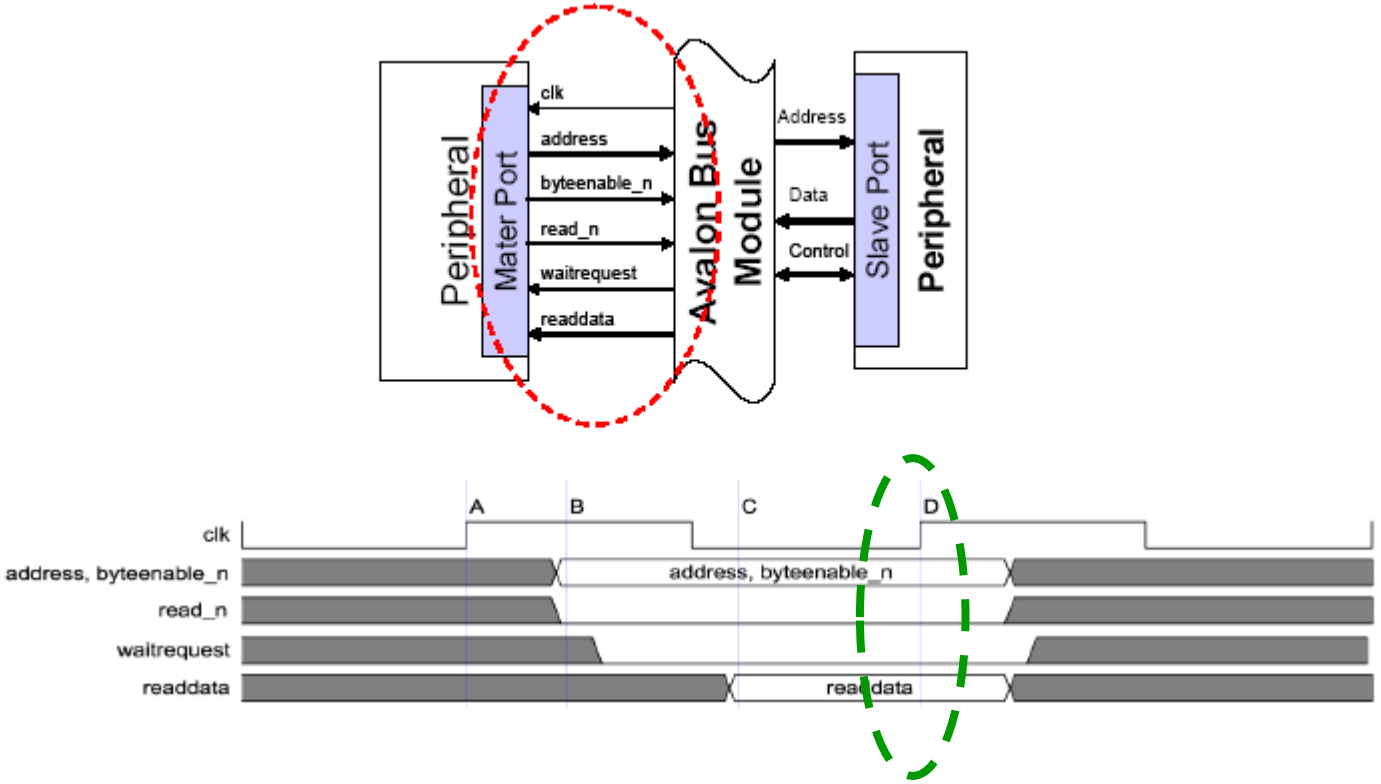
Avalon
master signals (2)

Signal Type	Width	Direction	Required	Description
writedata	8, 16, 32	out	no	Data lines to the Avalon bus module for write transfers. Not required if the master never performs write transfers. If used, write must also be used.
waitrequest	1	in	yes	Forces the master port to wait until the Avalon bus module is ready to proceed with the transfer.
irq	1	in	no	Interrupt request has been flagged by one or more slave ports.
irqnumber	6	in	no	The interrupt priority of the interrupting slave port. Lower value has higher priority.
endofpacket	1	in	no	Signal for streaming transfers. May be used to indicate an end of packet condition from the slave to the master port. Implementation is peripheral-specific.
readdatavalid	1	in	no	Signal for read transfers with latency and is for a master only. Indicates that valid data from a slave port is present on the readdata lines. Required if the master is latency-aware.
flush	1	out	no	Signal for read transfers with latency. Master can clear any pending latent read transfers by asserting flush.

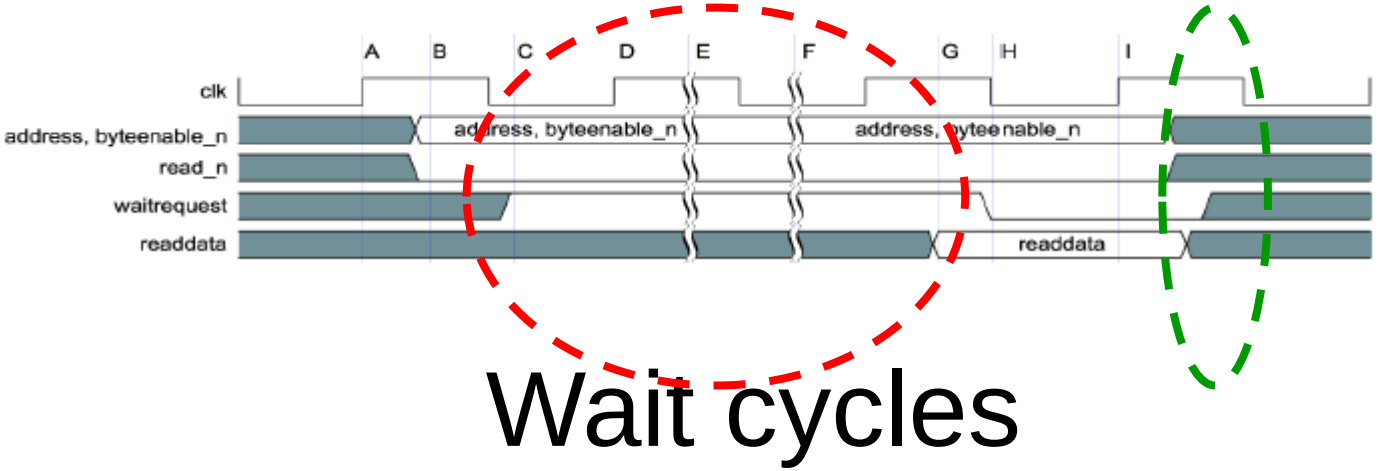
Avalon (Master view)
Basic fundamental transfers



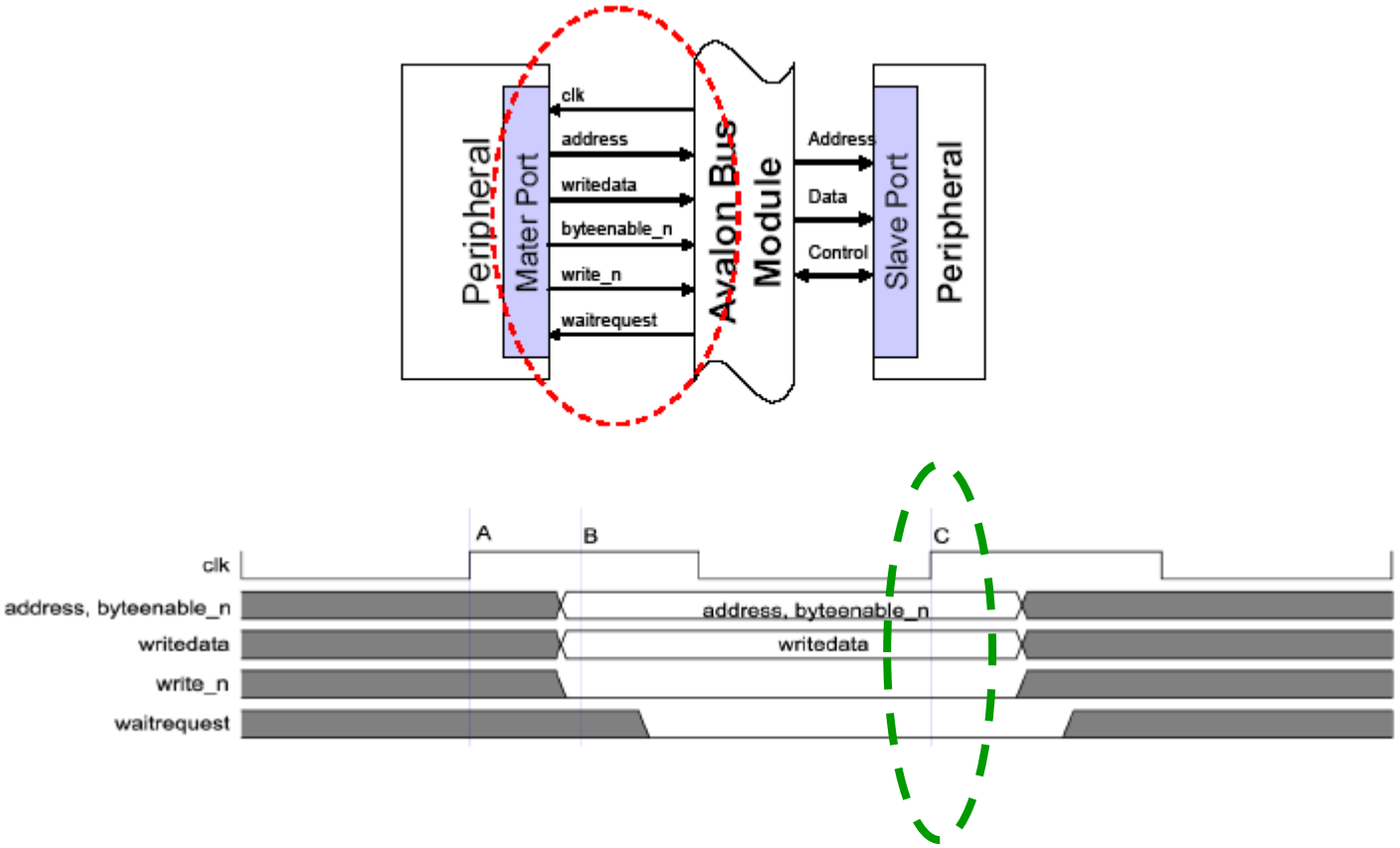
Avalon (Master view)
Read transfer, 0 wait



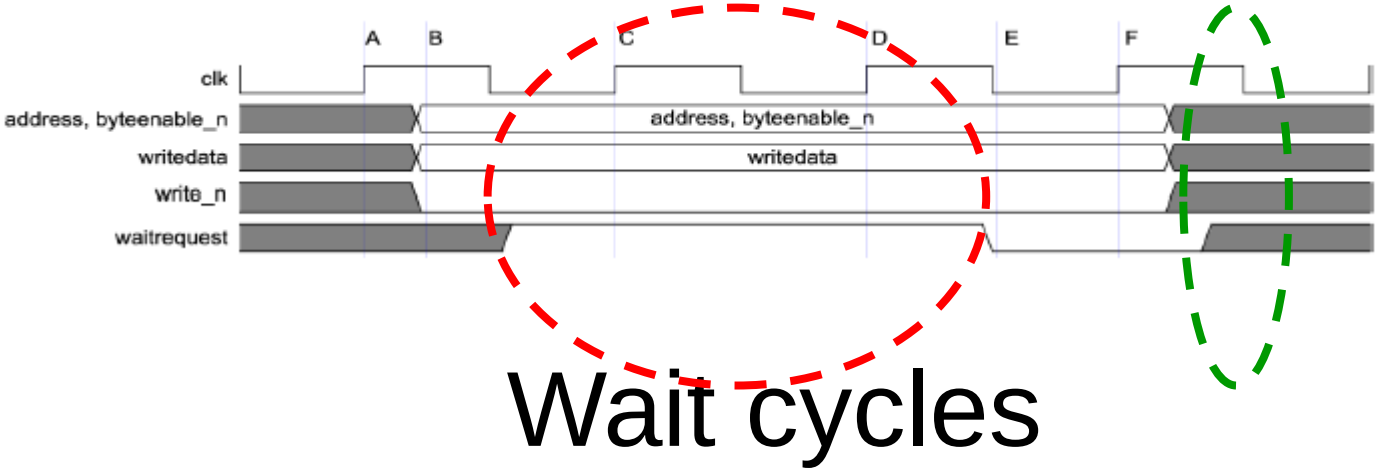
Avalon (Master view)
Read transfer, wait generated by slave/Avalon bus



Avalon (Master view)
Write transfer, 0 wait

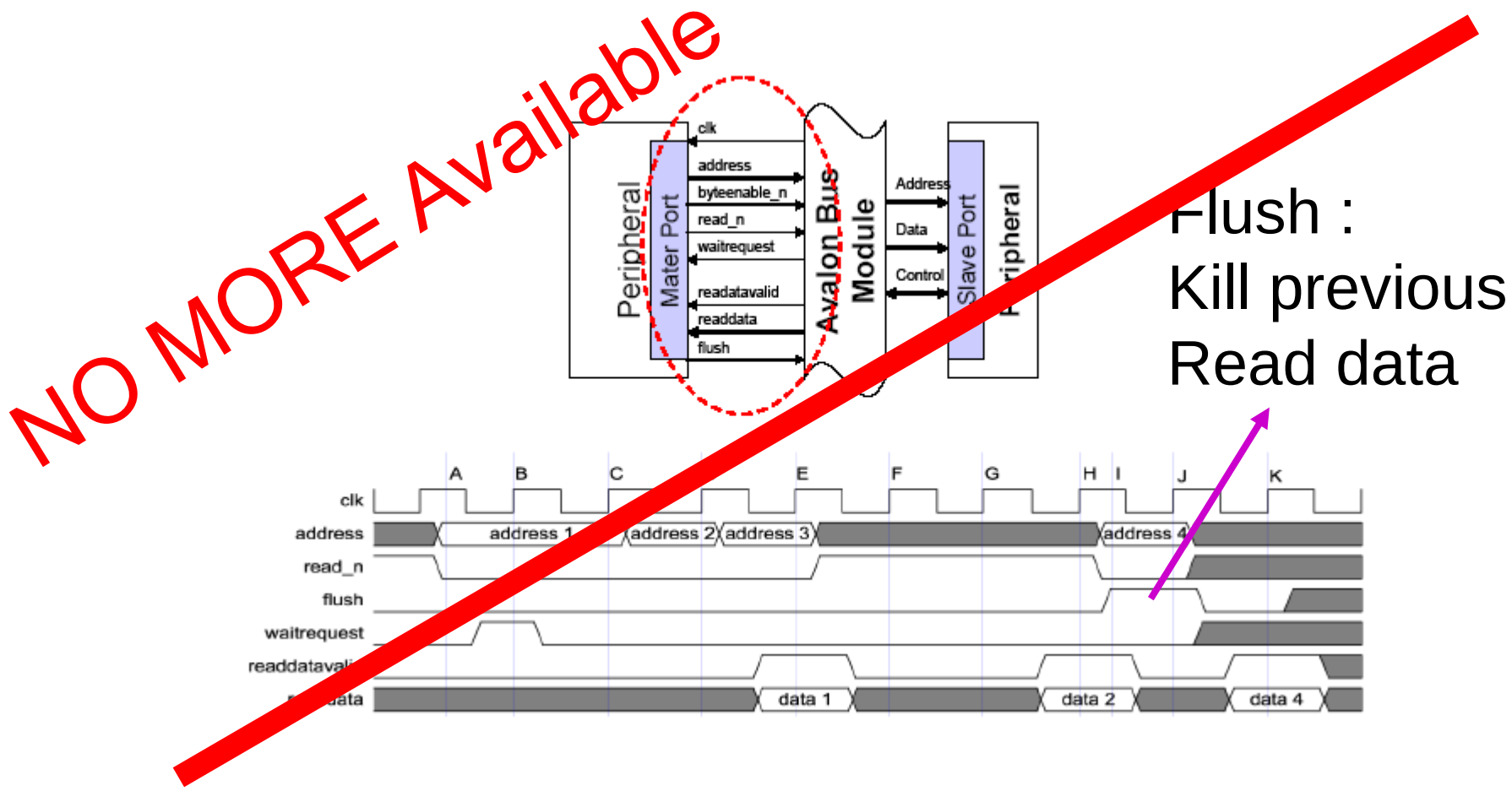


Avalon (Master view)
Write transfer, wait generated by slave

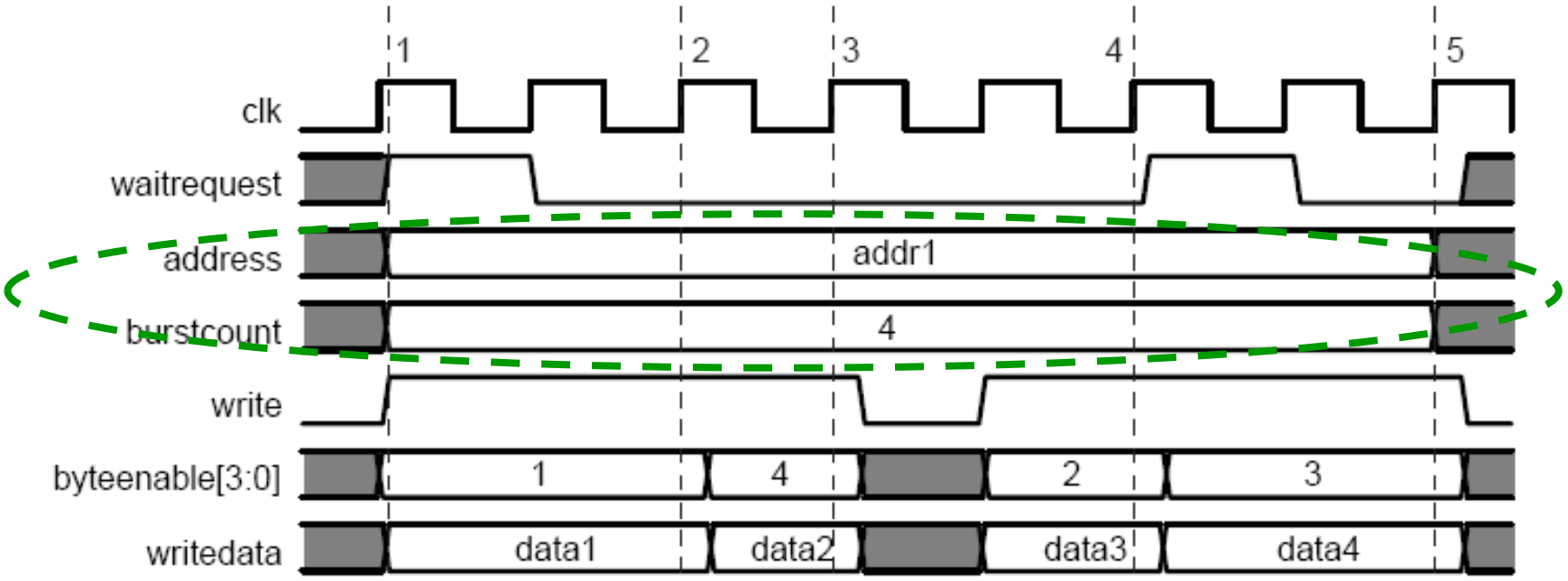


Avalon (Master view)

Read transfers with latency, and ***readdatavalid*** generated by slave

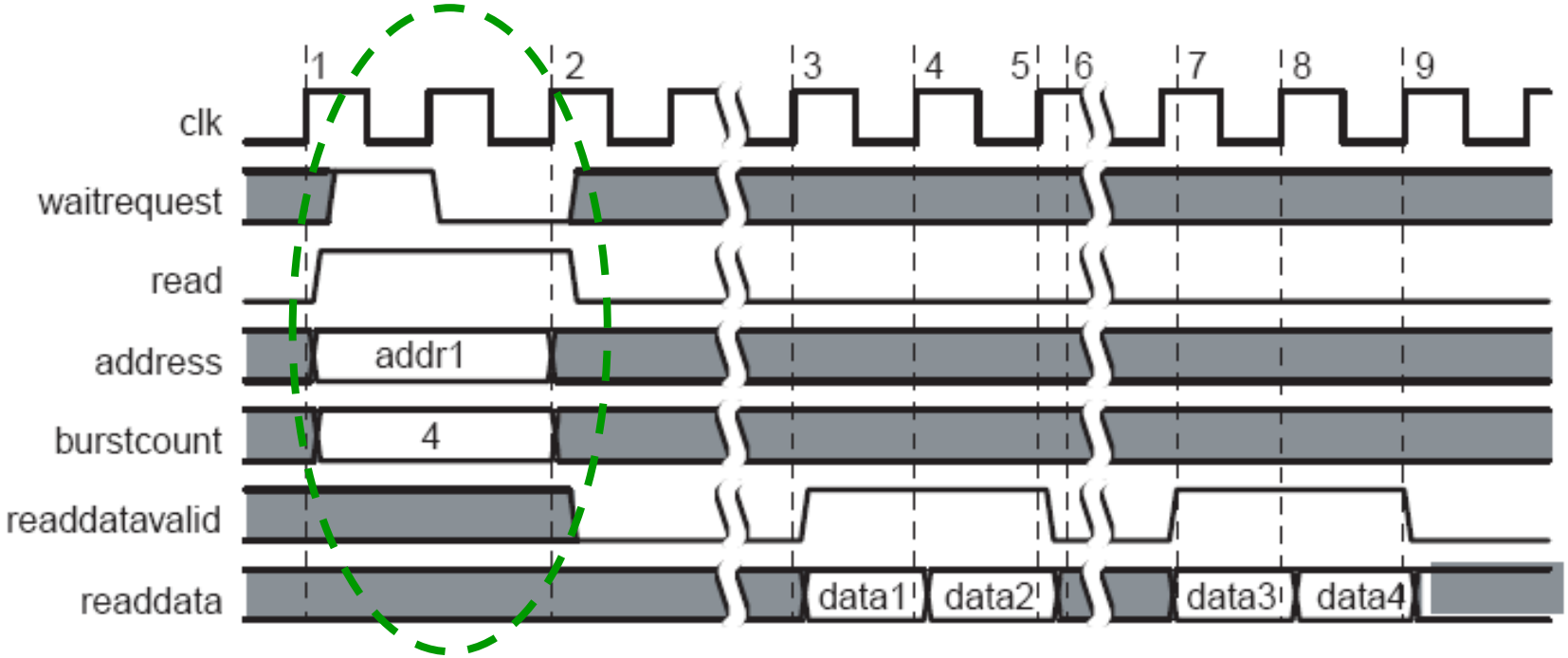


Avalon (Master view)
Burst Write transfers



Address and BurstCount available for the whole transfer
Write can be deactivated by the master
The number of burstcount needs to be generated

Avalon (Master view)
Burst Read transfer



Address and BurstCount available for the first cycle only
Read signal only for the first cycle
The number of burstcount ReadDataValid needs to be generated
The master could start a new transfer in 2

WaitReq generation

- The **WaitReq** signal can be generated by different sources, for different purposes:
  - Master access to a slave to extend the cycle
    - Generated by the slave itself or the Avalon bus
  - Multiple access needed by the Avalon to a slave, as ex. a 32 bits transfers to an 8 bits slave, need 1 master transfert for 4 slave transferts
  - Multi-master accesses to the same slave, one master can have the acces, the others are delayed with WaitReq

Bus avalon transfers resume

- Separate :
  - address, data in, data out
- Synchronous on clock’s rising edge
- Bus Internal or external wait request
- Transfers with latency available
- Multi-masters
- Arbitration at slave side

Avalon Address view

- 2 different views of addresses from master and slave, mode of decoding :
  - **Memory** (dynamic bus sizing)
  - **Register** (native transfers) ← ***deprecated***
- Example :
  - Master 32 bits data
  - Slave 8 bits data

Data bus seen on the Avalon Master side

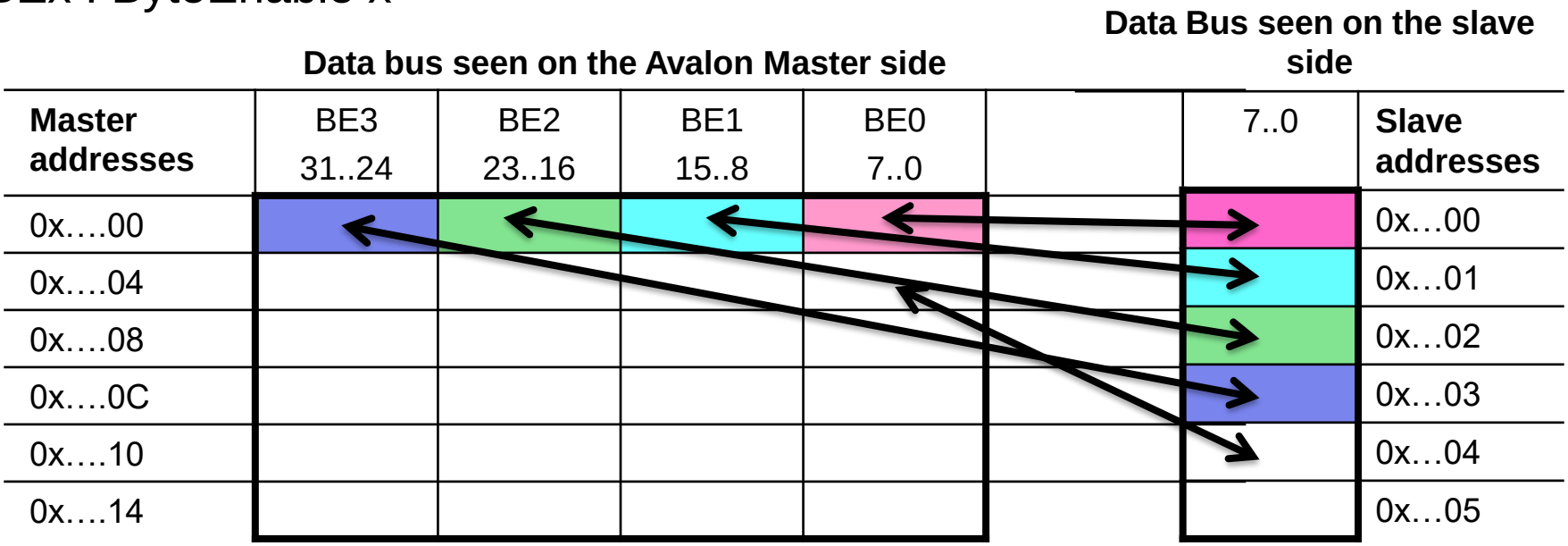
Master addresses	31..24 +3	23..16 +2	15..8 +1	7..0 +0
0x….00				
0x….04				
0x….08				
0x….0C				
0x….10				
0x….14				

Data Bus seen on the slave side

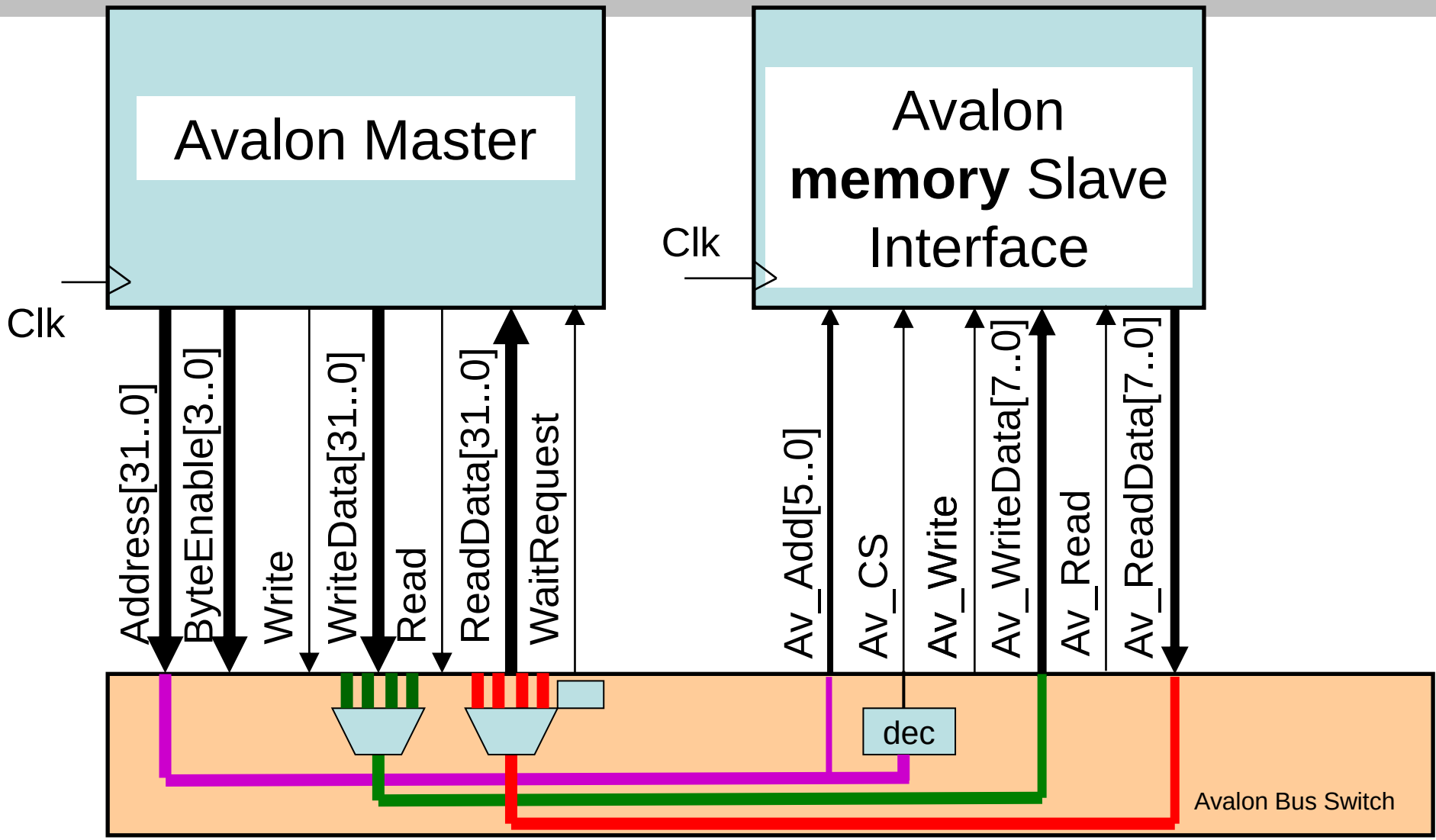
7..0	**Slave addresses**
	0x…00
	0x…01
	0x…02
	0x…03
	0x…04
	0x…05

Address view, Memory model

- **Memory model, dynamic bus sizing** :
  - No hole in the master address space
  - Need multiplexers on the data path
  - Master byte address = Slave byte address
  - 1 x 32 bits master transfer → 4 x 8 bits slave access by Avalon switch
- BEx : ByteEnable x

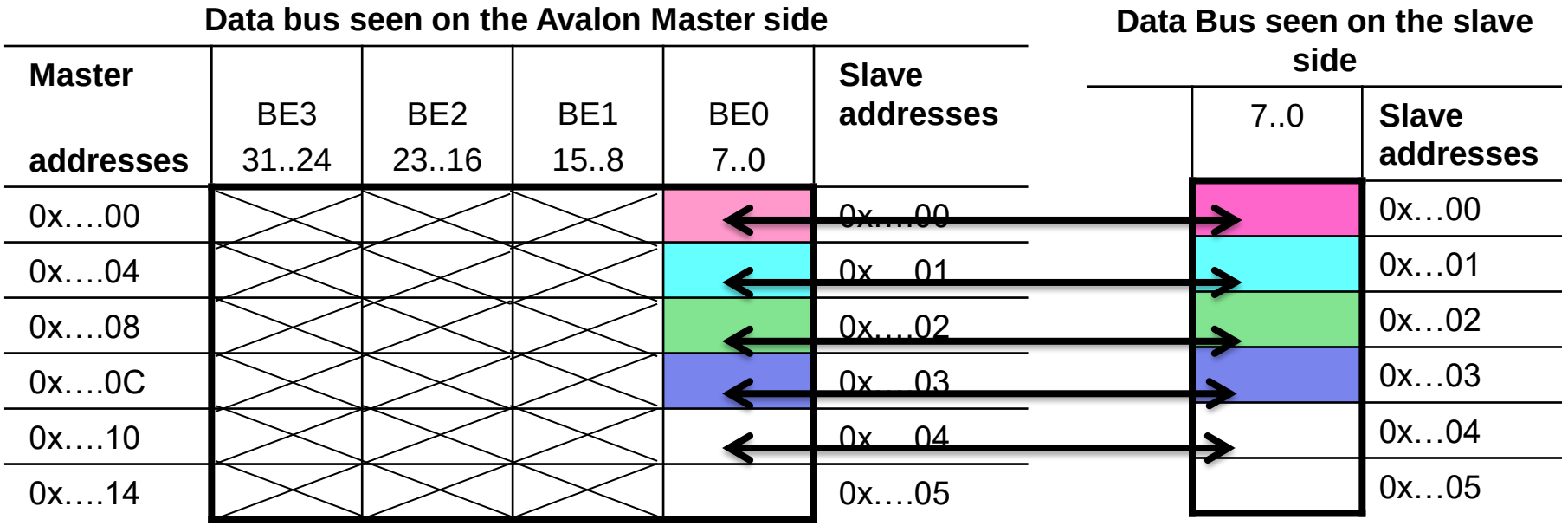


Memory model for Avalon memory slave

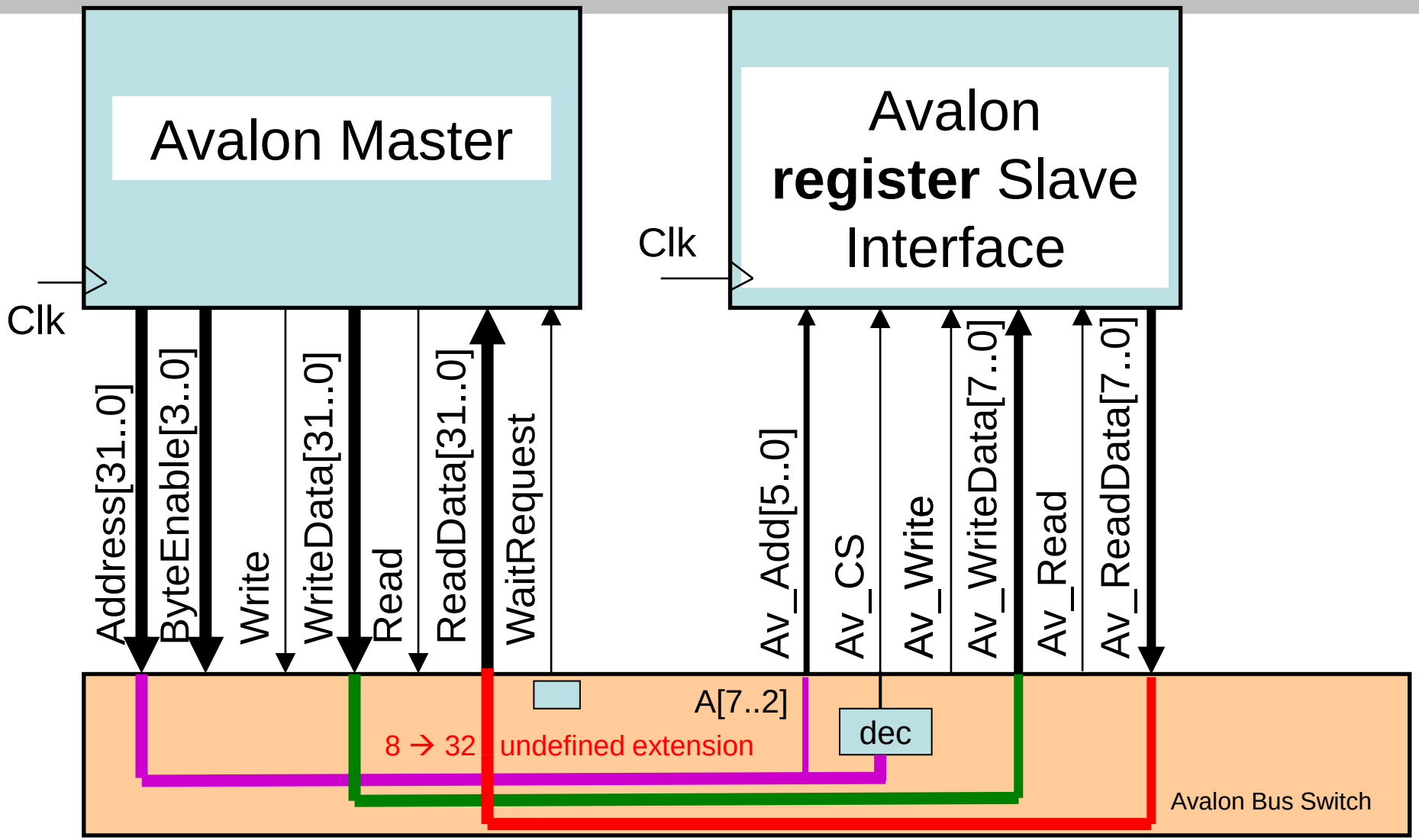


Address view, Register model (deprecated)

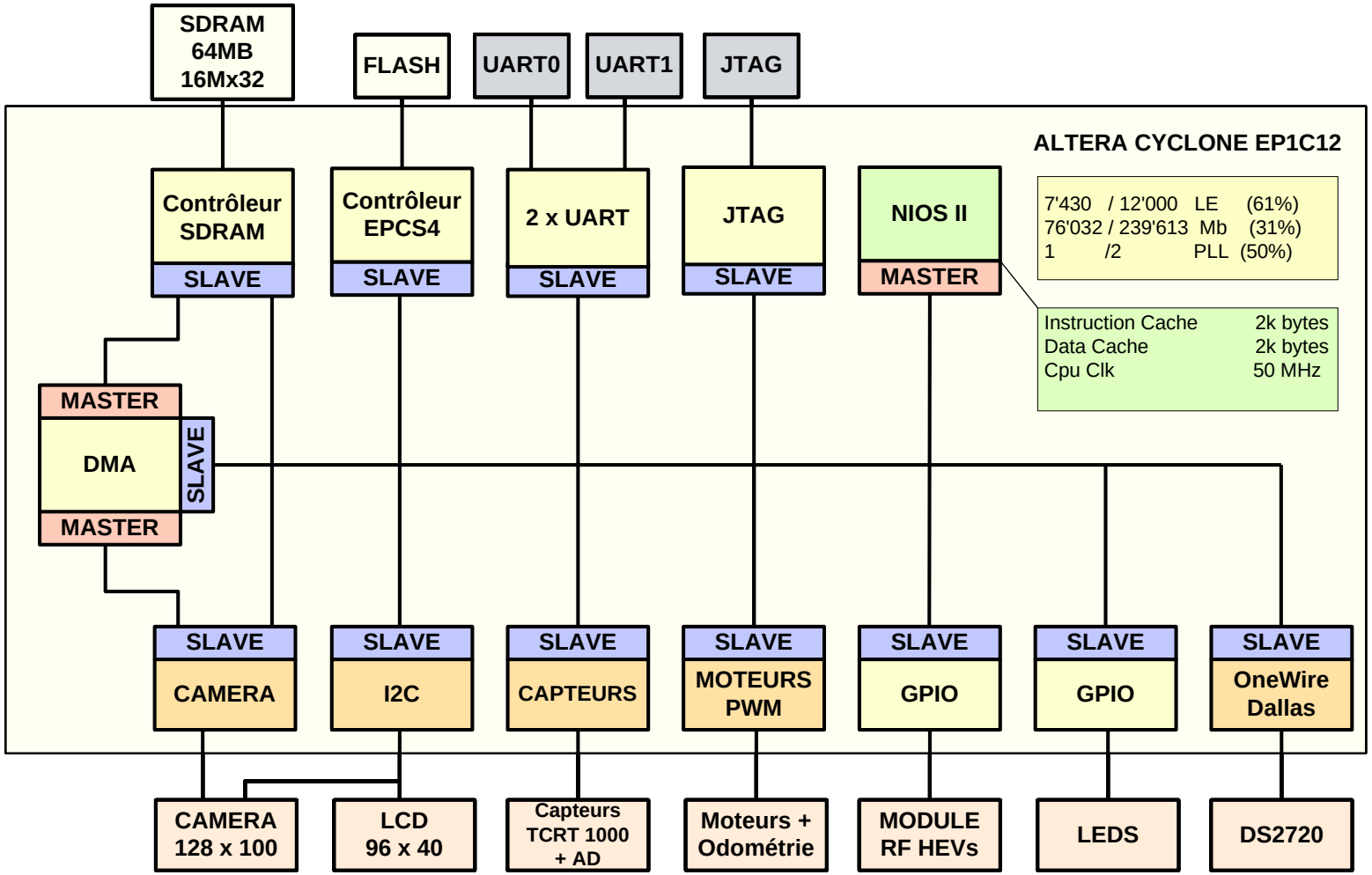
- **Register model**, **native transfer** :
  - Holes the master address space
  - NO multiplexers needed on the data path to align data
  - Master byte address ≠ Slave byte address
- Access by size of master bus (i.e. 32 bits), 8 bits available, highest bits undefined
- 1 master transfer = 1 slave transfer



Memory model for Avalon register slave



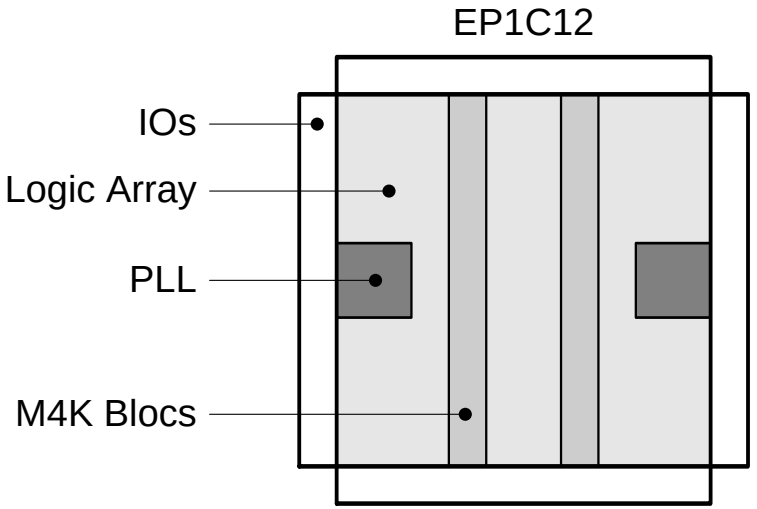
Embedded System on FPGA (example)



FPGA Architecture, ex. EP1C12 (1st Cyclone generation)

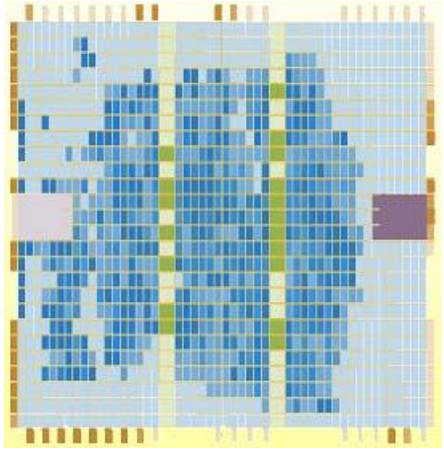
Architecture of EP1C12

- 12'000 logic Elements (LE)
- 52 x 4 Kbits RAM
- 2 x PLLs
- 180 IOs on 4 bancs
- Proprietary Configuration Bus
- JTAG Port

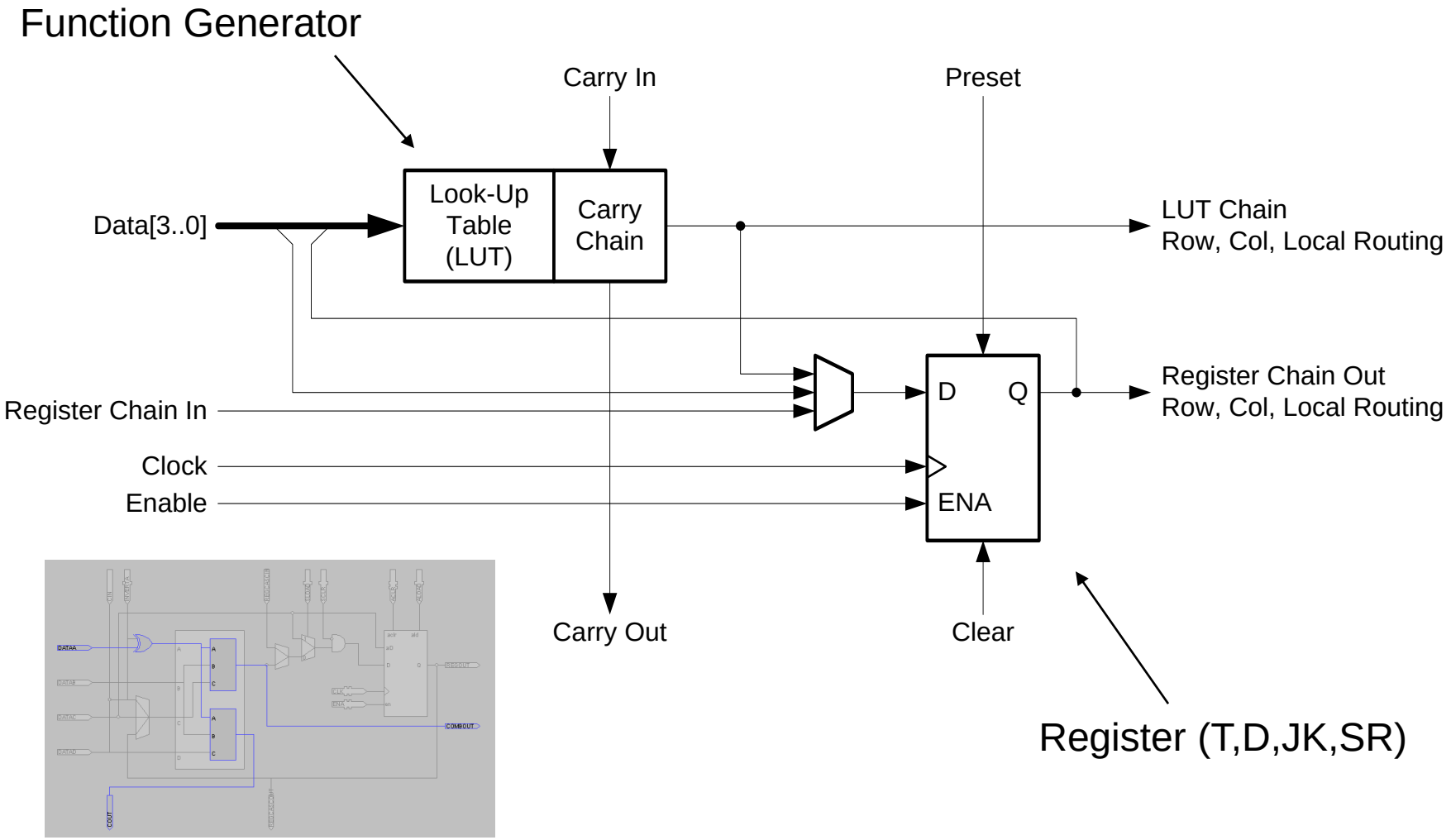


Quelques limites de fonctionnement

- multiplexor 16→1 : fmax LE = 275 MHz
- counter 64 bits : fmax LE = 160 MHz
- memory : fmax M4K = 220 MHz
- PLL : fmax PLL = 275 MHz

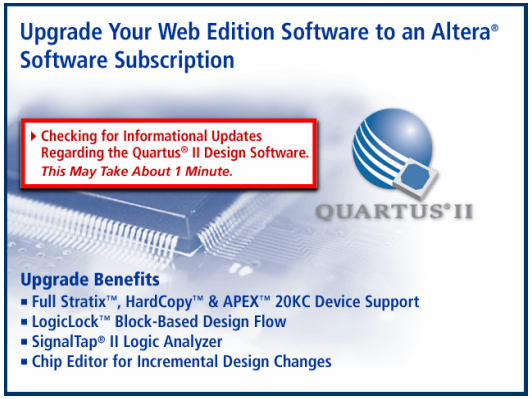


Logics Elements (LE)



Developments Tools from intelFPGA

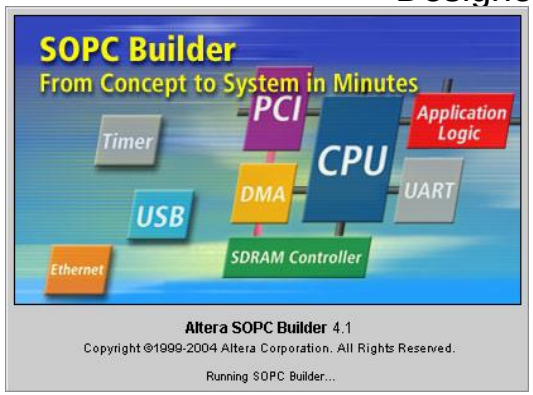
Quartus II → Hardware Description



- ❑ Schematic Editor, VHDL, …
- ❑ Synthesis + placement routing
- ❑ Simulation (graphical editor)
- ❑ Signal TAP

SOPC Builder → SOC NIOS II

2011 ⇒ Qsys
2018 ⇒ Platform Designer



- ❑ Configuration + SOC generation
- ❑ Programmable Interface library
- ❑ Own Programmable Interfaces.
- ❑ Generation SDK

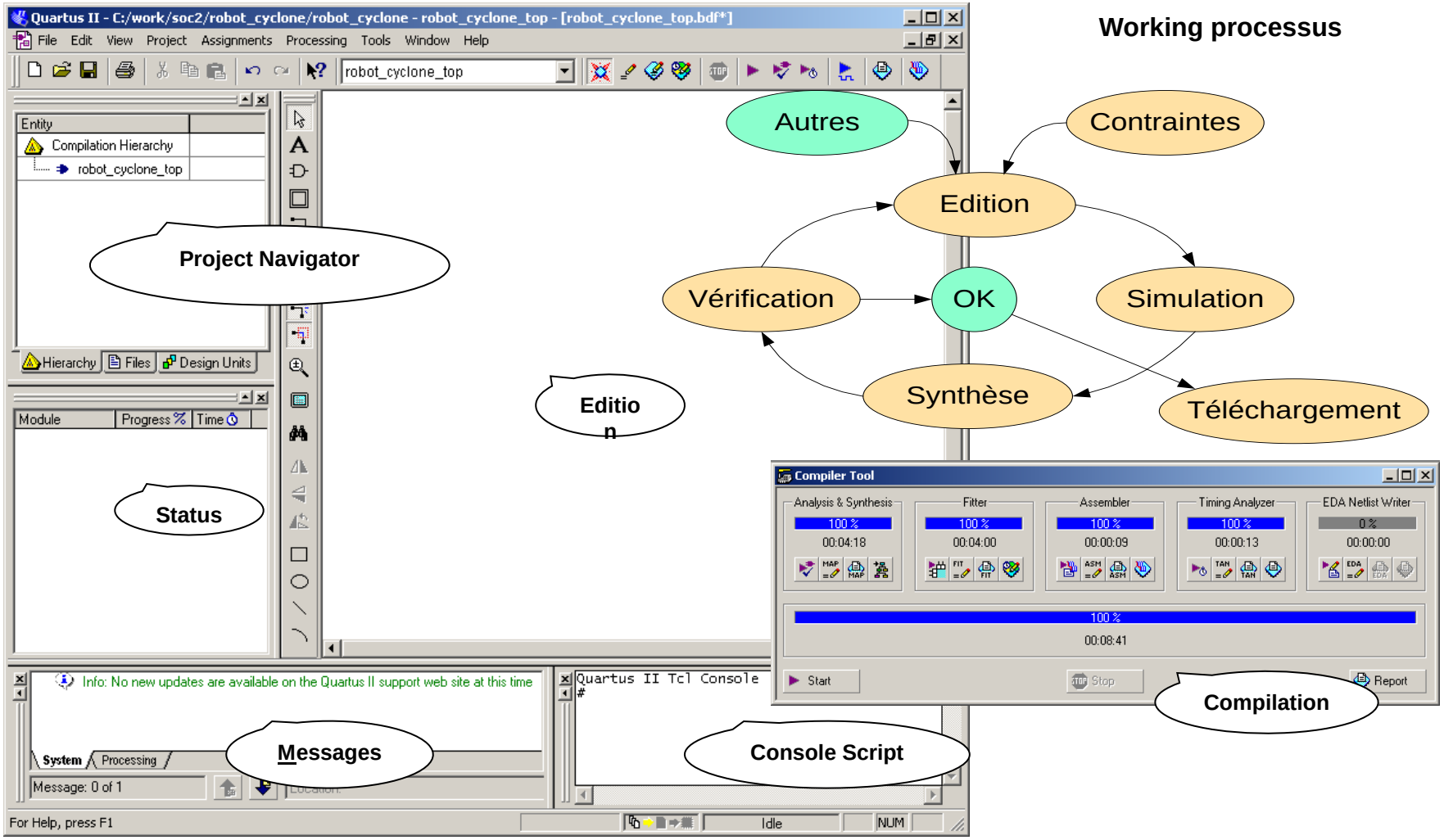
NIOS II IDE → NIOS II Code

2010 ⇒ SBP

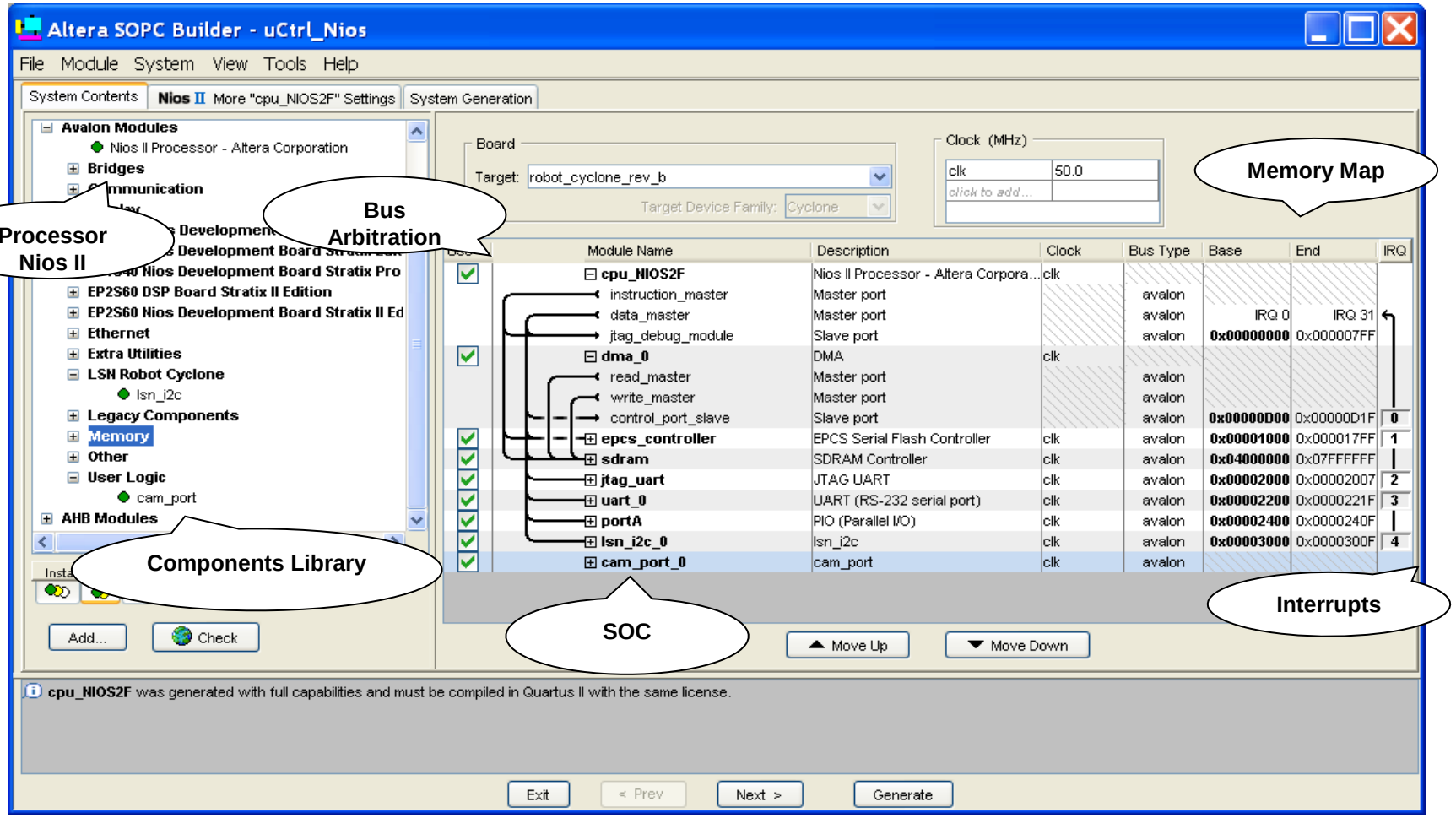
- ❑ Project management
- ❑ Compiler + Link Editor
- ❑ Debugger
- ❑ SOC Programmer

Developments Tools from intelFPGA

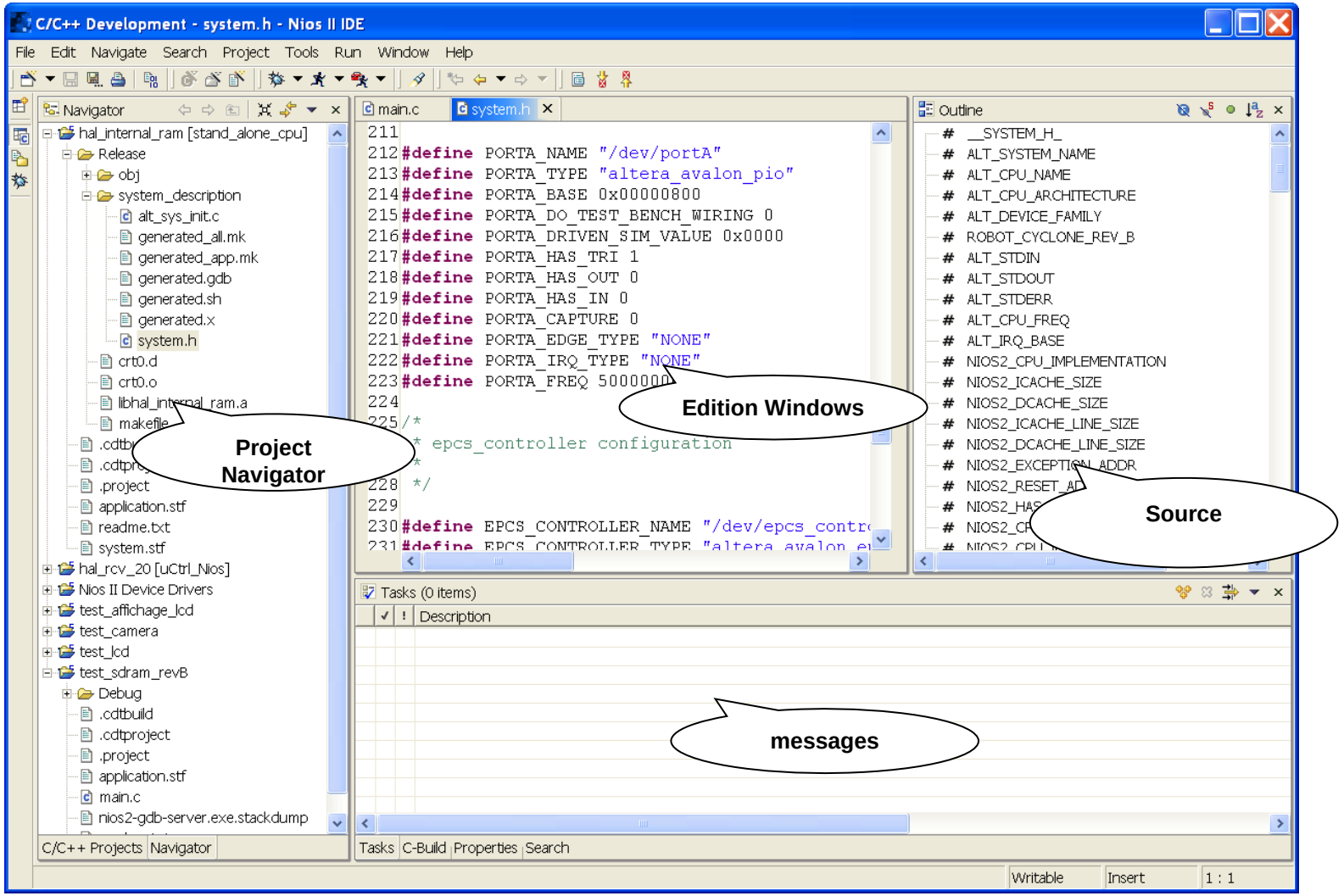
Quartus *II*



SOPC Builder (old) → Qsys → Platform Designer

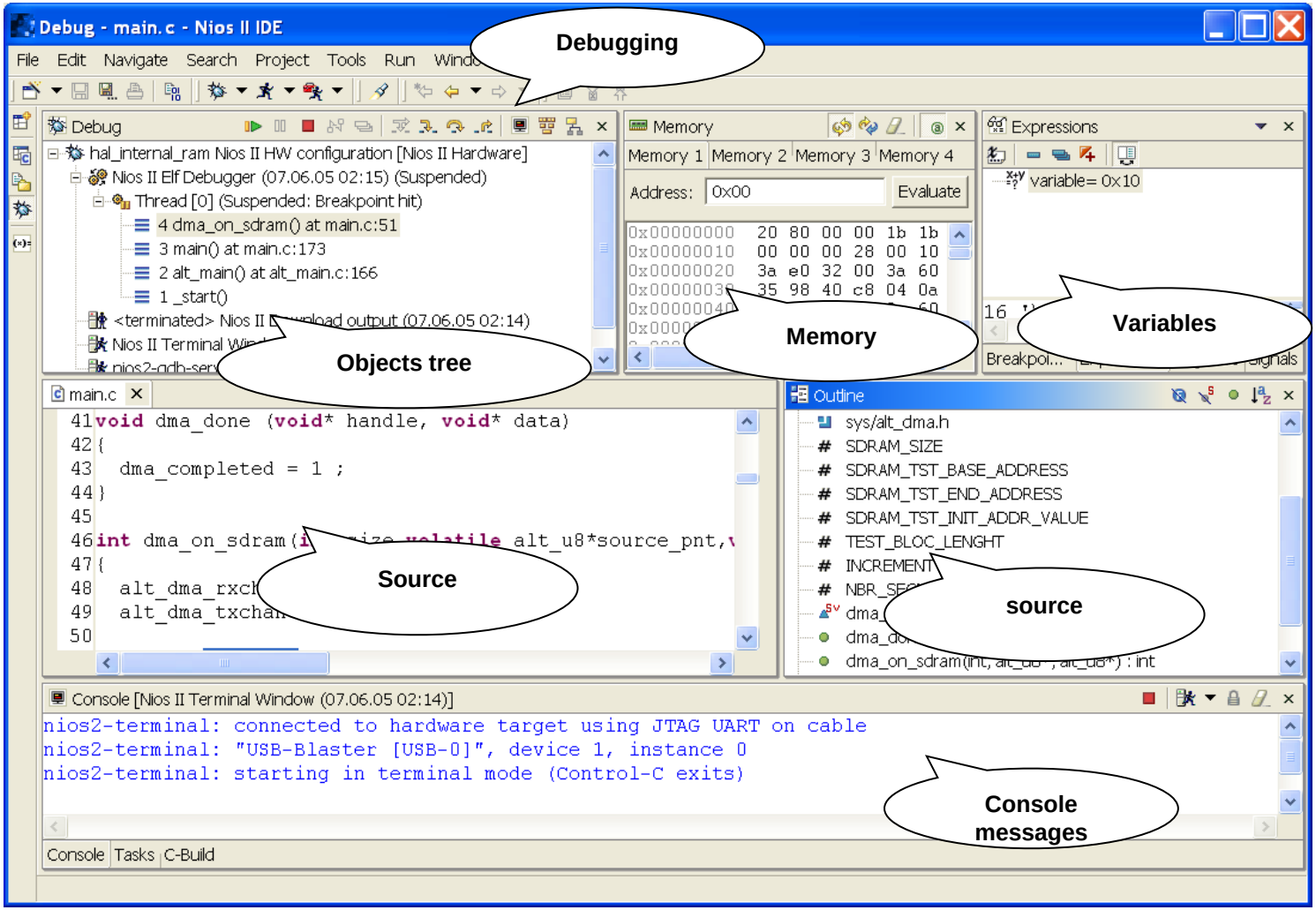


NIOS II IDE (C code development)



Developments Tools from intelFPGA

NIOS II IDE (debugger)



Conclusion

Some positives points of a softcore architecture

- **Fast implementation**
- **Modular Architecture**
- **Simplicity**
- **Good documentation**
- **Nice for teaching complex integrated embedded systems**
- **Ease of development of our own programmable interface on internal bus (i.e. Avalon in VHDL, Verilog)**
- **Full system on FPGA, easily adaptable**
- **Operating System included (uC/OS II)**

Some negate points

- **Quite big tools to develop a system**
- **Thus, new tools to learn**