Thinking and Designing Differently: The Asynchronous Alternative

Laurent Fesquet
Grenoble Institute of Technology

TIMA laboratory

Outline

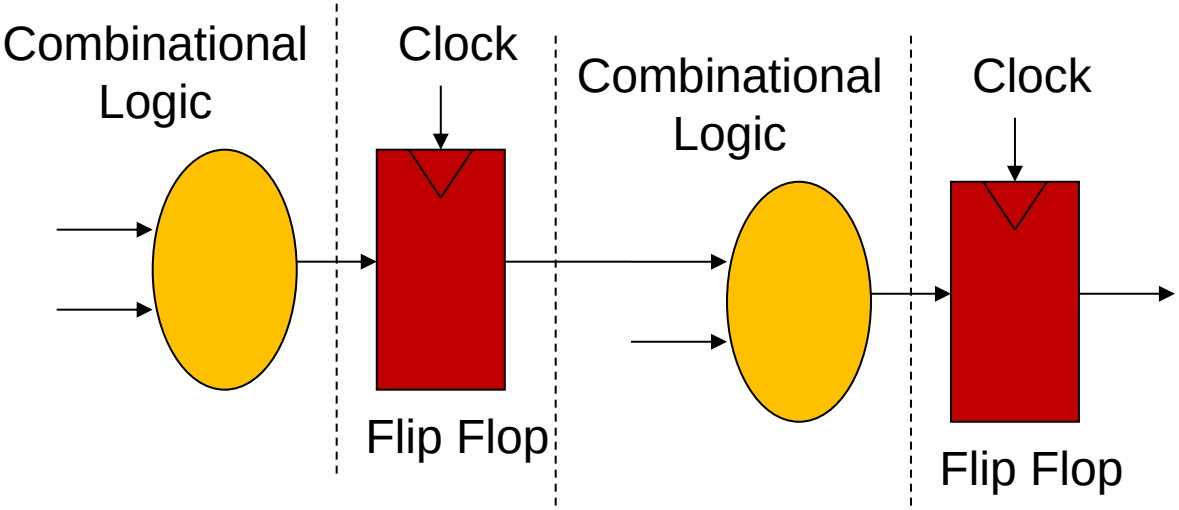
- Designing synchronous circuits
- Asynchronous circuit principles
- Asynchronous circuit design principles
- Asynchronous circuit classes
- Exploiting the asynchronous logic
- Success stories
- conclusion

Outline

- **Designing synchronous circuits**
- **Asynchronous circuit principles**
- **Asynchronous circuit design principles**
- **Asynchronous circuit classes**
- **Exploiting the asynchronous logic**
- **Success stories**
- **conclusion**

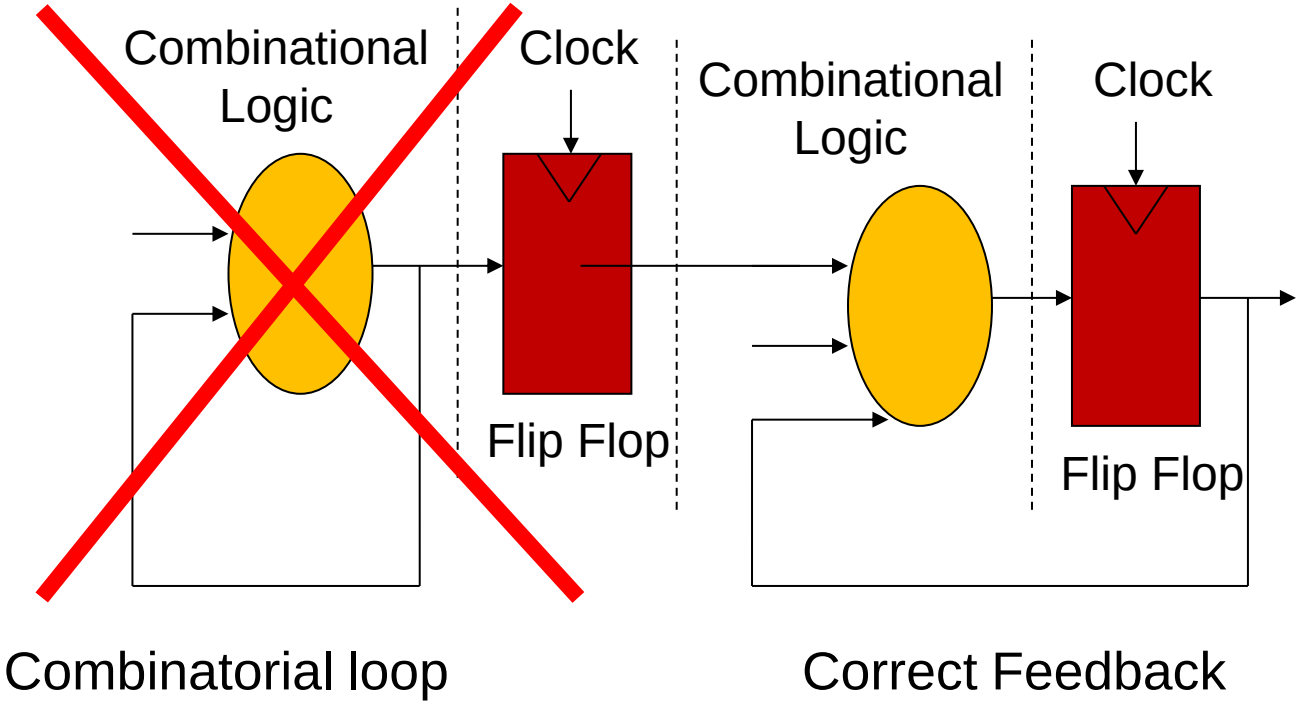
Designing synchronous circuits

- Synchronous circuit model



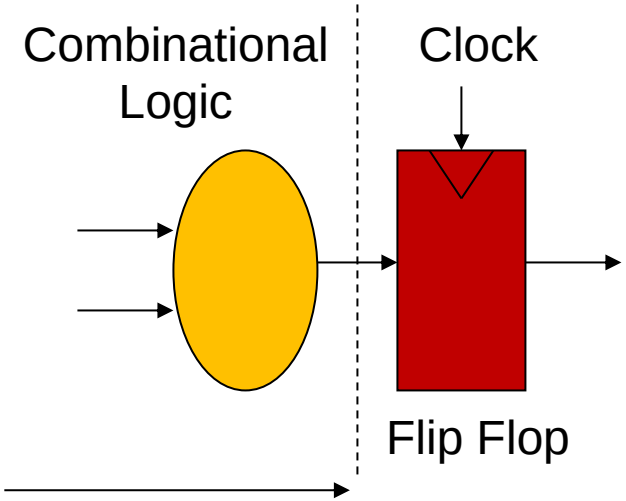
Designing synchronous circuits

- Synchronous circuit model



Designing synchronous circuits

- Synchronous circuits use timing assumptions

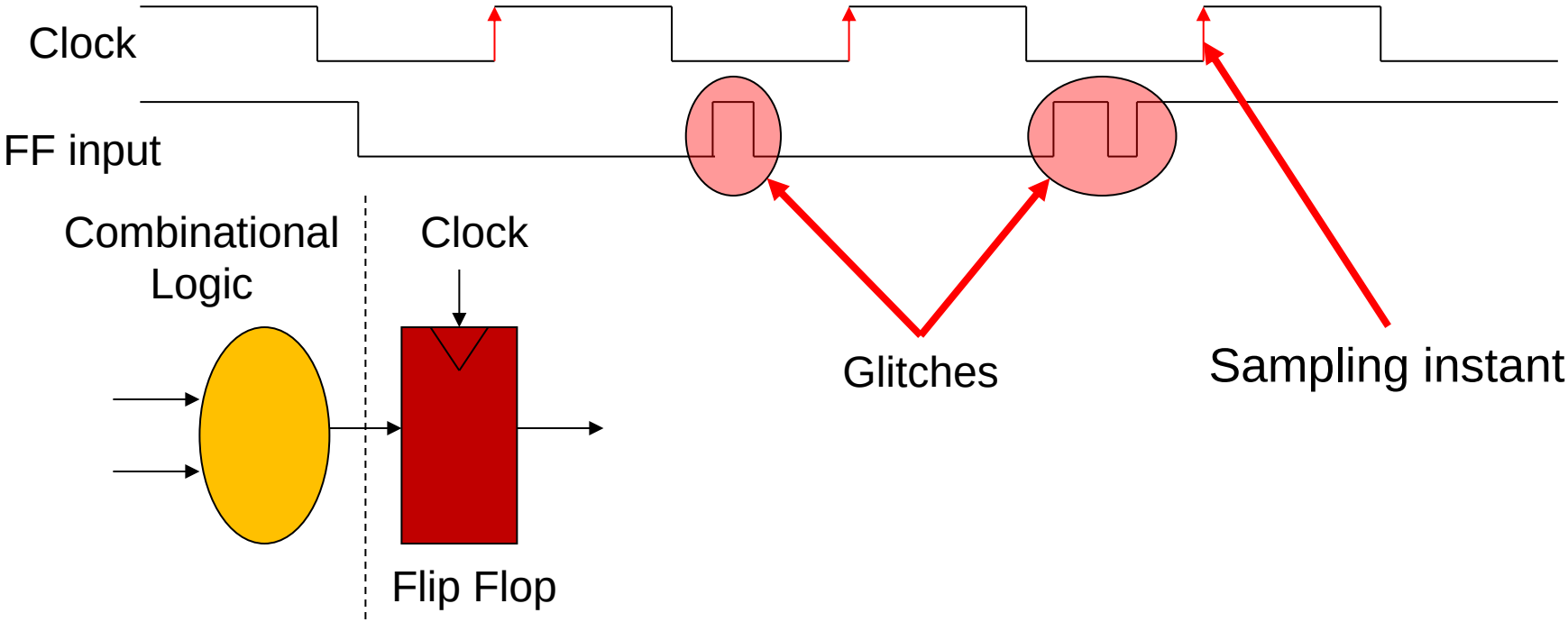


Critical path = Longuest path (worst case)

- **Correct behavior when Tcritical < Tclock**

Designing synchronous circuits

- Hazard is allowed (before sampling)
  - Combinatorial logic is stable before sampling



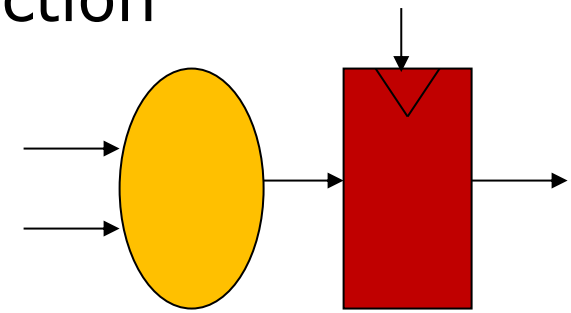
Outline

- Designing synchronous circuits
- **Asynchronous circuit principles**
- Asynchronous circuit design principles
- Asynchronous circuit classes
- Exploiting the asynchronous logic
- Success stories
- conclusion

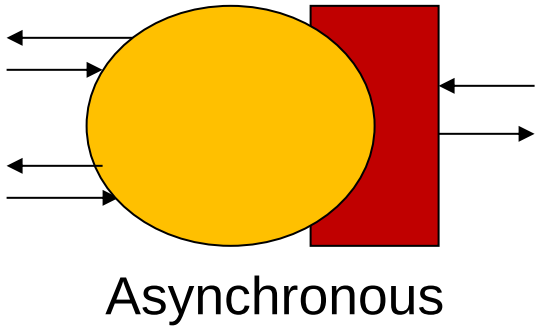
Asynchronous Circuits Principles

- At the hardware module abstraction

Every rising edge clock
triggers the computation



Data availability
triggers the computation

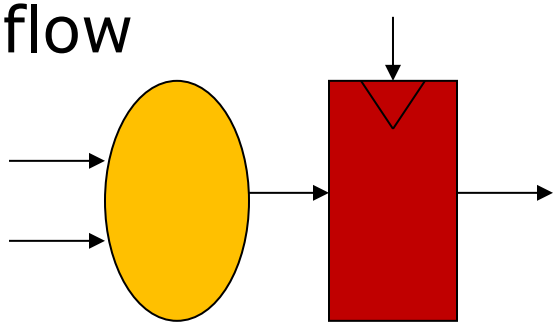


→ **Global Clock** is *replaced*
→ by **local channels** (handshaking)

Asynchronous Circuits Principles

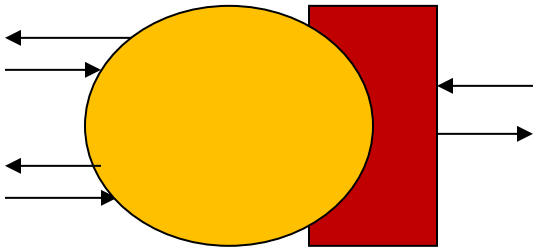
- Data-flow instead of control-flow

if rising_edge of clock **then**
    send output = f(inputs)
else
    output remains unchanged
end if



Synchronous

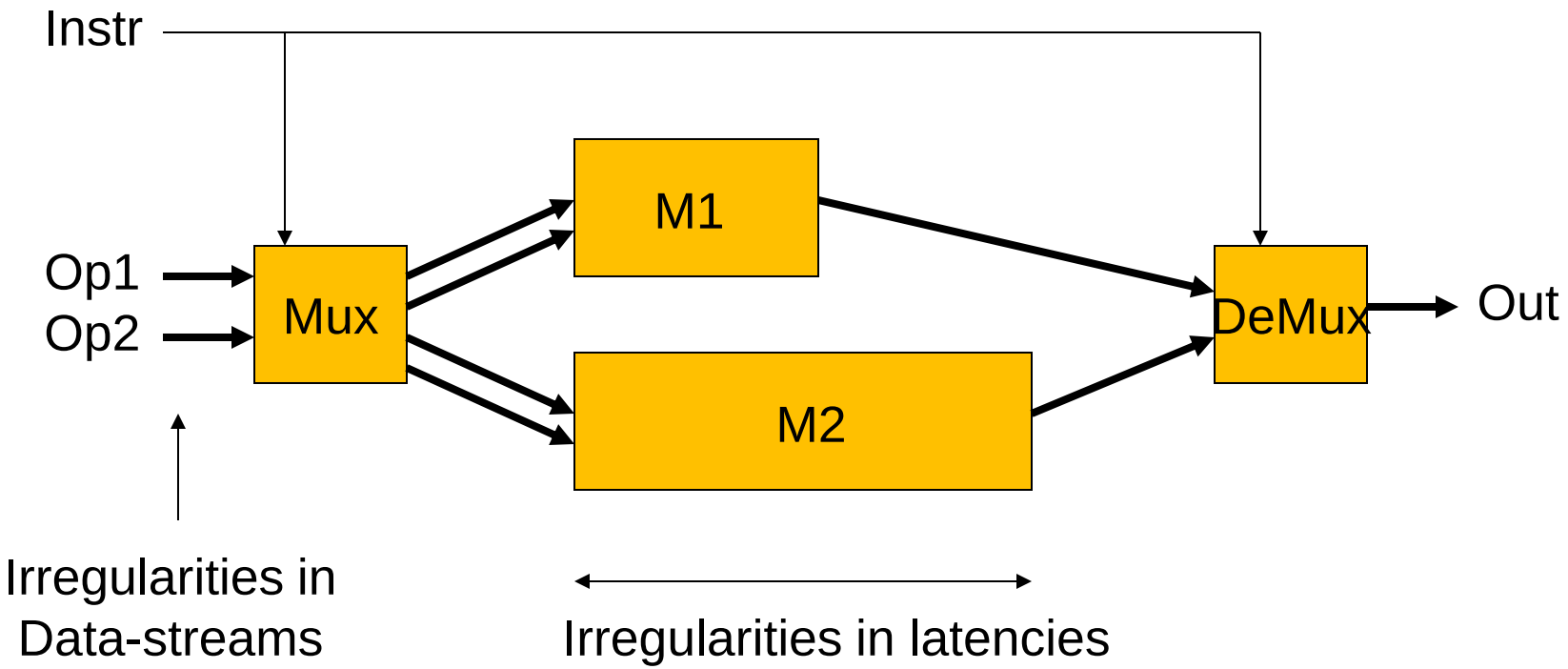
wait for valid inputs
    output = f(inputs)
complete input transactions
wait for output ready to receive
    send output
complete output transaction



Asynchronous

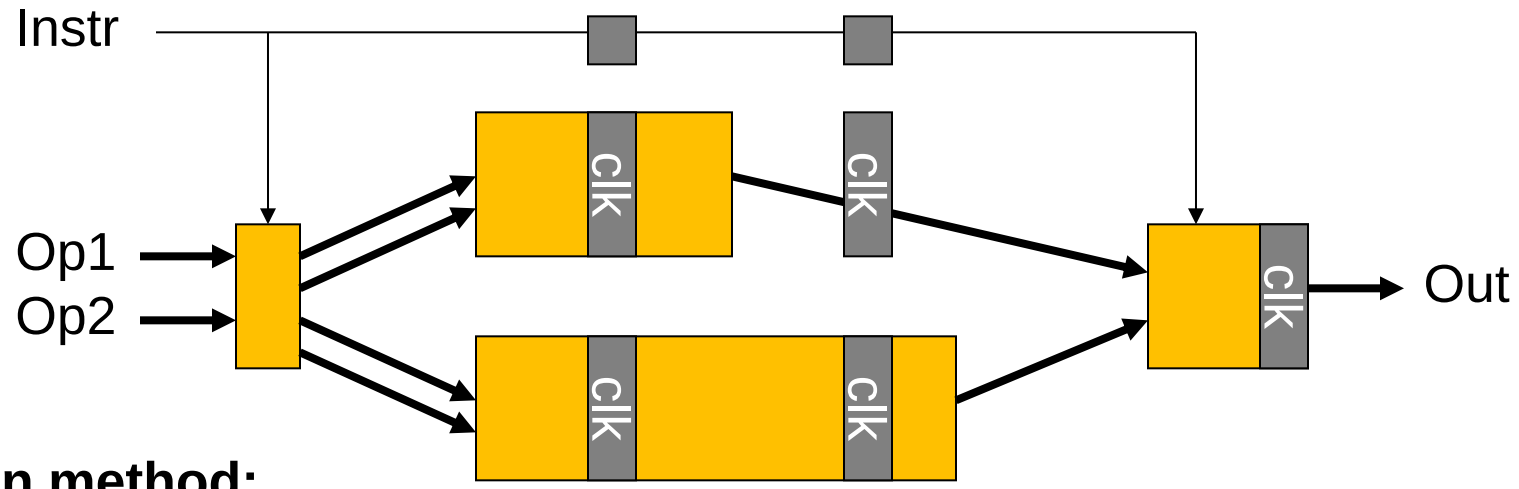
Asynchronous Circuits Principles

- Composing hardware modules



Asynchronous Circuits Principles

- Synchronous circuits : balance the pipelines
  (worst case approach)



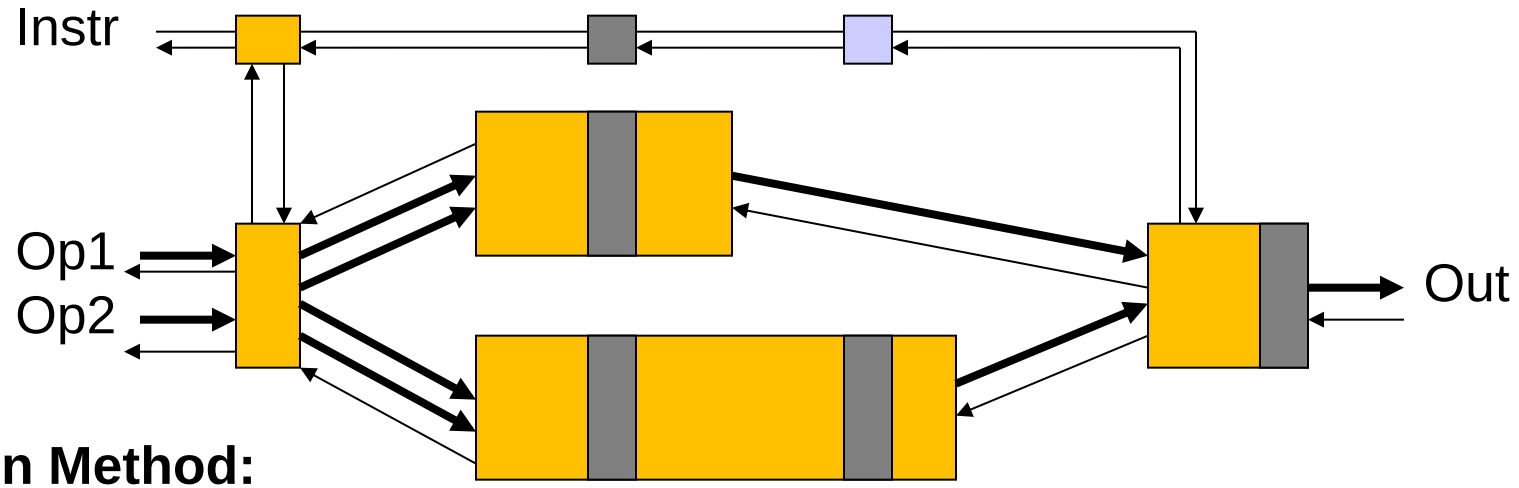
Design method:

Need to know the state of the whole architecture at each cycle

→ What happen if the system is very complex ?
→ Difficult to exploit input data stream irregularities
→ Circuit: add latency, increase power consumption

Asynchronous Circuits Principles

- Asynchronous circuits : ensure data flows



Design Method:
No need to know the state of the architecture
pipelining preserves the functional correctness

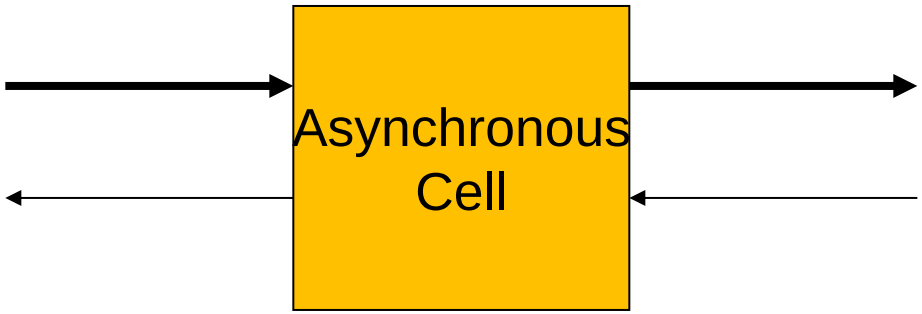
→Circuit: latency is always minimum, as well as power consumption
→Easy to compose a complex system using simple modules
→Free to exploit input data stream irregularities

Outline

- Designing synchronous circuits
- Asynchronous circuit principles
- **Asynchronous circuit design principles**
- Asynchronous circuit classes
- Exploiting the asynchronous logic
- Success stories
- conclusion

Asynchronous circuit design principles

A basic asynchronous cell



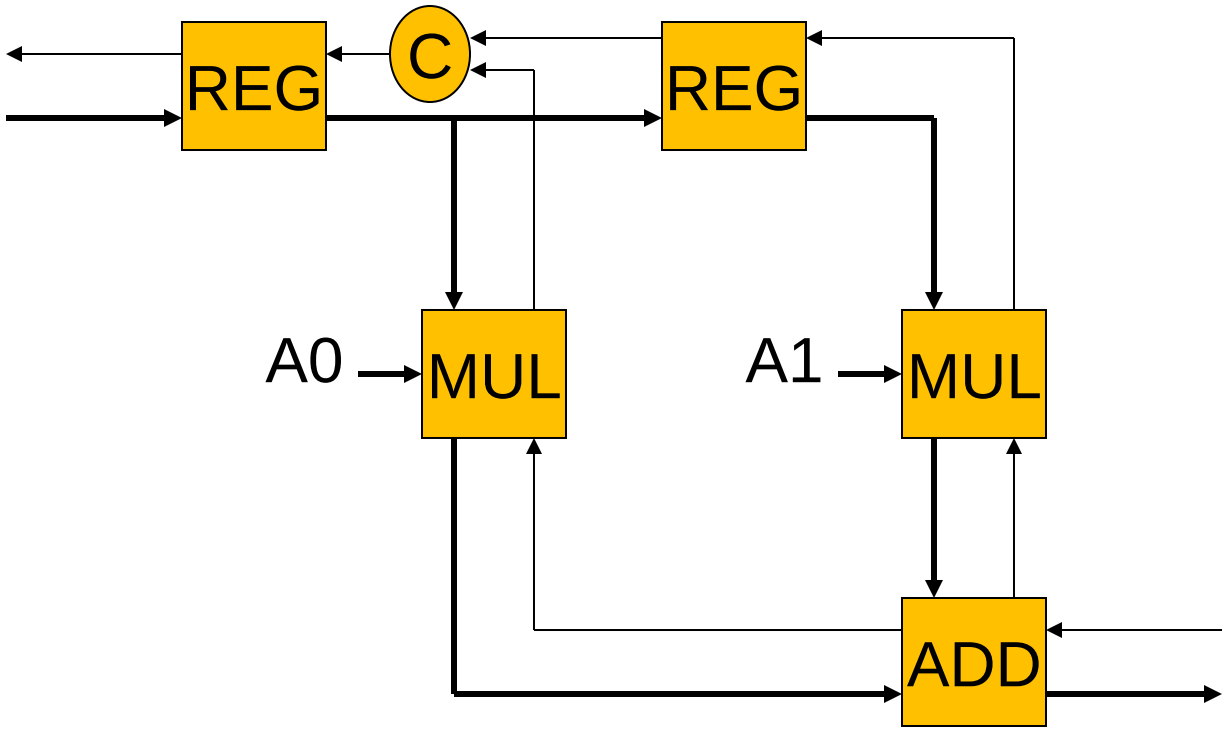
- Bit level
- Arithmetic function
- Complex function

- Maximal speed: minimal forward latency
- Maximal throughput: minimal cycle time
- Respect the handshake protocol

Design issues locally solved ⇒ **cells are easy to reuse**

Asynchronous circuit design principles

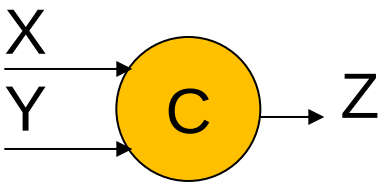
Designing a FIR filter

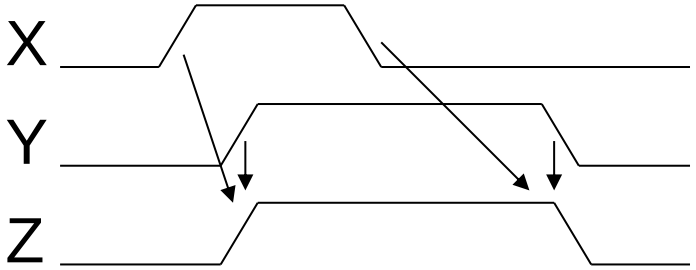


Asynchronous circuit design principles

The C-Element or Muller gate

Symbol

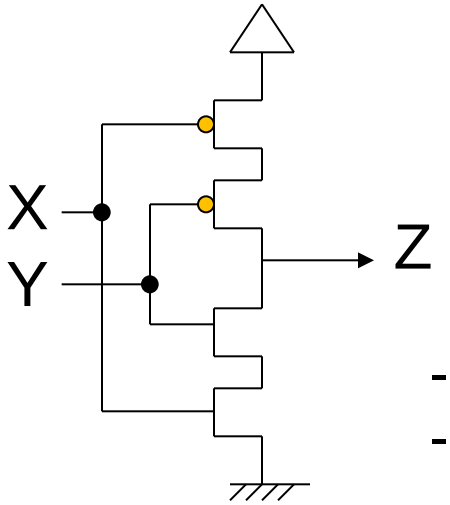




Truth table

X	Y	Z
0	0	0
0	1	Z^{-1}
1	0	Z^{-1}
1	1	1

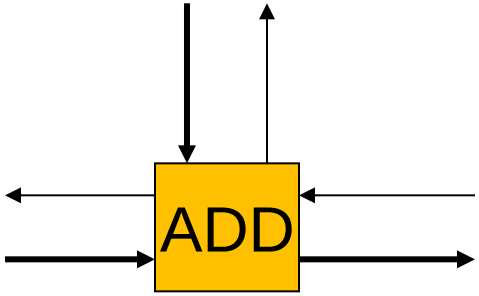
$Z = XY + Z(X+Y)$



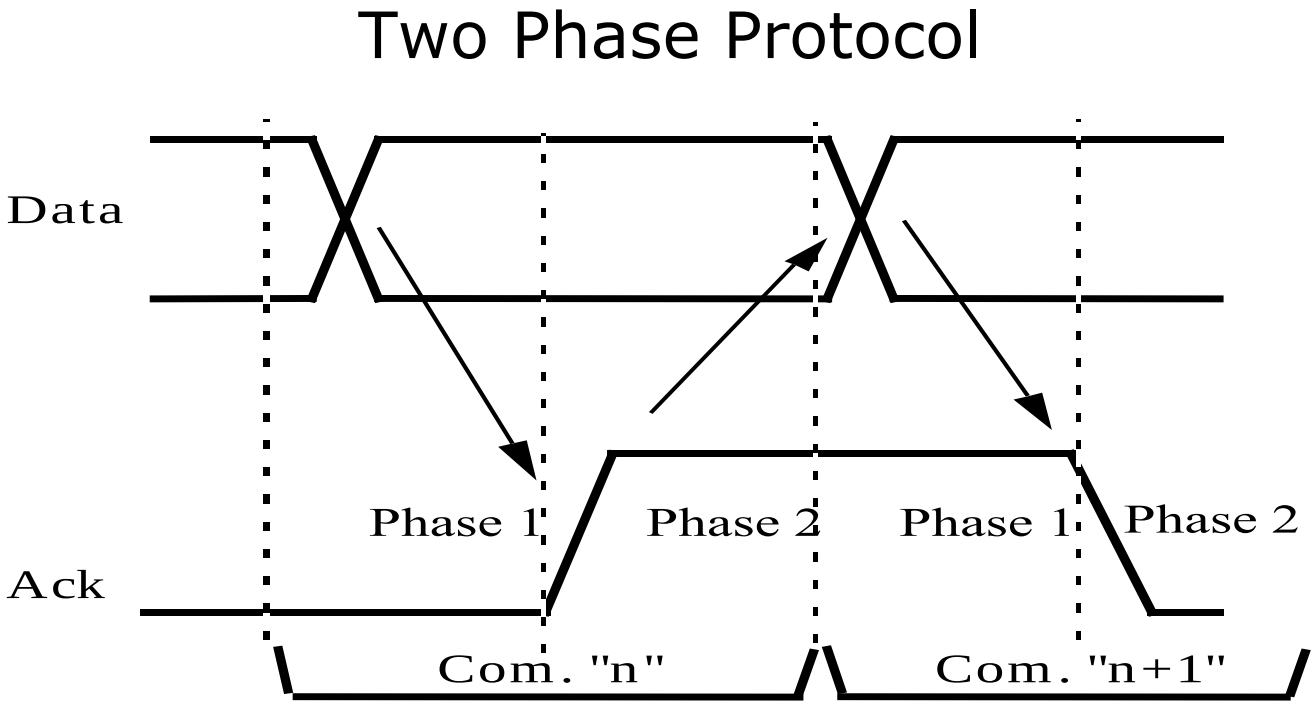
- State holding
- Reset

Asynchronous circuit design principles

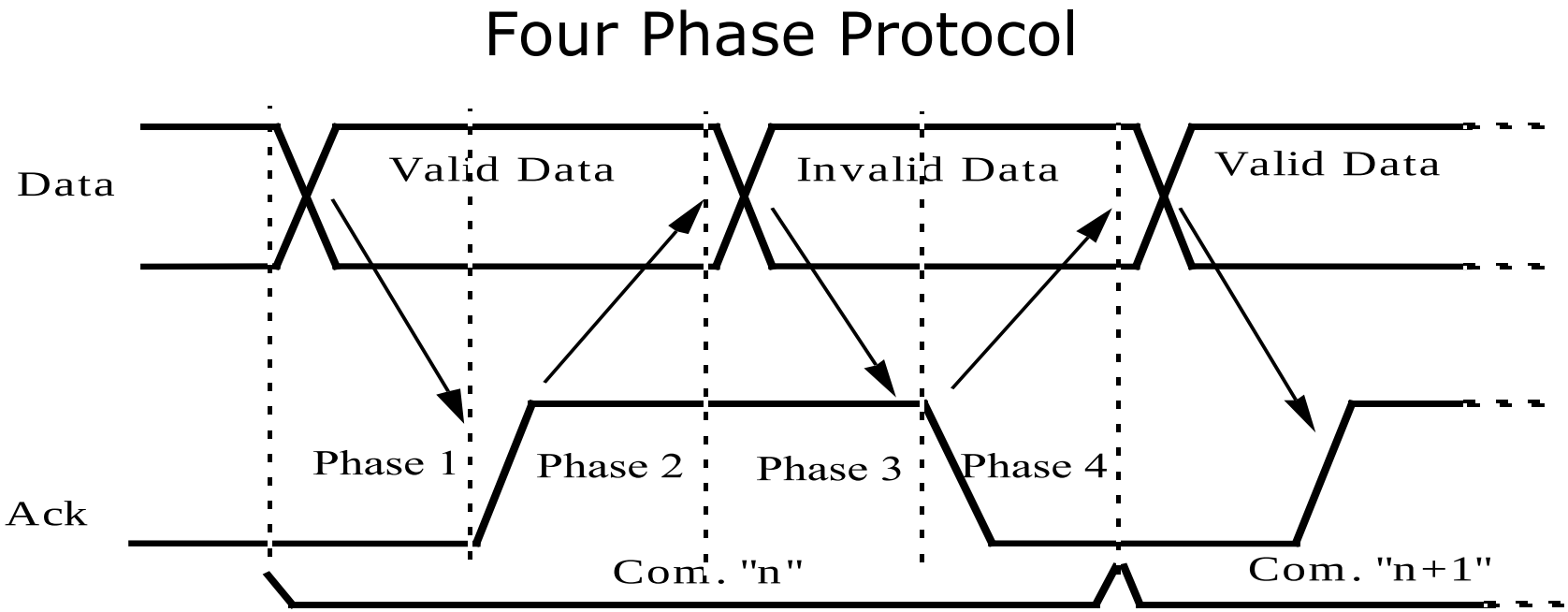
- Protocol
  - Two phases
  - Four phases
- Signaling
  - Data encoding / Request
    - Three states
    - Four states
  - Acknowledge



Asynchronous circuit design principles



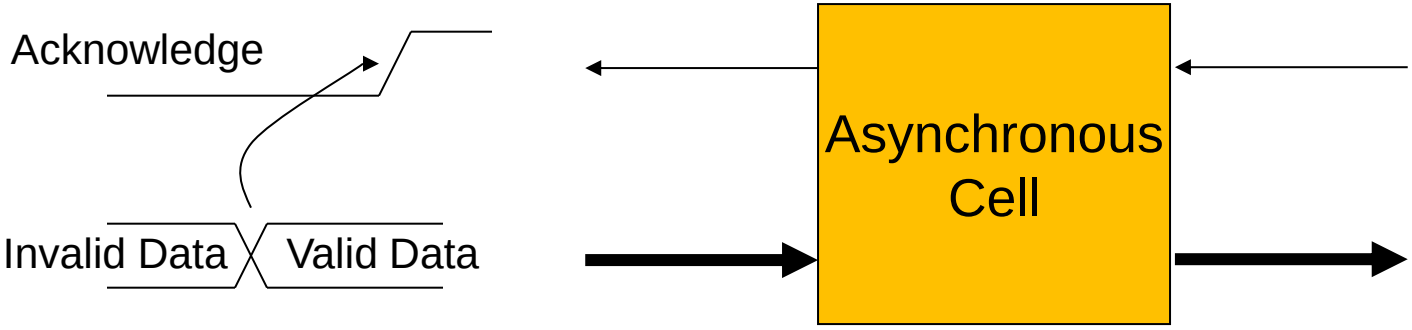
Asynchronous circuit design principles



➡ Several derivatives exist

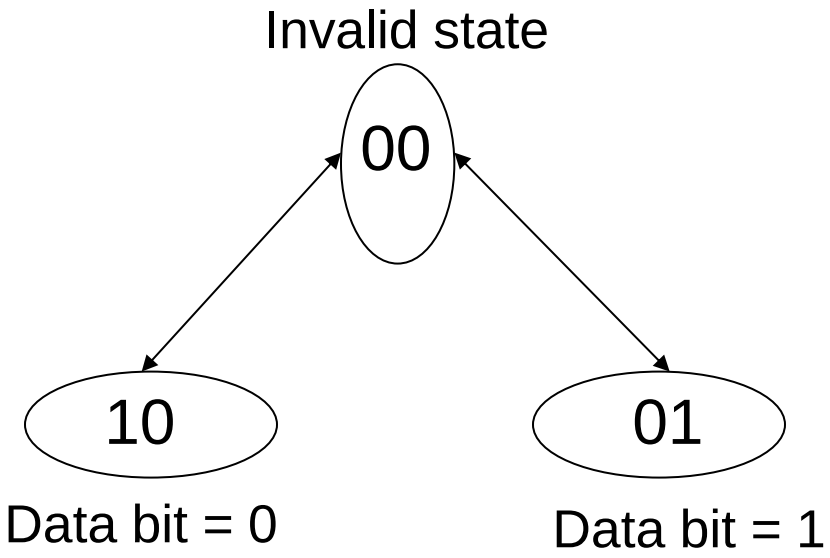
Asynchronous circuit design principles

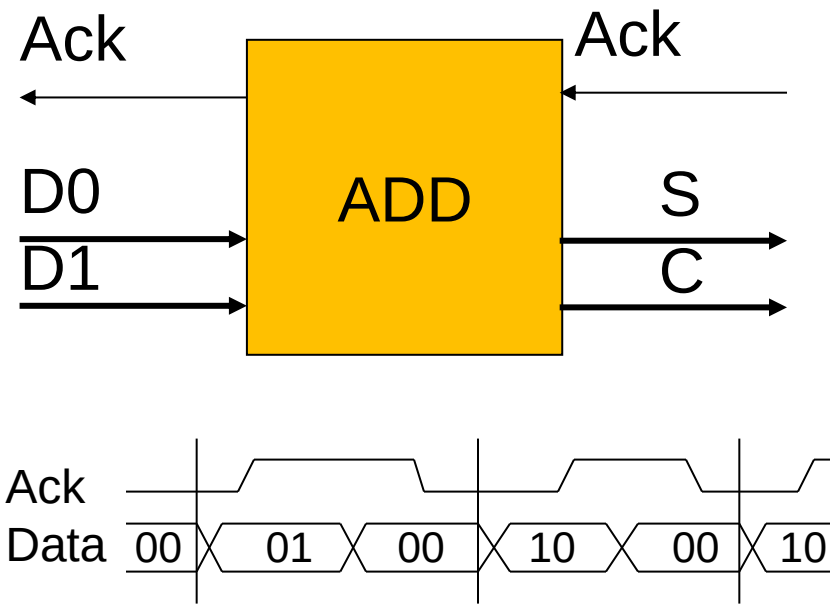
- 3-state encoding
  - Data Valid/Invalid Signaling
- 4-state encoding



Asynchronous circuit design principles

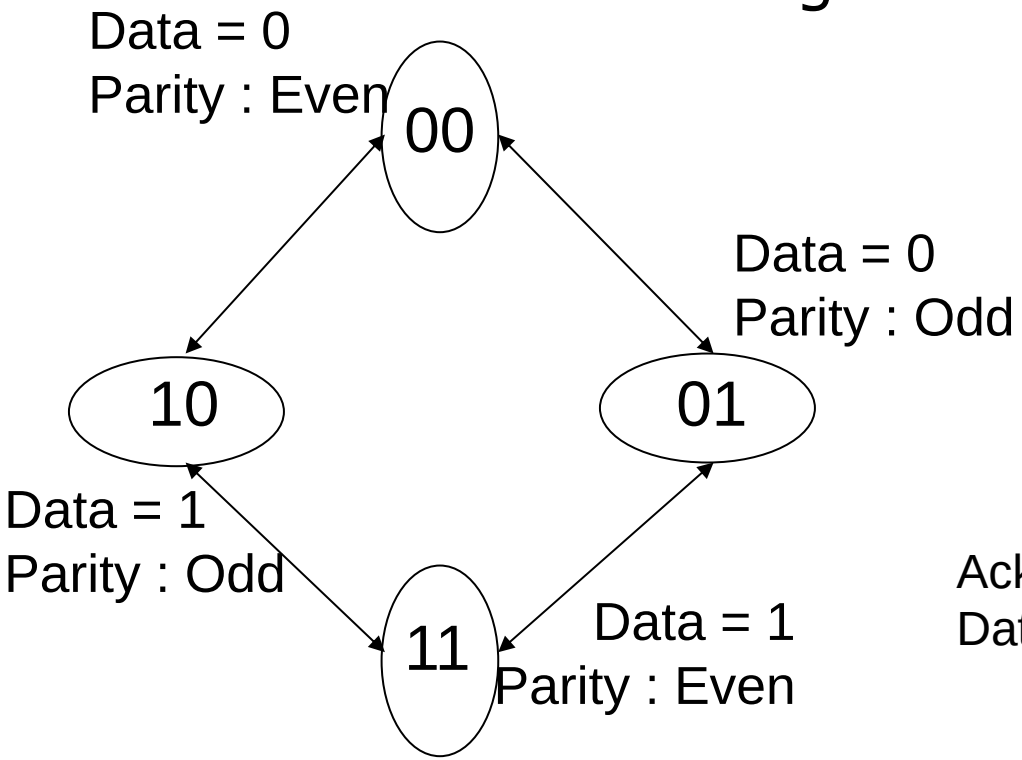
Data encoding: 3 states (dual rail)

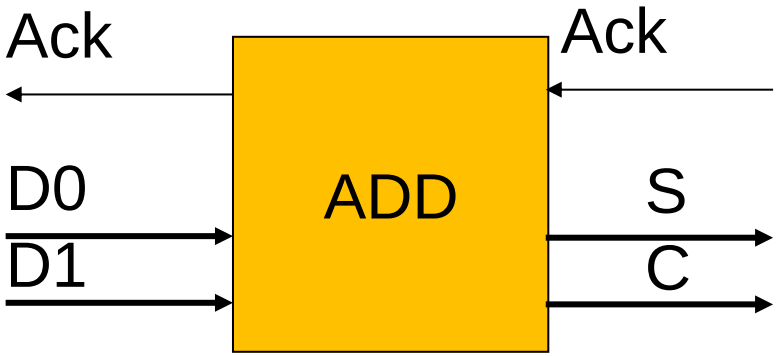


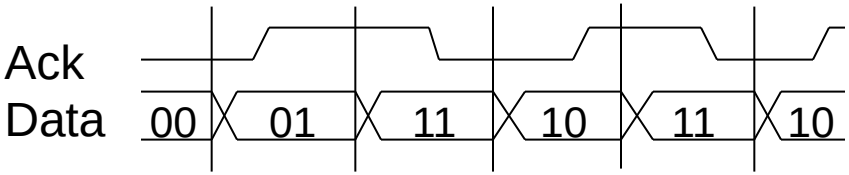


Asynchronous circuit design principles

Data encoding: 4 states (dual rail)



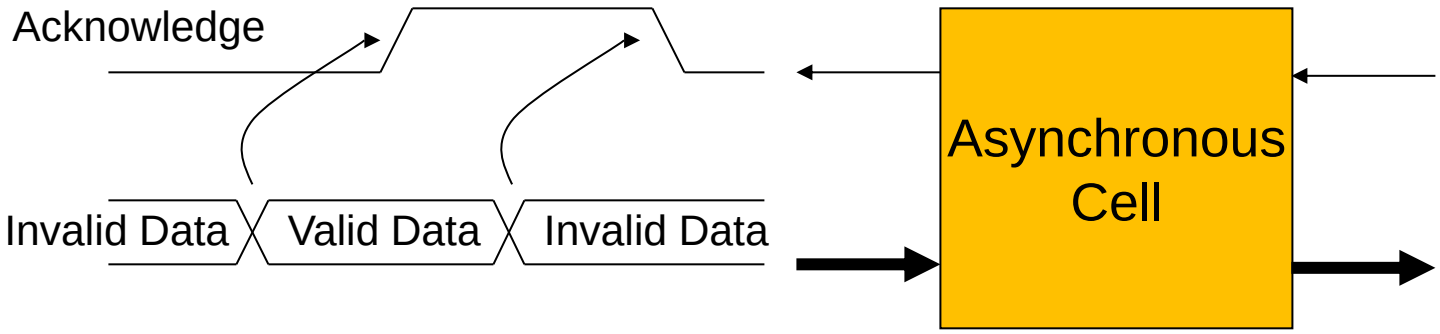




Asynchronous circuit design principles

Completion signal generation

- Internal clock (counter)
- Delay Cell
- Current sensing
- Data encoding

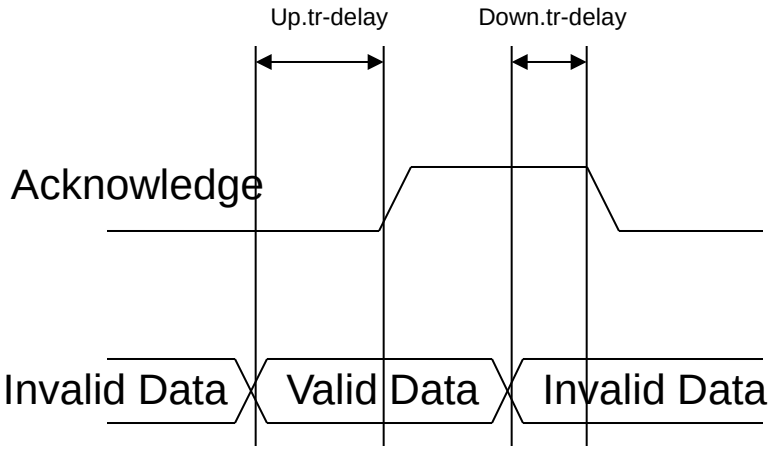


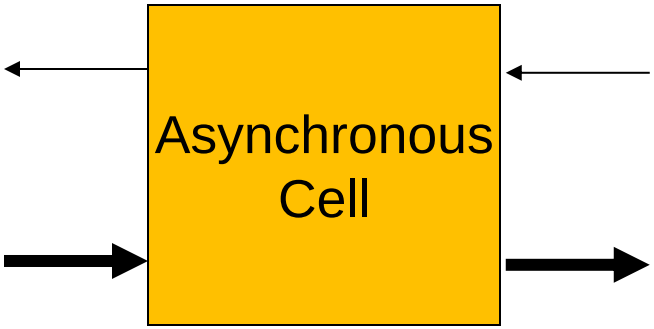
Asynchronous circuit design principles

Completion signal generation

- Use a delay cell

Asymmetric delay cell

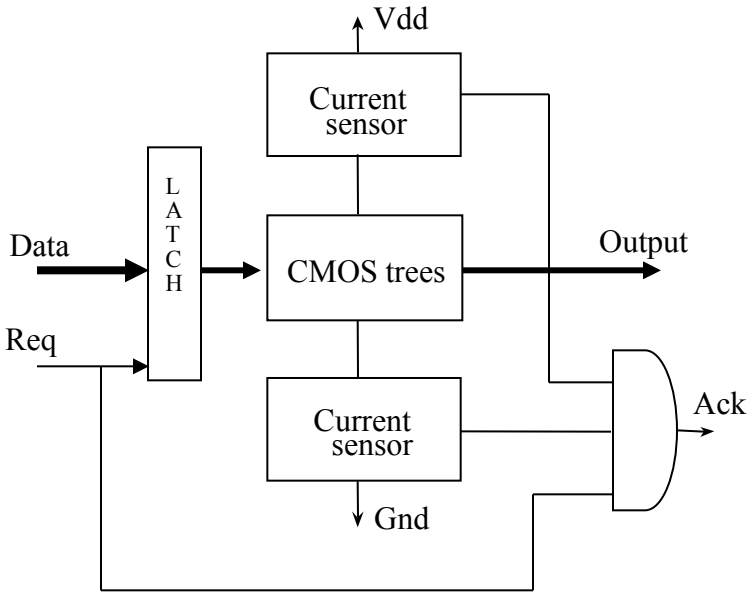


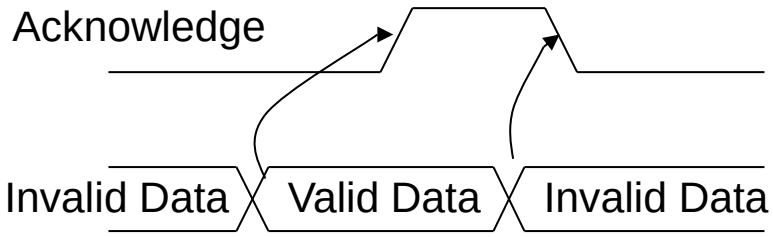


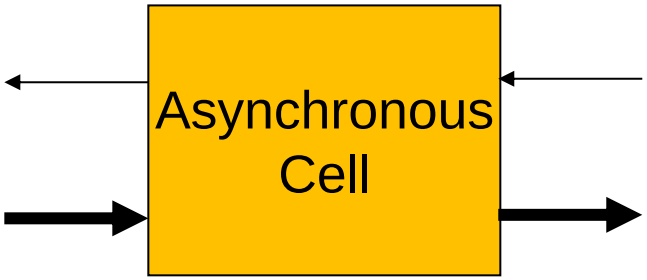
Asynchronous circuit design principles

Completion signal generation

- Current sensing





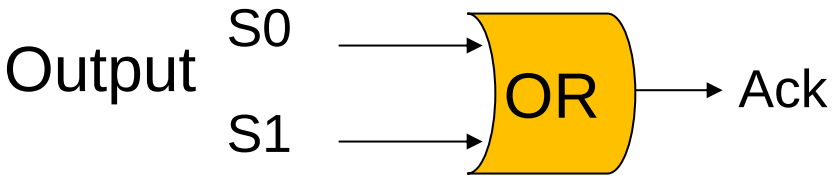


Asynchronous circuit design principles

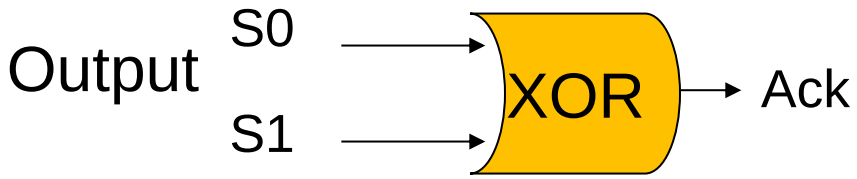
Completion signal generation

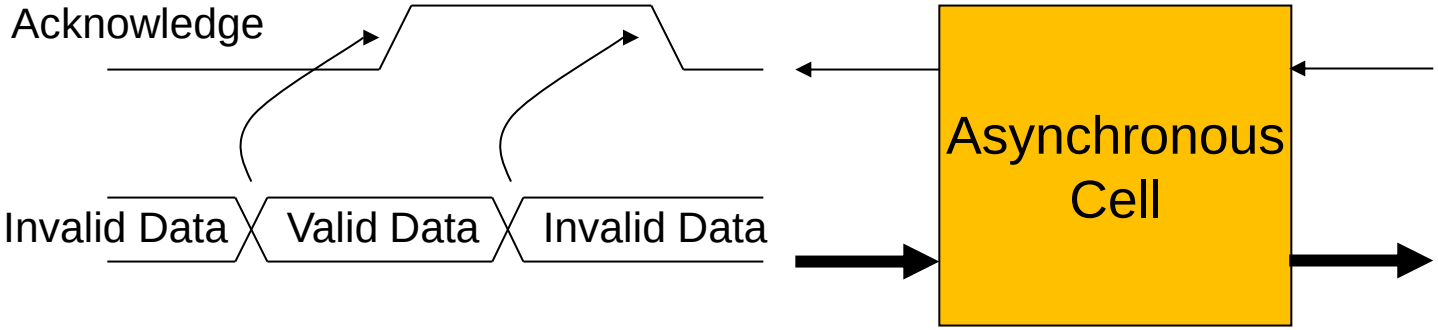
- Data encoding (dual rail)

3-state encoding



4-state encoding

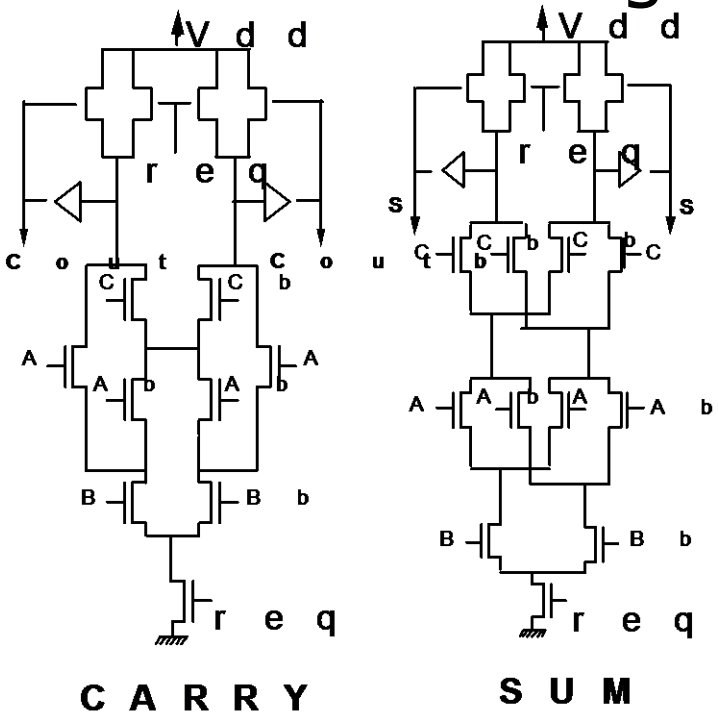


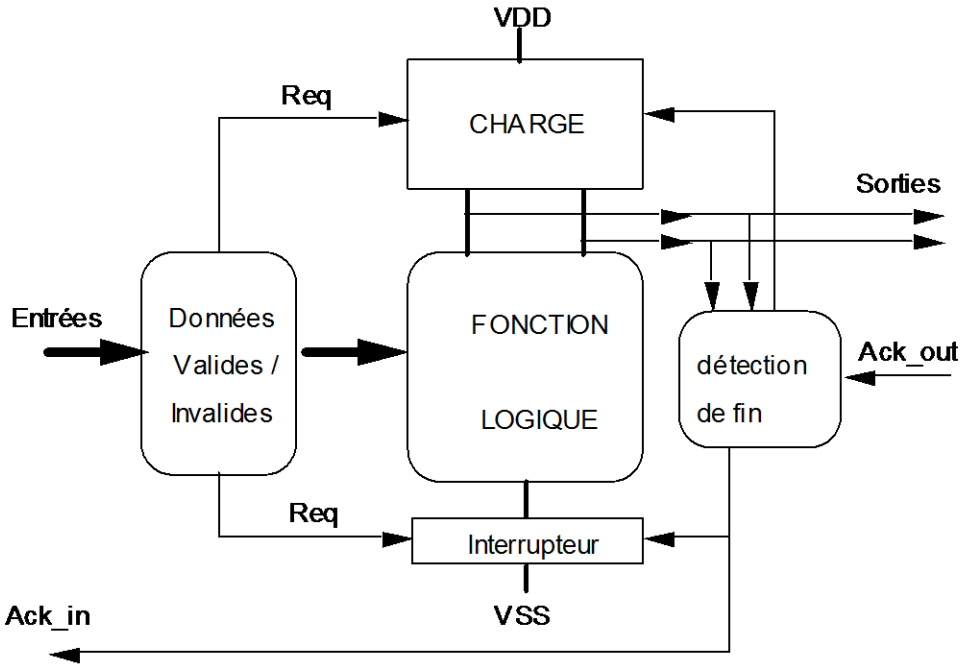


Asynchronous circuit design principles

Completion signal generation

- Data encoding





Asynchronous circuit design principles

- Conclusion
  - Asynchronous circuit communicate using an handshaking protocol
  - Data/Request have to be encoded
  - A completion signal is required

→ The implementation may be delay insensitive

→ Hazard free logic is required

→ Hardware overhead?

Outline

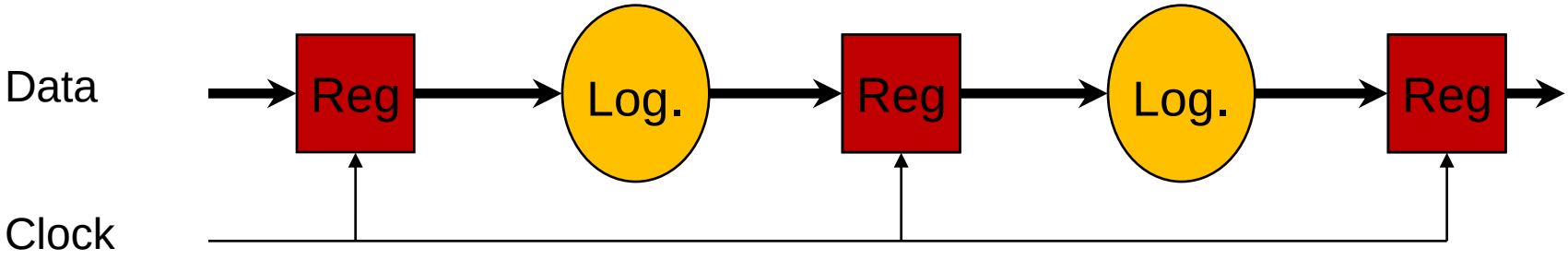
- Designing synchronous circuits
- Asynchronous circuit principles
- Asynchronous circuit design principles
- **Asynchronous circuit classes**
- Exploiting the asynchronous logic
- Success stories
- conclusion

Asynchronous circuit classes

- Hazard free logic
- QDI Circuits
  - Data path: a dual-rail OR Gate
  - Sequencing: the Q-Element
- Bounded delay circuits
  - Huffman circuits / Burst mode circuits
  - Sequencing: the Q-Element
- Micropipeline circuits

Asynchronous circuit classes

- Synchronous circuits

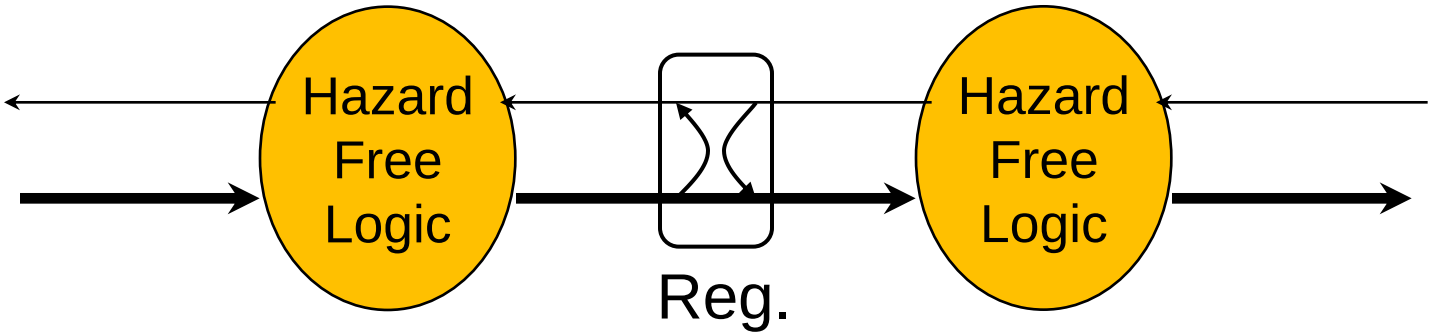


  - Time is discrete
    - → combinatorial logic is simple (hazards ignored)
    - → trivial communication mechanism
    - → worst case approach

→ **Global clock = Global timing assumption**

Asynchronous circuit classes

- Asynchronous circuits



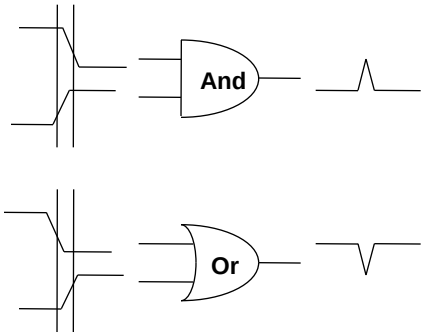
No global clock = No global timing assumption

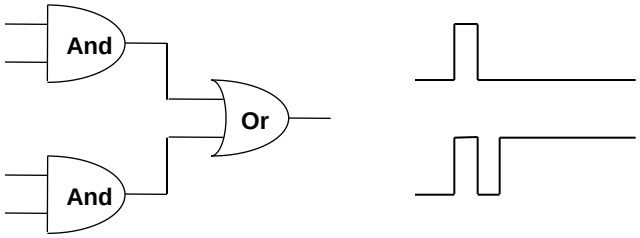
Sequencing is based on Handshaking

Hazard free logic is required

Asynchronous circuit classes

- Static hazards
- Dynamic hazards
- Combinatorial hazards

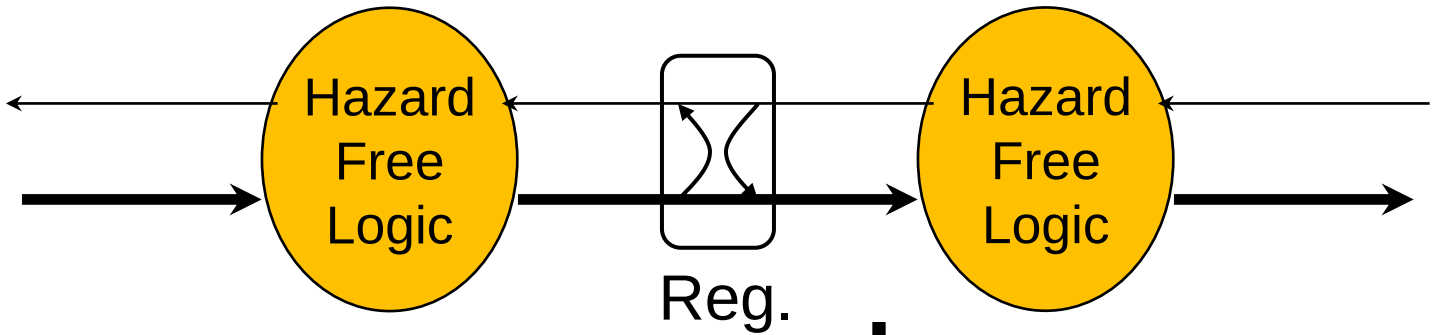




Asynchronous circuit classes

- Asynchronous circuits

No global clock
=
No global timing assumption



- Delay insensitive circuits
- Quasi delay insensitive circuits
- Speed independent circuits
- Huffman / Burst-mode circuits
- Micropipeline

Robustness & Complexity

are decreasing :

more timing assumptions

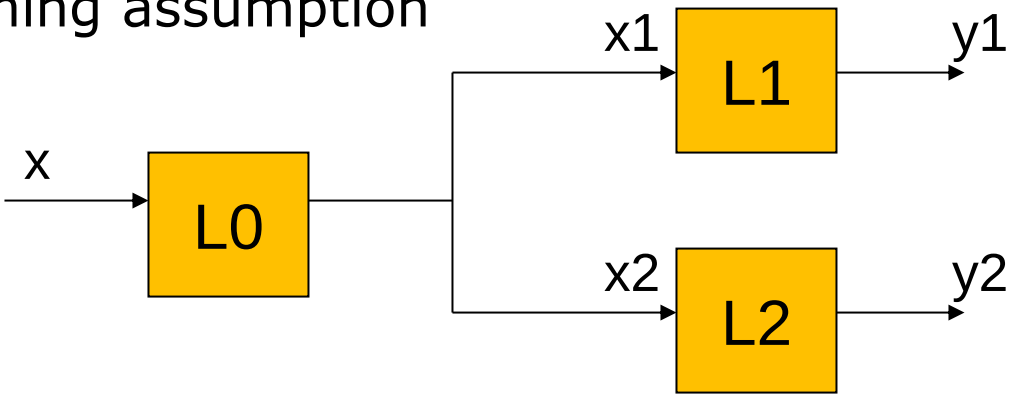
Asynchronous circuit classes

- Hazard free logic
- QDI Circuits
  - Data path: a dual-rail OR Gate
  - Sequencing: the Q-Element
- Bounded delay circuits
  - Huffman circuits / Burst mode circuits
  - Sequencing: the Q-Element
- Micropipeline circuits

Asynchronous circuit classes

QDI Asynchronous circuits

- Functionally correct without any assumption on the wire and gate delays (unbounded delay model) except...
- "Isochronic fork" timing assumption



→ Robustness is maximum (with respect to delay variations)

Asynchronous circuit classes

Speed Independent asynchronous circuits

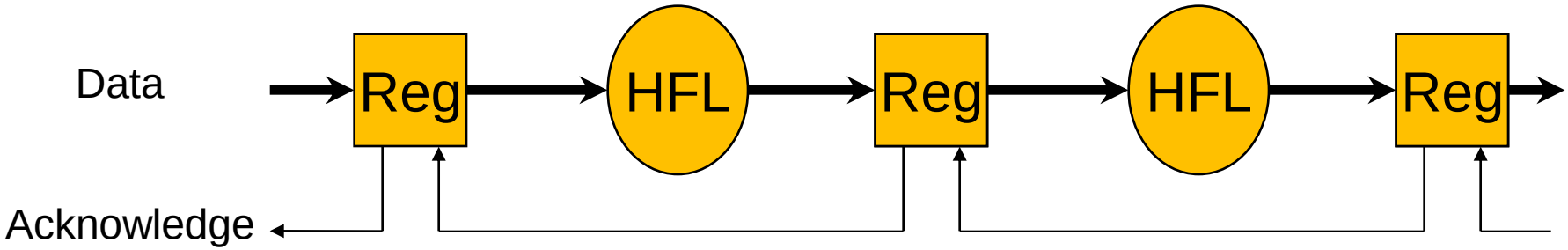
- Functionally correct whatever the delays in the gates (unbounded delay model)
- The wires are assumed to be zero delay
- => all wires respect the isochronic fork property

→ Less accurate than the QDI model

QDI asynchronous circuits

- Quasi Delay Insensitive:

⇒ hazard free control logic & hazard free data-paths



→ Time is no longer discrete

→ Hazard free logic

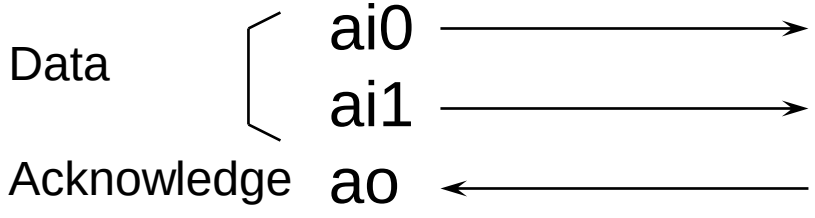
→ Handshake based communications

→ Mean time approach

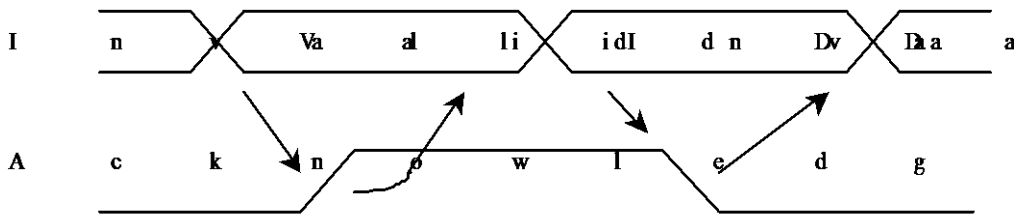
⇒ **No timing assumption**

QDI asynchronous circuits

- Implementing delay insensitivity: examples
  - Choice of a communication protocol (request - acknowledge)
    - 1 bit Channel
    - Data encoding



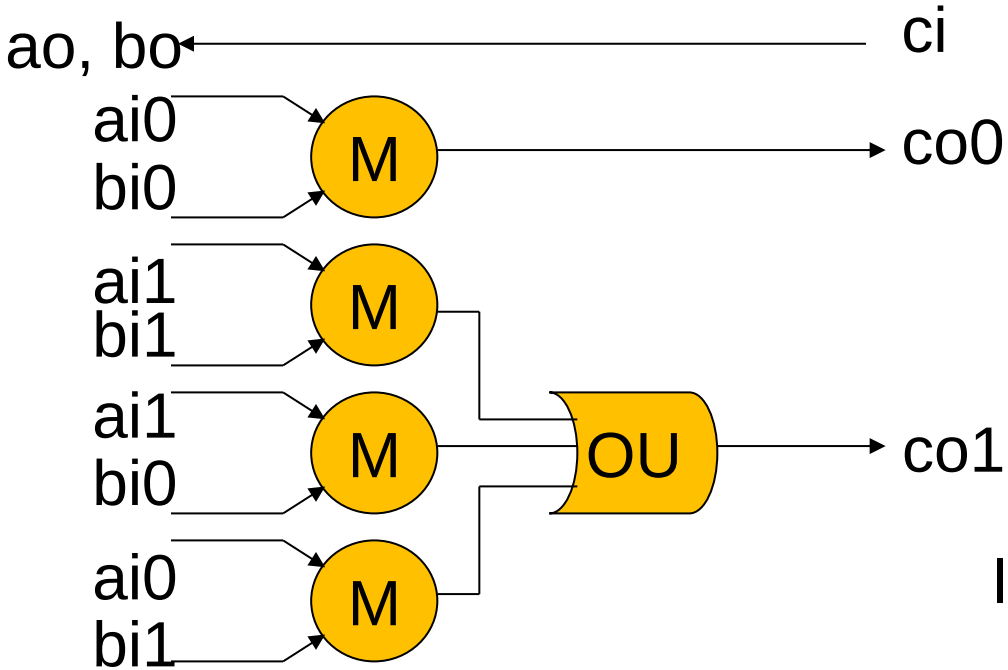
Data	ai0	ai1
0	1	0
1	0	1
Invalid	0	0

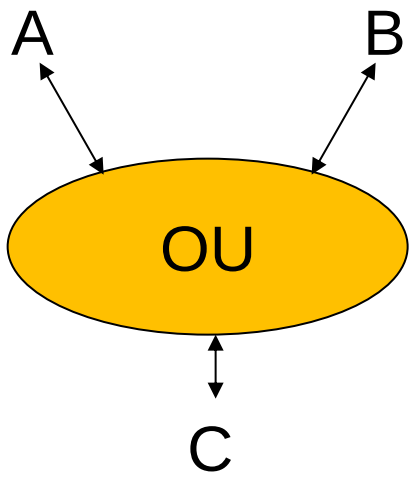


- 4 phase protocol

QDI asynchronous circuits

- An example: dual-rail OR Gate

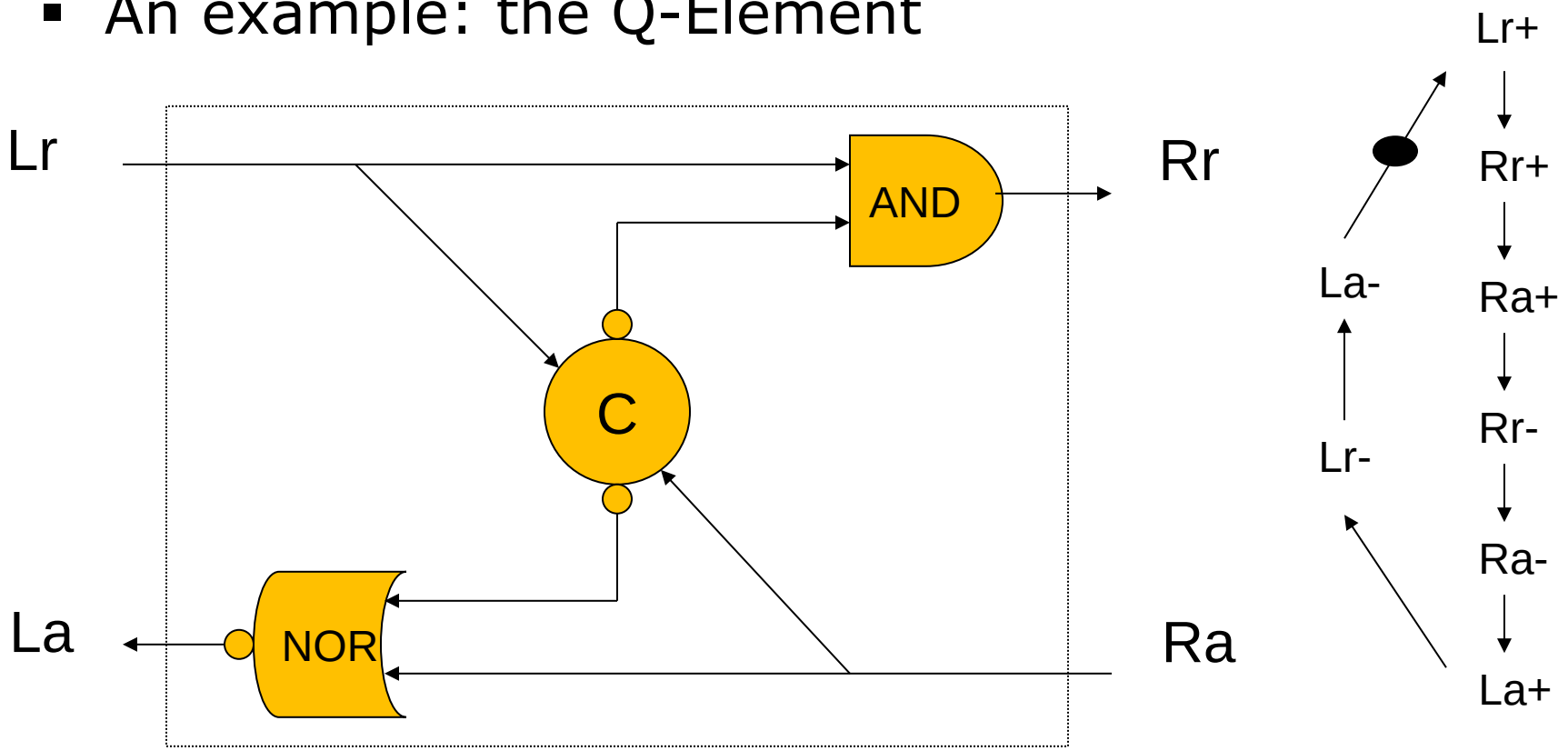




Implement both the function and the protocol

QDI Asynchronous circuits

- An example: the Q-Element

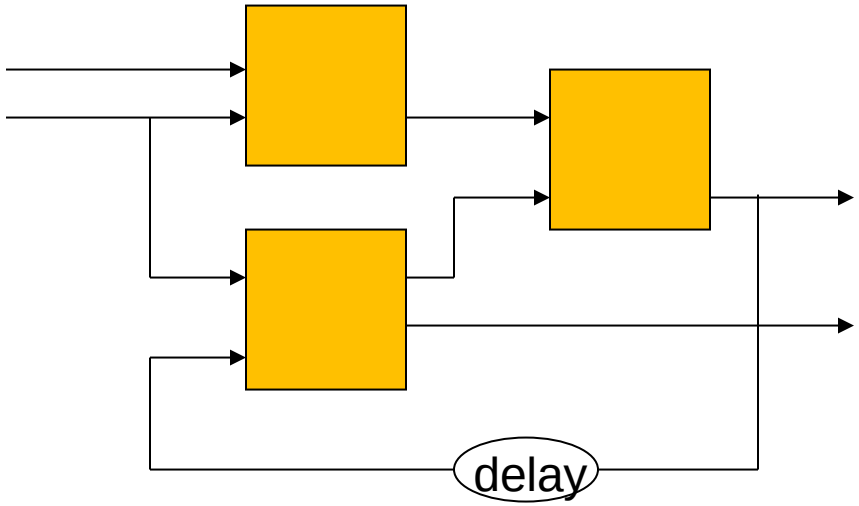


Asynchronous circuit classes

- Hazard free logic
- QDI Circuits
  - Data path: a dual-rail OR Gate
  - Sequencing: the Q-Element
- Bounded delay circuits
  - Huffman circuits / Burst mode circuits
  - Sequencing: the Q-Element
- Micropipeline circuits

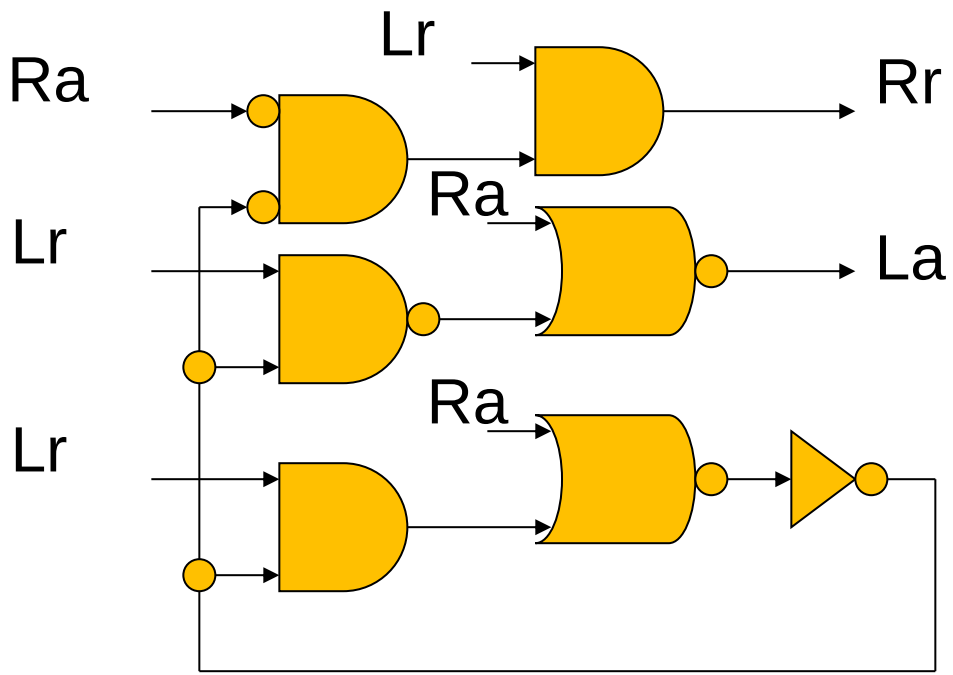
Huffman/Burst-mode asynchronous circuits

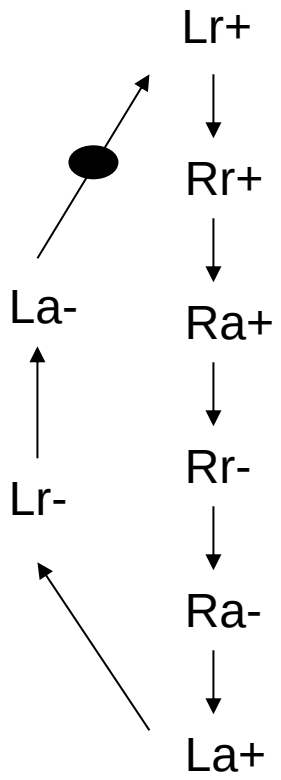
- The correctness depends on the gate/wire delays
- Based on the "bounded delay" model
- "Fundamental mode" assumption for the environment



Burst-mode asynchronous circuits

- An example: The Q-Element



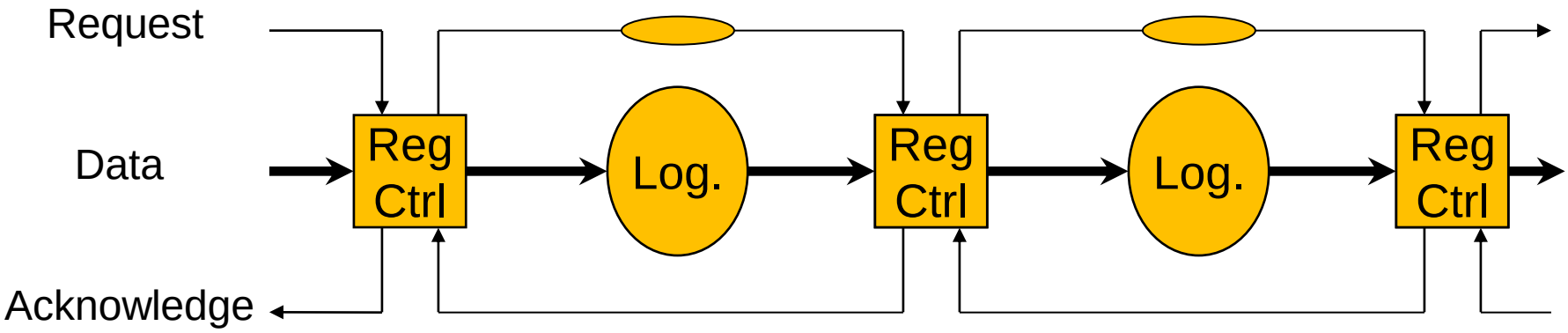


Asynchronous circuit classes

- Hazard free logic
- QDI Circuits
  - Data path: a dual-rail OR Gate
  - Sequencing: the Q-Element
- Bounded delay circuits
  - Huffman circuits / Burst mode circuits
  - Sequencing: the Q-Element
- Micropipeline circuits

Micropipeline asynchronous circuits

- Micropipeline

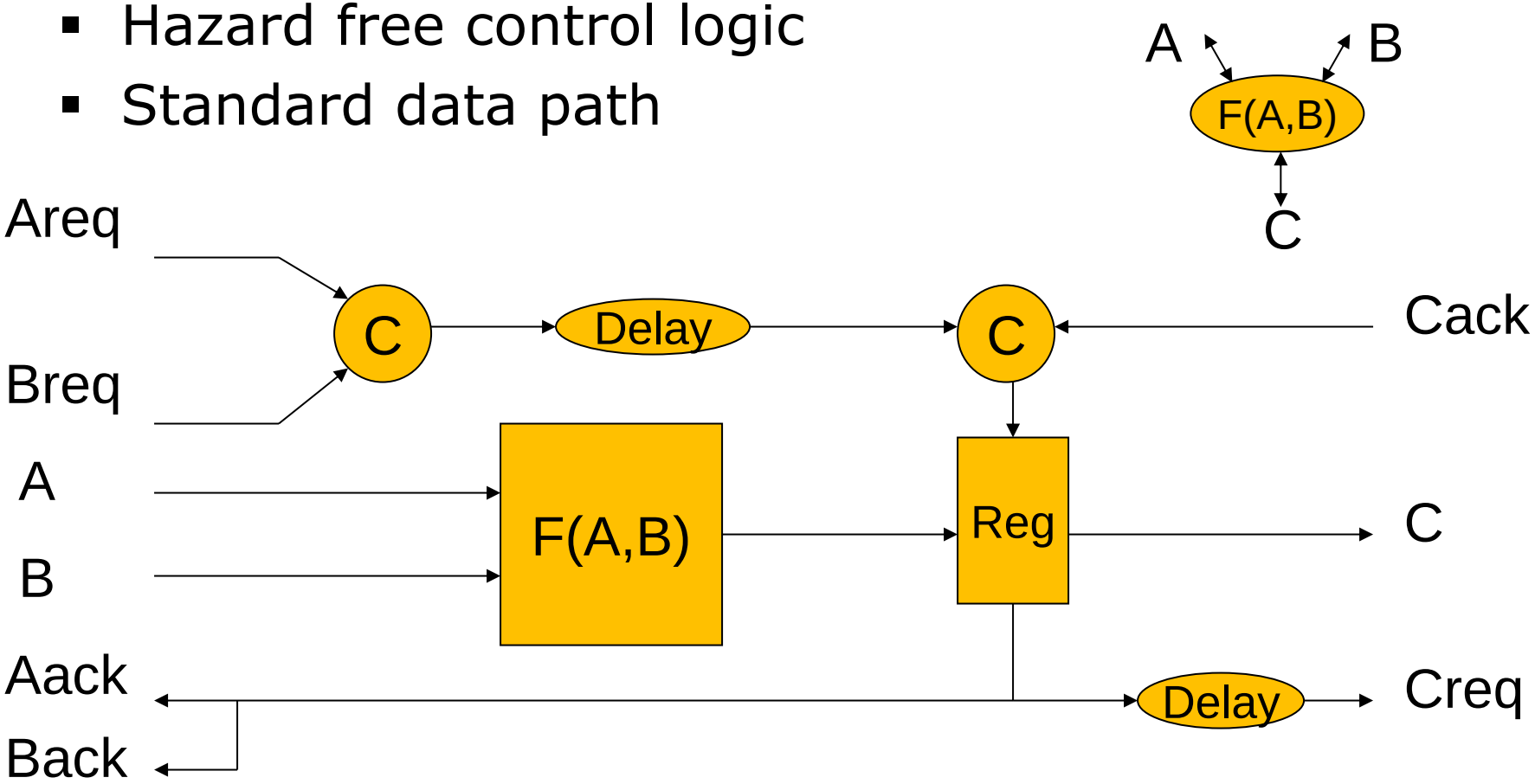


→ Time is discrete
→ Combinatorial logic is simple
→ Communication channels (handshake based)
→ Locally worst case approach

Local timing assumptions

Micropipeline asynchronous circuits

- Hazard free control logic
- Standard data path



Asynchronous circuit classes

- Conclusion
  - QDI circuits are the most robust with respect to delays (isochronic fork for some wires)
  - Huffman & Burst-Mode circuits use the bounded delay model and require the fundamental mode

→ **QDI : Data-paths & Controllers**
→ **Speed Independent / Burst-Mode: Controllers**
→ **Micropipeline : Standard data-path + QDI Controllers**

Asynchronous circuit classes

QDI

Pros

- Delay insensitive
- Fast
- Low power with some design effort
- Self-testable with certain logic style

Cons

- Larger area
- Synthesis of data-paths is complex

Micropipeline

Pros

- Low overhead
- Synthesis of data-path is performed using commercial tools
- Low power with some design effort

Cons

- Not delay insensitive
- Not very fast
- Some parts are difficult to test (delay fault)

Outline

- Designing synchronous circuits
- Asynchronous circuit principles
- Asynchronous circuit design principles
- Asynchronous circuit classes
- **Exploiting the asynchronous logic**
- Success stories
- conclusion

Exploiting the asynchronous logic

Asynchronous circuits

- Fast
  - speed only depends on the circuit complexity
- Low power, low noise
  - data driven (only the processing part consumes power)
  - Distributed processing in space and time
- Area
  - complexity depends on the asynchronous logic style
- Safe and secure
  - Robust to PVT variations
  - Less sensitive to DPA and FA

Exploiting the asynchronous logic

Modularity and Locality

SoCs design is easier: (reduced Time-to-Market)

- Distributed control (protocol implementation)
- Delay insensitive communications between modules
- Modules are independent from each other in terms of:
  - Functionality (global state not known)
  - Speed (maximum speed)
  - Activity (power)
  - Noise (uncorrelated current consumption)
- Scalability

Modularity
Reusability
Scalability

Exploiting the asynchronous logic

On-chip communication systems

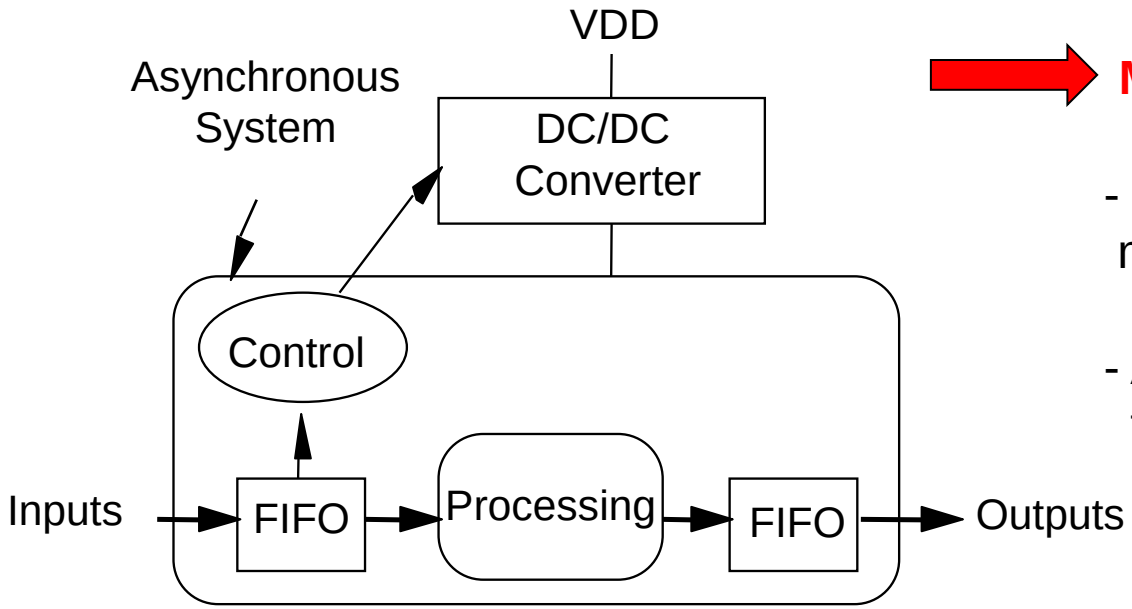
- Multiple Clock Domains (GALS)
  Metastability may occur at clock domain boundaries
  - Control the Mean Time Between Failure (MTBF)
    - Non adaptive synchronization
    - Adaptive synchronization → Probability of error is not zero
  - Avoiding metastability
    - Stretchable clocks → Probability of error is zero
- Fully asynchronous (GALA)


Issues: reliability and latency

Exploiting the asynchronous logic

Automatic performance regulation

- Computation-power controlled systems : $E = a.fCV^2$



Dynamic regulation of the power supply with respect to the processing power required.

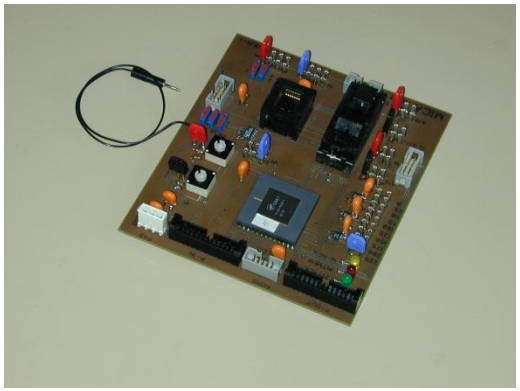
Minimal energy

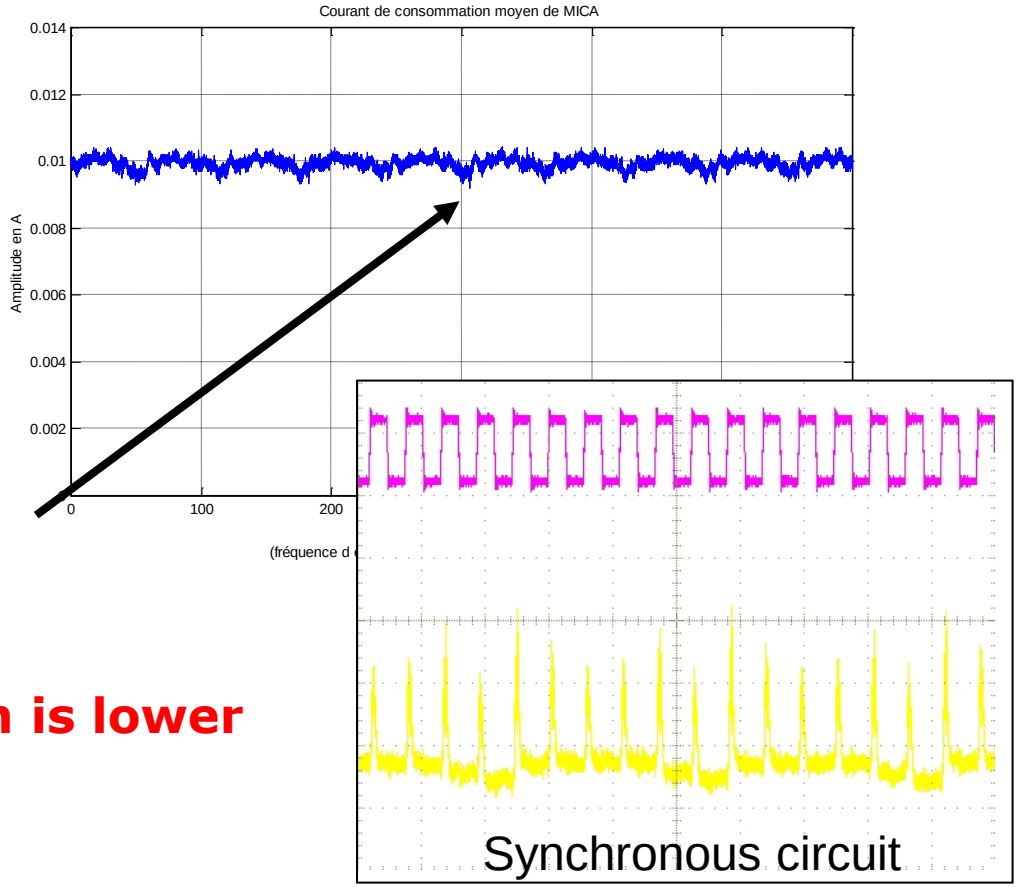
- Processing and data may be irregular

- A breakdown with respect to the synchronous approach

Exploiting the asynchronous logic

Noise





MICA: a QDI 8-bit microcontroller
-10 mA average current
- 0.6 mA amplitude variations

Mean power consumption is lower

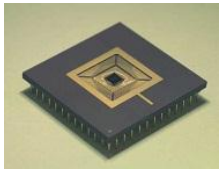
Smaller current peaks

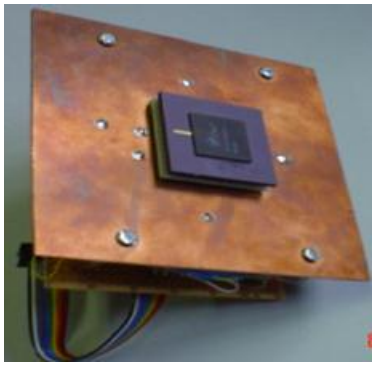
EM emission is lower

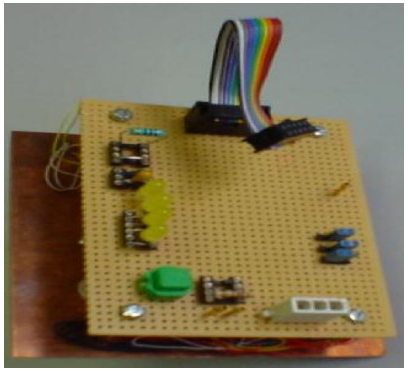
Exploiting the asynchronous logic

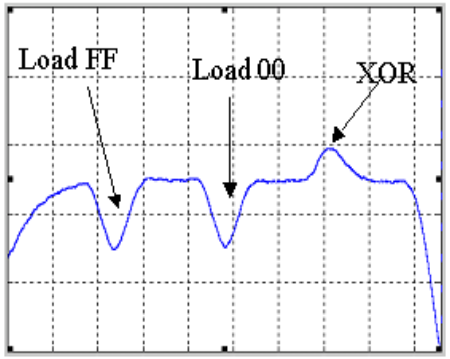
Security

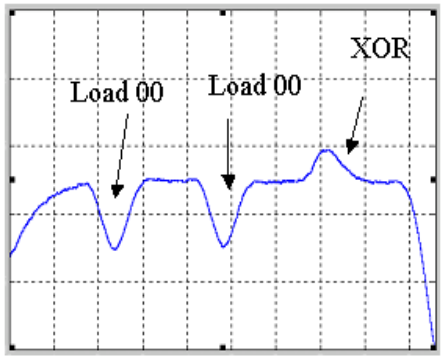
- Processor MICA



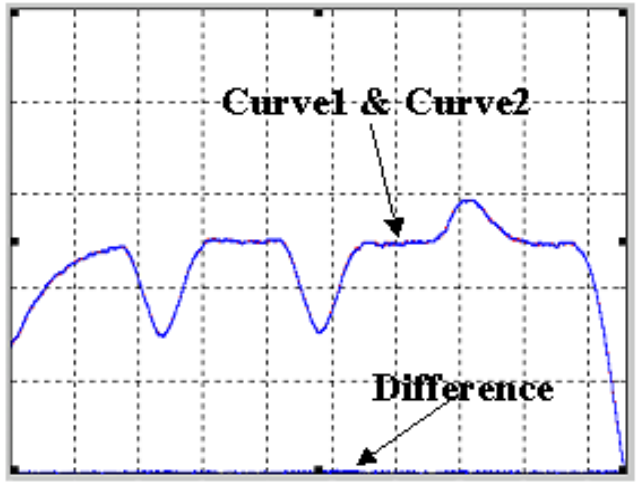








- Number of points : 100000
- 10000 computations

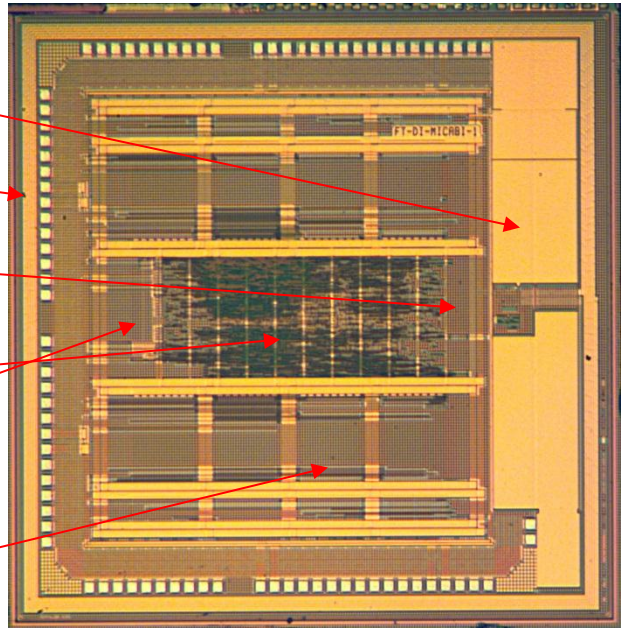


Outline

- Designing synchronous circuits
- Asynchronous circuit principles
- Asynchronous circuit design principles
- Asynchronous circuit classes
- Exploiting the asynchronous logic
- **Success stories**
- conclusion

SoC for Contactless Smart-Card

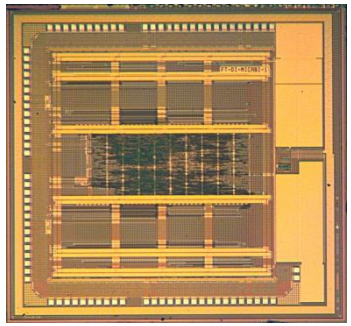
- Power reception system (on-chip coil)
- ISO 14443-B std compliant
- 8-bit CISC Asynchronous Microcontroller designed with standard cells (Mica)
- Rom
- Rams



Collaboration with France Telecom/R&D
CMOS 0.25 µm STMicroelectronics

SoC for Contactless Smart-Card

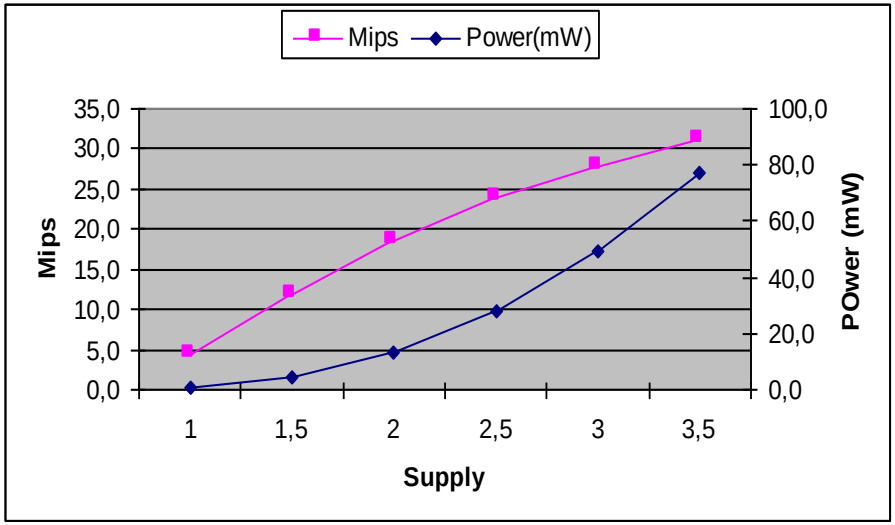
- Asynchronous Logic relaxed Design constraints
- Not sensitive to supply voltage variations
  → Power reception system (capacitances area, voltage regulation)
- Lower current peaks
  → The Micro-controller can be running during the communications without disturbing the load modulation.
- Maximum processing power delivered according to the power received



Collaboration with France Telecom/R&D
CMOS 0.25 µm STMicroelectronics

MICA information sheet

- CMOS 0.25 µm
- 1-of-4 DI codes for arith. and reg.
- 1-of-n DI codes for the control
- Complexity
  - 145 000 transistors
  - 1 M transistors with memories
  - 13 mm² with pads (prototype)
  - PGA120 package for the prototype
- Test
  - BIST (approx. 300 instr)
  - functional at 1er silicon between 3v et 0.65 v
- 24 Mips / 28 mW @ 2.5V
- 4,3 Mips / 800 µW @ 1V

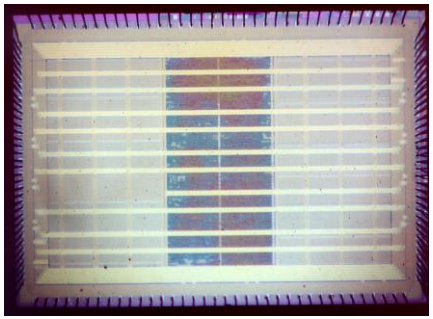


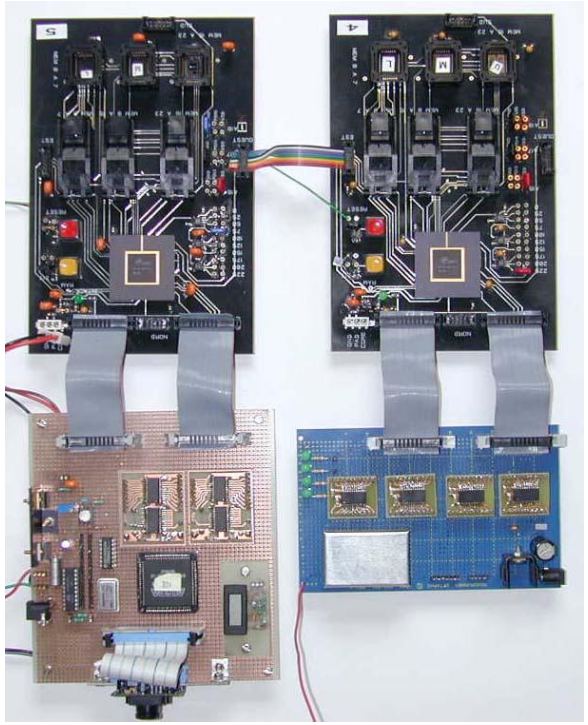
Supply(V)	Mips	Core Current (mA)	Power(mW)	Mips/Watt
1	4,3	0,8	0,8	5503,6
1,5	11,9	3,1	4,7	2560,2
2	18,6	6,7	13,3	1398,0
2,5	23,8	11,2	28,0	850,3
3	27,8	16,3	48,9	568,1
3,5	31,3	22,0	77,0	405,8

And many other circuits …

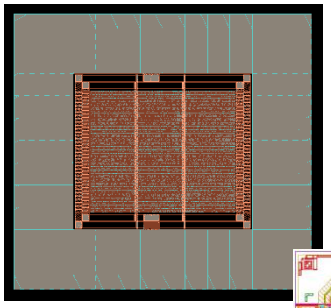
ASPRO (ASynchronous PROcessor)

- 16-bit RISC processor
- 140 MIPS
- QDI asynchronous logic
- Standard Cells
- 500 KTr for the core
- 6.3 MTr with memories
- Total area is 42 mm^2
- CMOS 0.25μm 6 metal layers STMicroelectronics

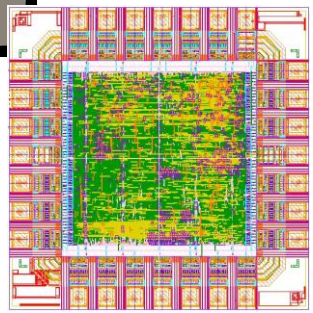




And many other circuits …

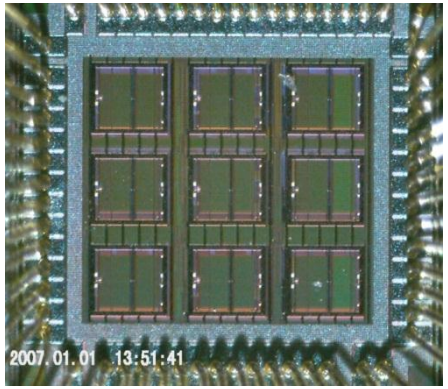


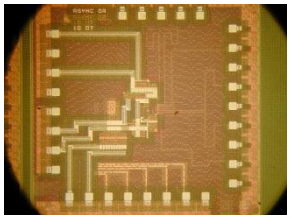
FA secured DES circuit (CMOS ST 65 nm)



DPA secured DES circuit
(CMOS ST 65 nm)

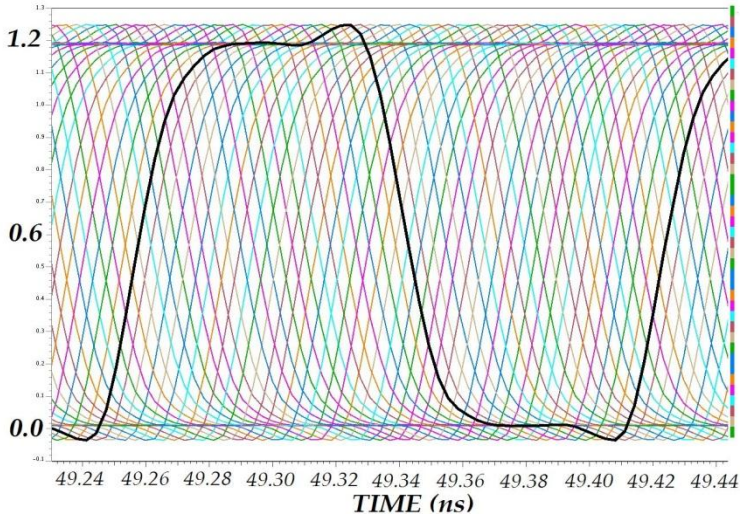
The first world secured FPGA (DPA)
(CMOS ST 65 nm)





And many other circuits …

Low-phase noise multiphase asynchronous oscillator
(CMOS ST 65 nm)



N° of stages	T/B	Freq. (GHz)	Consum. (mW)	PN at 1M (dBc)	PN at 10M (dBc)
3	2T/1B	3.95	0.454	-82.97	-109.07
6	4T/2B	3.95	0.908	-85.98	-112.08
12	8T/4B	3.95	1.817	-88.99	-115.09
24	16T/8B	3.95	3.635	-92	-118.1

N° of stages	T/B	Frequ. (GHz)	Comsu. (mW)	N° of phases	Resolution (ps)	PN at 1 MHz
9	4/5	6.41	1.94	9	17.3	**-82.9**
21	10/11	6.16	4.47	21	7.7	**-87.6**
41	**20/21**	**6.02**	**8.62**	**41**	**4**	**-90.7**

Outline

- Designing synchronous circuits
- Asynchronous circuit principles
- Asynchronous circuit design principles
- Asynchronous circuit classes
- Exploiting the asynchronous logic
- Success stories
- **conclusion**

Conclusion

Designing asynchronously means:

- **Channel-based** SoCs instead of clock-based
- **Delay-insensitive** or reduced timing assumptions
- **Event-driven** instead of clock-driven
  - → Low-power, low-noise
  - → Modularity, locality, scalability, reusability
  - → Reliability, safety, security
  - → Reduced design time (TTM)

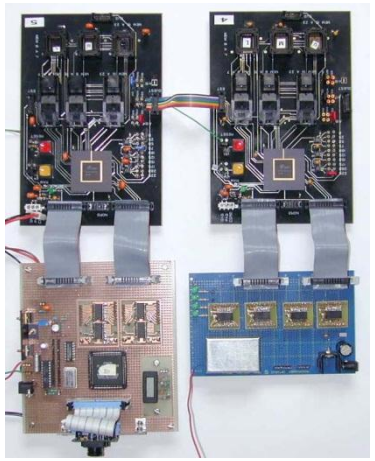
- **Are fully asynchronous systems the future?**
  - → Synergy between sensors, actuators, interfaces and processing

Towards Fully Asynchronous Systems?

- Many startups: Handshake Solutions, Fulcrum, Theseus logic, stilistix, Tiempo, …
- one spin-off from TIMA!

…

- Intel recently acquired Fulcrum microsystems



Thinking and Designing Differently: The Asynchronous Alternative

Thanks for your attention!