UiO : Department of Informatics
University of Oslo

IN3160

FPGA circuit technologies and configuration

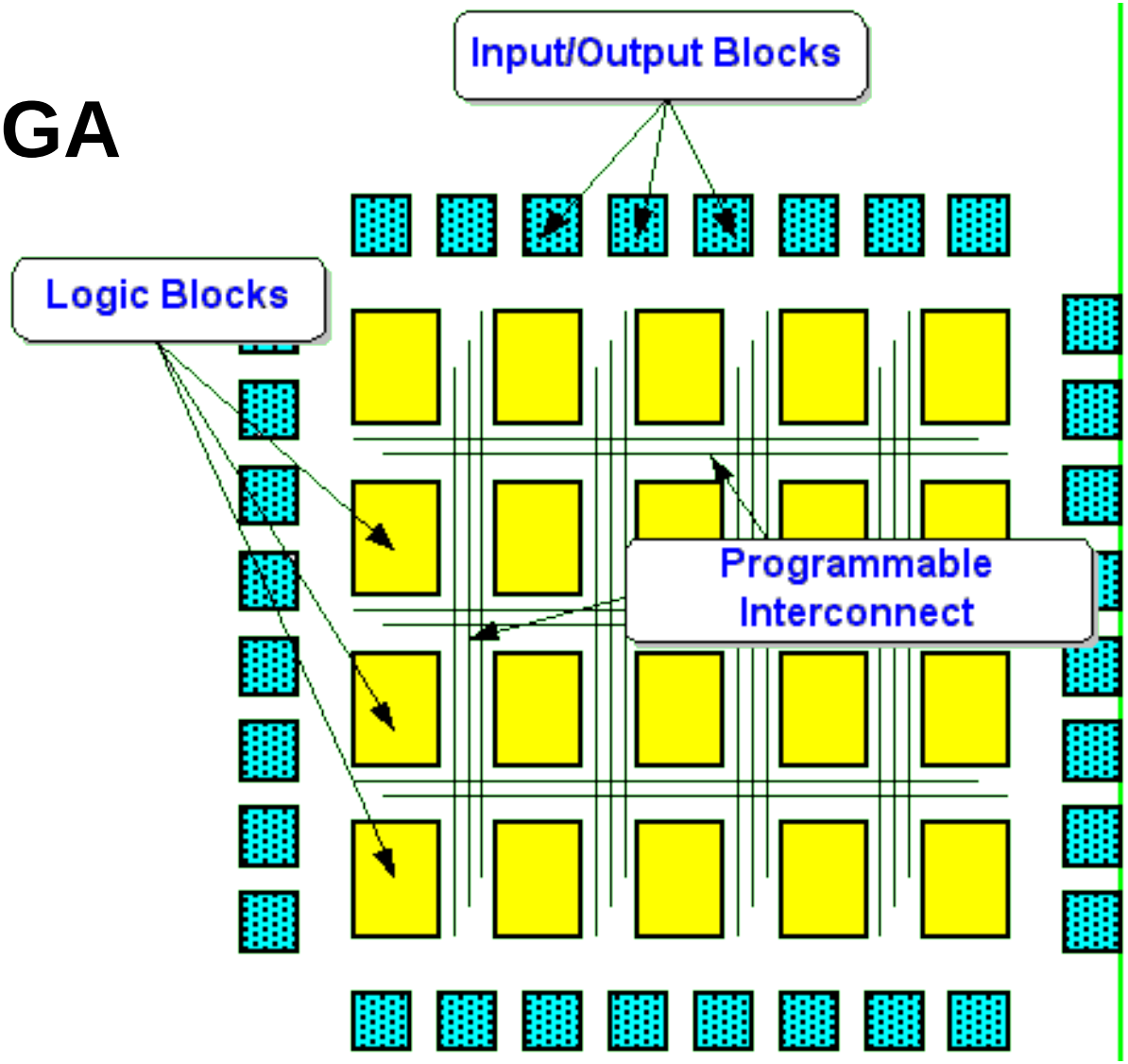
Outline

- Circuit technologies
- FPGA configuration
- Case study: Mars Rover

The difference between a microprocessor and programmable logic

- A micro processor is programmed with instructions (which are stored in RAM/ROM)
- A programmable logic circuit is programmed by a circuit description
- A programmable circuit consist of configurable blocks of logic and configurable connections between these block

FPGA


Input/Output Blocks
Logic Blocks
Programmable
Interconnect

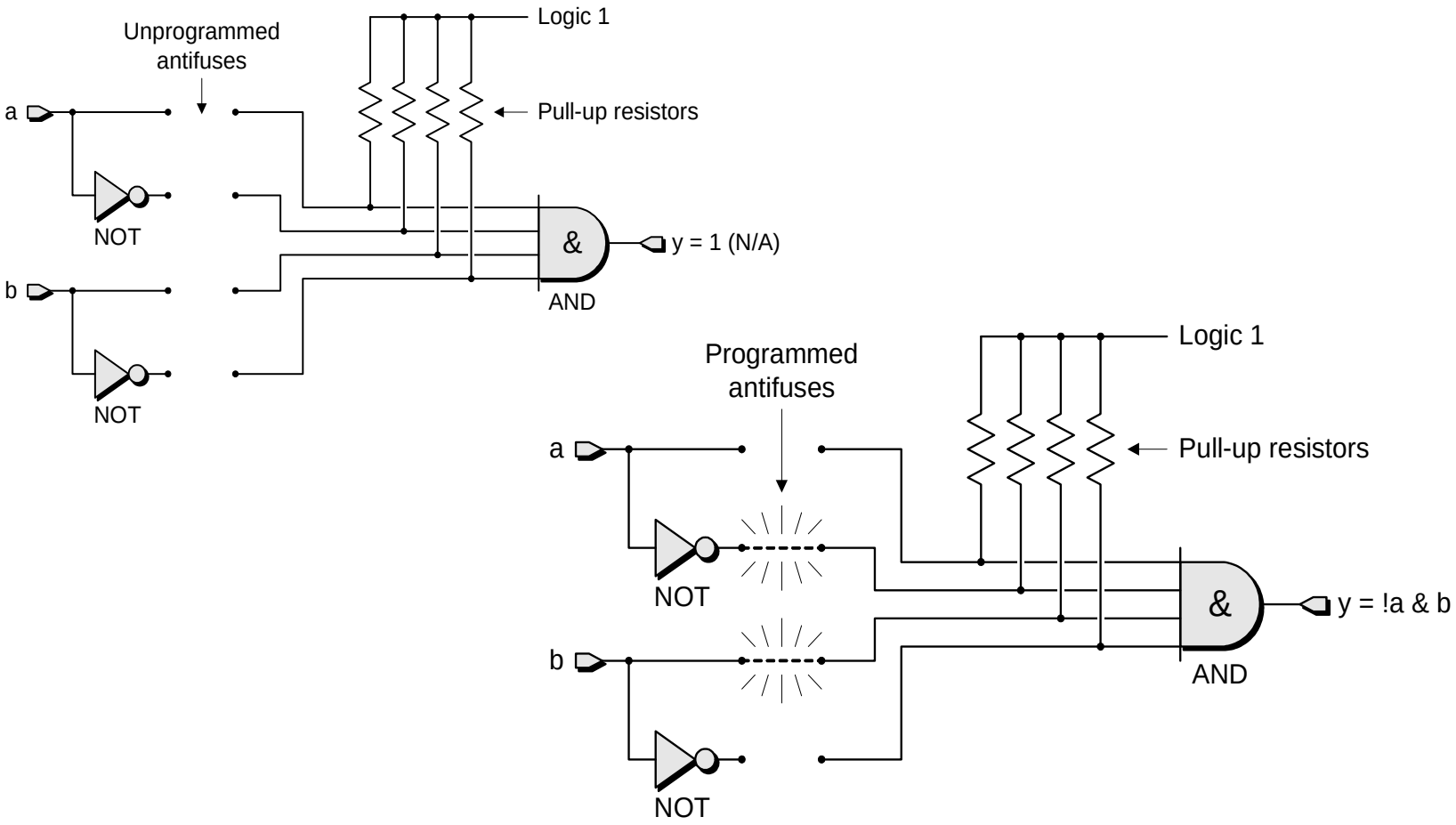
Circuit technologies

Feature	SRAM	Antifuse	E2PROM / FLASH
Technology node	State-of-the-art	One or more generations behind	One or more generations behind
Reprogrammable	Yes (in system)	No	Yes (in-system or offline)
Reprogramming speed (inc. erasing)	Fast	----	3x slower than SRAM
Volatile (must be programmed on power-up)	Yes	No	No (but can be if required)
Requires external configuration file	Yes	No	No
Good for prototyping	Yes (very good)	No	Yes (reasonable)
Instant-on	No	Yes	Yes
IP Security	Acceptable (especially when using bitstream encryption)	Very Good	Very Good
Size of configuration cell	Large (six transistors)	Very small	Medium-small (two transistors)
Power consumption	Medium	Low	Medium
Rad Hard	No	Yes	Not really

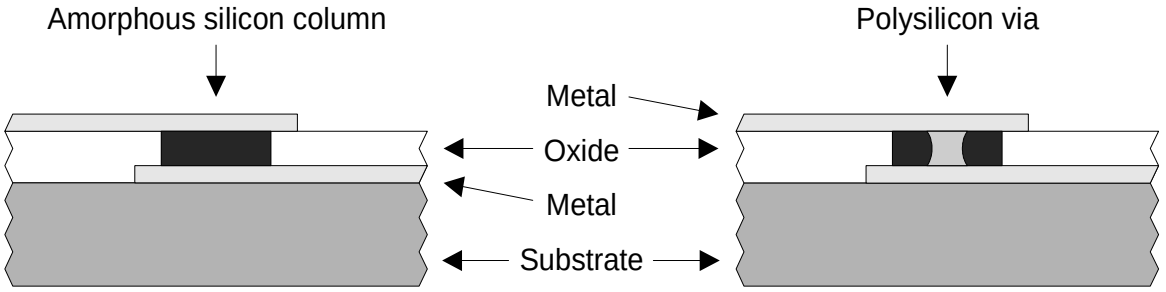
SRAM-based FPGAs

- Working principle:
  - SRAM memory inside the FPGA saves the configuration
- Advantages:
  - Can be programmed an unlimited number of times
  - Space for a lot of logic
  - Can easily change the functionality of the system
  - Does not need a special programmer or process
- Disadvantages:
  - Takes a lot of space (SRAM-cell with 5 transistors)
  - Volatile memory (configuration has to be saved externally)
  - Relatively high power usage
- We use SRAM-based FPGAs from Xilinx, which has SRAM-based FPGAs with the Artix, Kintex, Virtex and Zynq families. Intel (Altera) has similar FPGAs in their FPGA families.

Antifuse

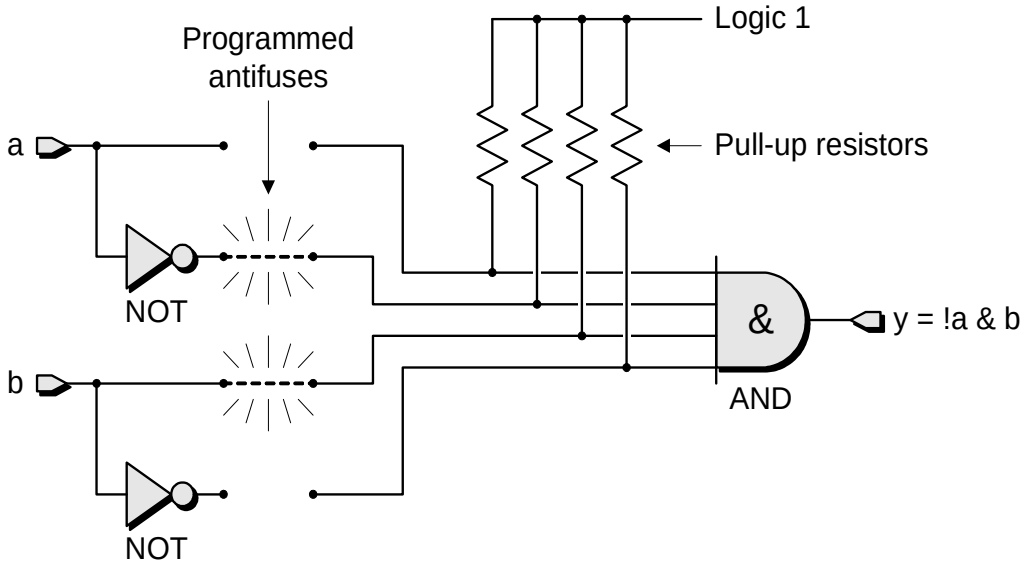

Unprogrammed
antifuses
Logic 1
Pull-up resistors
a
NOT
b
NOT
&
AND
y = 1 (N/A)
Programmed
antifuses
Logic 1
Pull-up resistors
a
NOT
b
NOT
&
AND
y = !a & b

Antifuse



(a) Before programming

(b) After programming

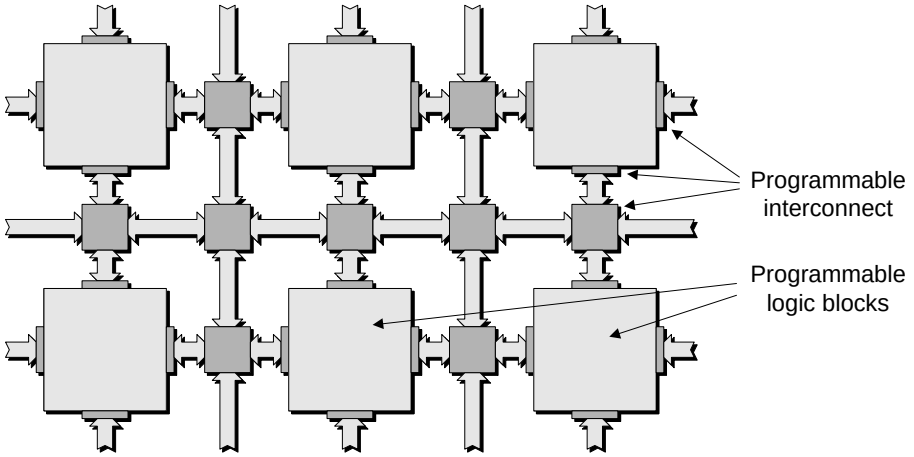


Antifuse

- Working principle:
  - Configuration is saved in the FPGA by making shorts using high voltage
- Advantages
  - Low impedance when fuse is “on” (small delay)
  - Low power usage
  - Compact technology (low space requirements)
  - Extra robust technology (high radiation resistance)
- Disadvantages
  - Has to be programmed with dedicated programmer
  - High programming voltage and power
  - Permanent programming (**one-time** programming)

Logic block complexity in FPGA

- Fine grained:
  - The blocks can be fully utilized in the design, but requires large routing resources
- Coarse grained:
  - A block can implement any arbitrary function (LUT), but the resources can often not be fully exploited



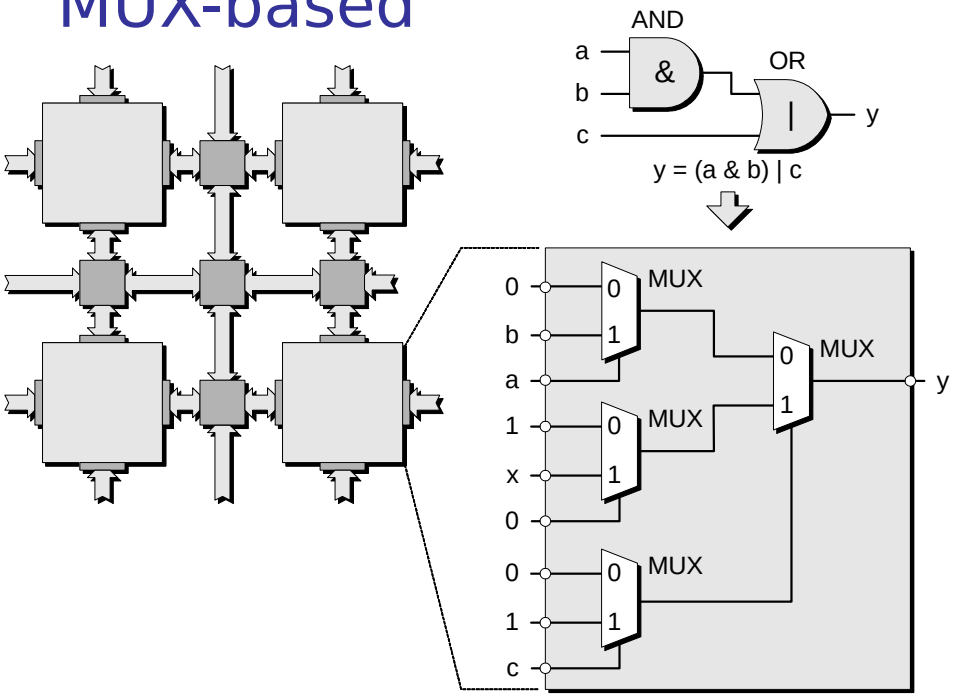
Coarse grained logic block

- The complexity of the coarse grained logic block increases with technological progress
- Example of a traditional coarse grained logic block:
  - Four 4-input LUT for combinatorial logic
  - Four multiplexers
  - 4 D-flip flops
  - Carry logic for efficient arithmetic (+ and -)

Example implementation in coarse grained logic block

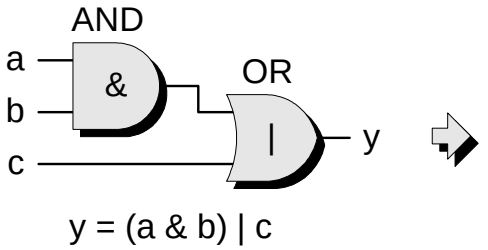
- Implementation of the function y = (a AND b) OR c

MUX-based



LUT-based

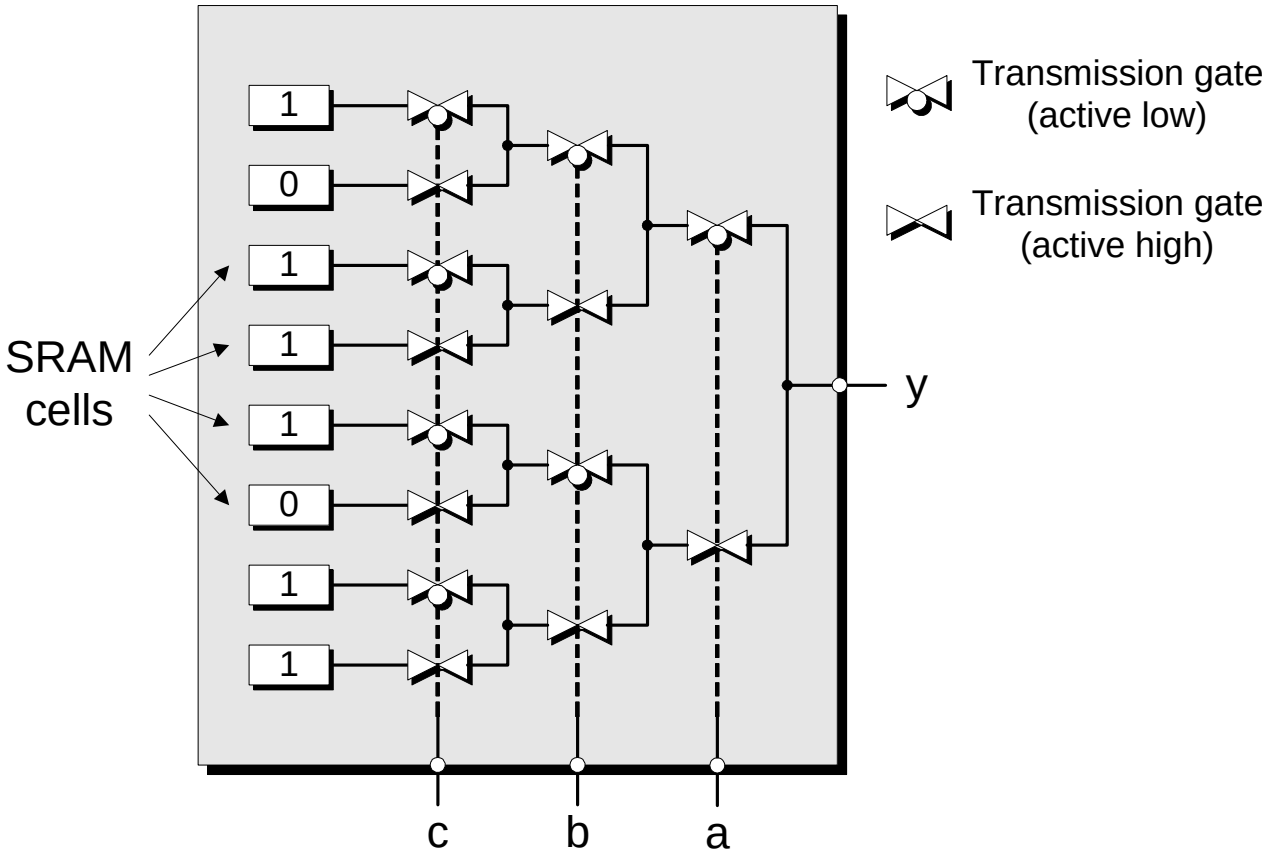
Required function



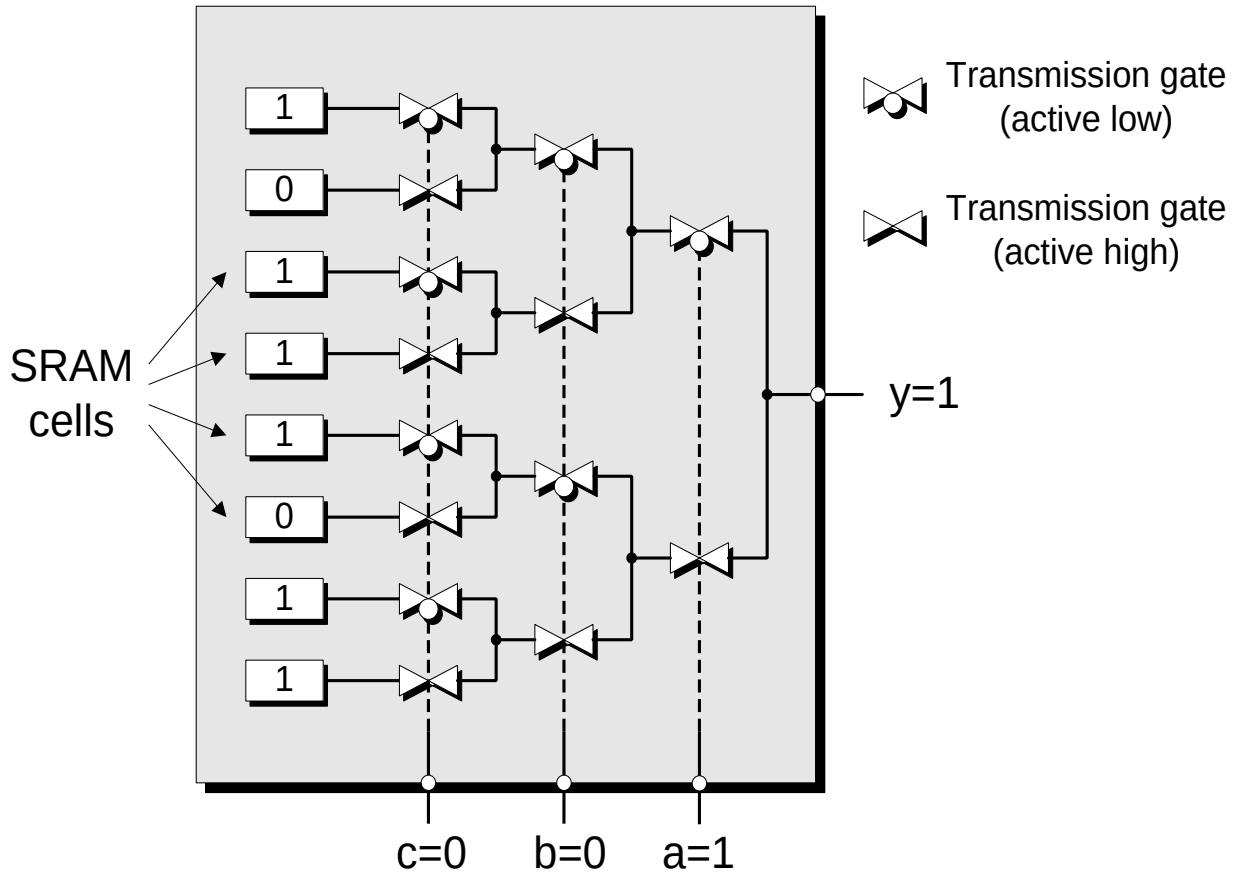
Truth table

a	b	c	y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

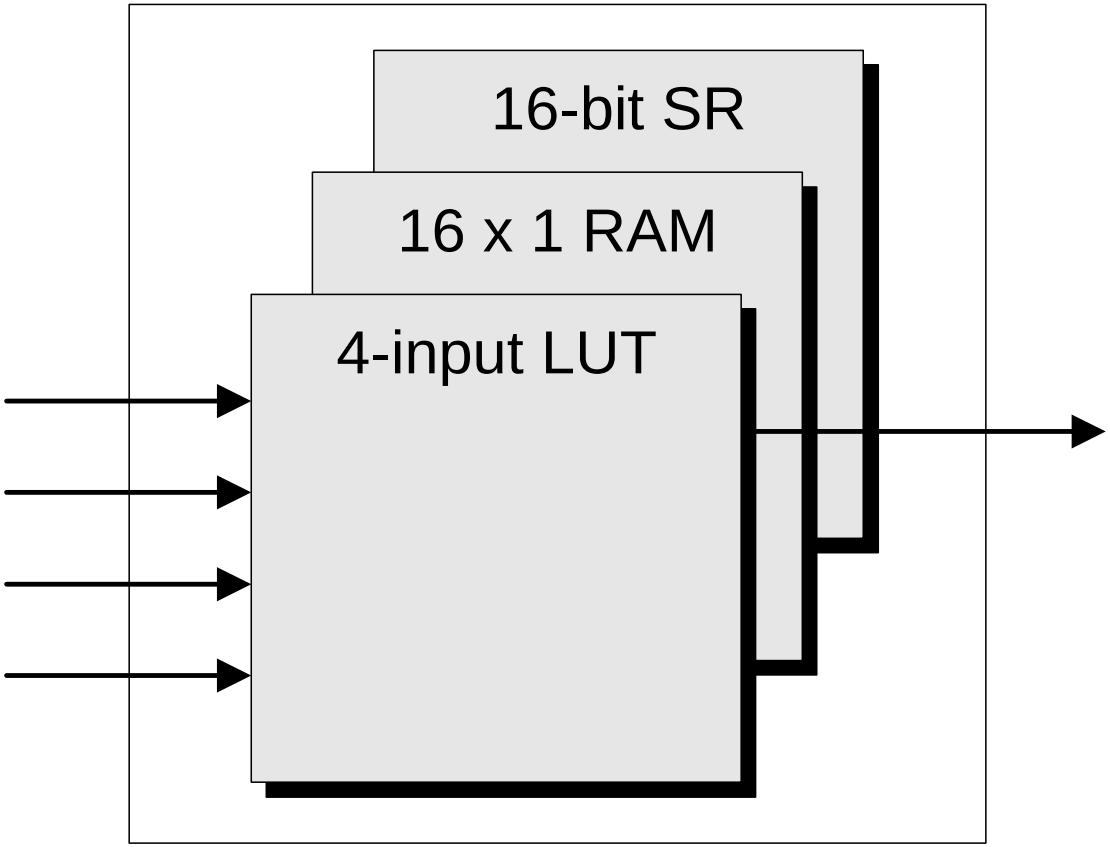
Typical LUT implementation



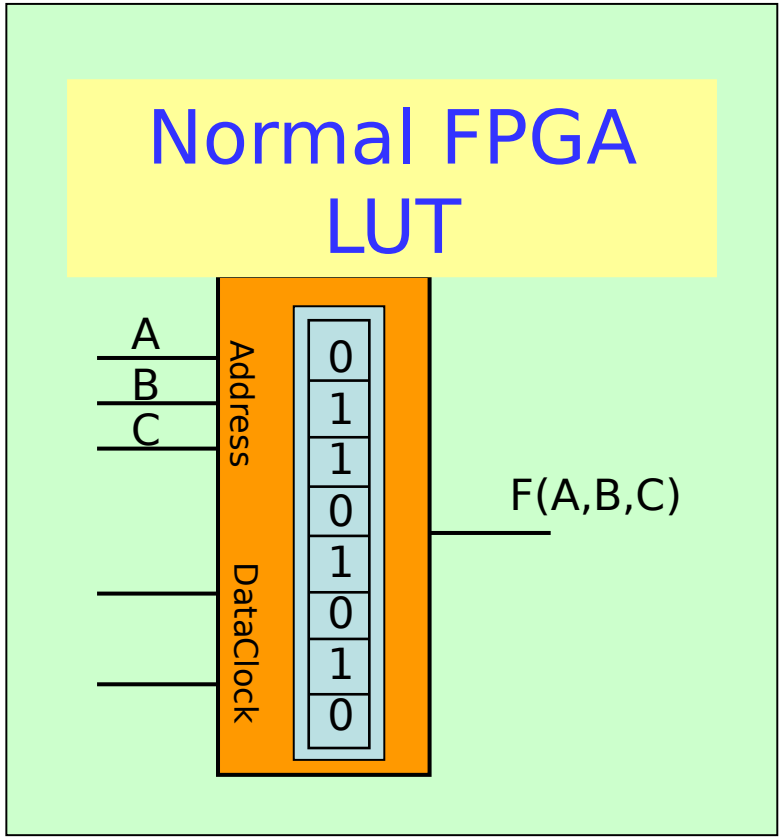
Typical LUT implementation cont.

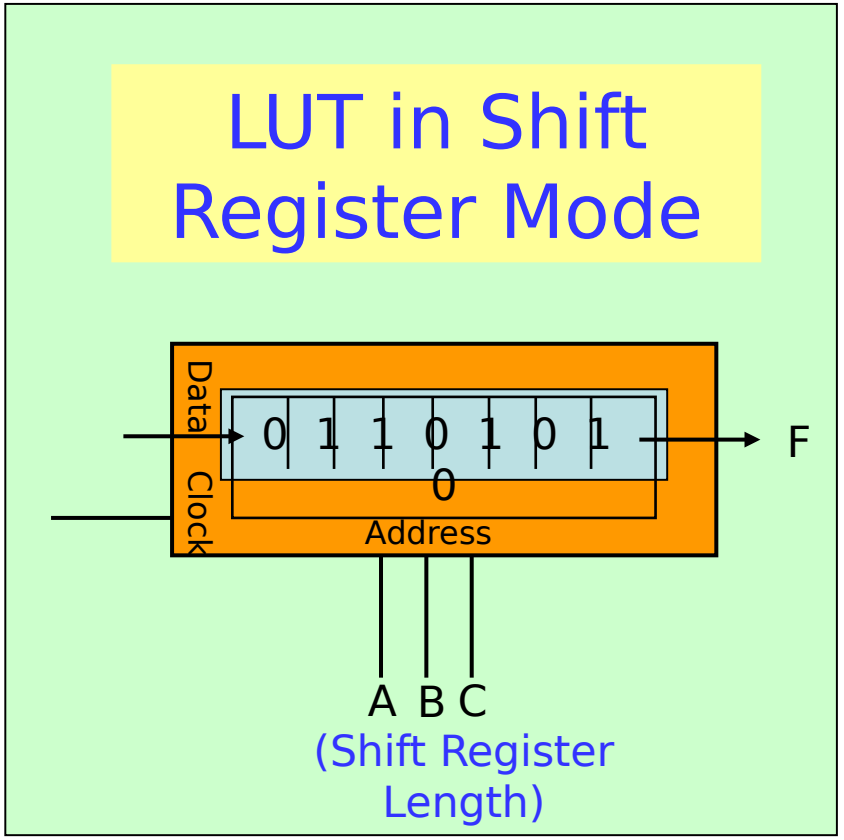


Different uses of LUTs

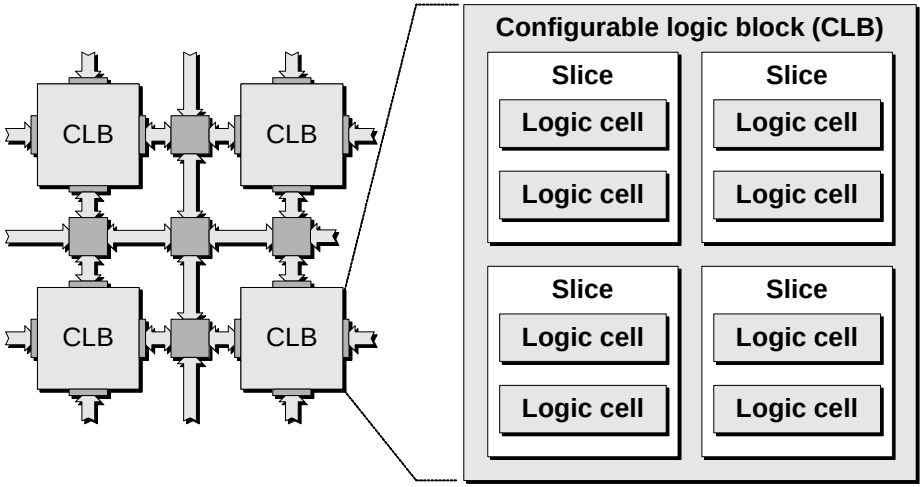


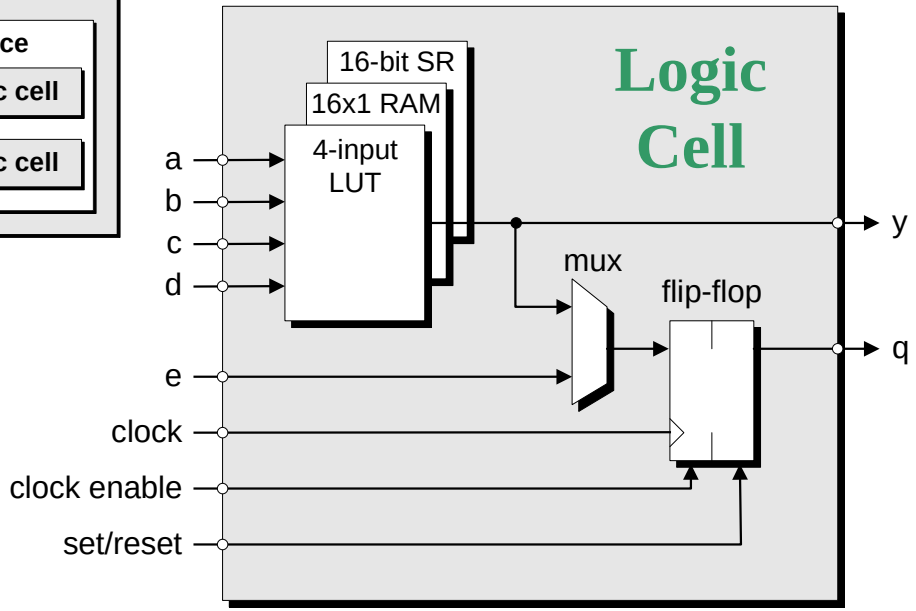
FPGA LUTs


Normal FPGA LUT
A
B
C
Address
DataClock
0
1
1
0
1
0
1
0
F(A,B,C)


LUT in Shift Register Mode
Data
Clock
0 1 1 0 1 0 1
0
Address
F
A B C
(Shift Register Length)

Xilinx terminology



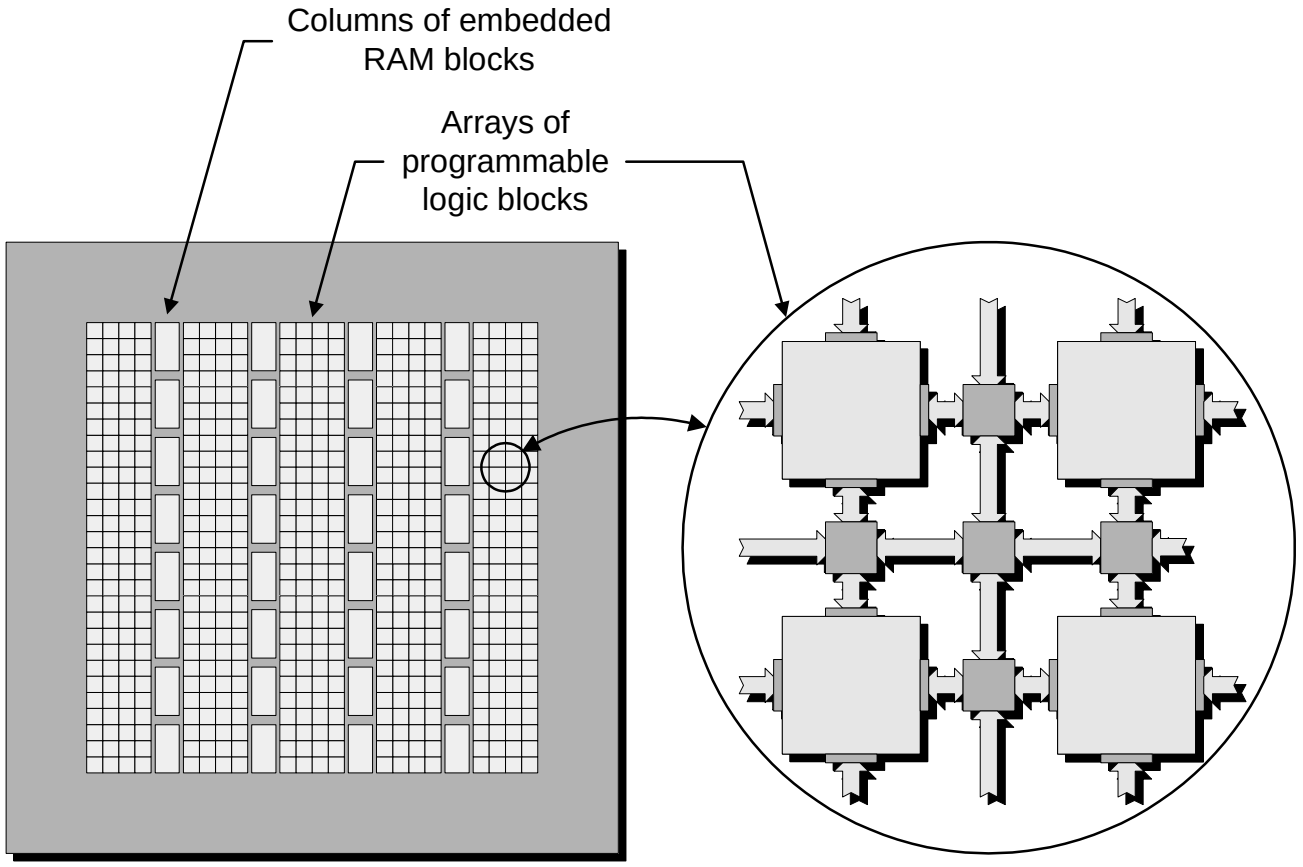


Additional functions in modern FPGAs

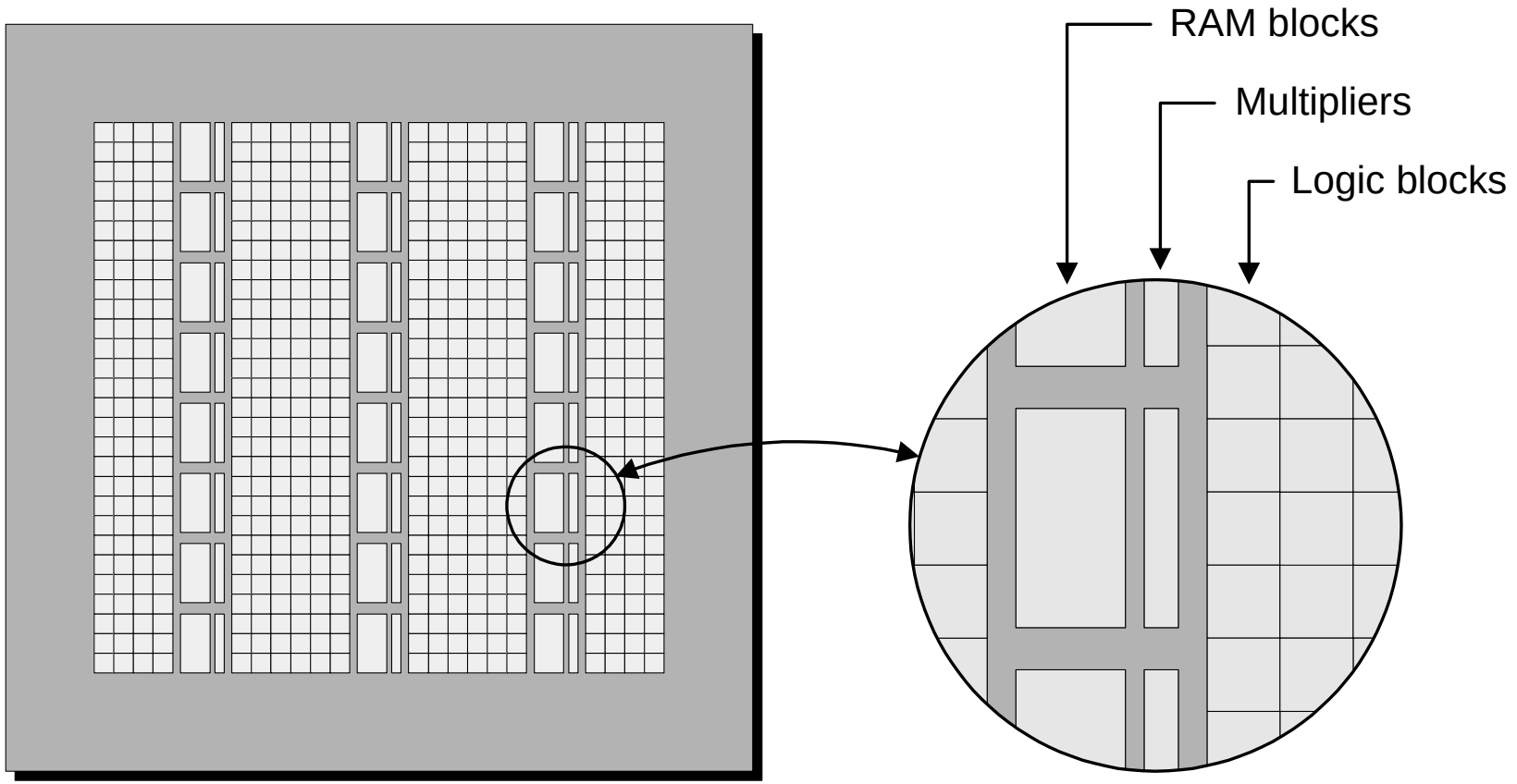
- Clock management and distribution
- Carry chains for fast arithmetic (+ and -)
- RAM blocks (in addition to distributed RAM with LUTs or through external memory interfaces)
- Function blocks (multipliers, DSP functions like multiply & accumulate, Ethernet MAC, PCI-E, etc.)
- Processor cores (ARM, RISC-V, MicroBlaze or other types)
- High speed serial transceivers

- This is in addition to LUTs and registers which should be utilized as efficiently as possible!

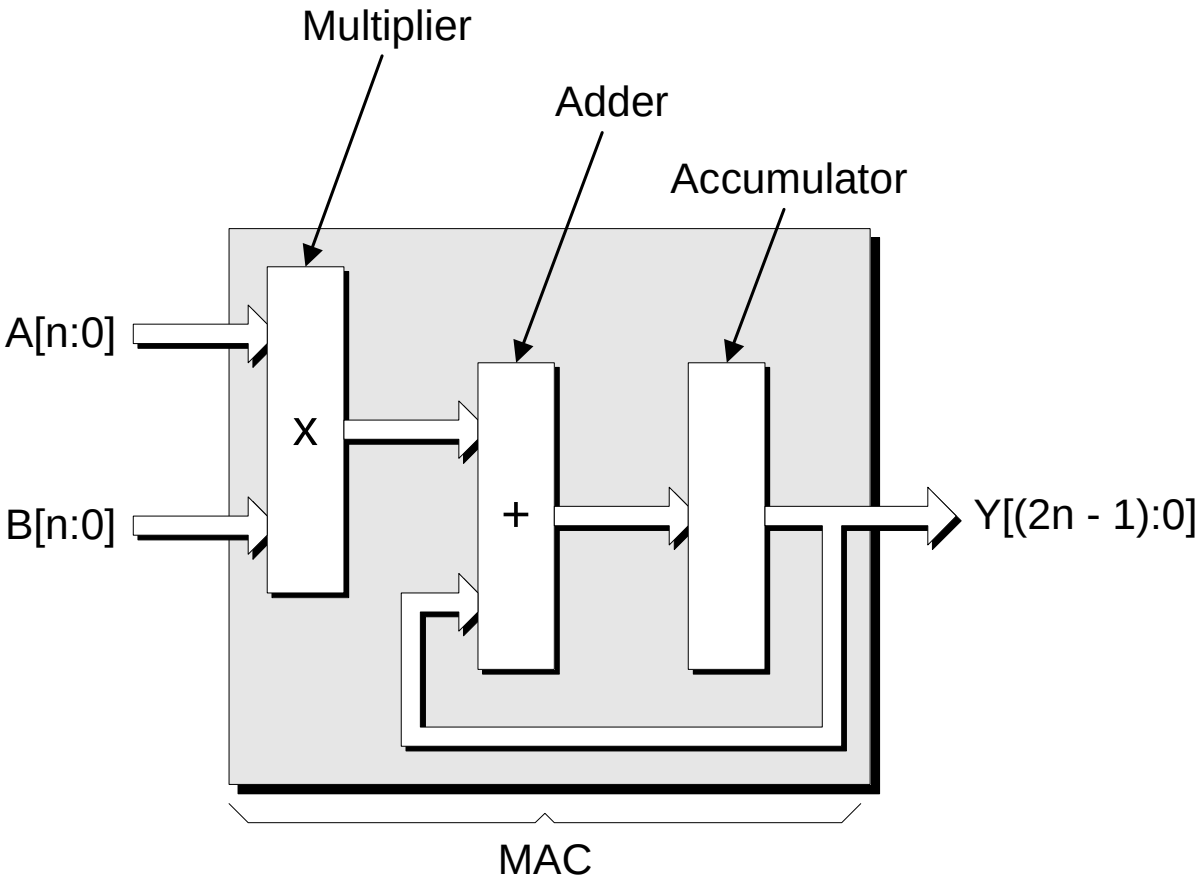
RAM blocks (Block RAM / BRAM)



Function blocks

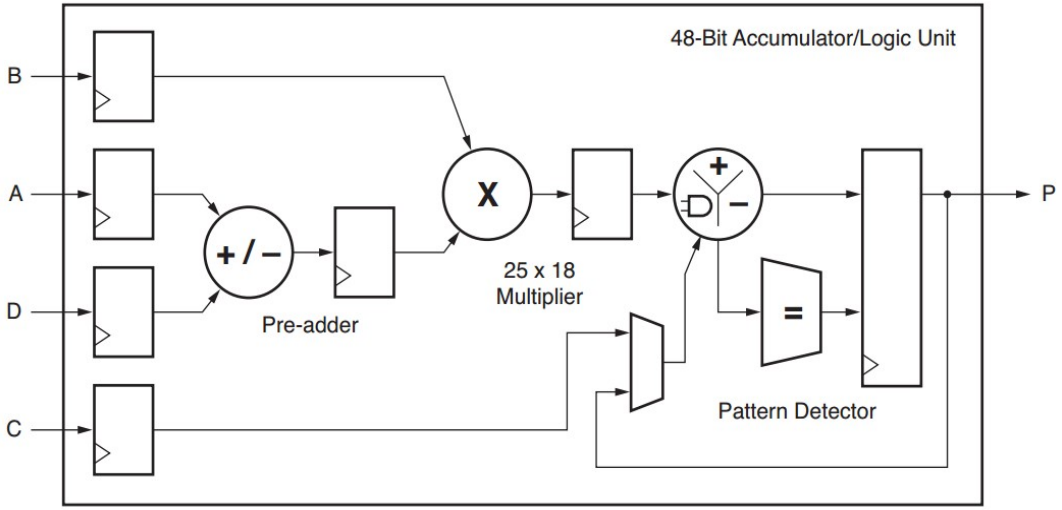


Multiply-and-accumulate (MAC)



DSPs (DSP48E1)

- Contains functionality for different arithmetic functions:
  - 25*18 two's-complement multiplier
  - 48-bit accumulator
  - 25-bit pre-adder
  - Dual 24-bit add/ subtract/accumulate
  - Pattern detector
  - Pipelining and cascading buses

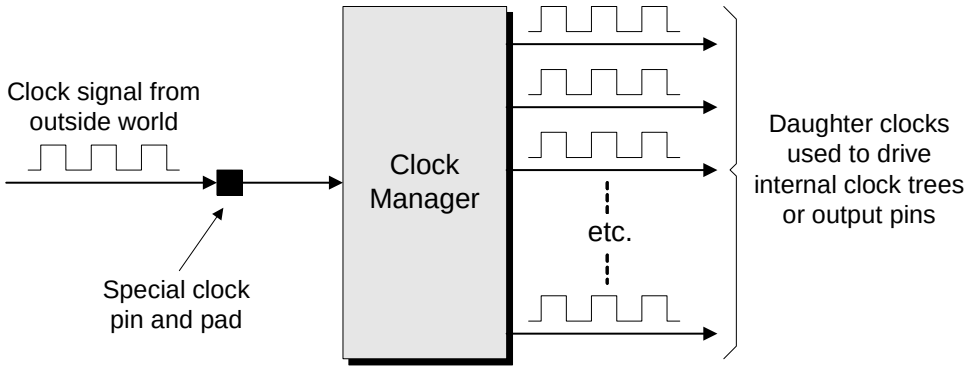


Processor cores

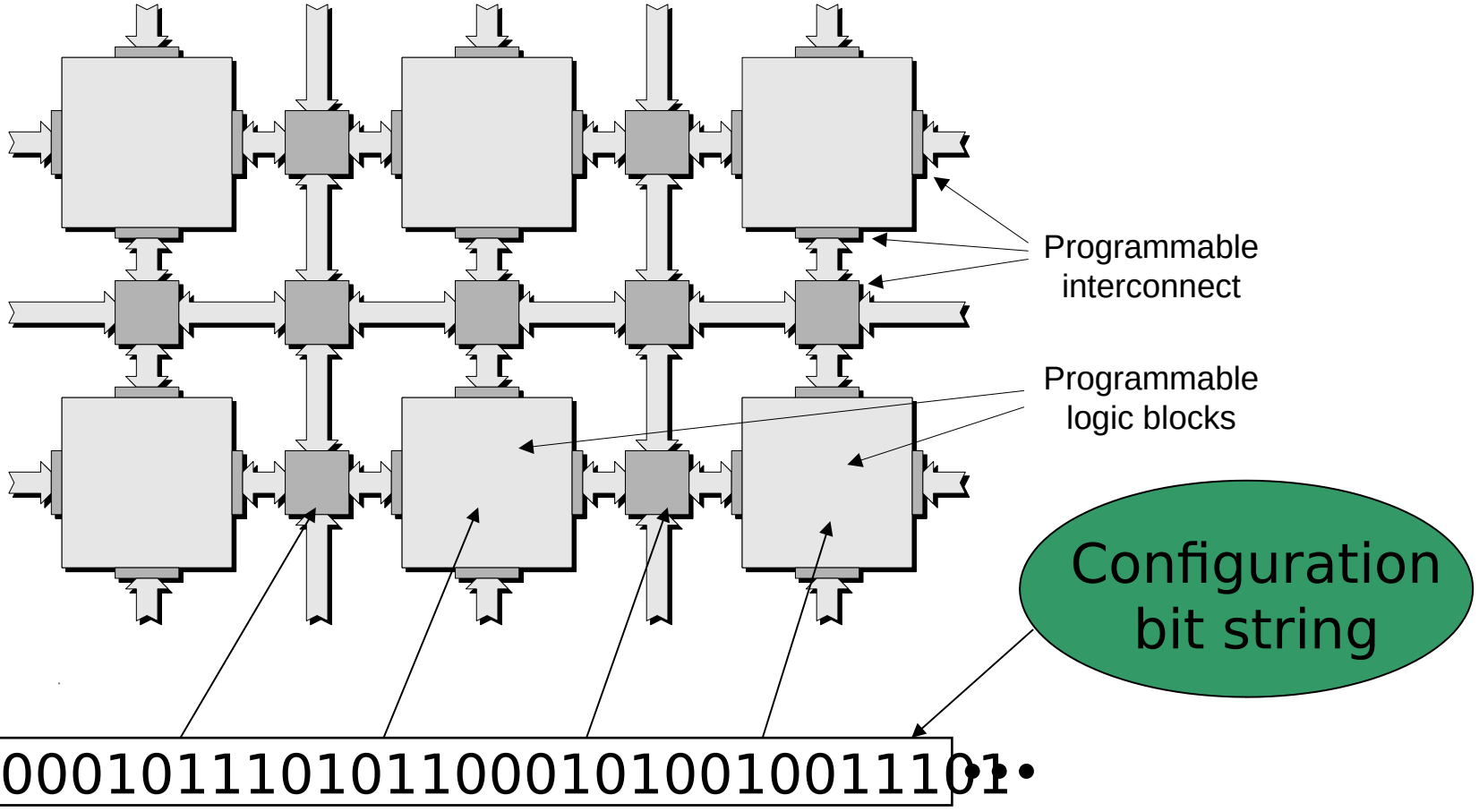
- Processor cores can be integrated with the FPGA logic
- Many designs require a processor, and adding it to the FPGA can remove the need for an external CPU
- Two different types
  - Soft core
    - Programmable logic is used in the FPGA to implement a simple microprocessor that interacts directly with the rest of the FPGA
    - Examples are Xilinx MicroBlaze, Intel (Altera) Nios II and RISC-V
  - Hard core
    - The processor is implemented on the silicon with the FPGA chip at production
    - High speed interfaces are used between the CPU and the FPGA fabric
    - Xilinx Zynq 7000 family have a dual core ARM Cortex-A9 (up to 1.0 GHz)
    - Xilinx Zync Ultrascale+ family have a quad core ARM Cortex-A53 (up to 1.5GHz), dual core ARM Cortex-R5 (up to 600MHz) and a Mali-400 MP2 graphic processor

Clock generation and distribution

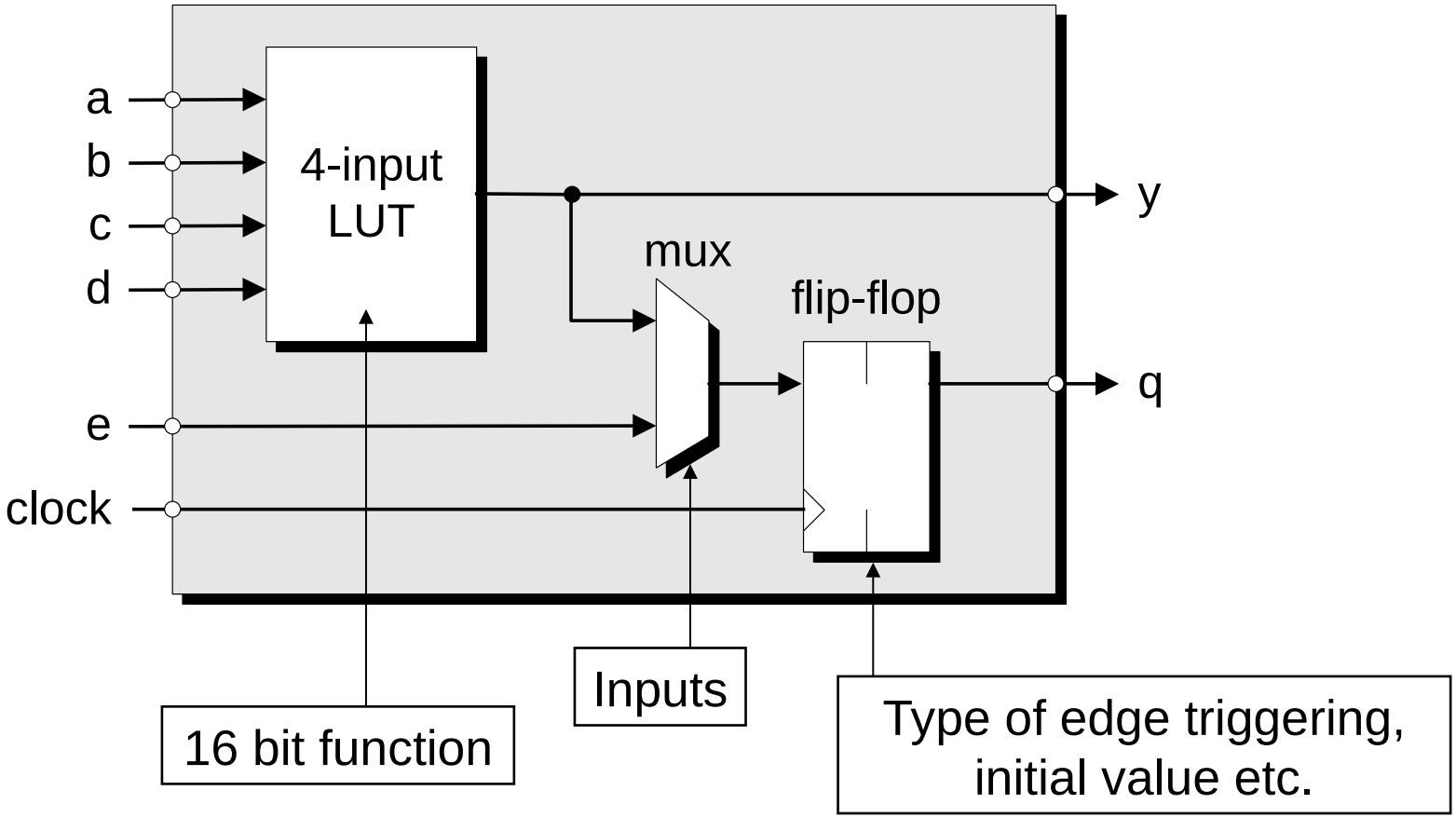
- Global clock trees distribute clock signals to synchronous elements across the device
  - Ensures that the edges of the clock signal arrives at almost the same time across the device (setup/hold time)
- A global clock is often generated externally, and an internal clock manager generates a number of derived clocks



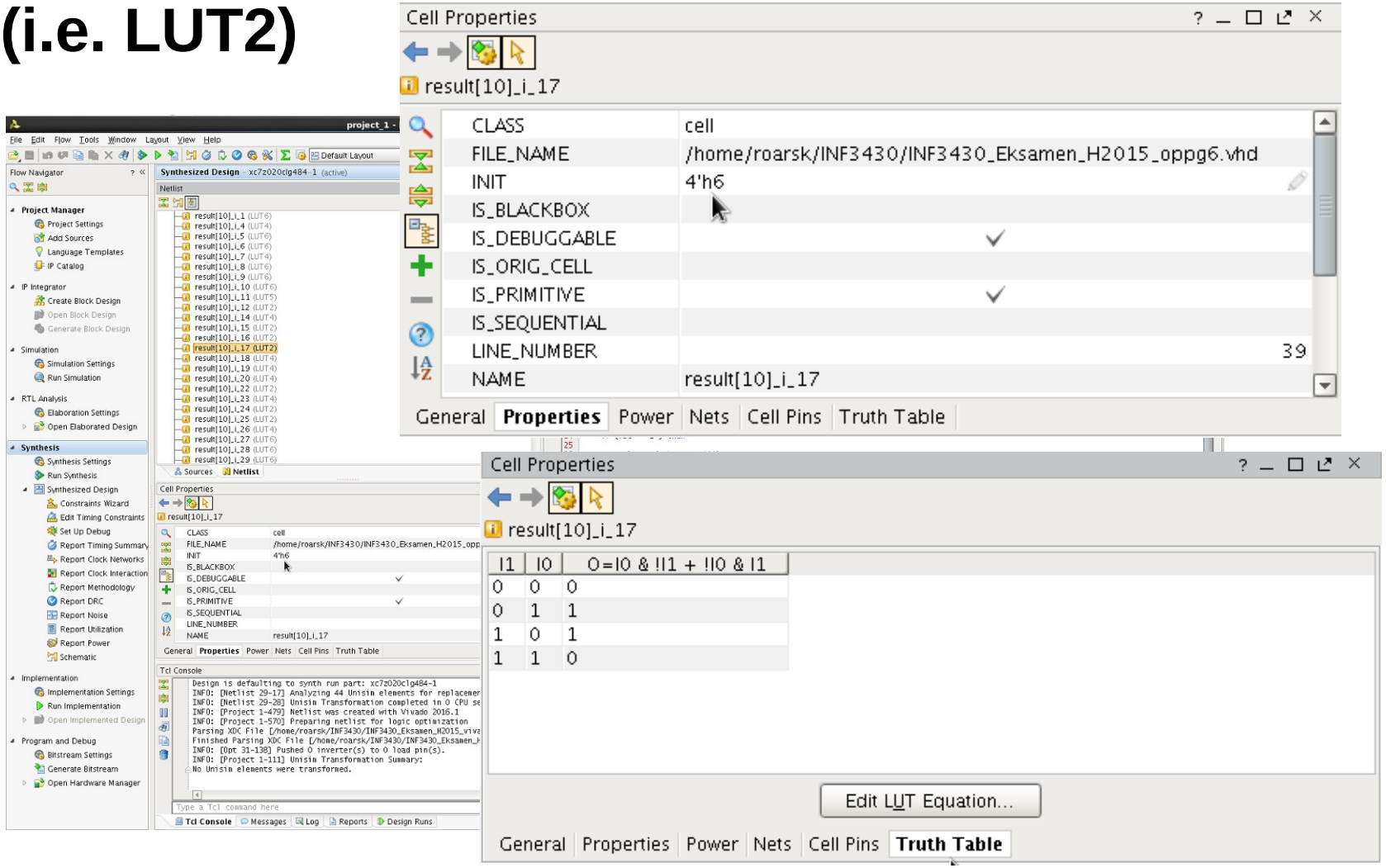
Configuration of the FPGA



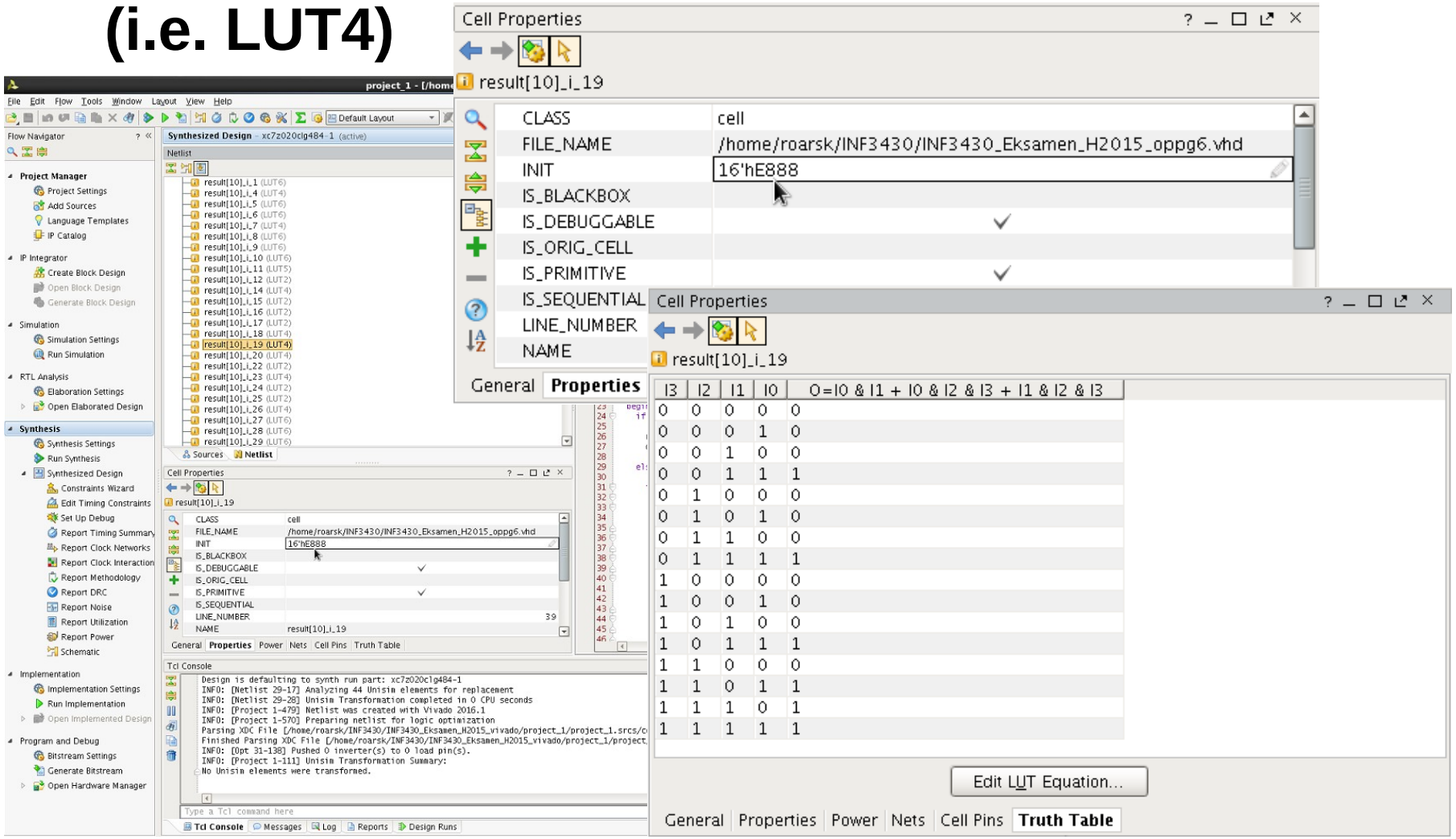
What do you need to configure?



Xilinx Vivado example with two input LUT (i.e. LUT2)



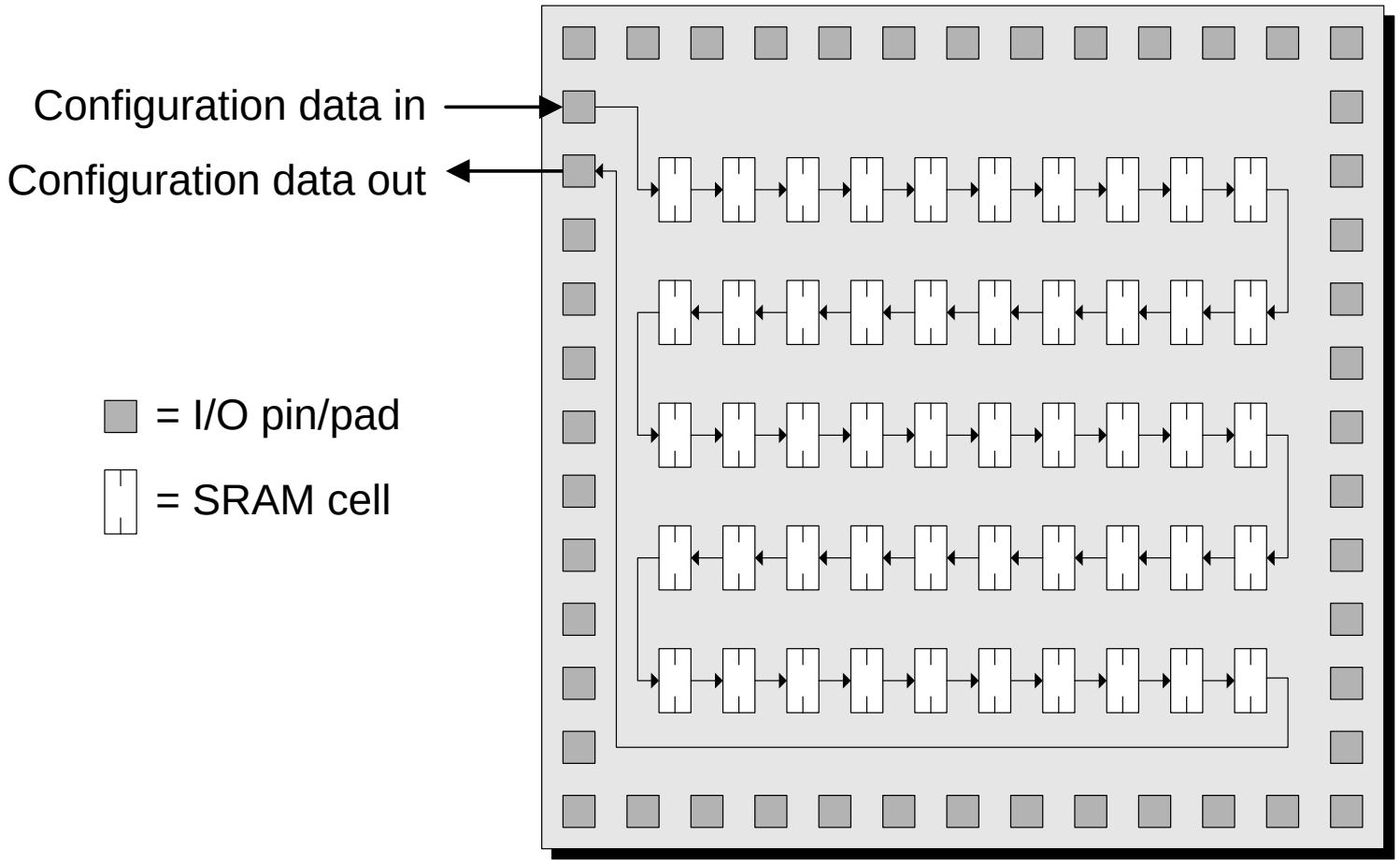
Xilinx Vivado example with four input LUT (i.e. LUT4)



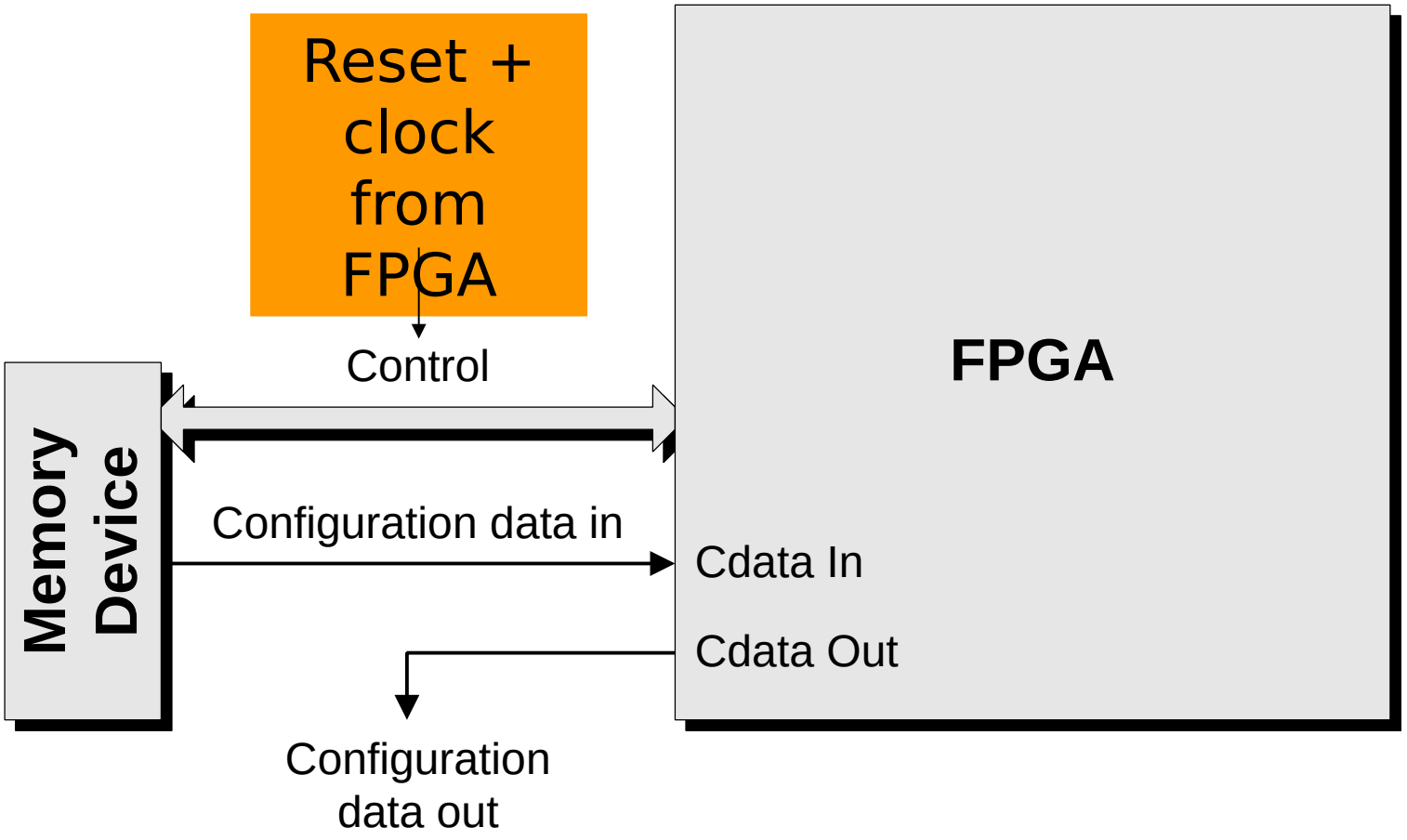
Configuration methods

- Configuration methods
  - Serial load with FPGA as master
  - Serial load with FPGA as slave
  - Parallel load with FPGA as master
  - Parallel load with FPGA as slave
- JTAG can also be used

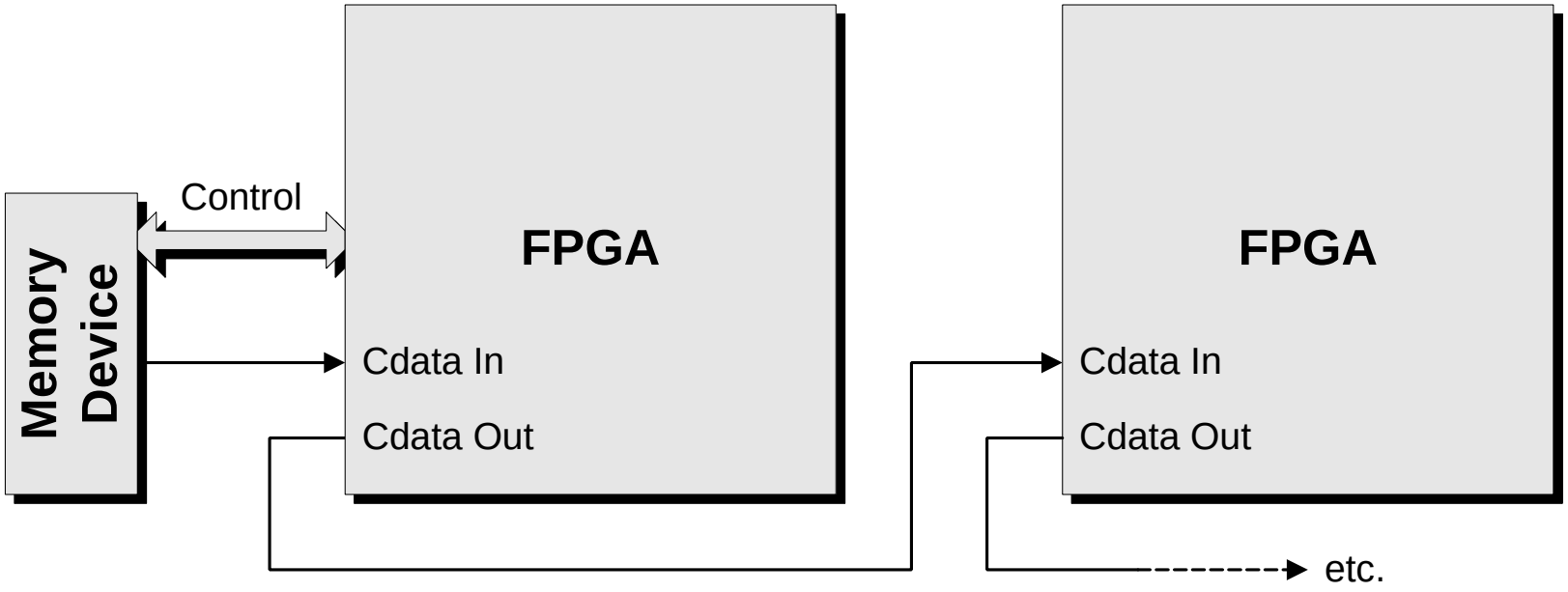
Serial download



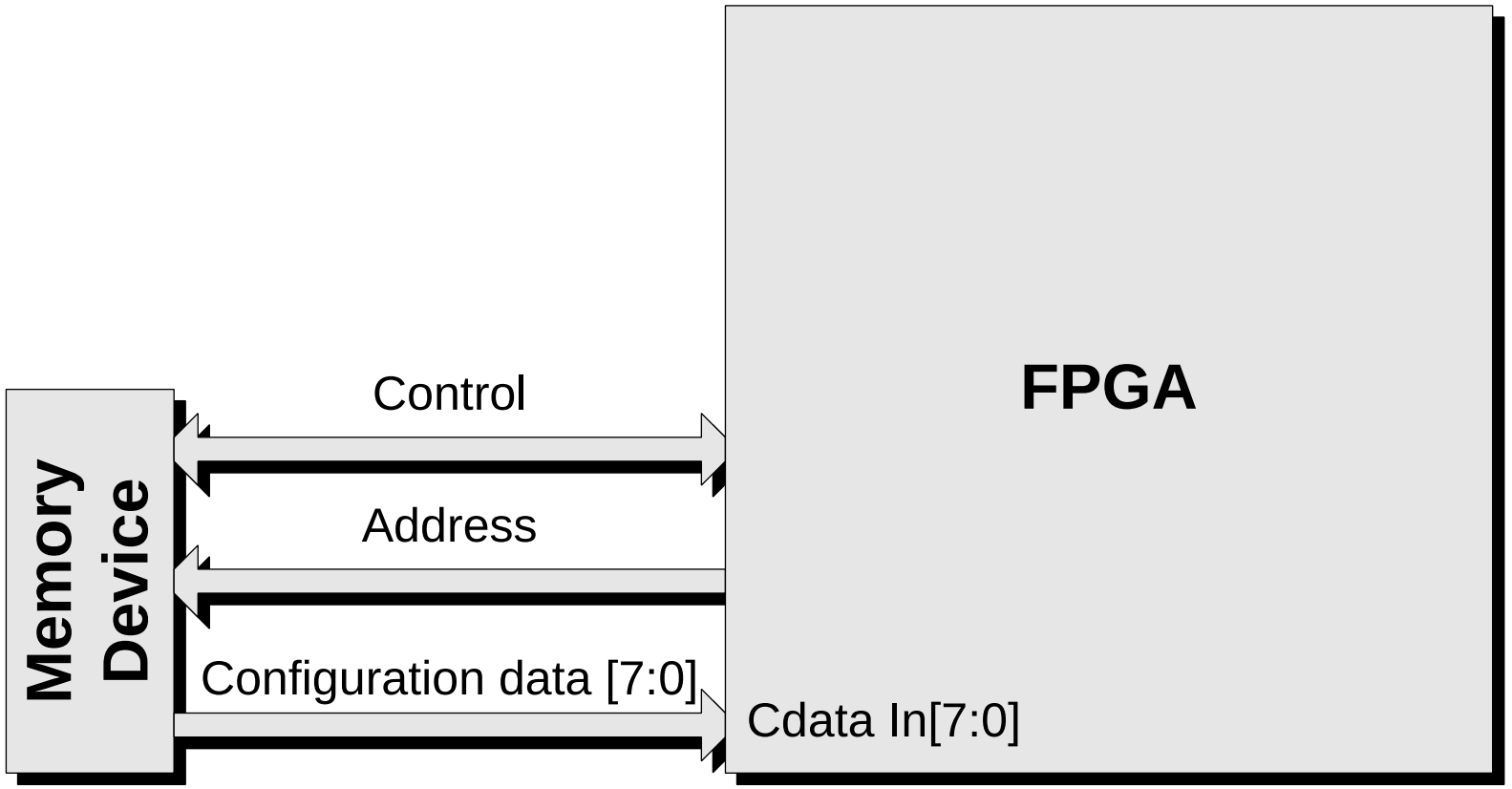
Serial download with FPGA as master



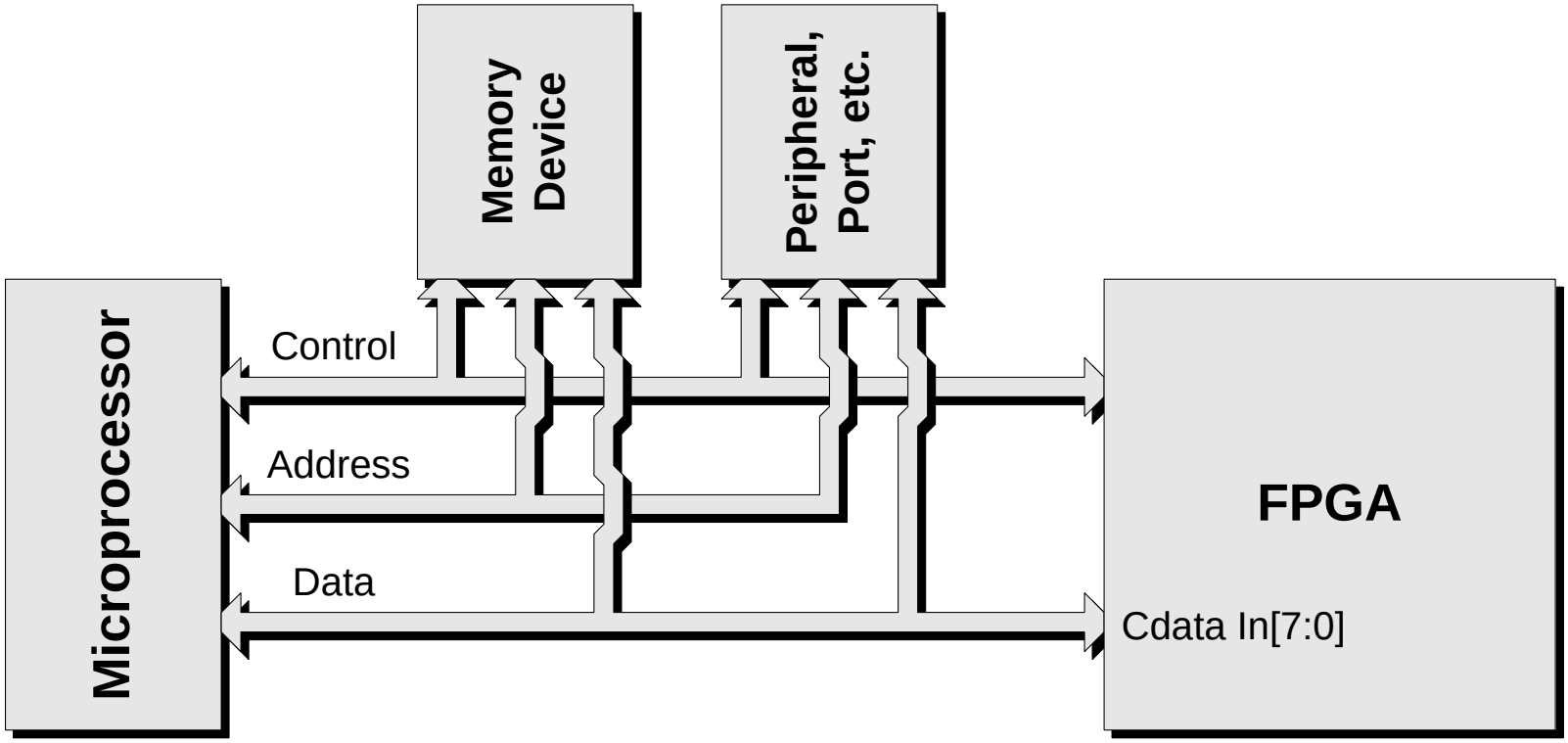
Daisy chaining FPGAs



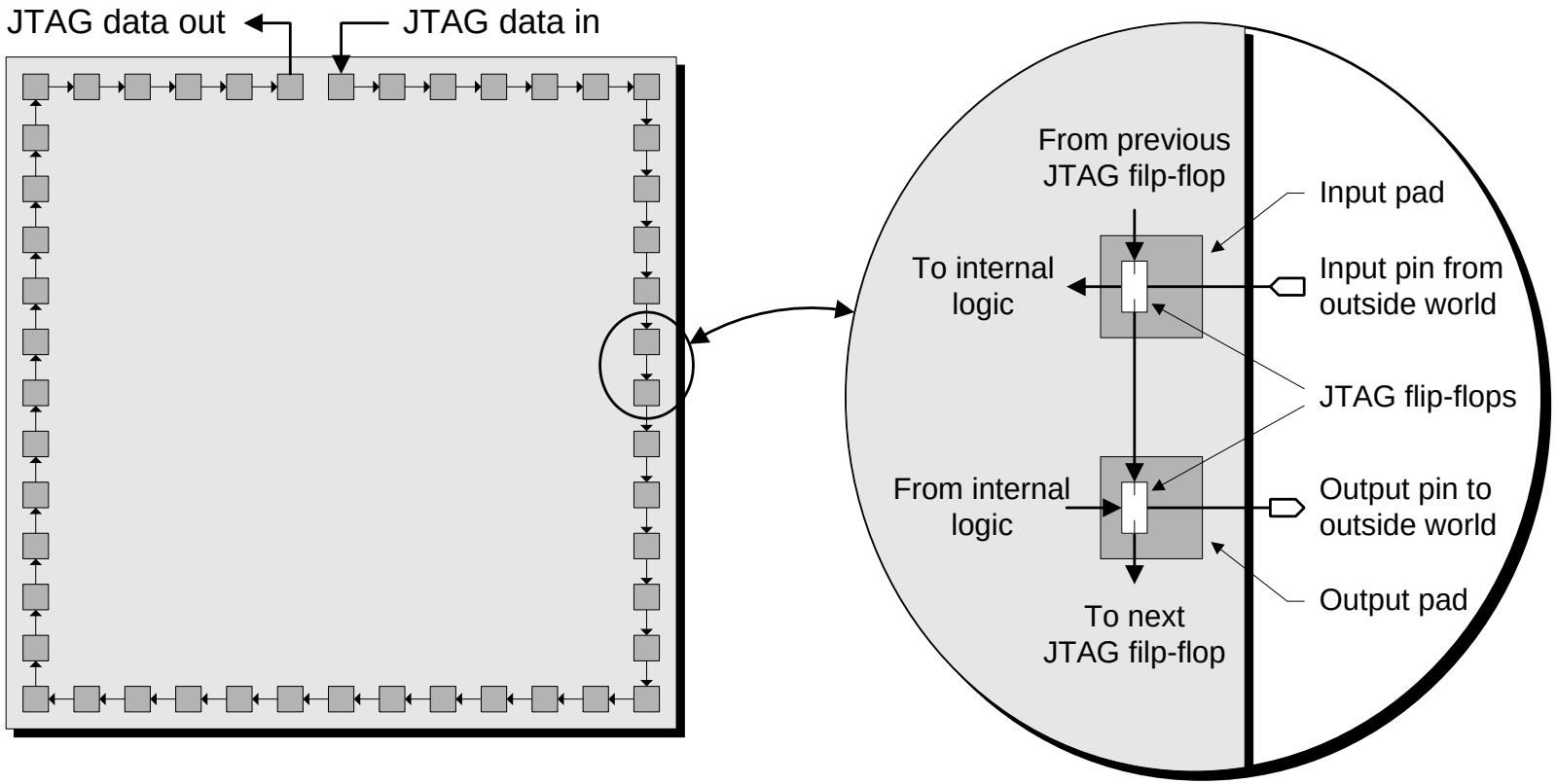
Parallel download with FPGA as master



Parallel download with FPGA as slave



Download using the JTAG port



Case: Mars Rover

JPL and its contractors eventually came up with a system of redundant avionics hardware implemented on about 1.2 million logic gates. That allows the rover's avionics to interface with all major scientific instruments, sensors and comms links along with the rover's drive train while also managing power in wake, sleep and dream modes.

Among NASA's Curiosity avionics contractors are Wind River (VxWorks real-time operating system) and **Microsemi Semiconductor** (**RTAX-S and RTSX-SU FPGAs**, high- and low-voltage power supplies, high reliability diodes and signal and power transistors).

WASHINGTON – The Curiosity rover now gearing up to explore a 96-mile-wide crater on Mars is by far the most complex machine ever to explore the surface of another planet. One big reason is the rover's avionics, which control everything from its 10 scientific instruments to communications, navigation, cameras and power management – which is where Curiosity's "dream mode" enters the picture.

Dream mode "is sort of the reptile brain for the rover," explained Jim Donaldson, the Mars Science Laboratory avionics chief engineer. Implemented in FPGAs, the rover's dream mode function monitors vital rover systems while its redundant main computers are in "sleep mode" to save power.

Donaldson said the biggest challenge engineers at NASA's Jet Propulsion Laboratory (JPL) faced in developing rover avionics was development and implementation of the FPGAs that have provided Curiosity with a quantum leap in functionality as it explores the Red Planet. From an engineering standpoint, Donaldson said the biggest challenge was scaling JPL's FPGA design practices to achieve the higher levels of complexity needed to put a largely autonomous rover inside Gale Crater, which is believed to harbor the conditions needed for microbial life.