ICS901/A2HW: System On Chip Design and Integration

Tarik Graba
<tarik.graba@telecom-paris.fr>
October 2026

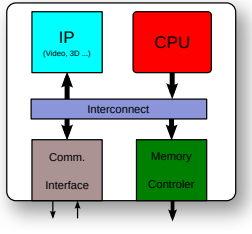
Plan

Plan

What is an SoC?

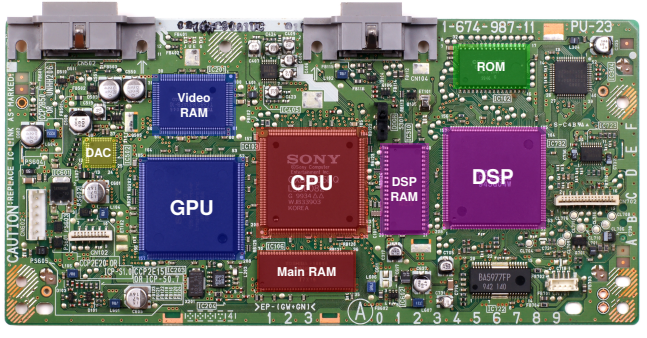
SoC

An SoC (System on Chip) is an integrated circuit that embed a processor core and hardware blocs designed for a specific application.



Why integrate?

An Embedded System (a classic game console from the 90s)

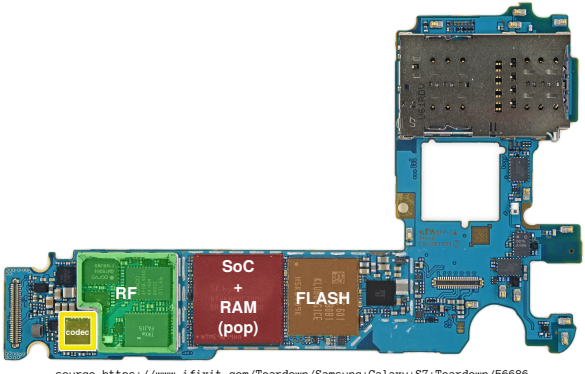


source https://en.wikipedia.org/wiki/PlayStation_technical_specifications

- General purpose Processor (32-bits MIPS CPU @33MHz)
- Discreet Dedicated Accelerators (GPU + DSP)
- Memories (16Mbits, 1Mbits, 4Mbits), ROM (4Mbits)
- Interfaces …

Why integrate?

Modern Embedded System (a Smartphone)



source https://www.ifixit.com/Teardown/Samsung+Galaxy+S7+Teardown/56686

- SoC (64bits, 2.2GHz) + GPU + DSP + Network
- Unique Shared Memory (4GBytes),
- Flash (32GBytes)
- Interfaces …

What is an SoC?

Compared to a general purpose computer?

In a general purpose “of the shelf” computer (your PC for example), there is also:

- a processor;
- memory;
- peripherals …

Also, recent “processor” are often SoCs.

They integrate:

- GPU, Interface controlers (PCI, USB…)

What is an SoC?

Compared to a general purpose computer?

The main difference comes from **specialization**.

- SoC are generally dedicated to a specific function:
  - video processing, audio processing, network, wireless…
- An SoC will **integrate dedicated modules** for those functions
- A specialized SoC will only integrate the necessary modules for its targeted usage.

Efficiency vs. General Purpose

What is an SoC?
Where do we find specialized SoC?

- Smartphones
  - graphics/image, networking, wireless, security…
- TV, set-top boxes
  - display control, image processing (and networking…)
- Cameras
  - sensor control, image/signal processing
- Smart-cards
  - security/cryptography
- …

Why integrate?

- Smaller
  - New applications (smartphones, smartwatches…)
- Performances
  - Faster
  - Less Energy
  - Thus new applications
- Less components in the Embedded System
  - Reduces the costs
  - Easier System integration

Why integrate?
Smaller

Miniaturisation thanks to the integration of new features on the same silicon die.

This reduction in size allowed the emergence of new applications.

- from smartphones to smartwatches
- IoT…

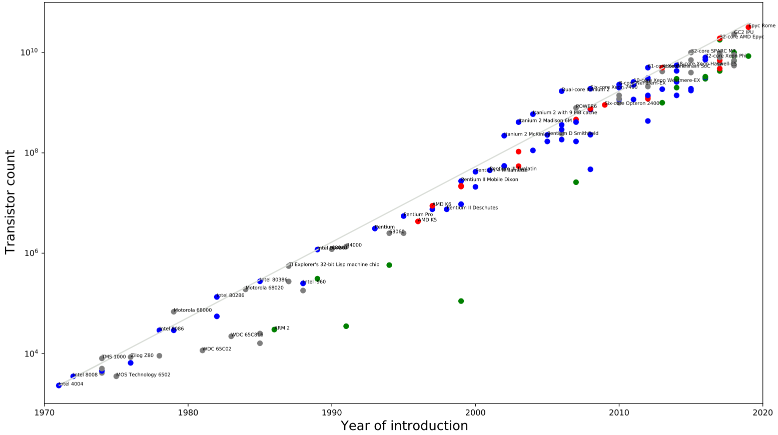
What is an SoC?

Why can we integrate more?

…thanks to the Moore Law!

⇒ Integration density doubles every 18/24 months!

Evolution of Transistor count in General Purpose Processors



Data from: `https://en.wikipedia.org/wiki/Transistor_count`

Why integrate?
Cheaper

Reduction of circuits costs

Thanks to Moore Law we can either:

- integrate more features at a constant cost;
  - Advanced Network Processing, rich media processing, Machine Learning…
- reduce the costs for existing features;
  - Low budget smartphones, Low cost IoTs

Why integrate?
Cheaper

Reduce the costs at a System Level

Less components:

- Embedded Systems are easier to build;
  - Simpler and less expensive PCB (Circuit Boards)
- Reduced constraints for PCB conception
  - High speed communication links
  - Low power

Why integrate?
Increase performances

Two directions:

- Raw performances:
  - Processing speed (IPS: Instructions per Second)
  - Operating Frequency (MHZ -> GHz)
- Power Consumption
  - Energy/Power per computation

Generally you chose one of the two directions depending on the targeted application

Why integrate?
Increase performances

The dissipated power in a digital circuit:

$$P = \alpha C f V_{dd}^2$$

With:

- f the operating frequency,
  and [α] the activity rate,

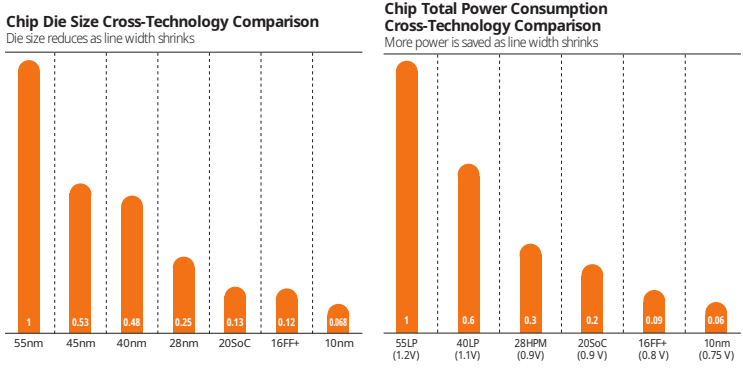
imposed by the application, and

- C the equivalent capacity
  (related to the size of/number of gates in the circuit)
- V_{dd} the power voltage,

related to the "physics" of the circuit.

Why integrate?
Increase performances

We take advantage of the advances in Circuits (IC) fabrication technologies.

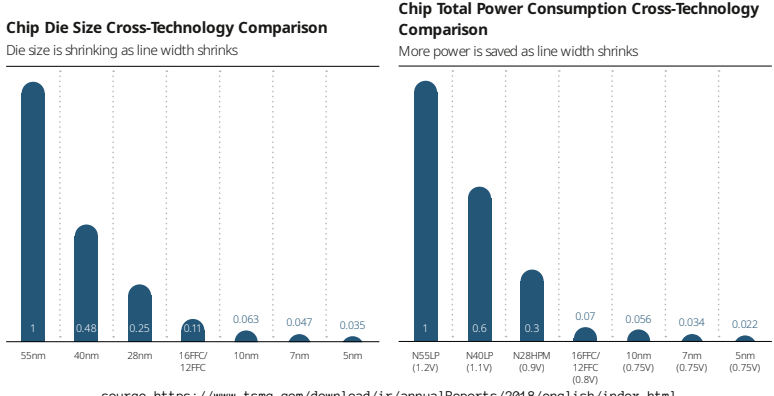


source http://www.tsmc.com/download/ir/annualReports/2015/english/index.html

Still Moore's Law!

Why integrate?
Increase performances

We take advantage of the advances in Circuits (IC) fabrication technologies.



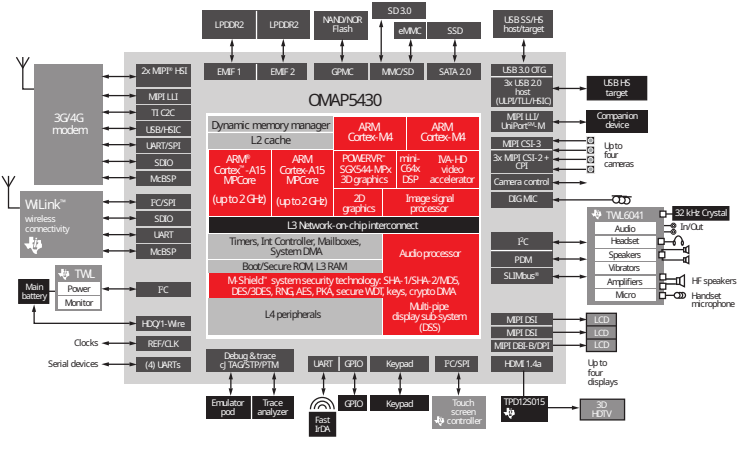
source https://www.tsmc.com/download/ir/annualReports/2018/english/index.html

Still Moore's Law!

Plan

How complex is an SoC?

Example: OMAP5430 (Texas Instruments 2011)



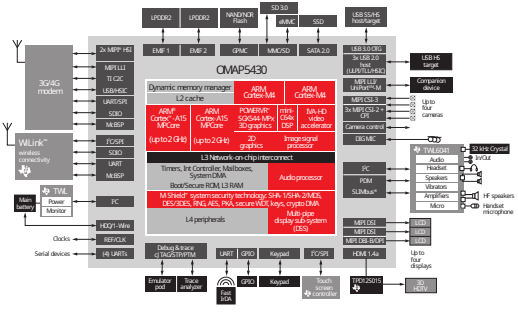
Source `http://focus.ti.com/pdfs/wtbu/OMAP5_2011-7-13.pdf`

How complex is an SoC?

Example: OMAP5430 (Texas Instruments 2011)

Application Platform

- 2 dual core CPU (Central Processing Unit) (2 GHz)
- DSP, GPU, video codec
- DDR/Memory controlers, IOs
- …



http://focus.ti.com/pdfs/wtbu/OMAP5_2011-7-13.pdf

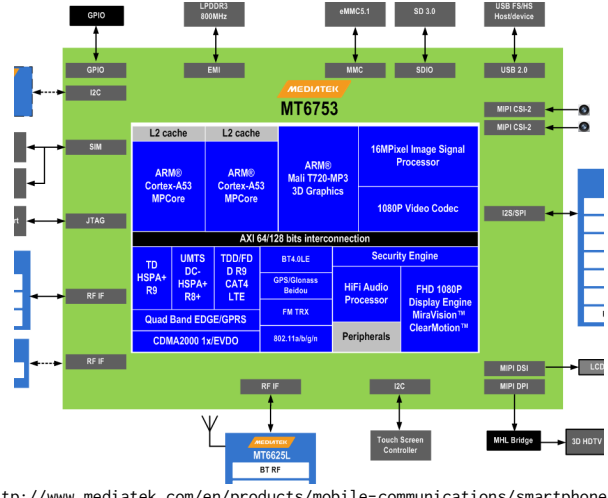
TELECOM Paris

Do not miss your market!

- Texas Instruments is an historical semiconductor company
  - processors, DSPs, sensors…
  - leader in mobile phone chipsets in late 1990 and early 2000
- The OMAP platform from Texas Instruments was used by many wireless products
  - 2G/3G phones from Nokia
  - first generations of smartphones (Nokia, Droid…)
- 4G modem not ready when the market wanted it
  - No OMAP SoC with integrated 4G modem
  - Concurrence got it in time (mainly Qualcom)
- *"UPDATE 3-Texas Instruments eyes shift away from wireless"* [REUTERS Sep. 25 2012]

How complex is an SoC

Example: MT6753 (Mediatek 2015)



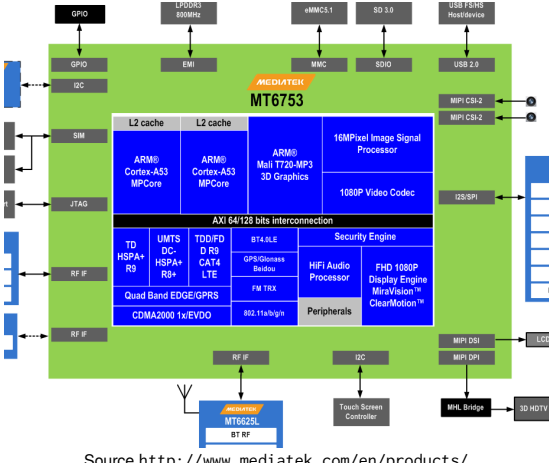
Source http://www.mediatek.com/en/products/mobile-communications/smartphone1/mt6753/

How complex is an SoC

Example: MT6753 (Mediatek 2015)

Application Platform

- **8 cores** CPU (1.3 GHz)
- DSP, GPU, video codec
- **GSM/LTE Modem**
- DDR/Memory controlers, IOs
- …



Source http://www.mediatek.com/en/products/mobile-communications/smartphone1/mt6753/

How complex is an SoC Design?

Hardware Reuse/Hardware Libraries

Hardware development is hard! It must be conducted:

reliably, guaranteeing the functionality and the performances before even fabrication

- Hardware bugs are definitive
- Exhaustive verification and validation must be carried on

rapidly, before concurrent/COTS solutions are available

- Time to market is the main economic driver

without reinventing the wheel and concentrating on the differentiating features

- Most evolutions are incremental

Is it possible, to have libraries for hardware elements to build an SoC?

How complex is an SoC Design?
Hardware Reuse

Reusable Hardware blocs are called **IPs (*Intelectual Property*)** blocs.

They can be:

- RTL sources
- Netlists
- Placed & Routed HW blocs

To reduce the development risk and the time to market, hardware blocks that have already been validated are reused.

- There is a market for *standardized* IP Design
- Some companies are specialized in the design of some IPs

How complex is an SoC Design?
Hardware Reuse

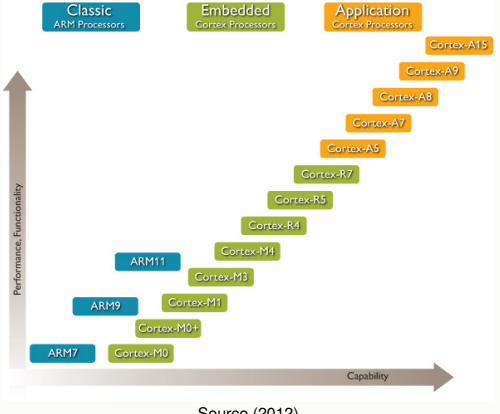
What kind of IPs do you need to build an SoC:

- at least:
  - Processors,
  - On chip communication infrastructure (Bus Fabric),
  - On chip memories and memory controllers.
- Generic Hardware Accelerators
  - GPU, DSP, …
- Specialized blocs
  - Dedicated physical interfaces (PHY)
  - Analog modules

How complex is an SoC Design?

IP vendors (examples)

ARM



Source (2012)
http://arm.com/products/processors/index.php

- CPU cores (from low footprint to HPC)
- Interconnect
  - CoreLink IP family
- Physical IPs
  - Embedded memories, standard cells
- Software support
  - OS support, compilers, libraries

How complex is an SoC Design?
IP vendors (examples)

EDA vendors sell also IPs

- Synopsys DesignWare IP
  - Physical IPs
  - PHY interfaces (DDR, USB,…)
  - Wireless, Analog (ADC)
  - Embedded CPUs…
- Candence
  - Physical IPs
  - PHY interfaces (DDR, USB,…)
  - Analog (ADC,DAC, Power Management…)

Plan

What is an SoC?
Do not forget software

Software is mandatory

- If you can develop your application using *COTS : commercial off-the-shelf* hardware then no need to develop a new SoC
- Software is easy to deal with
  - Software developers are easier to find
  - Software is easier to update and adapt and debug
  - Less risky
- Clients will ask for software
  - There are expectations (drivers, OS support, code examples)
  - Reuse and integration with software design practices

Plan

Systems On Chip
- Introduction: SoC vs Embedded Systems
- Time to market and Design Reuse
- Hardware and Software

Interrconnect and on-chip communication

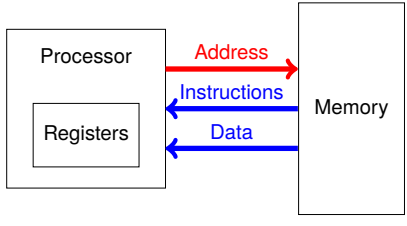
On-Chip communication protocols
- The AXI protocol
- The Avalon protocol

Programmable SoC
- Softcore Processors
- Hard Processors
- Tools and Methodology

Communication in an SoC
CPU/Memory

A processor fetch data and instructions from a memory to his internal registers.

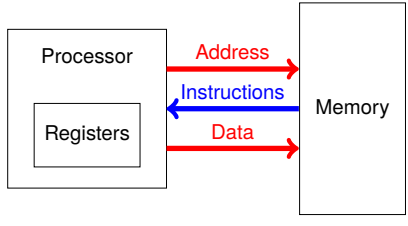
1. he presents the address of a data/instruction
2. he waits for the data/instruction to be available and stores it into his internal registers.
3. interprets the instruction/process the data internally



Communication in an SoC
CPU/Memory

A processor store data in memory from his internal registers.

1. he presents the address and the data and indicates that he want to write them
2. he waits for the data to be written



Communication in an SoC
Memory Mapped Peripherals

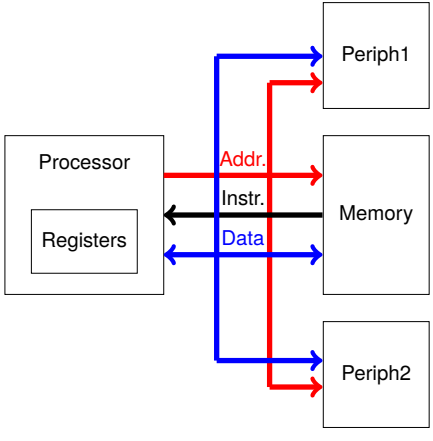
In a modern SoC, peripherals (targets) are also memory mapped.

- Each peripheral is assigned a range of addresses
- When a processor reads/writes data from/to a peripheral, he presents the corresponding address

In a modern SoC, there is more than one initiator

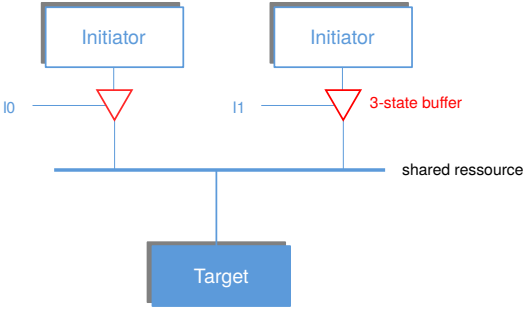
- Multi-processor systems
- Accelerators, DMAs (Direct Memory Access)

How to connect all these elements?



Multiple accesses to the same resource

Tri-state buffers



- The Initiators are connected to the shared resource through 3-states buffers
- An *Arbiter* selects which Initiator has access to the bus
- The access can be bi-directional

Multiple accesses to the same resource

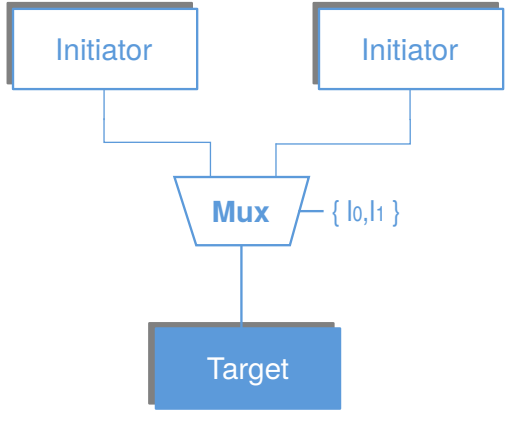
Tri-state buffers

Not ideal for on-chip interconnect

- Performances are hard to guarantee
  - Capacitive Load
  - Depends on routing and the number of connected elements
- We do not benefit from the Transistor density evolution (Moore Law)
  - Wire density does not increase as fast as transistors density
  - See *Routing Congestion in VLSI Circuits.*

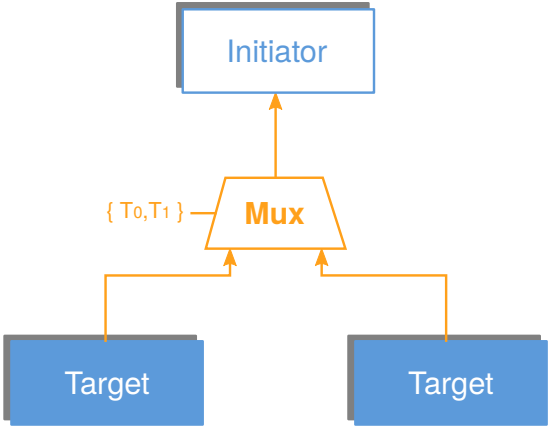
Multiple accesses to the same resource
Multiplexed Access

- Active CMOS multiplexers are used to select between Initiators
- An *Arbiter* selects which Initiator has to the resource
- Mono-directional access



Multiple accesses to the same resource

Multiplexed Access

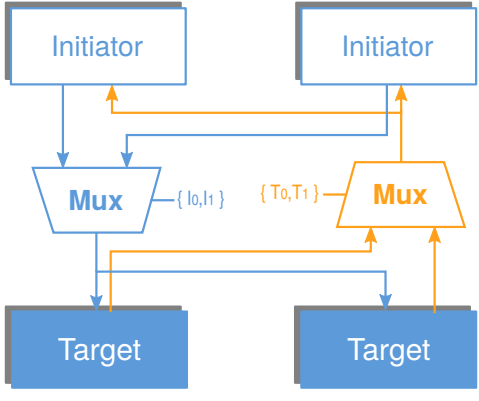


- The return PATH (from targets to initiators) is also multiplexed.
- An address decoder selects the target.
- Signals are still mono-directional

Hardware complexity is related to the number of slaves and the size of the address bus used for the decoding.

Multiple accesses to the same resource

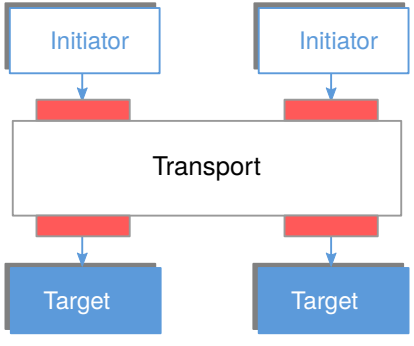
Circuit Switched Network



When arbitration and decoding is done, one initiator has direct access to one of the targets.

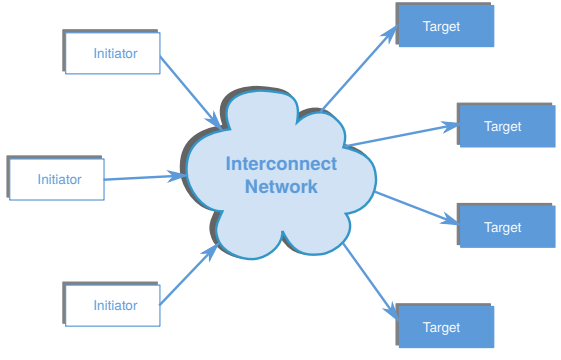
Multiple accesses to the same resource

Bus Fabric



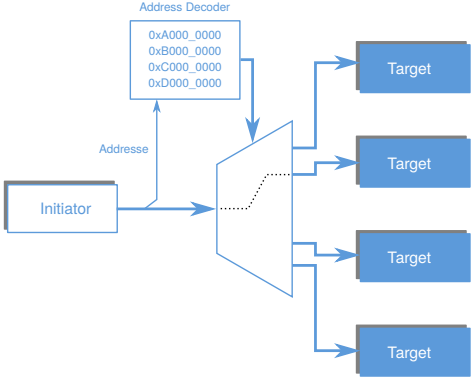
The initiators send requests through the interconnect and the targets respond. The routing mechanism is hidden.

Multiple accesses to the same resource
Bus Fabric

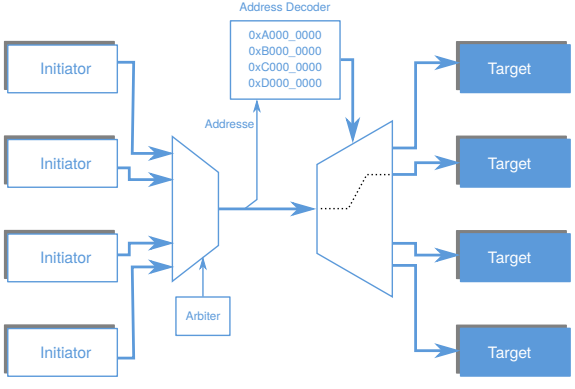


It can even seen as a Network!

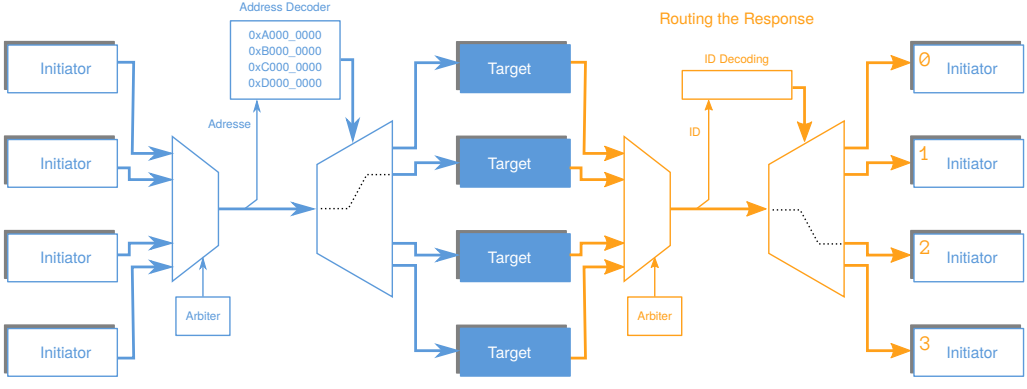
Multiple accesses to the same resource
Example



Multiple accesses to the same resource
Example



Multiple accesses to the same resource
Example



Plan

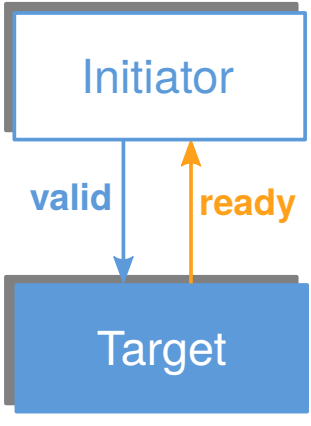
On-Chip communication protocols
Standardized protocols

- Protocols have been defined to guarantee interoperability
  - Set of functionalities to allow the communication between initiators and targets
  - Usable for different levels of performances
  - High bandwidth or/and low latency
- Point-to-point protocols
  - Define the communication sequence between an initiator and a target
  - At the interconnect interface
  - Independent of routing strategies

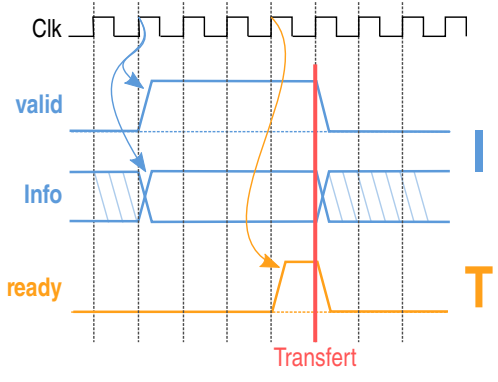
On-Chip communication protocols
A handshake

Common On-Chip protocols use synchronous handshake

- The initiator start a transaction by setting a `valid` signal,
- The target indicates when it can accept the transaction by setting a `ready` signal,
- The transfer occurs when we have both `valid` and `ready` at a clock edge.

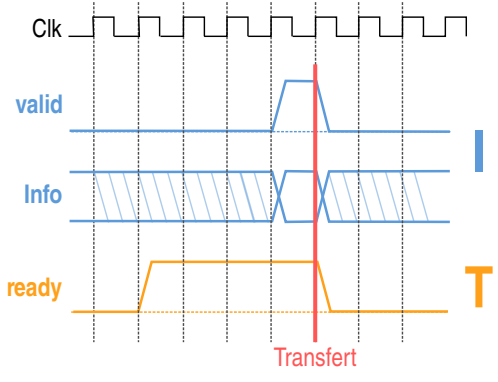


On-Chip communication protocols
A handshake



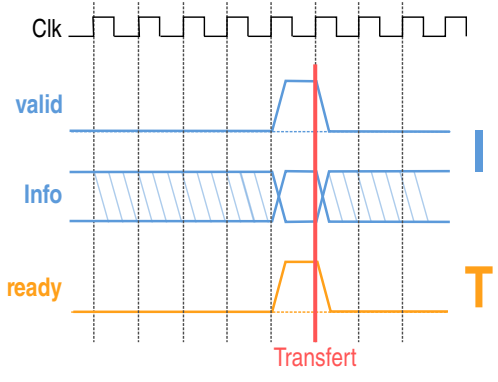
- Valid before Ready

On-Chip communication protocols
A handshake



- Valid after Ready

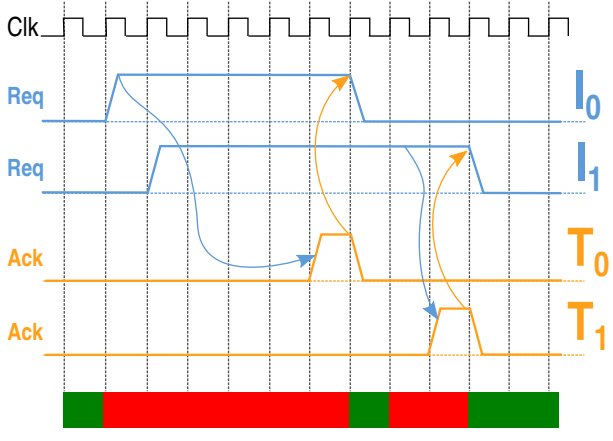
On-Chip communication protocols
A handshake



- Valid and Ready at the same time

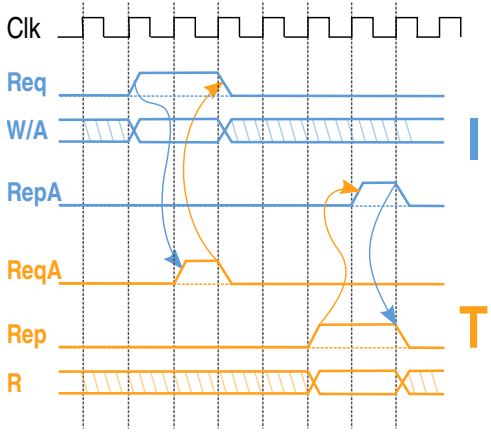
On-Chip communication protocols
Blocking transactions

- The bus is monopolized by one initiator as long as the transfer is not completed



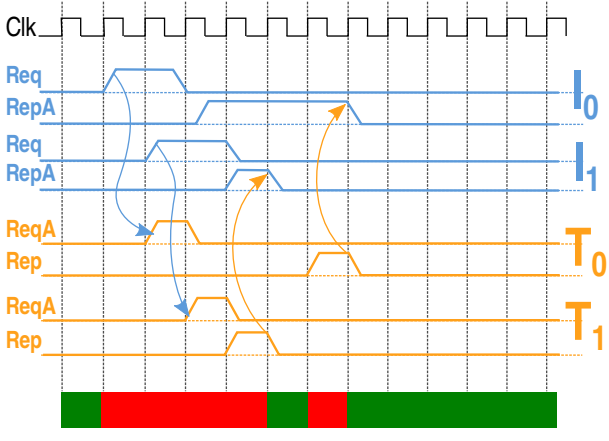
On-Chip communication protocols

Separate Requests and Responses



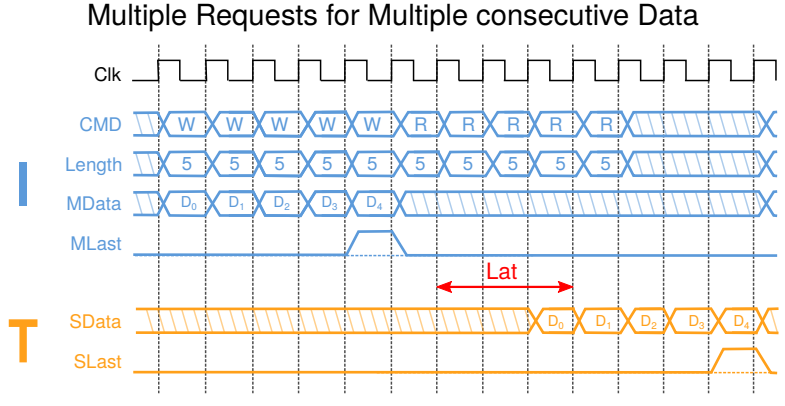
On-Chip communication protocols
Separate Requests and Responses

- The bus availability is better



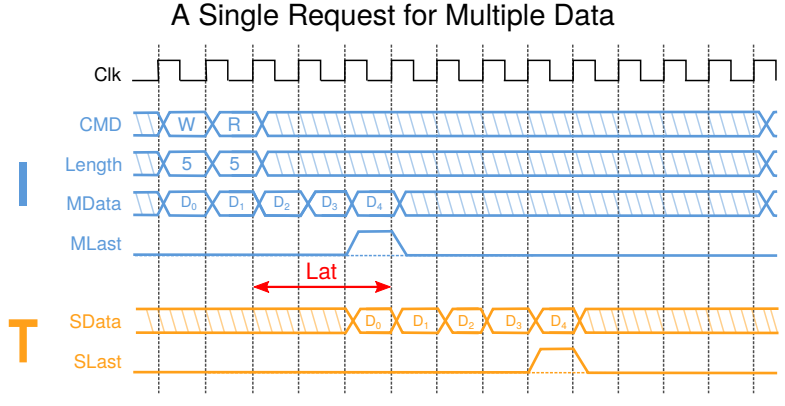
On-Chip communication protocols

SRMD *vs.* MRMD for Burst Accesses



On-Chip communication protocols

SRMD *vs.* MRMD for Burst Accesses



On-Chip communication protocols

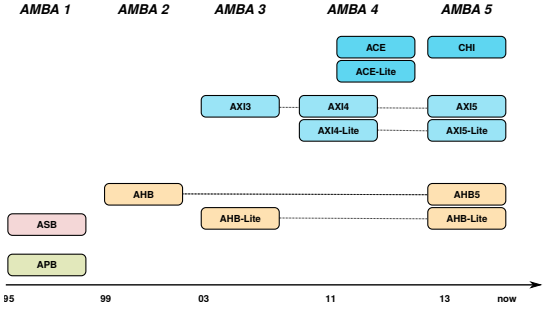
Some On-Chip Protocol

	sync. handshake	Req/Resp	SRMD
ARM AXI	+	+	+
ARM AXI-Lite	+	+	o
OpenCore OCP	+	+	+
ARM AHB/APB	+	o	+
Whishbone	+	o	o
Altera Avalon	+	pipeline	burst ext.

Plan

On-Chip communication protocols

AXI: Advanced eXtensible Interface



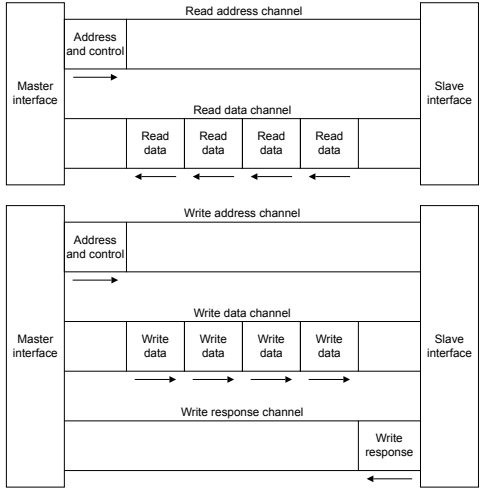
- AMBA: Advanced Microcontroller Bus Architecture
- ARM proprietary, now industry standard
- Evolved to adapt to the complexity of modern SoC

On-Chip communication protocols

AXI: Advanced eXtensible Interface

- Separate read and write channels
- Separate requests/responses
- Designed for Multi-Core SoC with high bandwidth requirements
  - Burst modes
  - Mutli-master IDs
  - Special access modes

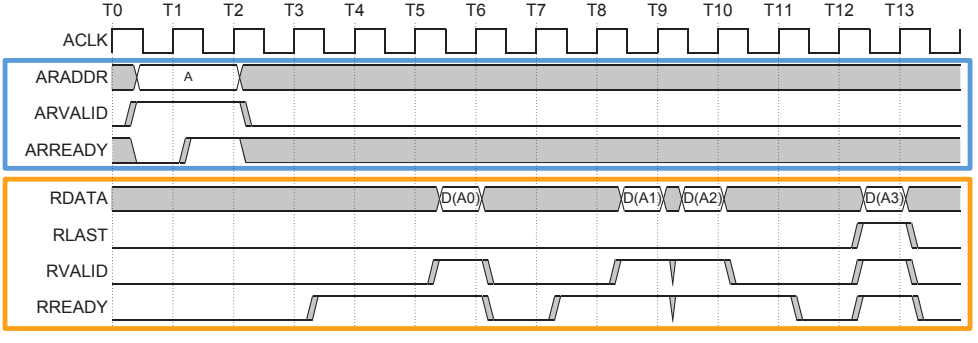
The specifications are publicly available [link].



From *AMBA AXI and ACE Protocol Specification*

On-Chip communication protocols

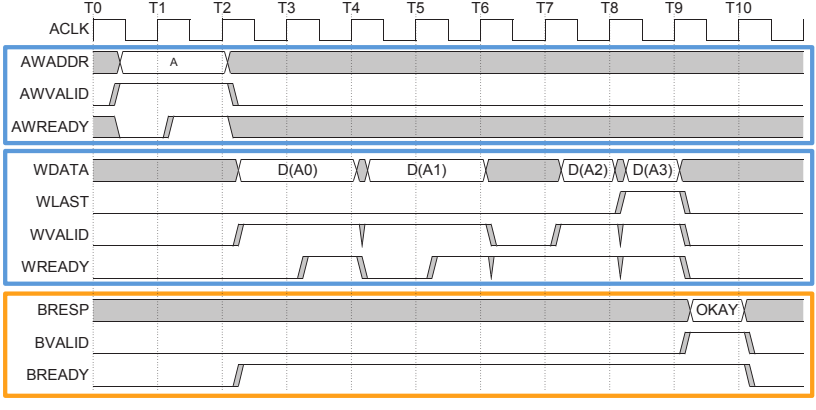
AXI: Advanced eXtensible Interface



From *AMBA AXI Protocol v2.0 (2010)*

On-Chip communication protocols

AXI: Advanced eXtensible Interface



From *AMBA AXI Protocol v2.0 (2010)*

Plan

On-Chip communication protocols
The Avalon Protocol

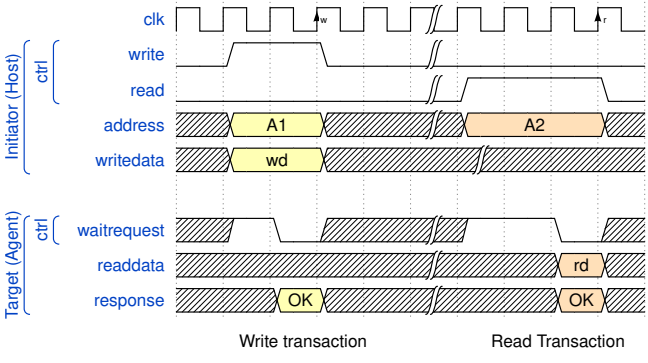
- Developed by Altera (now Intel) for FPGA SoC
- Simple Memory Mapped applications
- Extended to support Pipeline and Burst transfers
- Used for Altera's IPs and Nios-II processor

The specifications are publicly available [link].

On-Chip communication protocols

The Avalon Protocol

Simple transactions:

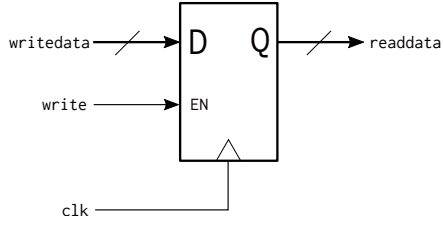


- An initiator (Host) starts a read or write transaction by setting the corresponding signal
- A target (Agent) can set the waitrequest signal if it can not respond immediately (the initiator has to wait)
- Transfers occur at the active edge clock when read or write is active and waitrequest is not asserted

Control signals are not all required.
For example, for a simple register:

- `waitrequest` is not needed (we can always accept or return data at the clock edge)
- `read` signal is not needed
- ...

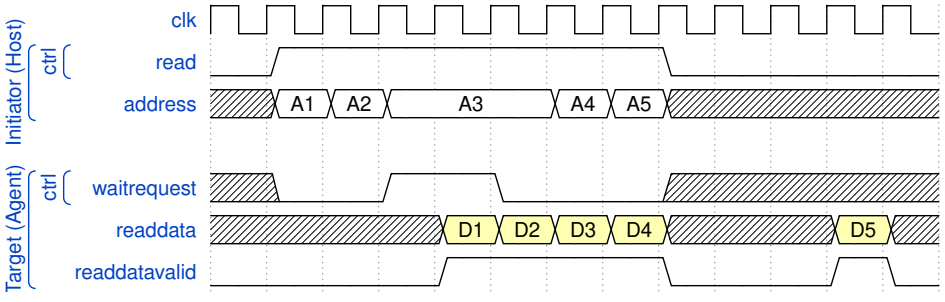
This is a compliant Avalon Agent (Target)



On-Chip communication protocols

The Avalon Protocol

Pipelined Read transactions:

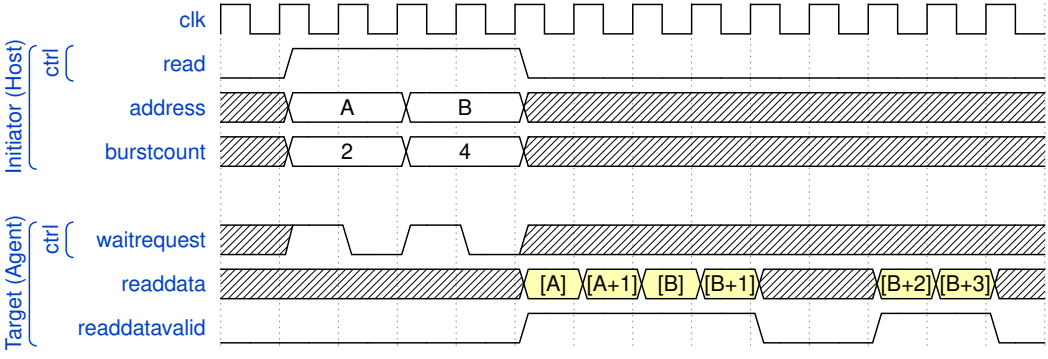


- A target (Agent) can respond to a read request several cycles after accepting the request
- ‘readdatavalid‘ signal is used to indicate that the data is available
- the initiator (Host) must know that this mode is used

On-Chip communication protocols

The Avalon Protocol

Burst Read transactions:



- Single Request for consecutive addresses

Plan

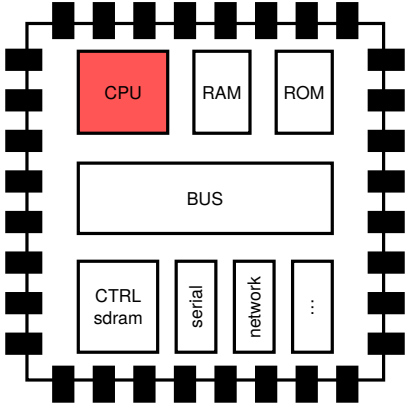
Systems On Chip
- Introduction: SoC vs Embedded Systems
- Time to market and Design Reuse
- Hardware and Software

Interrconnect and on-chip communication

On-Chip communication protocols
- The AXI protocol
- The Avalon protocol

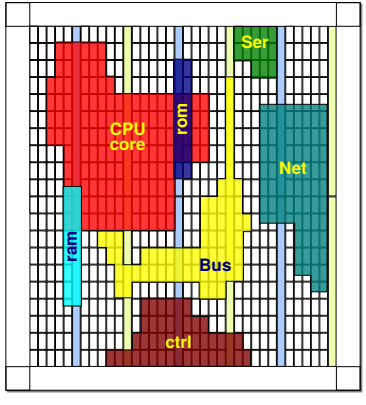
Programmable SoC
- Softcore Processors
- Hard Processors
- Tools and Methodology

Programmable SoC
SoC in an FPGA



Programmable SoC

SoC in an FPGA

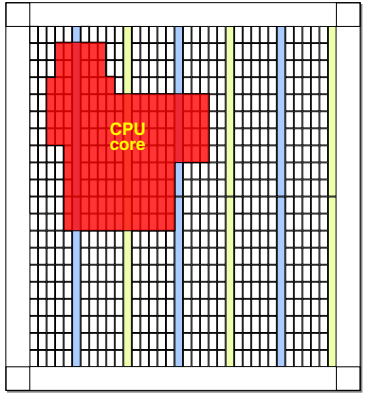


Programmable SoC
SoC in an FPGA

- Use the SoC methodology for an FPGA target.
  - FPGA is a good target for digital systems
  - Modern FPGAs are large enough
  - Modern FPGAs are fast enough
- For prototyping
  - FPGA are programmable, which is ideal for testing
  - Short development cycles
- For small volume production
  - The cost of ASIC manufacturing is very high
- Upgradable
  - Bug correction, feature enhancement

Plan

Programmable SoC
Softcore Processors



Use the FPGA logic resources to implement Processors and Peripherals

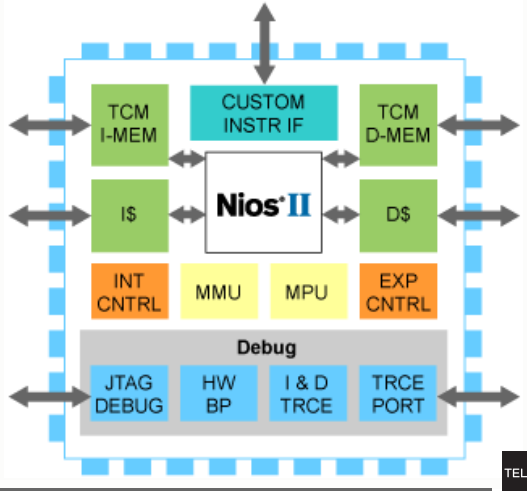
Using FPGA resources to implement a Processor

+ Can target several families of standard FPGAs
+ The Processor Core can be configured/modified for the application
− Often we need a *specific* Core (designed specifically for the FPGA target)
− The performances are limited compared to an ASIC design
− It consumes the FPGA resources instead of using them exclusively for the application

Softcore Processors
Examples of platforms

Intel/Altera Nios II

- Exclusively for Intel/Altera FPGAs
- 32-bit Processor
- Configurable (pipline depth, custom instructions, cache size, …)
  - Has OS support (Linux/RTOS)
- Proprietary (RTL sources can not be freely distributed or used for other purposes)
- Licence Fees (not free)



Softcore Processors
Examples of platforms

Xilinx Microblaze

- Exclusively for Xilinx FPGAs
- 32-bit Processor
- Configurable (pipline depth, custom instructions, cache size, …)
  - Has OS support (Linux/RTOS)
- Proprietary (RTL sources can not be freely distributed or used for other purposes)
- Licence Fees (not free)

Softcore Processors
Examples of platforms

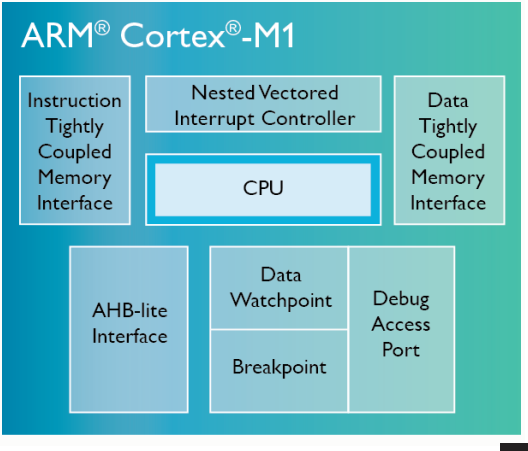
Lattice Mico32

- Designed for Lattice
- 32-bit Processor
- RTL sources are free
- No Linux support

Softcore Processors
Examples of platforms

ARM Cortex-M1

- 32-bit Processor
  - The Cortex-Mx Microcontroller Family is very well known to software developers
- Configurable
- Implementations for several FPGA families from several vendors
- RTOS support (no Linux)
  - Rich software support and tooling
- Non free



Softcore Processors
Examples of platforms

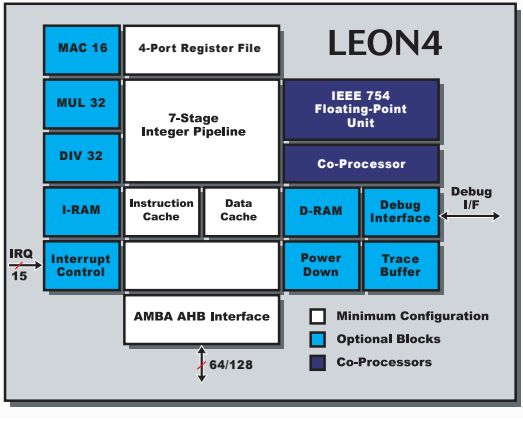
RISC-V

- Open source ISA (RISV-V Foundation)
  - `https://riscv.org`
- Several implementations
  - 32/64/128 bit
  - 32-bit instructions or a 16-bit compressed instruction set
  - Several profiles ranging from μ-controllers to HPC
- Actively developed software environment
  - Commercial and community support
  - Several free and commercial implementations
  - `https://riscv.org/risc-v-cores/`
- A trend
  - Intel/Altera Nios-V
  - Xilinx/AMD Microblaze-V

Softcore Processors
Examples of platforms

Cobham/Gaisler Leon

- SPARC 32bits initially developed for the European Space Agency
  - maybe not the bes choice for an embedded application
- Linux support
- Open source (version 3)
- Hardened version for space applications
- ASIC versions are available



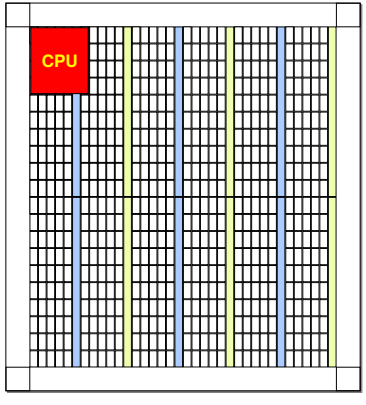
Softcore Processors
Examples of platforms

OpenCore/OpenRisc …

- Open source Processors
- HW/SW not well maintained
- Community support
  - Little interest compared to the RISC-V

Plan

Hard Processors



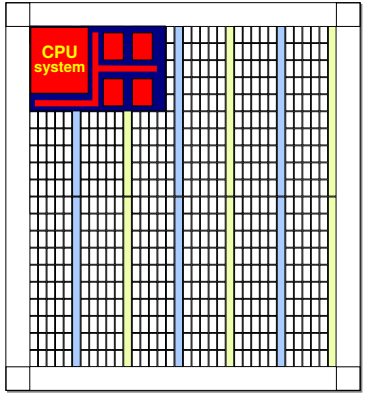
Embed a Hard Processor on the FPGA die.

Hard Processors

hard core:

- − Only some specific FPGA families include a hard processor
- − The Processor can not be modified or adapted
- + High performances (equiv. to an ASIC)
- + Standard Processors (PPC, ARM…)
- + The FPGA resources are entirely usable for the application
  - − Still some resource are used for the interconnect and base peripheral

Hard Processors
Hard System



Embed a complete Processor System in the FPGA Die

A complete Hard Processor System in an FPGA

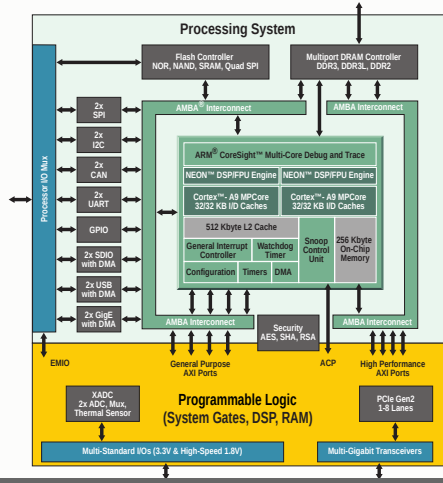
- + Software support is easier
  - + The system is functional without configuring the FPGA
  - + OS and Libraries are easier to support
- + The FPGA resources are uses to implement HW accelerators
- − Steeper learning curve
  - + Design methodology

Xilinx Zynq SoC

Complete Hard System

- Dual Core ARM Cortex-A9 (32-bit)
- DDR memory controller, MMC, Flash…
- Serial interfaces, Network interfaces…

The FPGA matrix is connected directly the System AXI bus fabric



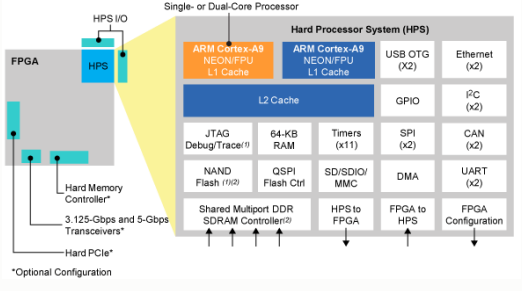
Hard Processors
Hard System examples

Intel/Altera CycloneV SoC

Complete Hard System

- Dual Core ARM Cortex-A9 (32-bit)
- DDR memory controller, MMC, Flash…
- Serial interfaces, Network interfaces…

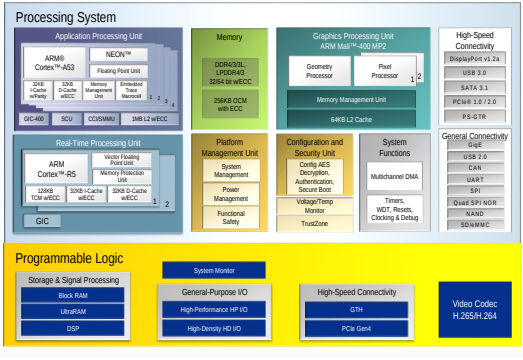
The FPGA matrix is connected directly the System AXI bus fabric



Hard Processors
Hard System examples

Xilinx Zynq Ultrascale+ MPSoC

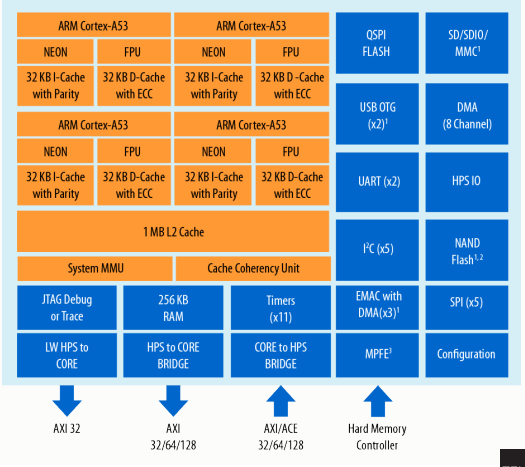
- Dual Core ARM Cortex-A53 (64-bit) plus a Dual Core ARM Cortex-R5 (32-bit) (real-time)
- Mali-400 GPU, Video Processing Unit
- DDR memory controller, MMC, Flash…
- Serial interfaces, Network interfaces…
- PCIe, SATA, DisplayPort, …



Hard Processors
Hard System examples

Intel/Altera Stratix10-SoC

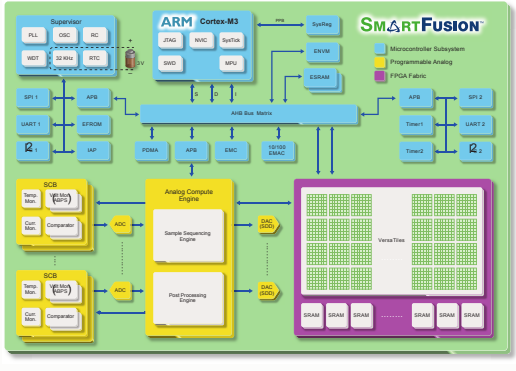
- Dual Core ARM Cortex-A53 (64-bit)
- DDR memory controller, MMC, Flash…
- Serial interfaces, Network interfaces…



Hard Processors
Hard System examples

Microsemi Smatfusion/2

- 32-bit ARM Cortex-M3 (Microcontroller Grade)
- Serial interfaces, Network interfaces…
- ADC, DAC
- Programmable Analog Array



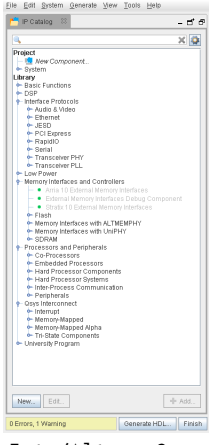
Plan

Tools and Methodology
IP oriented Design

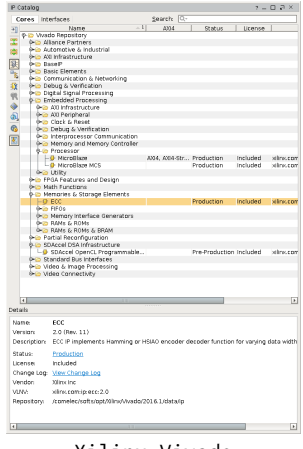
- Packaged HW IPs
  - Standard On-Chip protocols
  - Metadata (IP-xact, ad-hoc)
- IP Libraries
  - Basic building blocs
    - Interconnect, protocol translators…
  - Common peripherals
    - GPIOs, communication interfaces…

Tools and Methodology

IP oriented Design



Inte/Altera Qsys



Xilinx Vivado

TELECOM Paris

Tools and Methodology
IP oriented Design

Users can:

- Assemble existing IPs
  - IP configuration
  - System configuration
    - Interconnect architecture
    - Address map
- Develop its own ones
  - Respecting the platform On-Chip protocol
  - Package the IP for the tool
    - Templates are generally provided

Tools and Methodology
Software support

The tools often include means to:

- Generate software drives
  - Hardware Abstraction Layer (HAL)
  - OS integration
  - BSP: Board Support Package
- Software Libraries

The vendor provide a Software Development Kit with the ability to generate software projects from the Hardware structure.