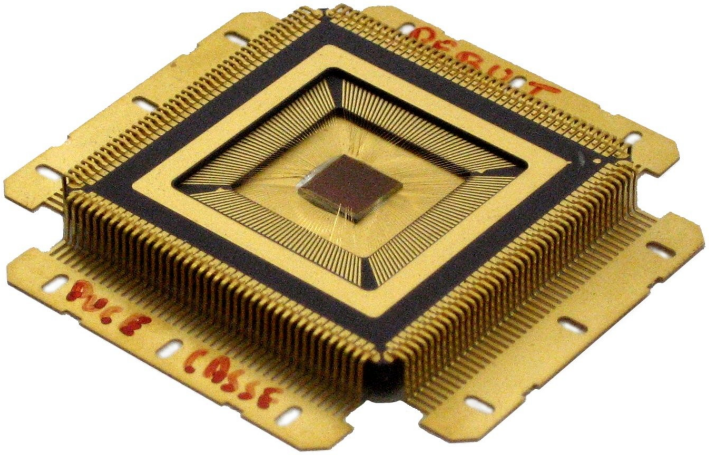
Evolving Digital Hardware
EHW Module 2008



Andy Greensted
Department of Electronics
ajg112@ohm.york.ac.uk

www.bioinspired.com/users/ajg112/teaching/evoHW.shtml

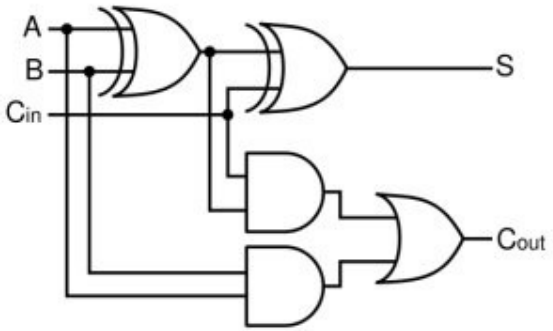
Lecture 2

Lab Overview

Over 4 lab sessions you will implement two evolutionary digital hardware systems.

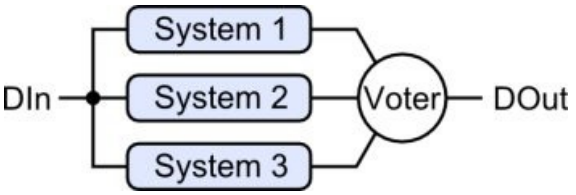
Lab 1

- Evolution of 1 and 2 bit adder circuits
- Explore using a VRC (Virtual Reconfigurable Circuit)
- Explore different evolutionary techniques



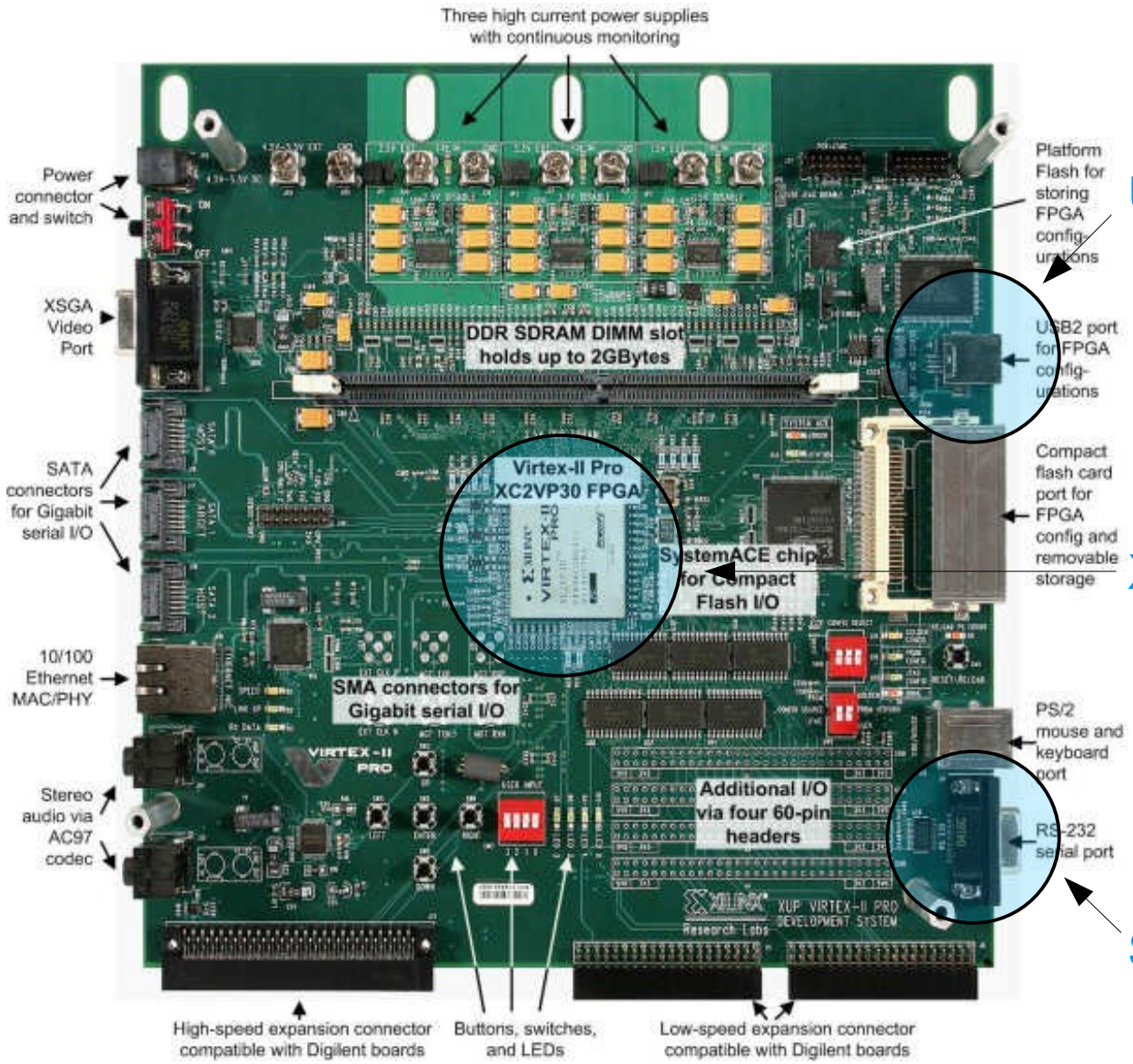
Lab 2

- 4 bit Parity Generator
- Continuous Evolution for Fault Tolerance



The FPGA Development Board

Digilent XUP-V2Pro (Xilinx University Program Virtex 2 Pro)

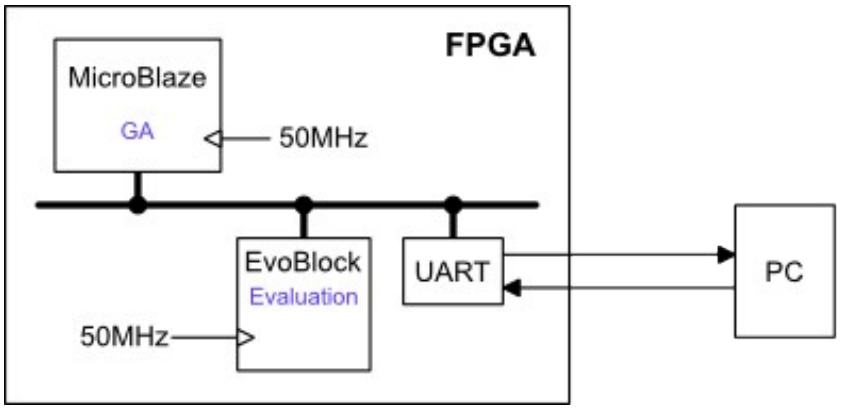


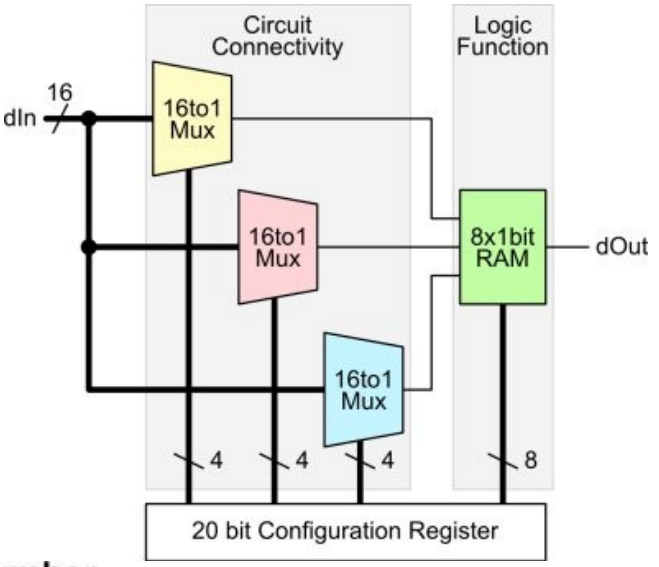
USB Configuration Port

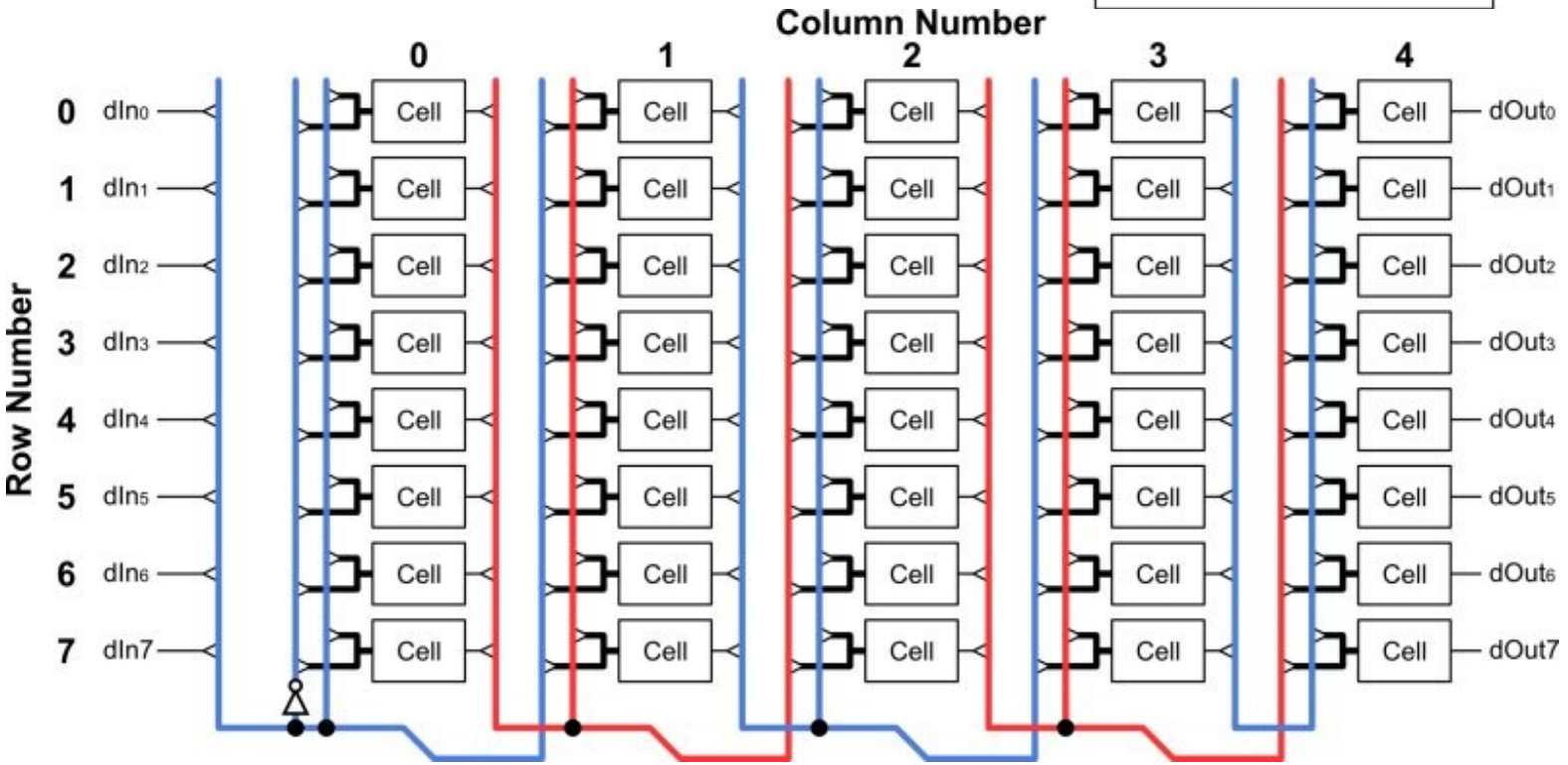
Xilinx FPGA (Virtex2)

Serial Port

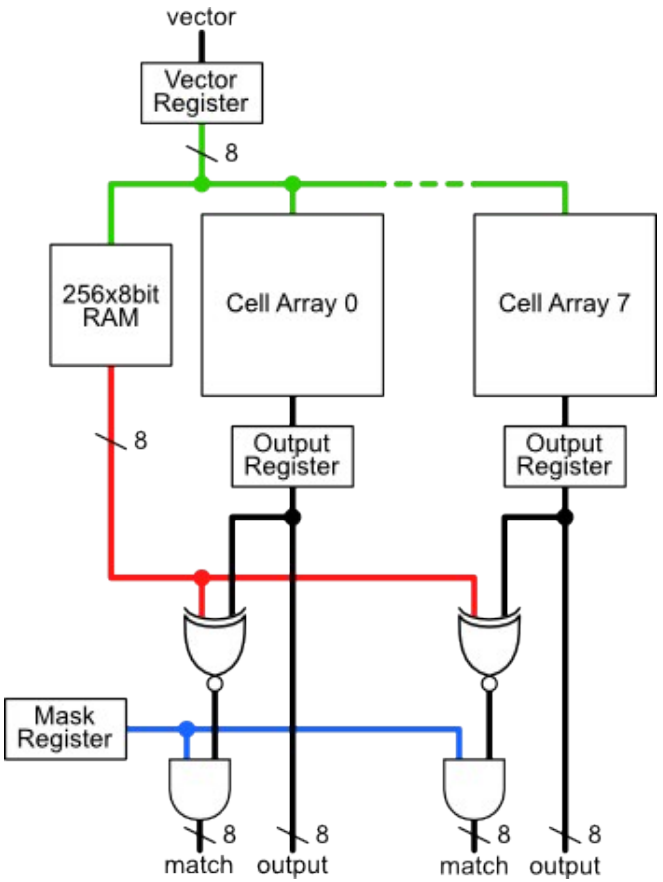
Lab 1 (Session 1 & 2)

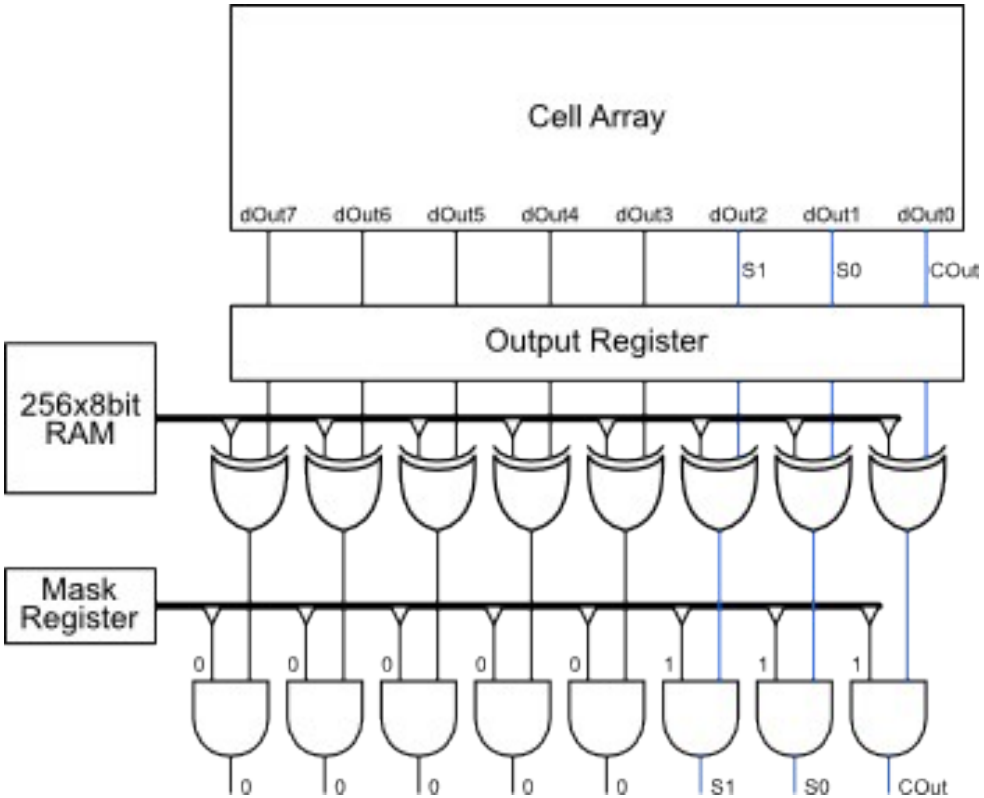




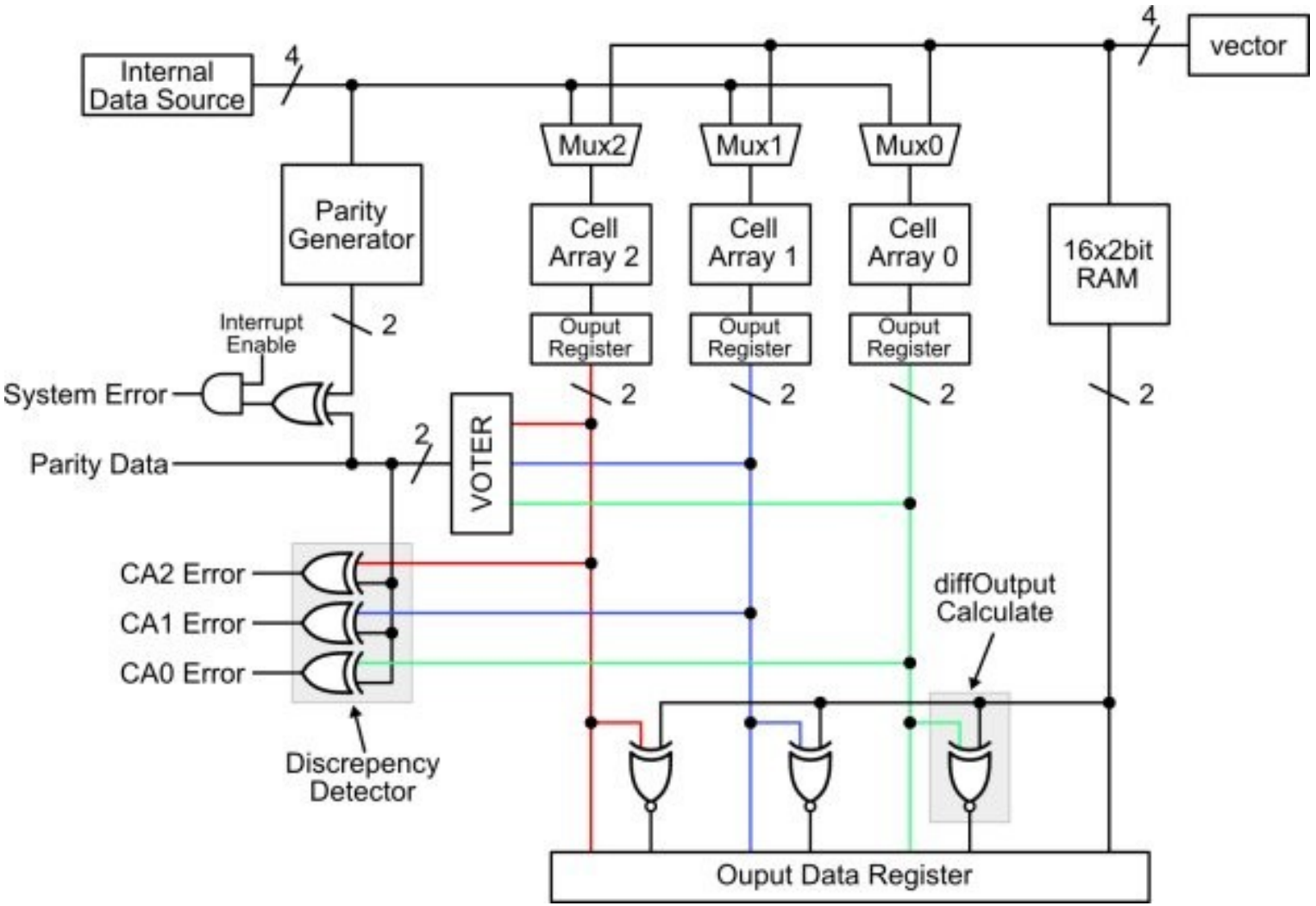


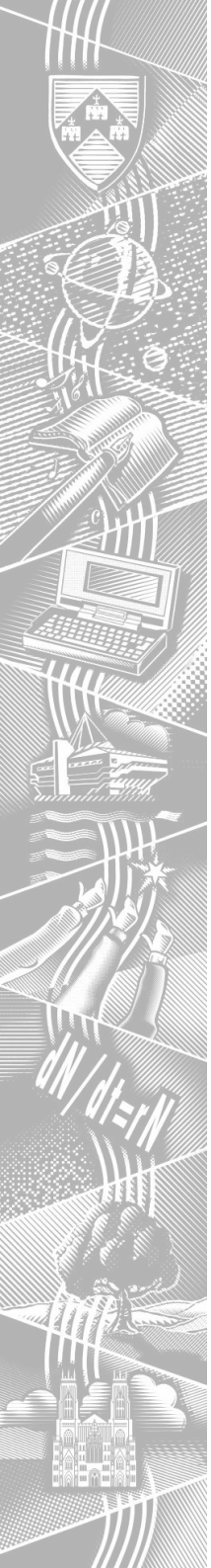
Lab 1 (Session 1 & 2)





Lab 2 (Session 3 & 4)



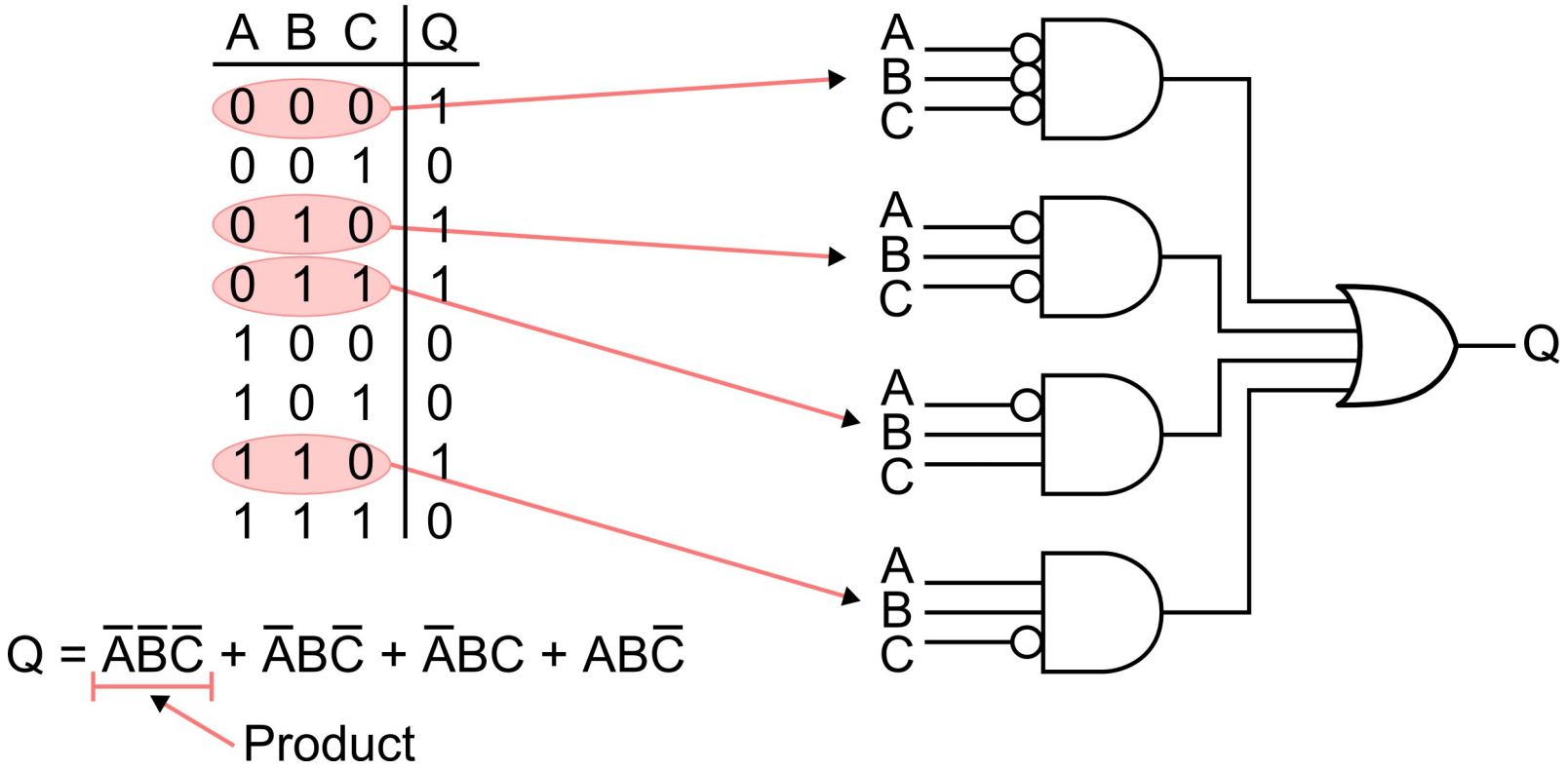


Part 2

An Introduction to FPGAs

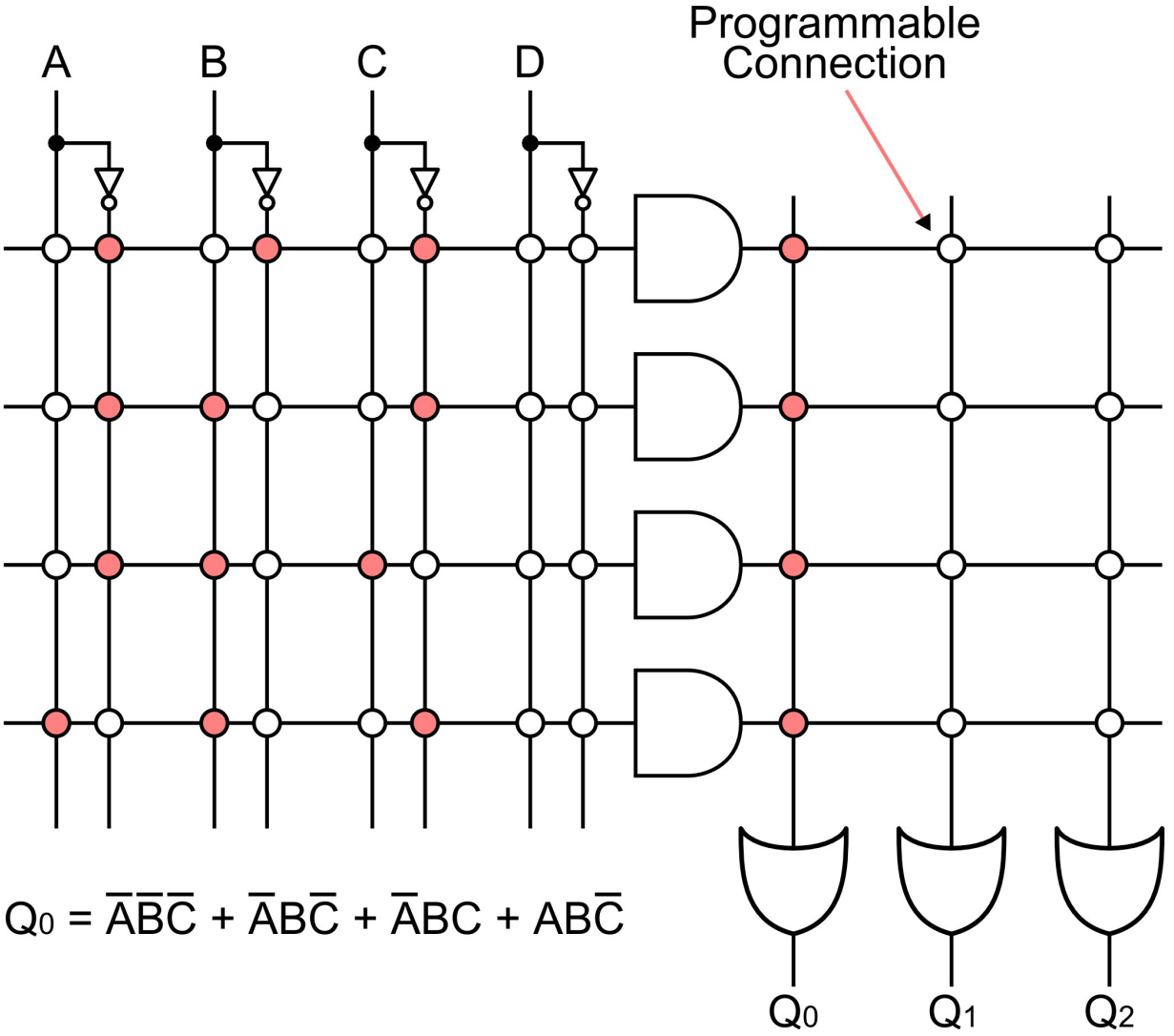
Sum of Products

Any combinatorial function can be represented in a *sum of products* form.



This generic form, allows us to create a general purpose combinatorial logic chip.

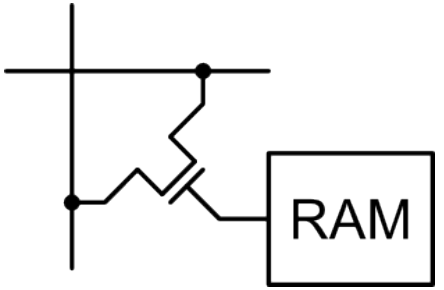
Programmable Logic Devices (PLDs)

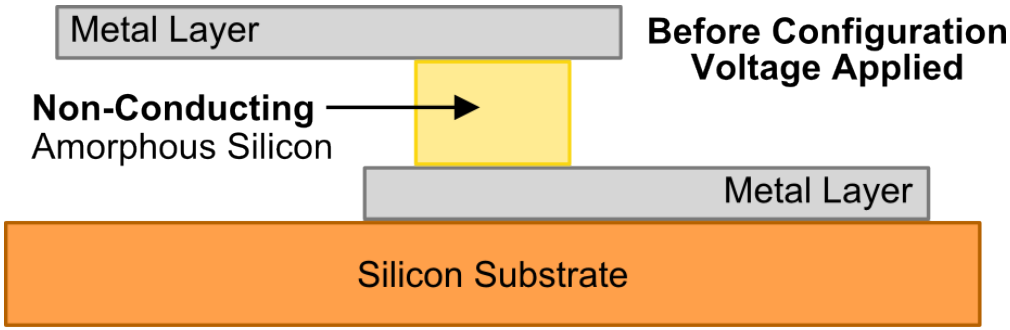


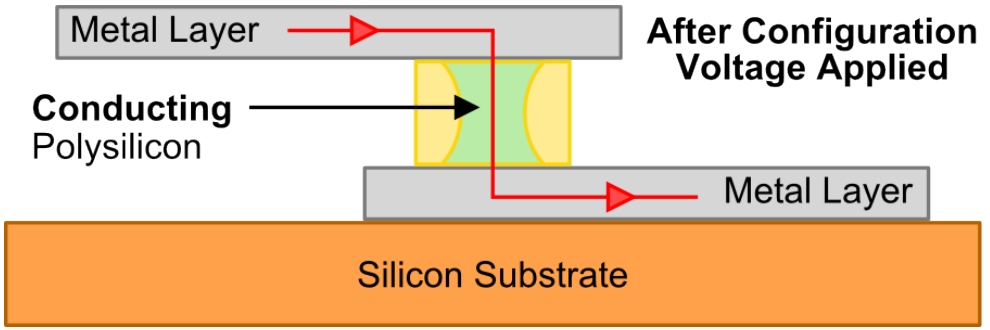
Various forms available: PLAs, PALs & GALs

Connections

The interconnects of programmable logic can be made in a number of ways
- Antifuse & fuse based
- Transistor Based
    - EEPROM (Non-volatile)
    - SRAM (Volatile)



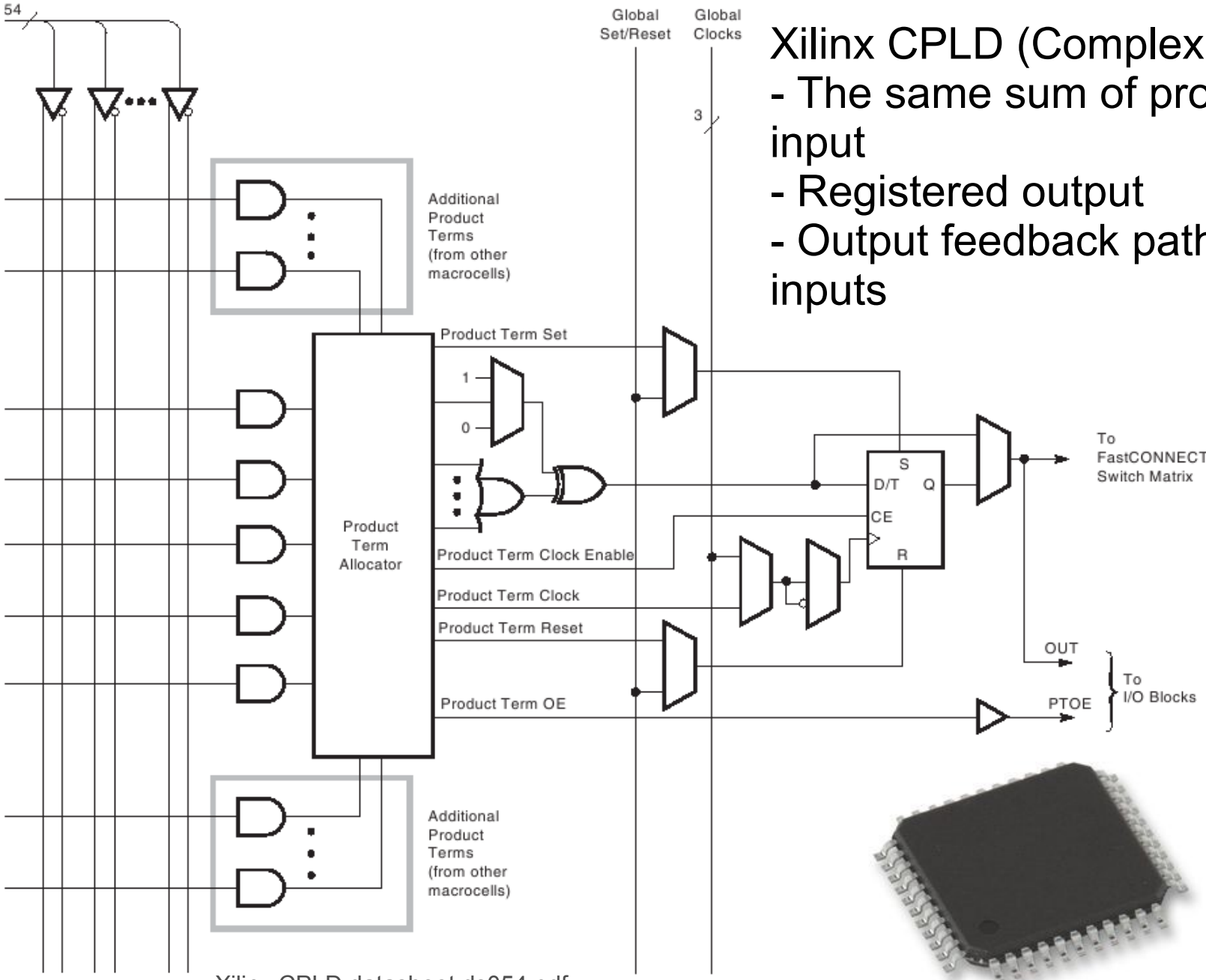




Antifuses start non-conducting.

The application of a programming voltage makes the connection conducting.

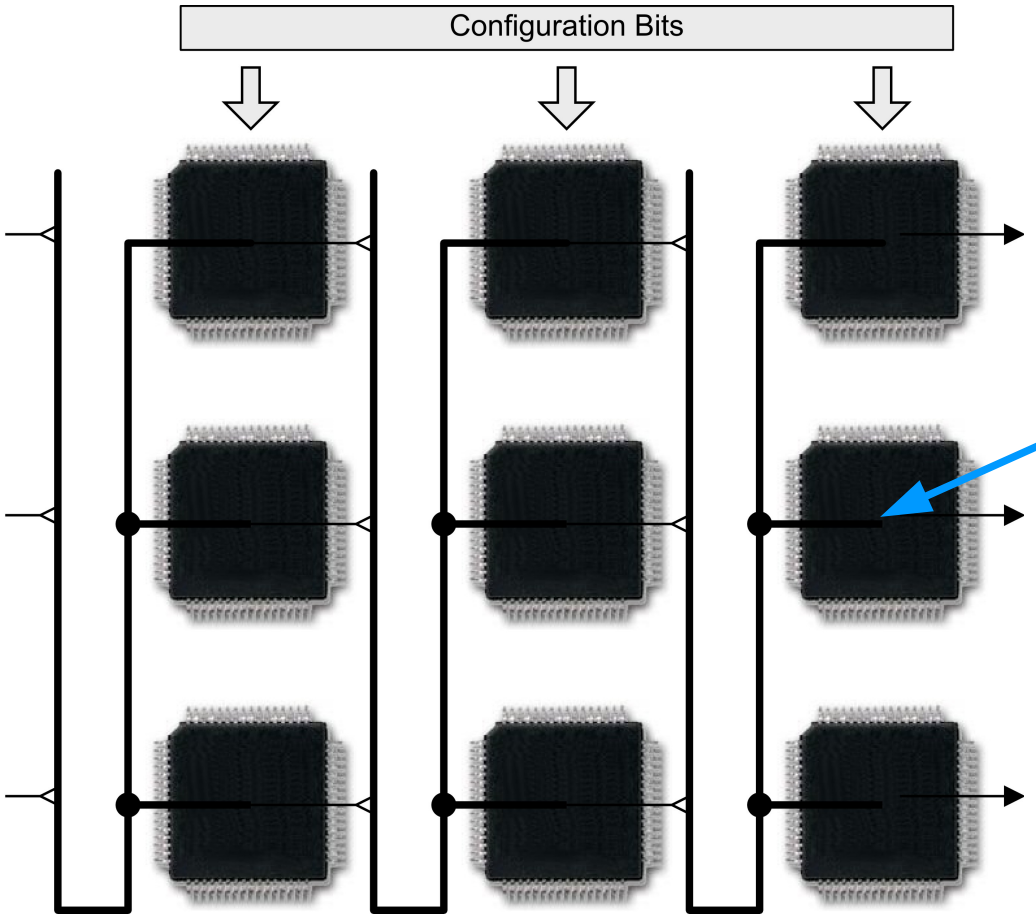
CPLDs

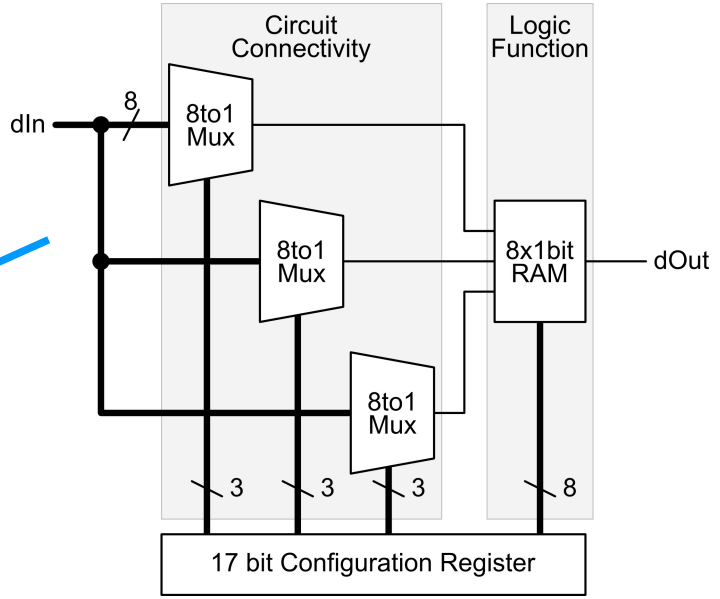


Xilinx CPLD (Complex PLD)
- The same sum of products input
- Registered output
- Output feedback paths into inputs

Xilinx CPLD datasheet ds054.pdf

A First Architecture for EHW



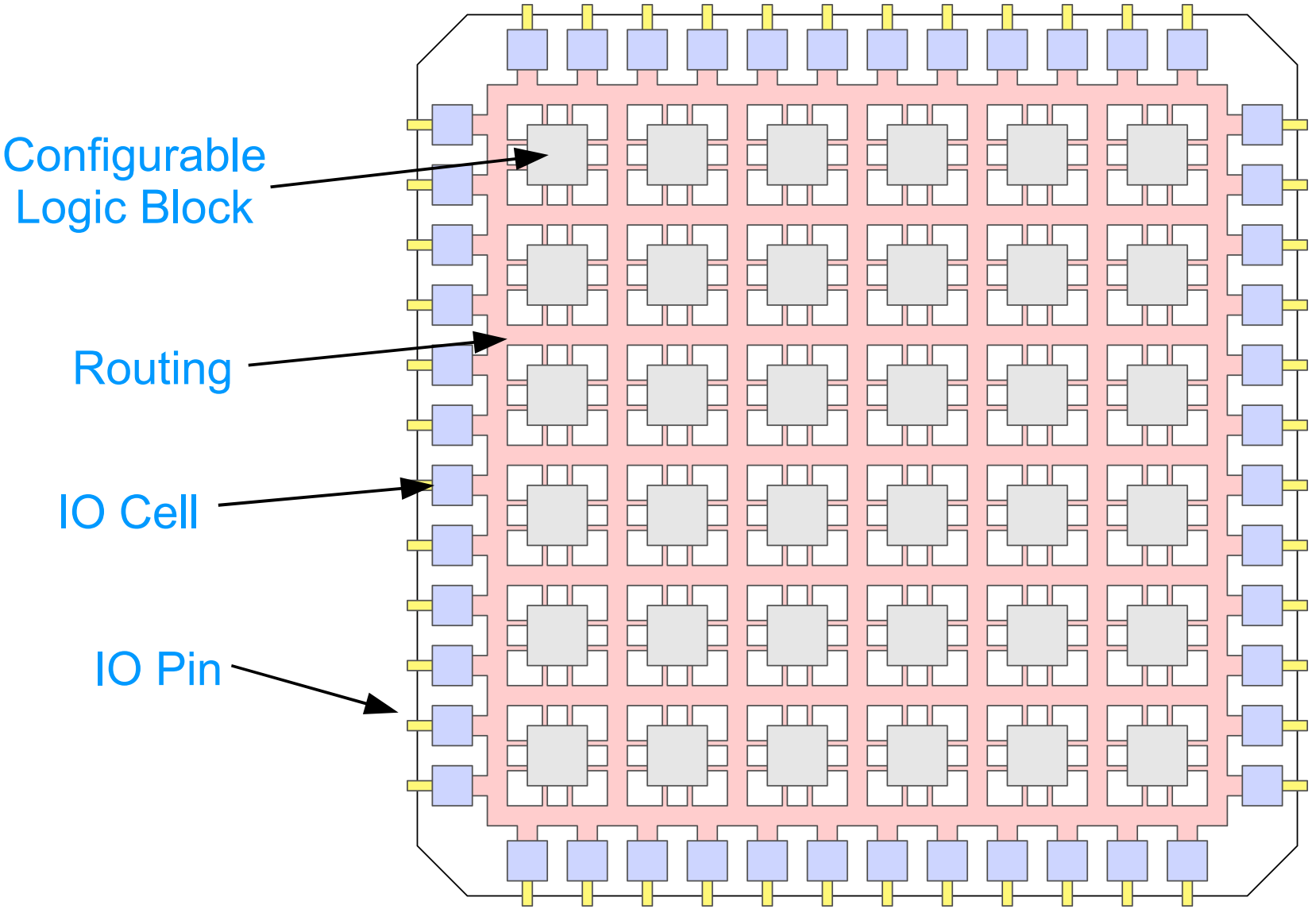


The configurable cell array used in the lab could be recreated using a CPLD for each of the configurable cells.

However, this approach isn't very flexible.

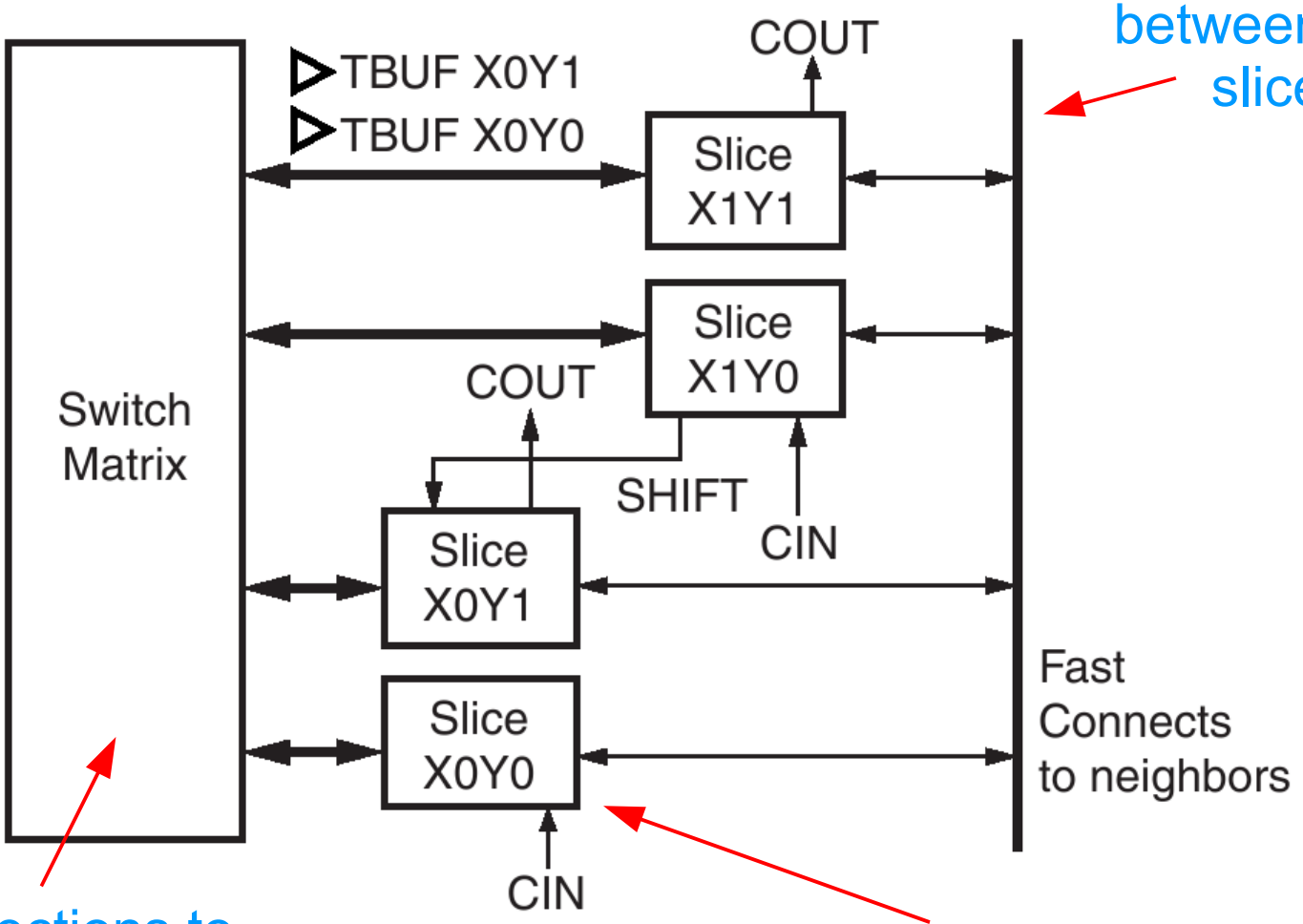
FPGAs

Field Programmable gate arrays are highly configurable digital circuits

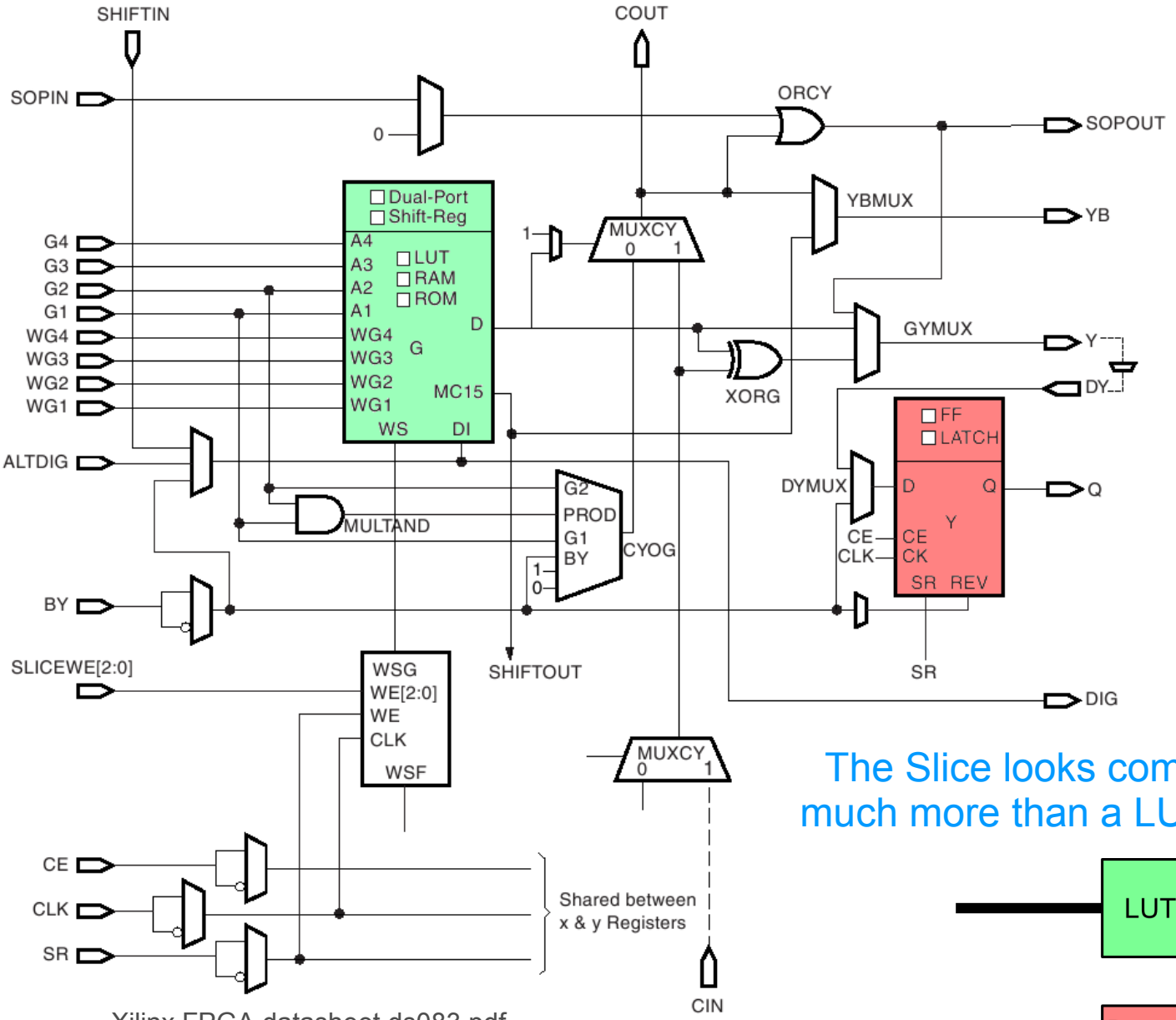


The Virtex II CLB

Xilinx FPGA datasheet ds083.pdf

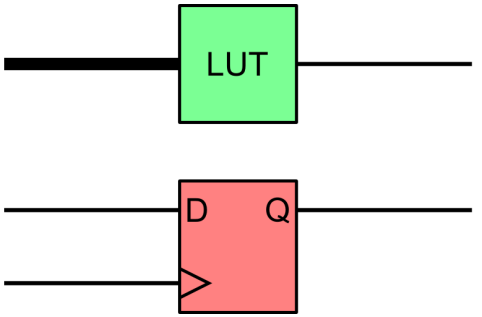


The Virtex II Slice

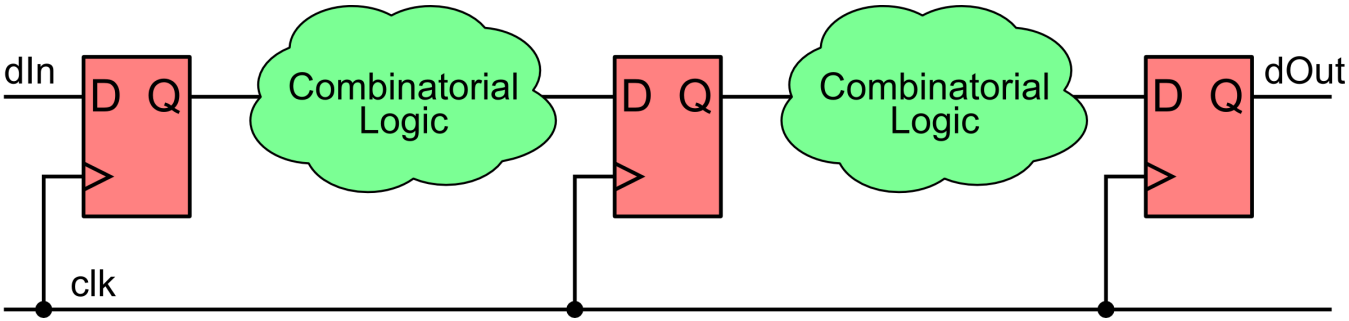


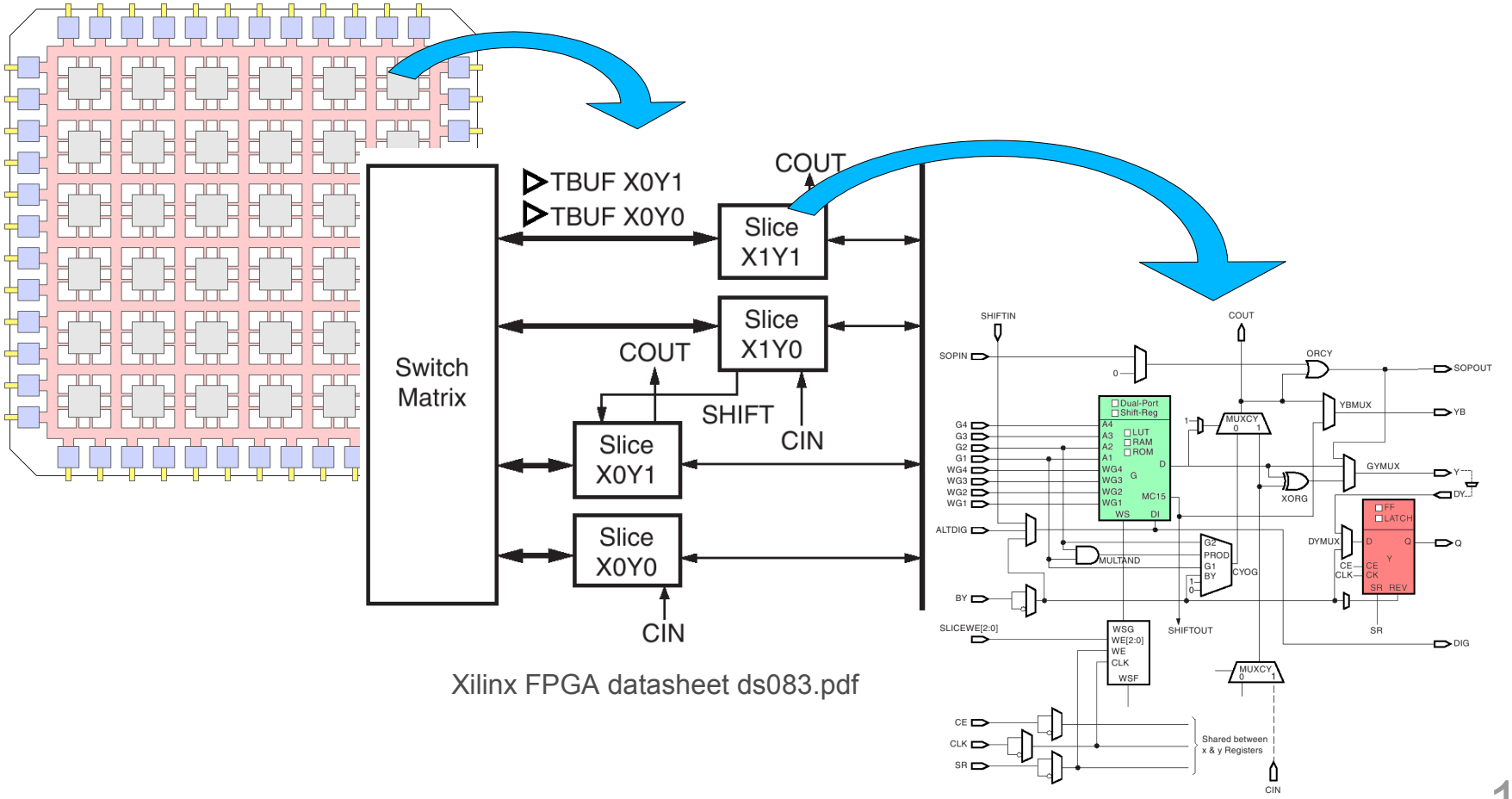
Xilinx FPGA datasheet ds083.pdf

The Slice looks complex, but it is not much more than a LUT and a Flip Flop.



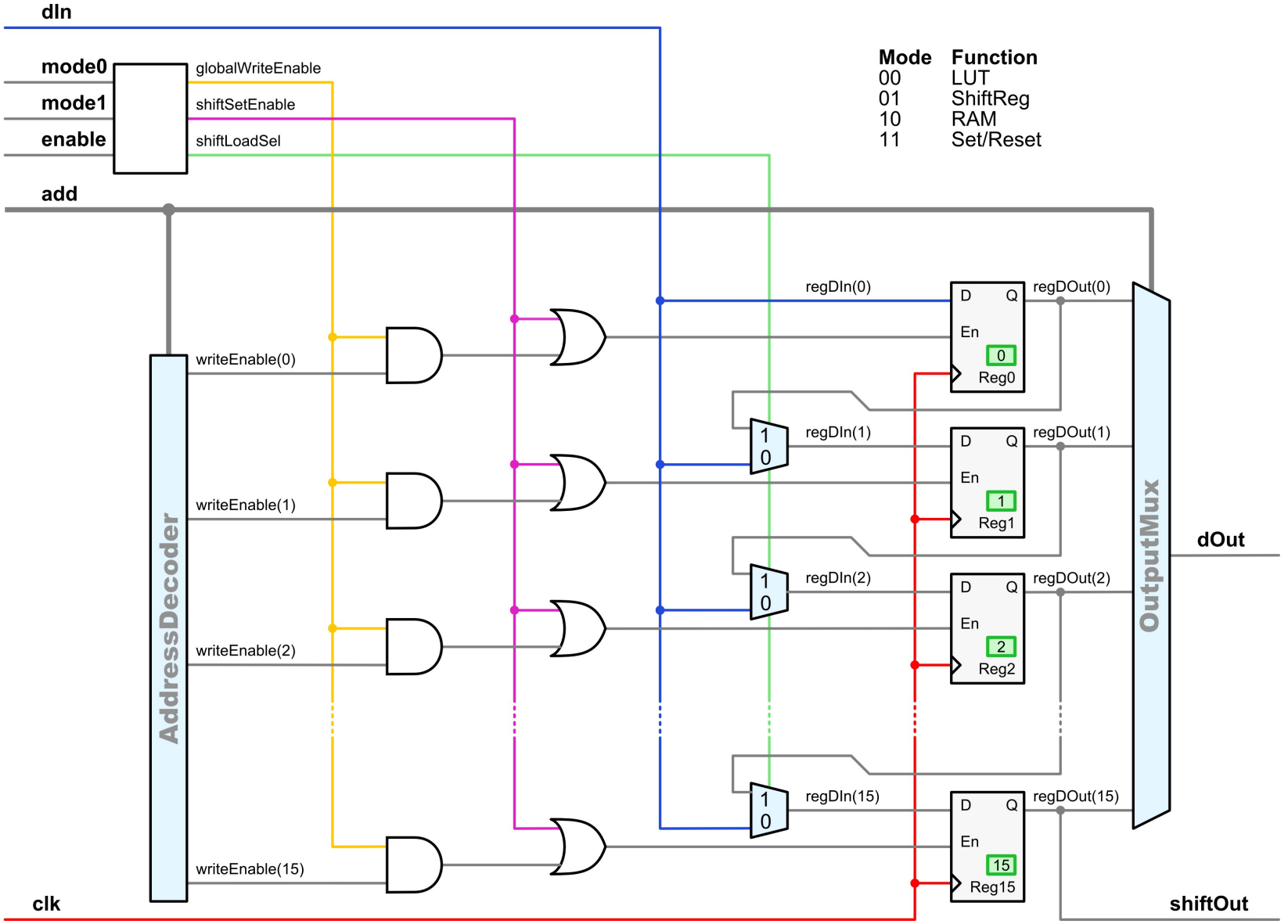
FPGAs





Xilinx FPGA datasheet ds083.pdf

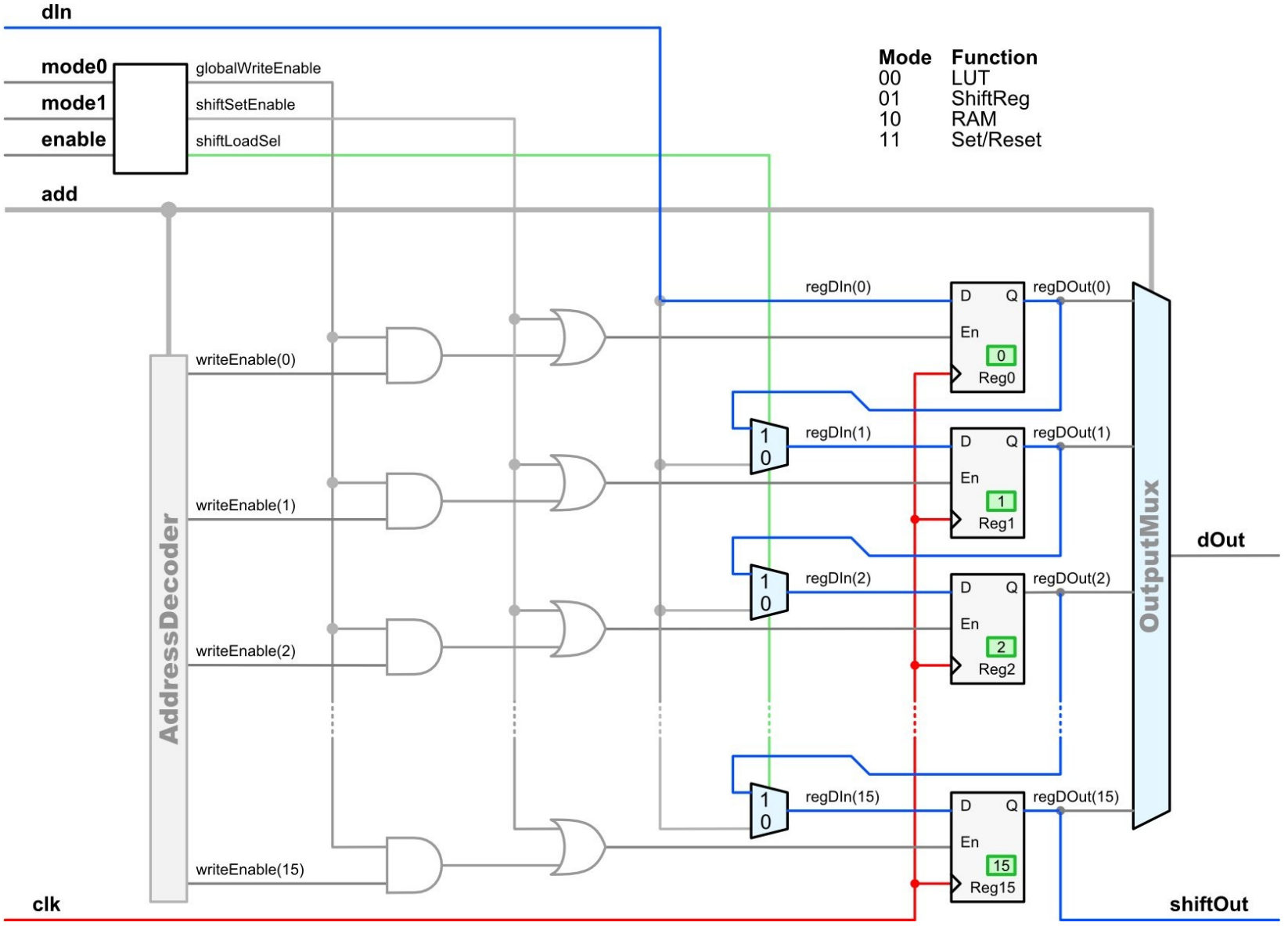
A Closer Look at Look Up Tables



An example LUT.

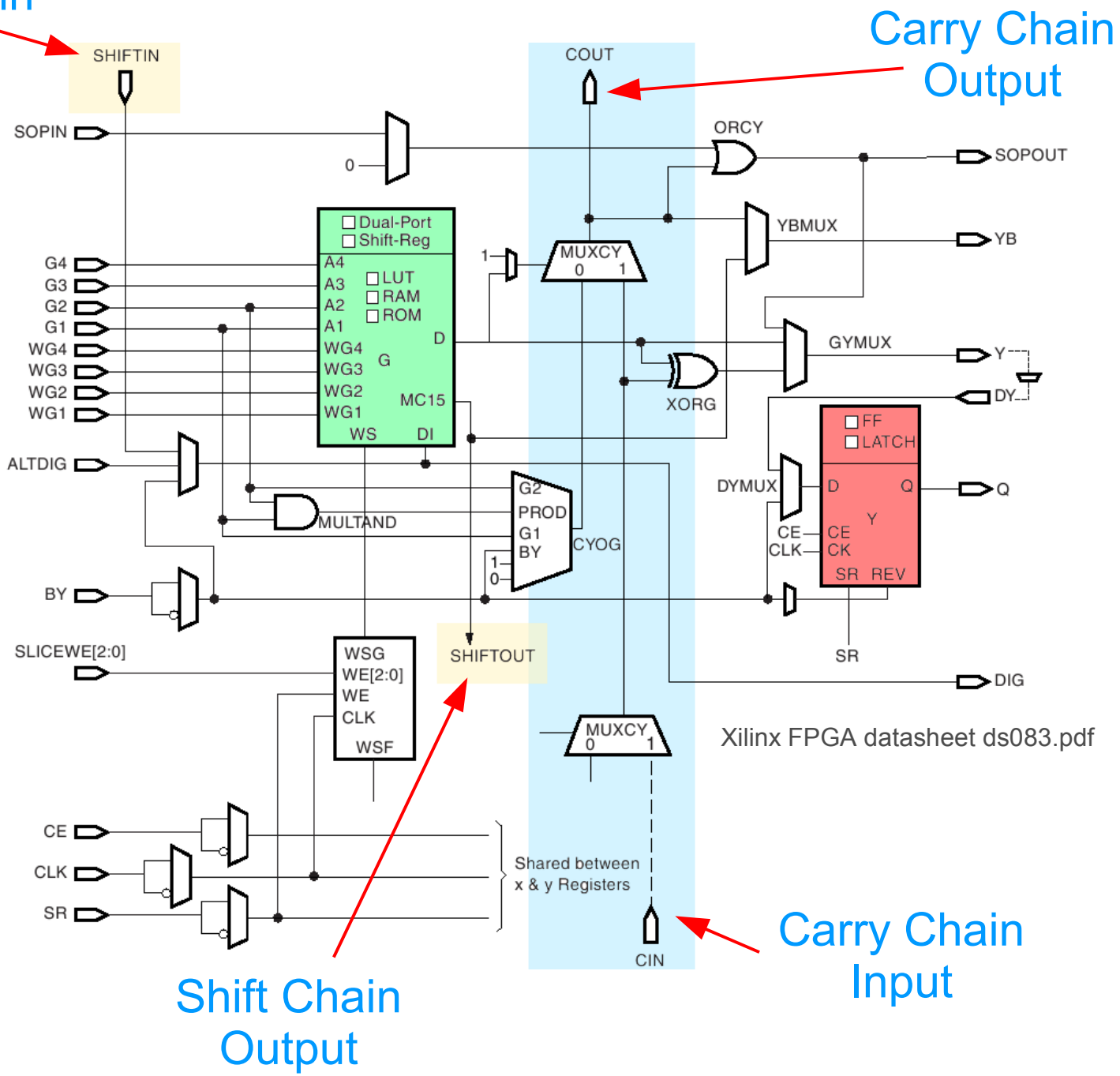
(This one is from the RISA architecture, a reconfigurable device designed at York)

LUT Shift Registers



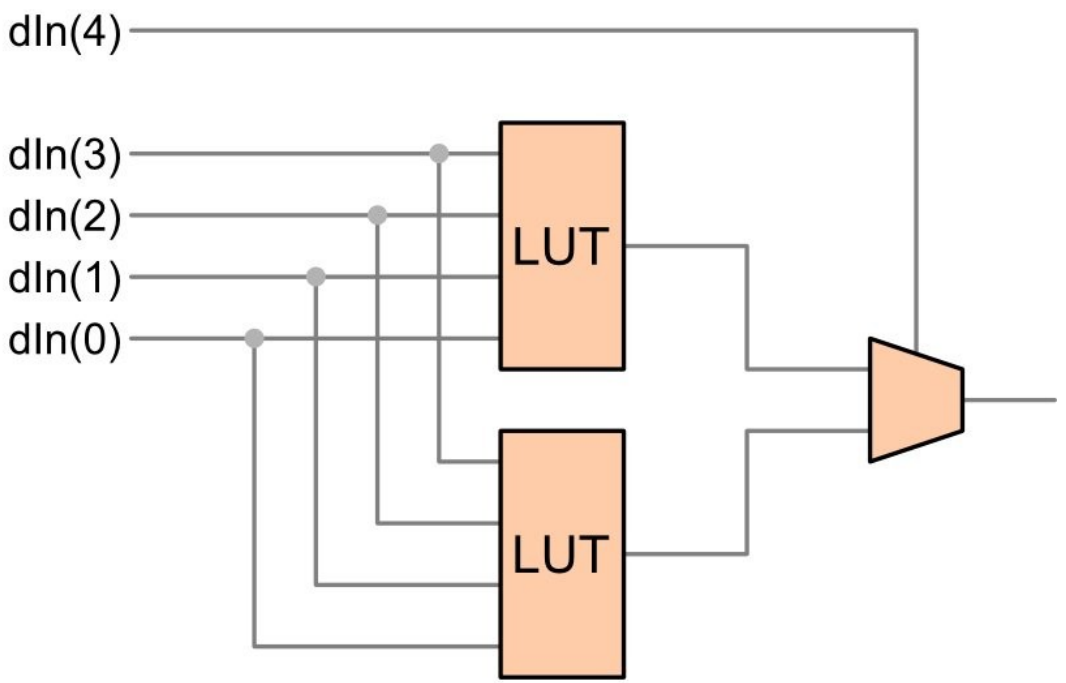
The LUT can be configured to work as a shift register, 1-bit wide, 16-bits long. Using the output mux, it is possible to select different shift register lengths.

Slice Chains



LUT Size

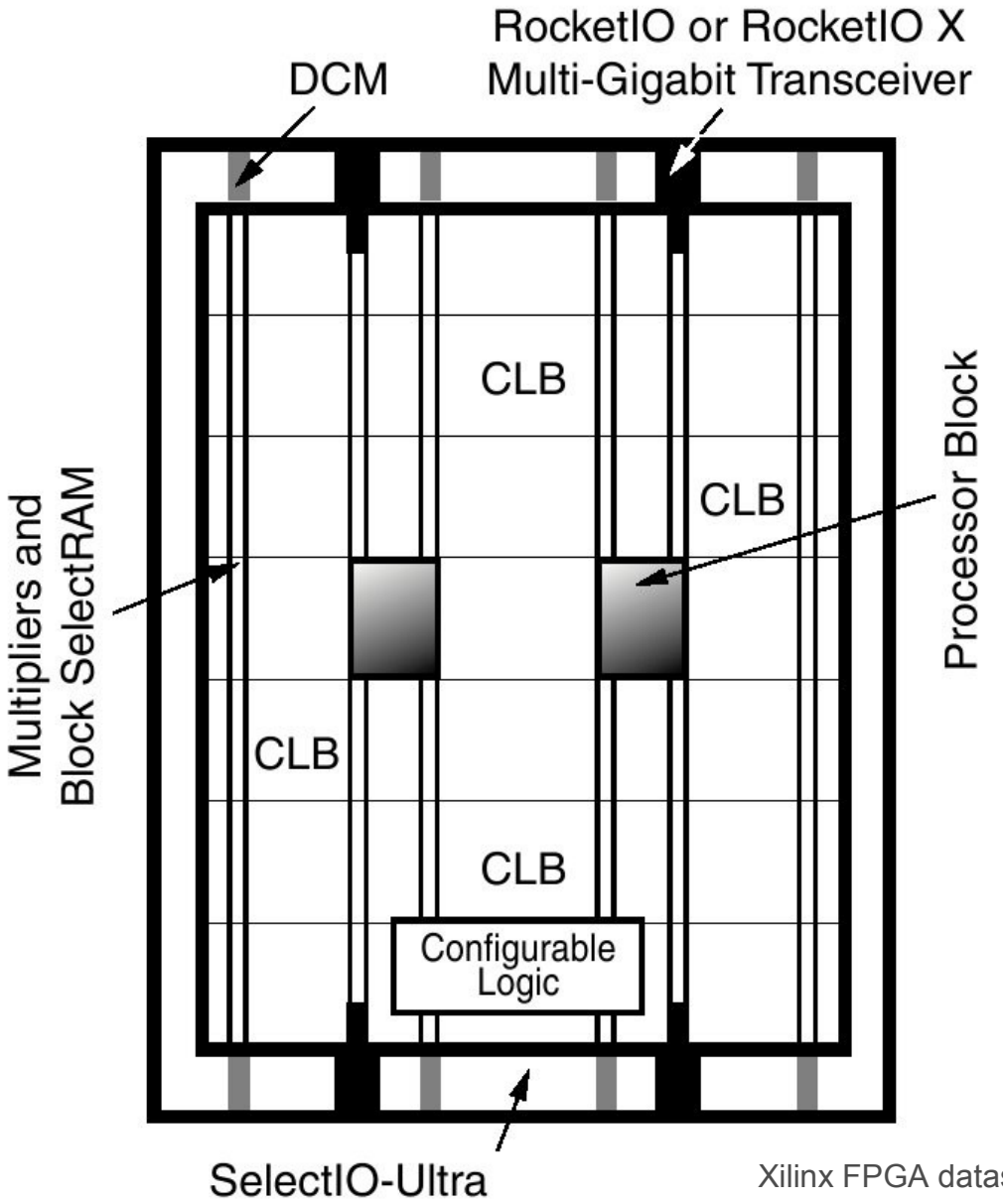
dIn(4)	dIn(3)	dIn(2)	dIn(1)	dIn(0)
0	0	0	0	0
0	0	0	0	1
0	0	0	1	0
0	0	0	1	1
0	0	1	0	0
0	0	1	0	1
0	0	1	1	0
0	0	1	1	1
0	1	0	0	0
0	1	0	0	1
0	1	0	1	0
0	1	0	1	1
0	1	1	0	0
0	1	1	0	1
0	1	1	1	0
0	1	1	1	1
1	0	0	0	0
1	0	0	0	1
1	0	0	1	0
1	0	0	1	1
1	0	1	0	0
1	0	1	0	1
1	0	1	1	0
1	0	1	1	1
1	1	0	0	0
1	1	0	0	1
1	1	0	1	0
1	1	0	1	1
1	1	1	0	0
1	1	1	0	1
1	1	1	1	0
1	1	1	1	1



Larger LUTs can be created by combining standard LUTs with multiplexers.

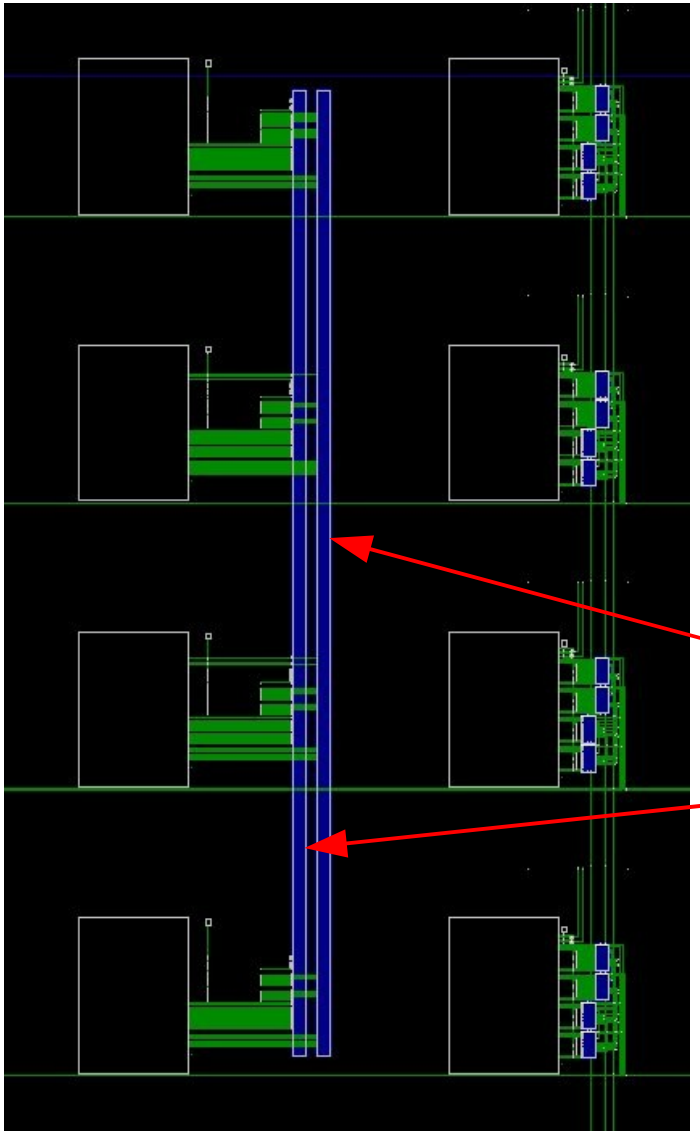
- Larger LUTs require less routing, but result in more wasted logic,
- Small LUTs will need combining with routing, but cause less wasted logic.

Extending Functionality



Xilinx FPGA datasheet ds083.pdf

Block RAM & Multipliers



To save logic and routing resources, regularly used functions are implemented as Hard Macros. These make more efficient use of die area, and will be faster than implementing the same functionality in the FPGA fabric.

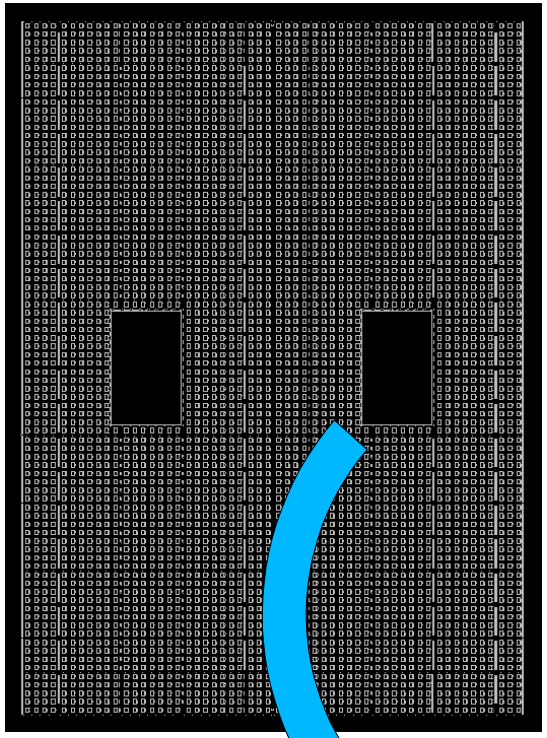
Two Such Components are:

Block RAM

Multipliers

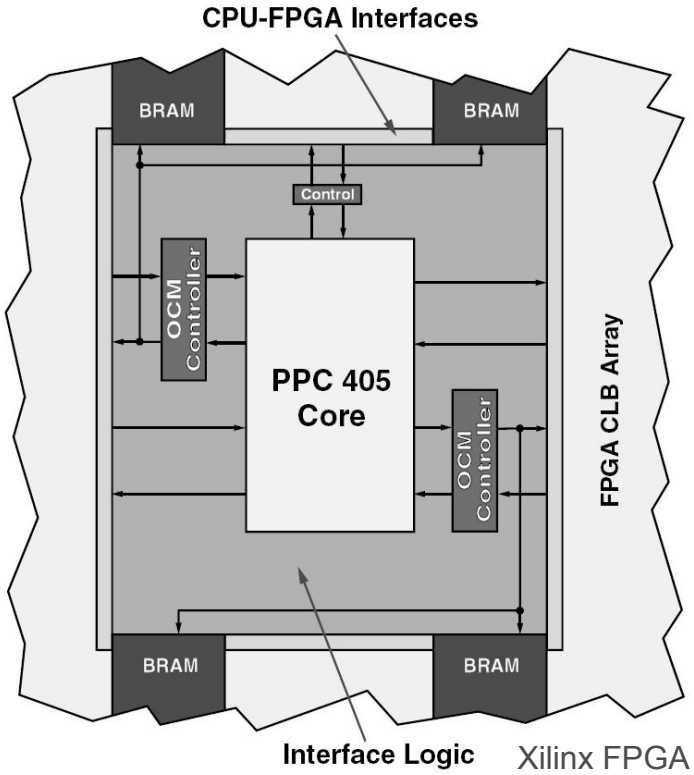
These components are connected directly into the FPGA routing resources.

Power PC Cores



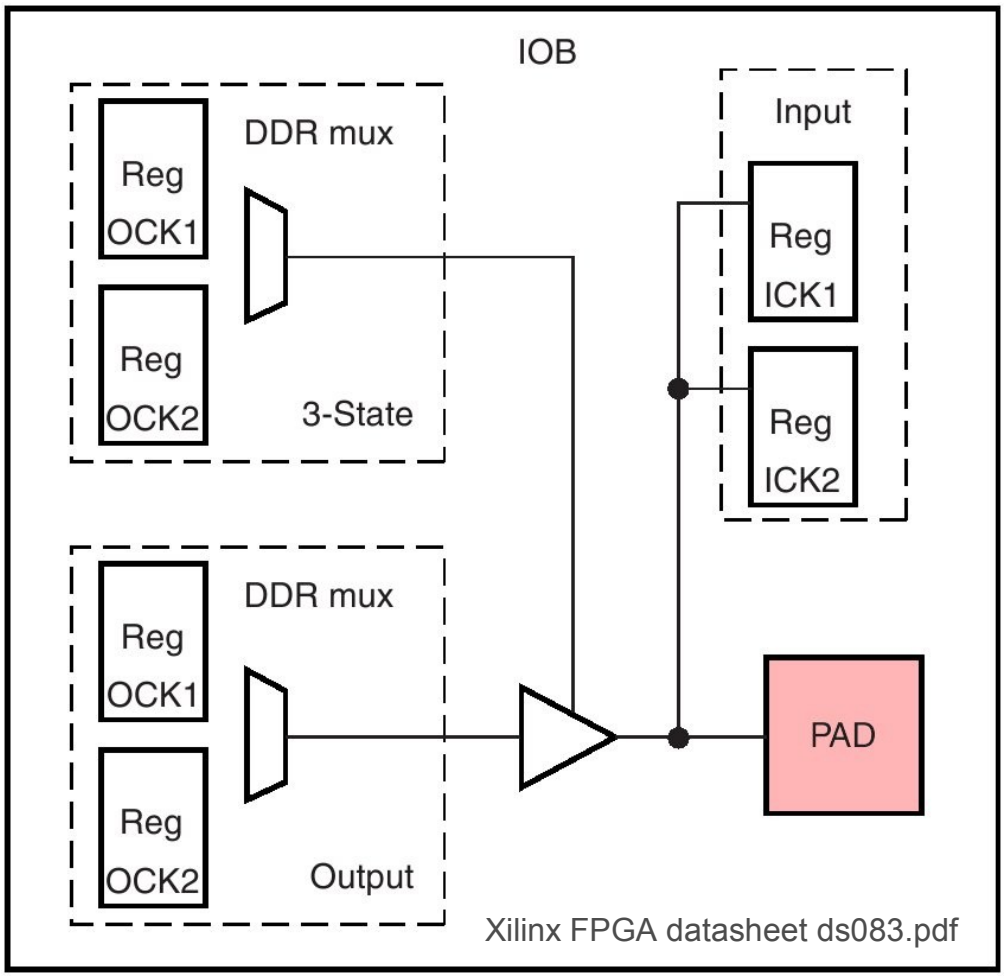
Hard Power PC cores provide an efficient way of implementing software elements into an FPGA design.

Soft cores can also be used, but these will use the device's logic and routing resources.



Xilinx FPGA datasheet ds083.pdf

IO Blocks

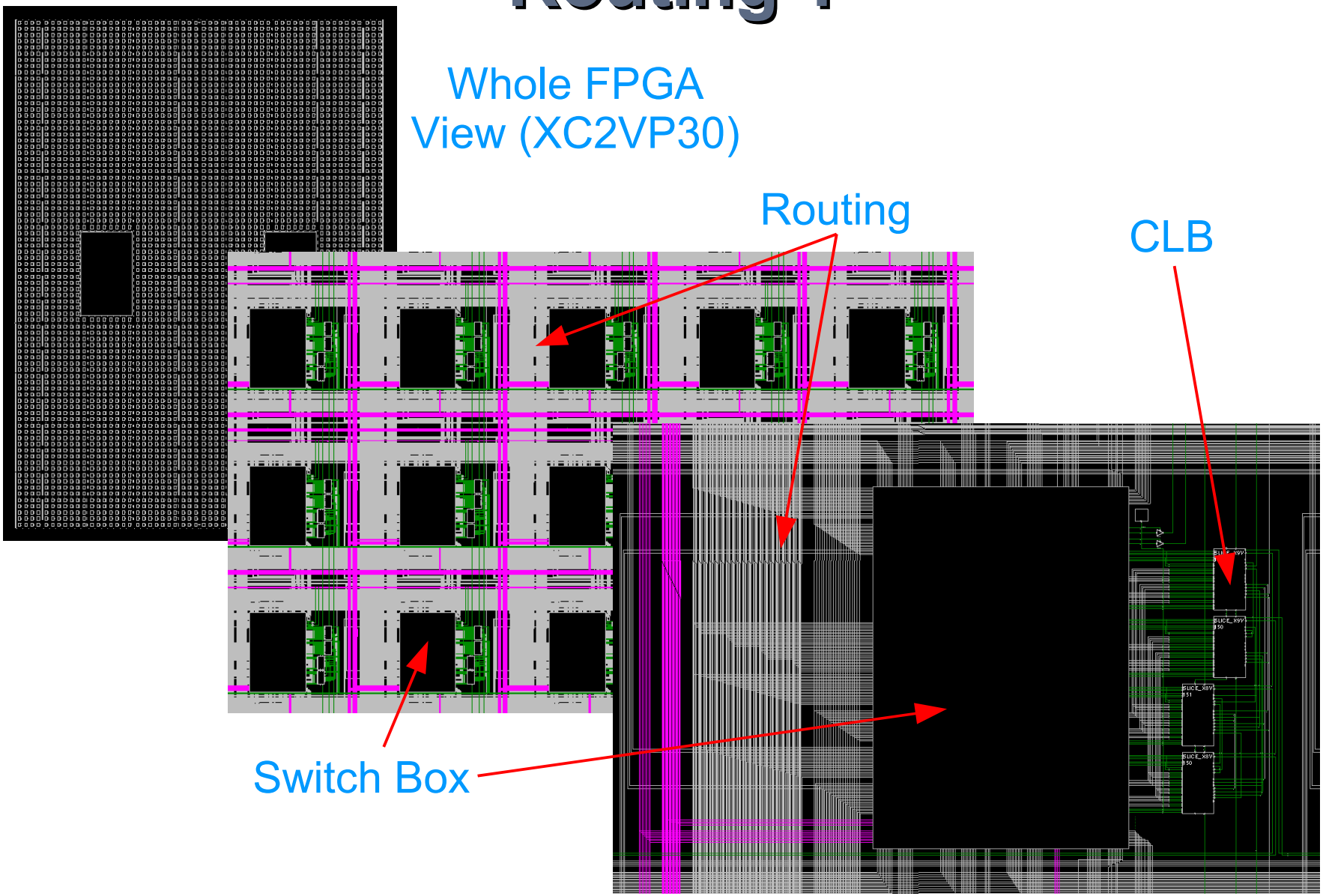


LVTTL
LVCMOS33
LVCMOS25
LVCMOS18
LVCMOS15
PCI33_3
PCI66_3
PCIX
GTL
GTLP
HSTL_I
HSTL_II
HSTL_III
HSTL_IV
HSTL_I_18
HSTL_II_18
HSTL_III _18
HSTL_IV_18
SSTL2_I
SSTL2_II
SSTL18_I
SSTL18_II
LDT_25
LVDS_25
LVDSEXT_25
BLVDS_25
ULVDS_25
LVPECL_25
LDT_25_DT
LVDS_25_DT
LVDSEXT_25_DT
ULVDS_25_DT
LVDCI_33
LVDCI_25
LVDCI_DV2_25
LVDCI_18
LVDCI_DV2_18
LVDCI_15
LVDCI_DV2_15
GTL_DCI
GTLP_DCI
HSTL_I_DCI
HSTL_II_DCI
HSTL_III_DCI
HSTL_IV_DCI
HSTL_I_DCI_18
HSTL_II_DCI_18
HSTL_III_DCI_18
HSTL_IV_DCI_18
SSTL2_I_DCI
SSTL2_II_DI
SSTL18_I_DCI
SSTL18_II_DCI

As FPGAs increase in size, they are combining and performing more roles. Therefore, it is very important that they are able to connect to many different signalling standards.

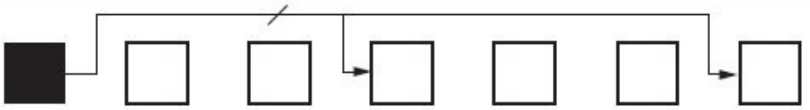
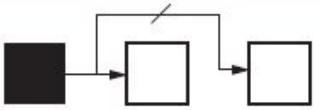
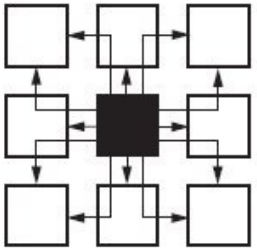
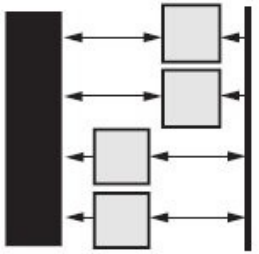
Being able to adjust IO parameters such as drive strength, drive speed & termination resistance is both useful and important.

Routing 1



Routing resources make up a major portion of an FPGA's die area

Routing 2

24 Horizontal Long Lines 24 Vertical Long Lines	
120 Horizontal Hex Lines 120 Vertical Hex Lines	
40 Horizontal Double Lines 40 Vertical Double Lines	
16 Direct Connections (total in all four directions)	
8 Fast Connects	

Xilinx FPGA datasheet ds083.pdf

Xilinx FPGAs use a number of different routing resources to improve the efficiency of logic inter-connections.

FPGA Quick Summary

- FPGAs are built from a grid of cells called Configurable Logic Blocks (CLBs).

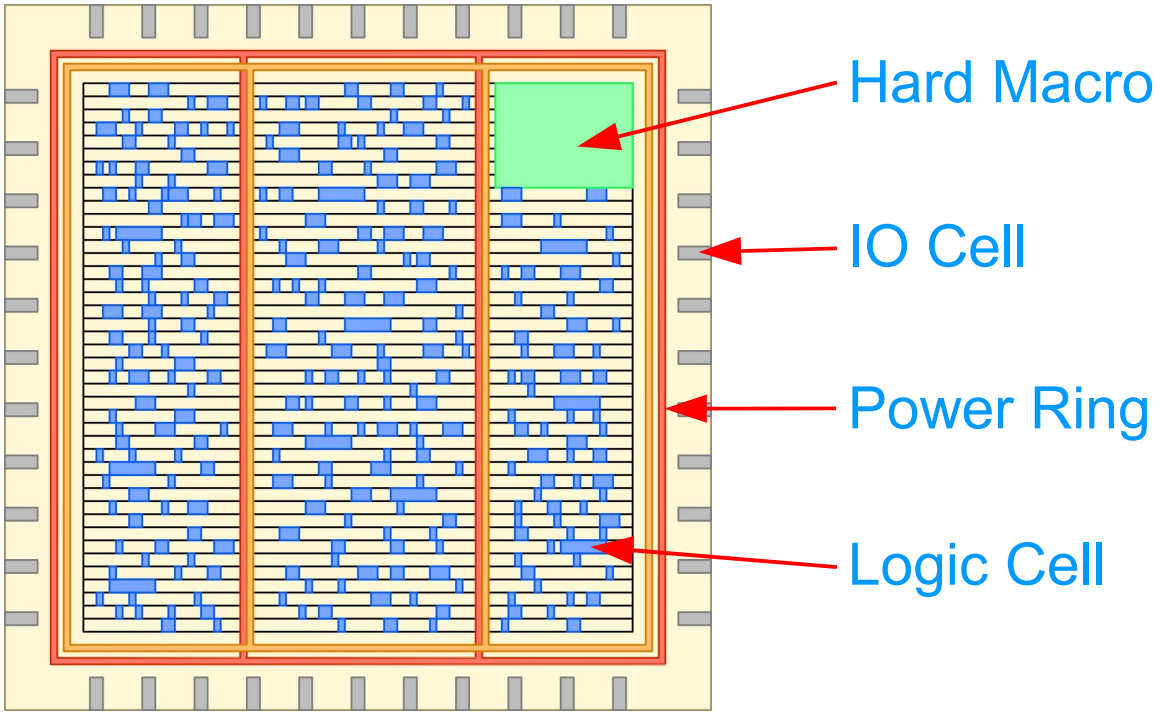
- Each CLB contains a number of Lookup Tables (LUTs) and Flip Flops. These are arranged into slices.

- A variety of routing resources provides connectivity for short (fast) distances and long (slow) distances.
    - Short, Hex & Long wires

- A number of 'hard macros' improve space efficiency by providing extra functionality without having to implement them in CLBs.
    - Multipliers
    - DCMs (Digital Clock Managers)
    - PPCs

- IO Blocks can be configured to easily interface to a large number of different signal standards.

FPGAs vs ASICs

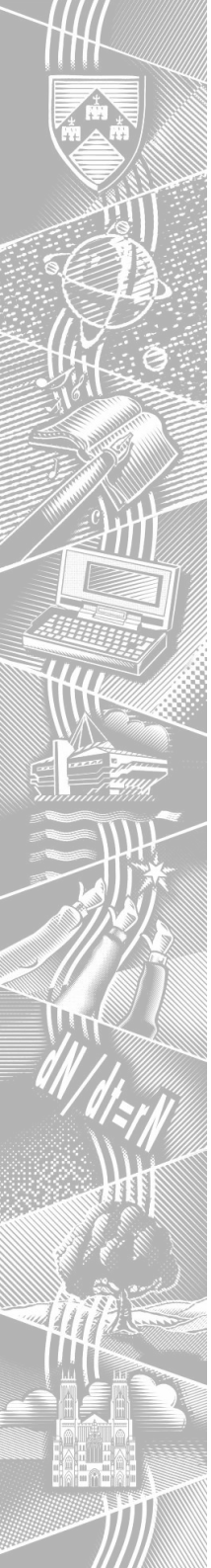


Cell-based Application Specific IC (ASIC)

If FPGAs are so great, why haven't they removed the need for application specific ICs?

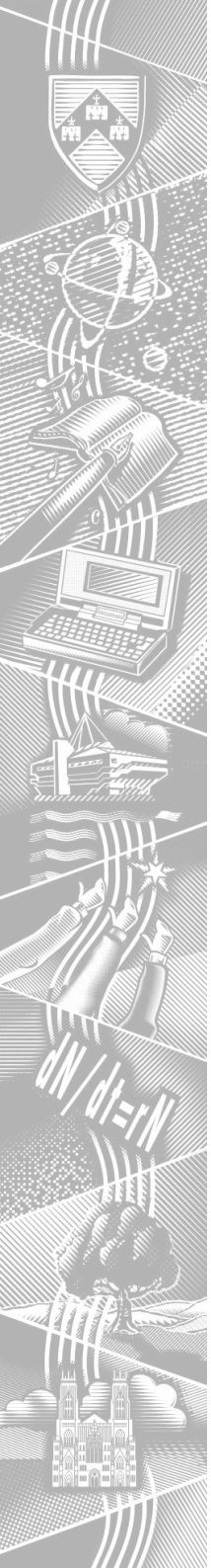
ASICs
- Cheaper at large volumes
- Do not require configuration at power-up
- Long development time

FPGAs
- Cheaper for small runs
- Can be updated with new (improved) configurations
- Very quick development time



FPGA Software

FPGA Design Flow



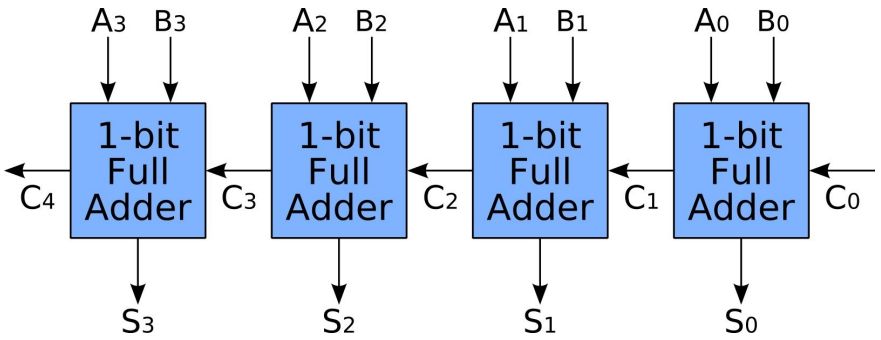
Design Entry	Describe the circuit in HDL, Cores, & SW
Parsing	Check HDL for syntax errors
Synthesis	Inference of design blocks (Muxs, RAMs, Adders etc...)
Implementation	
Translate	Translate the logical design into FPGA primitives (gates, flip flops)
Map	Map inferred logic into FPGA Elements (CLBs, IOBs)
Place and Route	Place the elements and connect.
Bit Stream Generation	Create the configuration bitstream
Device Configuration	Download the bitstream into the FPGA

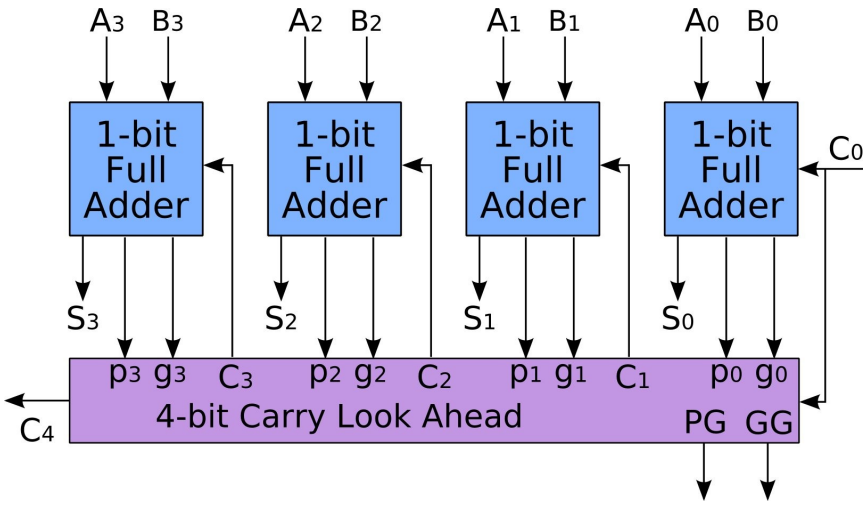
The complete FPGA design flow can be undertaken using Xilinx ISE

Synthesis

```
Adder : process(dInA, dInB)
begin
   result <= dInA + dInB;
end process;
```

Using a higher-level behavioural description allows the synthesis tools to choose the most appropriate implementation.
- If speed is critical a carry look ahead adder may be used.
- If area is critical a ripple adder may be used.



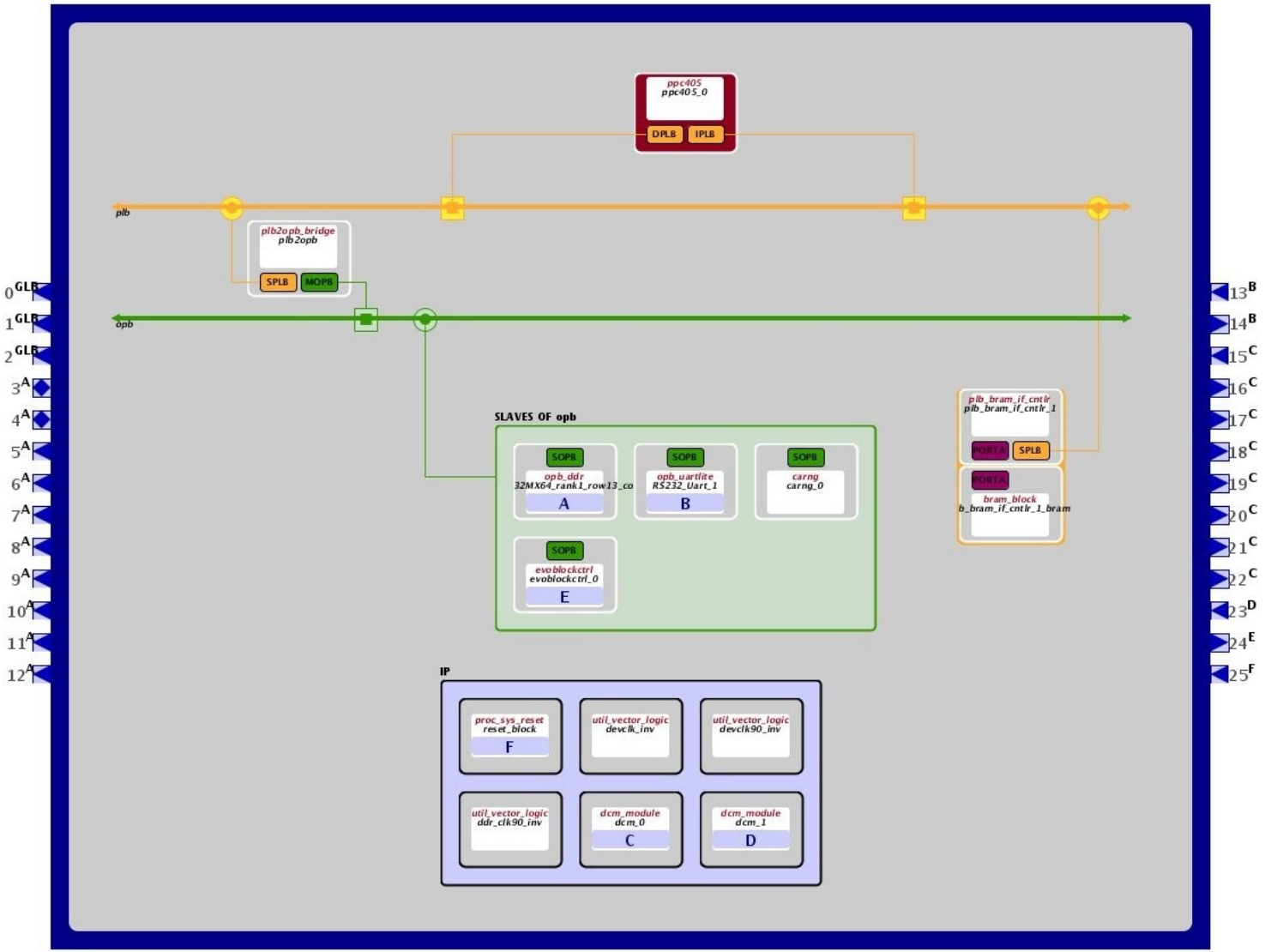


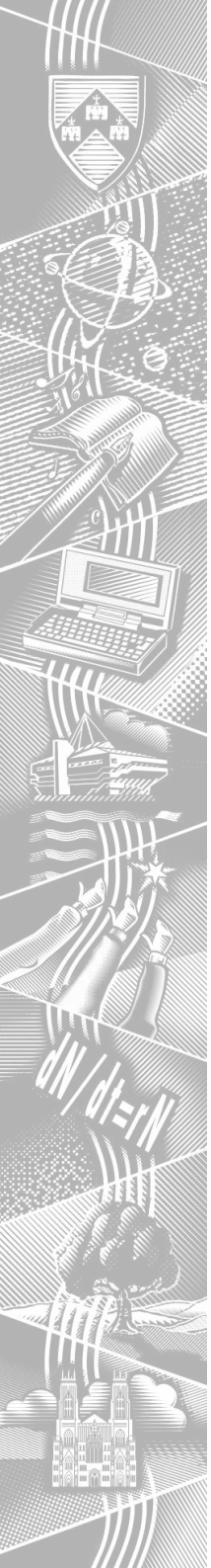
4 Bit ripple adder
- Small footprint
- Slow Operation

4 Bit carry look ahead adder
- Large footprint
- Fast Operation

Embedded Development Kit (EDK)

Xilinx EDK provides a simple method for creating embedded systems with both hardware and software elements.
EDK can use either a hard PPC core of a soft Microblaze core.





FPGAs for Evolvable Hardware

FPGAs For Evolving Hardware

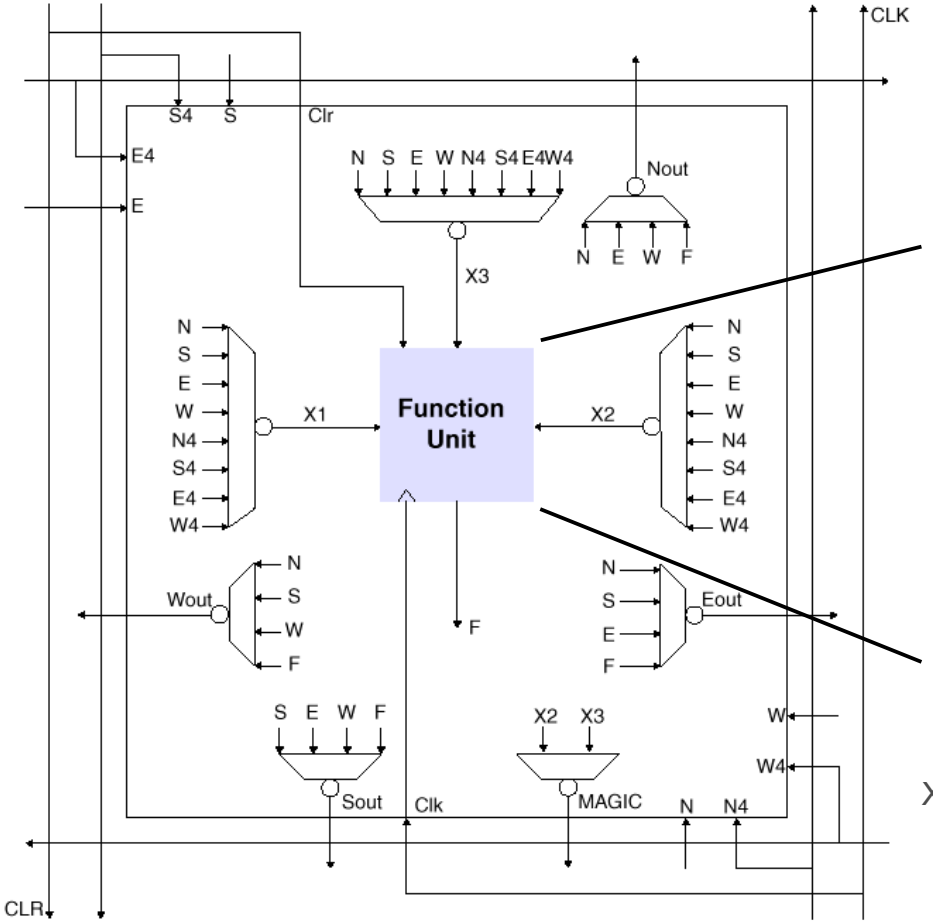
FPGAs should be the perfect platform for evolving digital circuits.

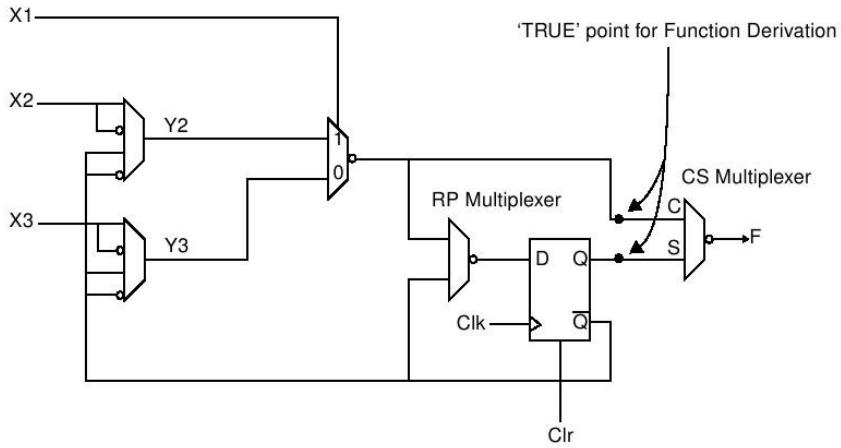
- They are able to implement a vast range of circuits.
- They are reconfigurable, allowing many circuits to be tested.

However,

- Creating a configuration requires a long software tool chain.
- Configuration bit details are proprietary.
- Partial device configuration is impossible or very coarse.
- Long configuration times would slow down evolution.
- Arbitrary adjustment to a bit stream can lead to device damage.

The Xilinx XC6200

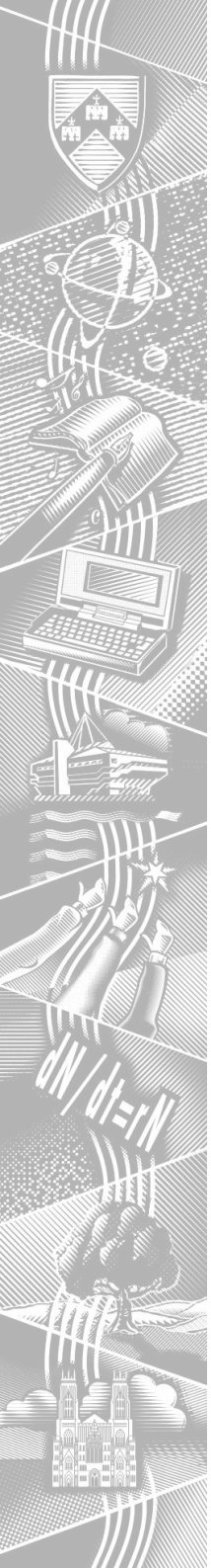




Xilinx XC6200 datasheet

The Xilinx XC6200 is no longer manufactured, but when it was available it was very popular for implementing evolved circuits.

- Complete configuration information was available.
- Fine grained reconfiguration was possible.

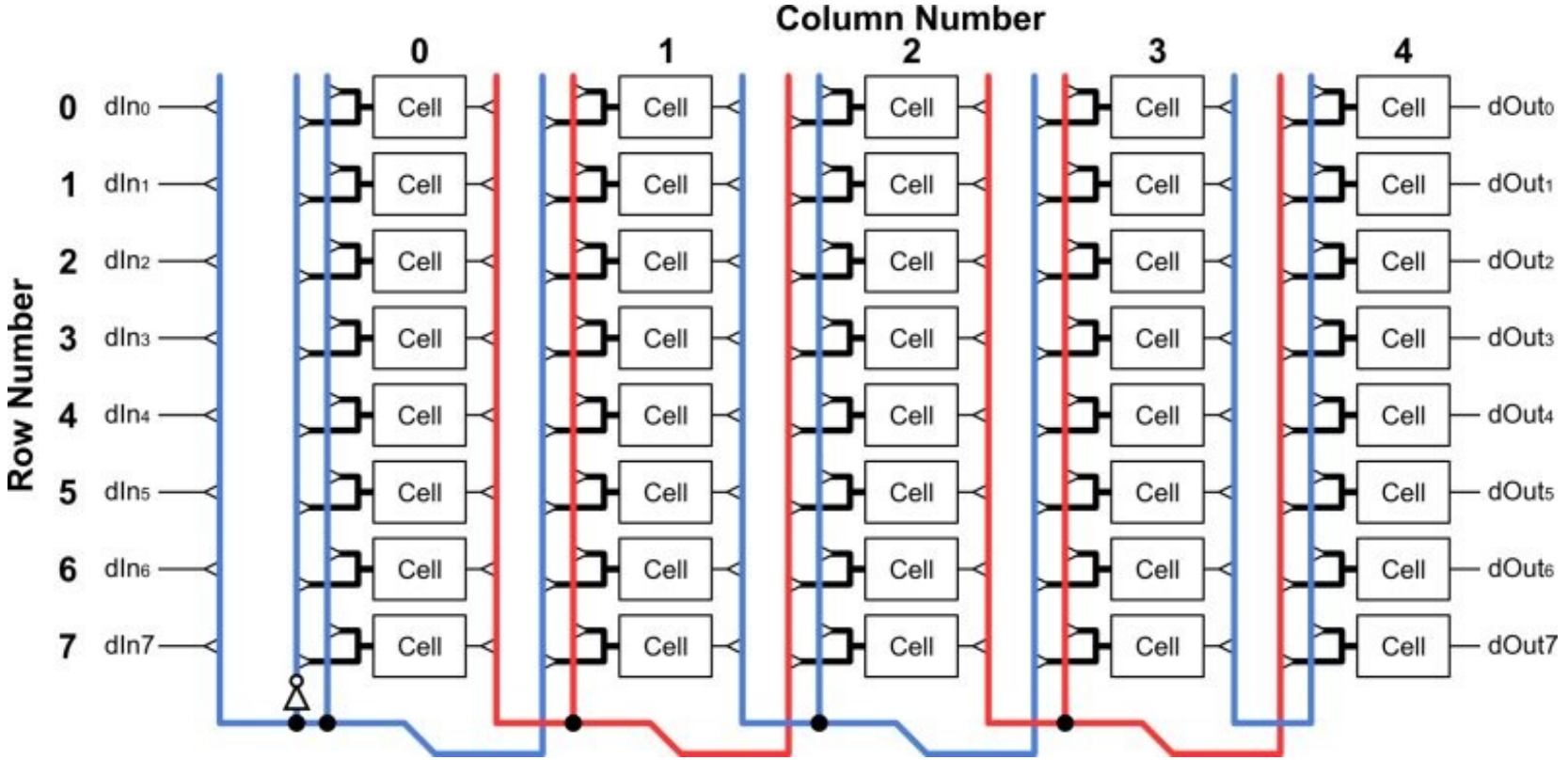


The JBit API

The JBits API is a set of Java-based functions that can be used to examine and adjust a Xilinx FPGA bitstream. However,

- JBits is only available for a limited number of FPGAs types.
- Despite the low-level control, JBits does not reveal any real information about the bitstream structure.

Overlay Architectures
The VRC (Virtual Reconfigurable Circuit)

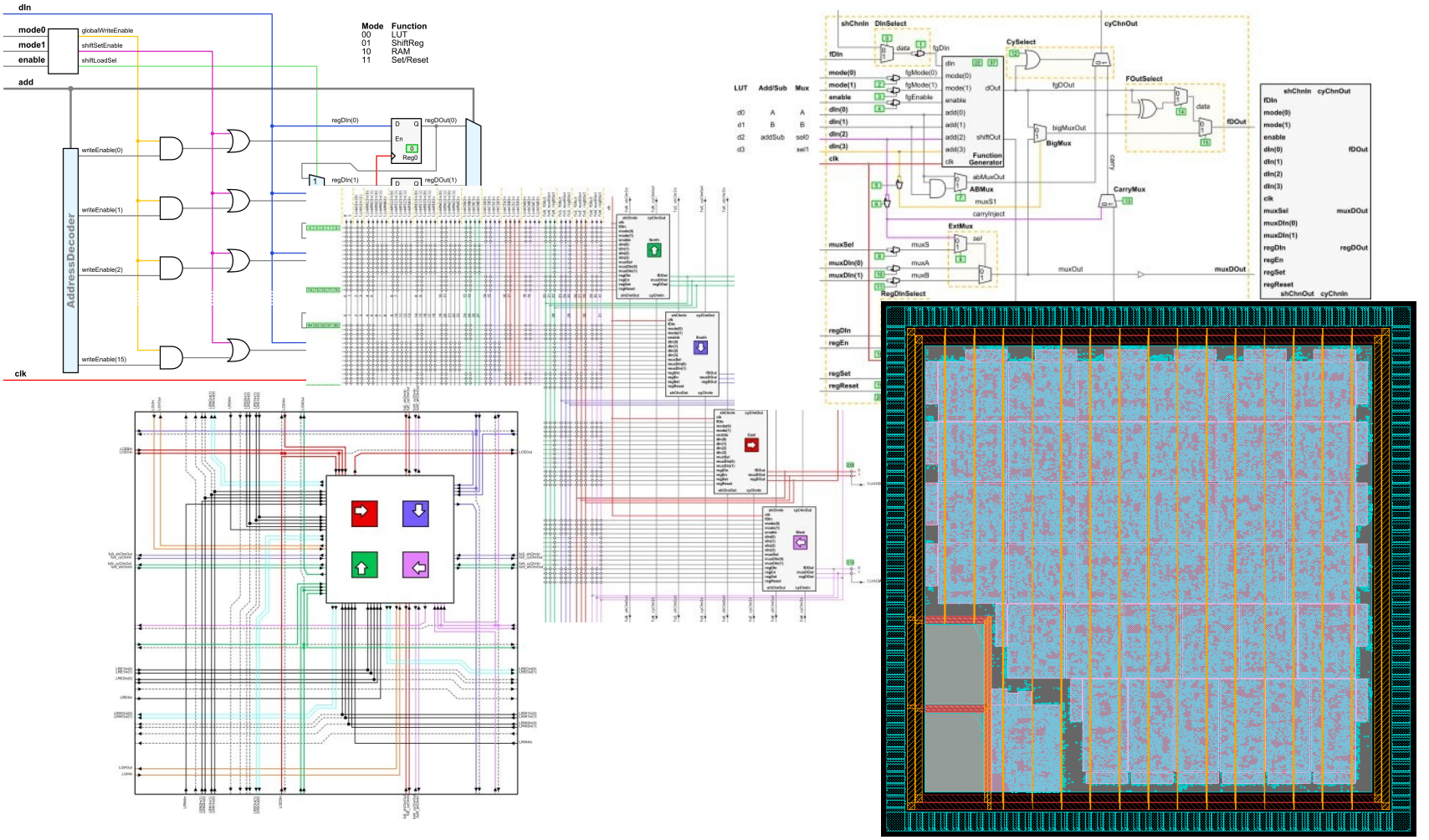


The most common approach is to create a custom reconfigurable architecture within an FPGA.

- This approach is flexible as it allows you to create exactly what you need.
- But, it is very inefficient. A small portion of reconfigurable fabric is created in proportion to that of the underlying FPGA.

(This is the approach taken in the labs)

ASICs



Creating an ASIC allows complete control over the chips circuitry

- This approach is complex and time-consuming.
- ASIC fabrication is very expensive and allows very little chance of architecture adjustment.