



Programmable Logic Devices

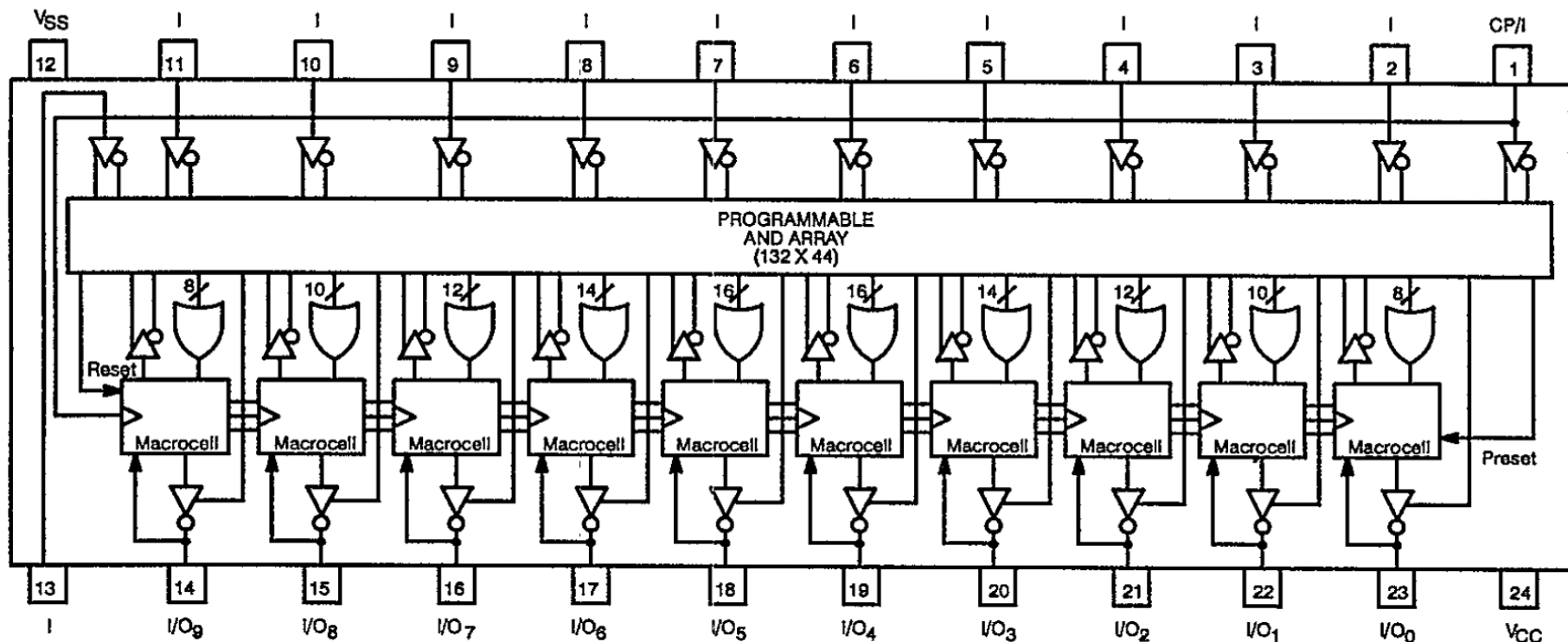
Aims & Objectives

- To give an introduction into Programmable Logic Devices in use commercially today
 - And how they are constructed internally
- Specifically, we'll talk about Field Programmable Gate Arrays (FPGA's)
- We'll derive a fine-grained FPGA to illustrate this
- Then look at other commercial devices and a little into the future

Programmable Logic Devices

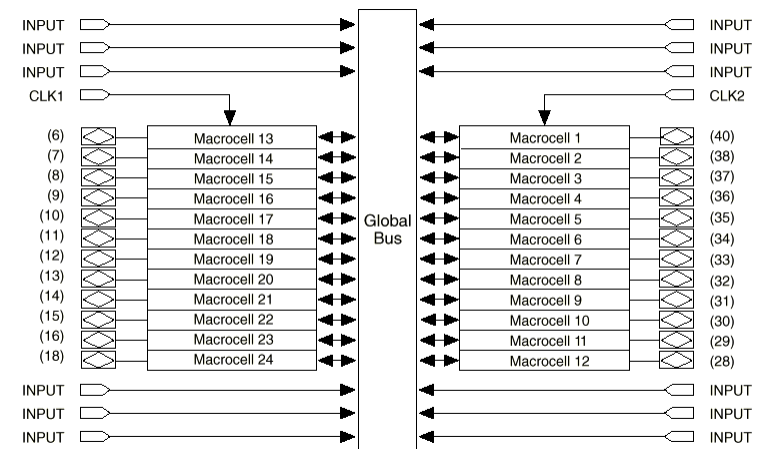
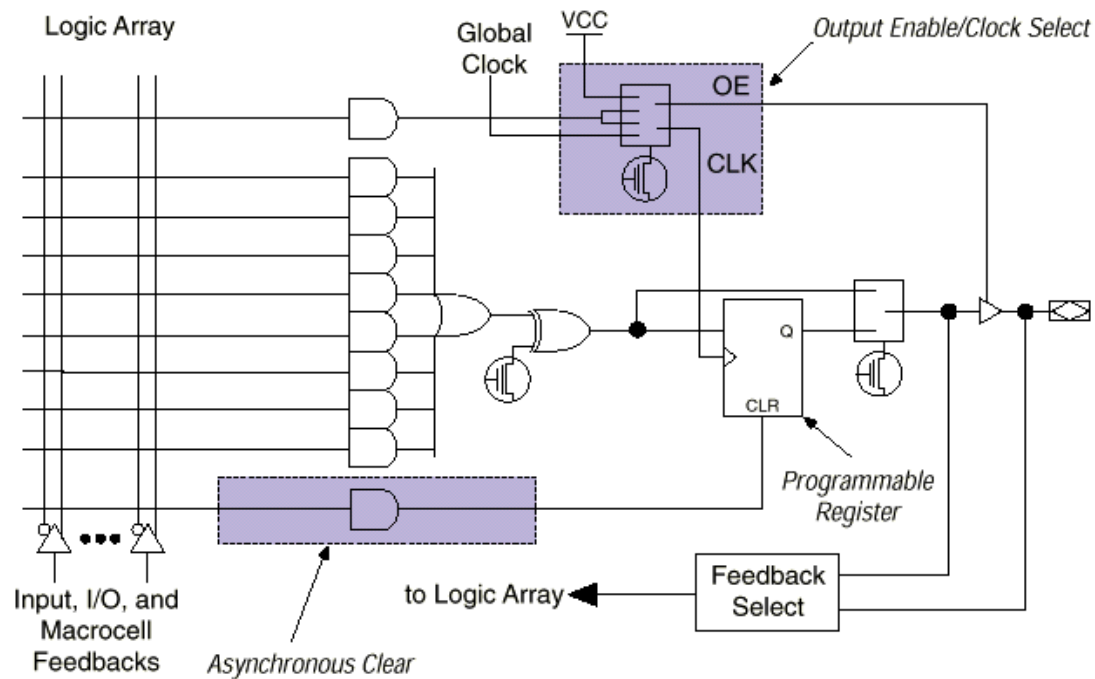
- Programmable logic devices are a class of 'chips' that can be programmed to perform some system function

Logic Block Diagram (PDIP/CDIP) and Pin Configurations



Programmable Logic Devices

- Programmable logic devices are a class of ‘chips’ that can be programmed to perform some system function
- Example- Altera EP910

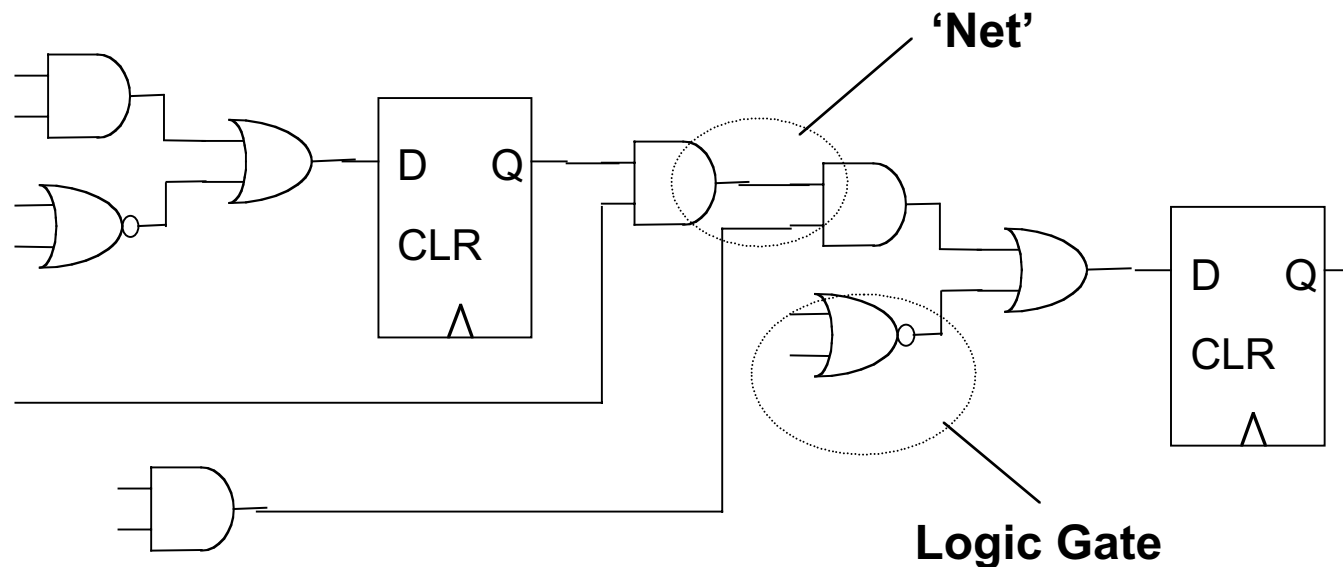


Problems with Simple PLD's

- Size
 - Simple PLD's typically have a small gate count (<1000 gates)
 - Simple combinatorial logic and registers
 - Complex devices available- suitable for some applications
- Functionality
 - Circuit typically only involves sum of products representation
- Programming
 - Often cumbersome
- For these reasons, we need something a bit better
 - Ideally a chip which can implement any given digital system

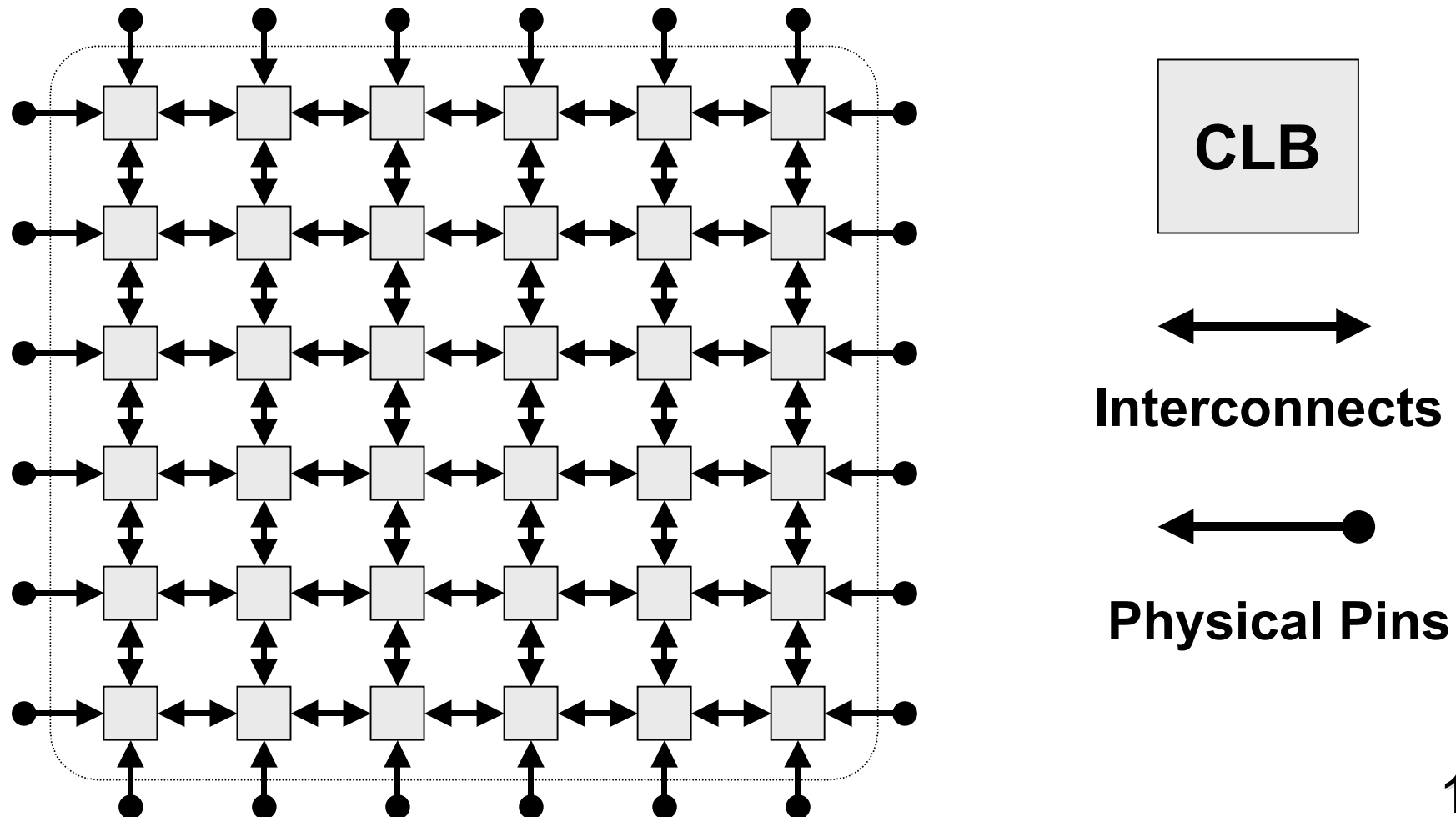
Anatomy of a Digital Logic Circuit

- Digital Logic Circuit (however complex) is just a bunch of primitive gates, FF's, etc. connected together
 - Connections are termed *nets*



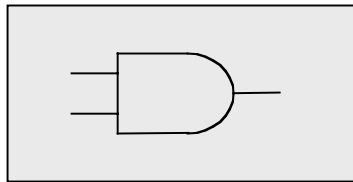
Anatomy of a Programmable Device

- Basically, our programmable devices look as follows:

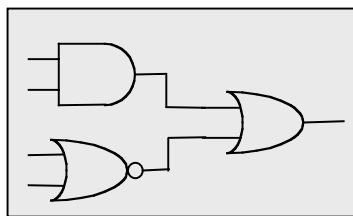


Terminology- I

- CLB= Configurable Logic Block
 - Degree of ‘complexity’ is called the *granularity*
 - Can implement a single logic function (fine grained)



- Or a block of logic functions (course grained)



- Also, may incorporate registers (‘D’ Type FF’s)

Terminology- II

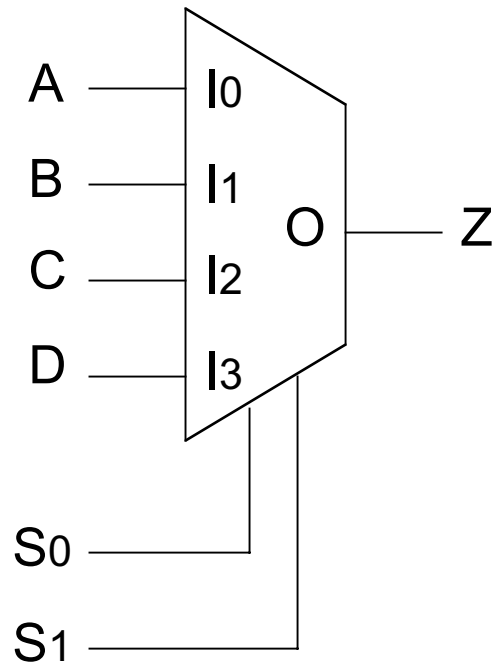
- Interconnects
 - Used to connect the CLB's together
 - May be one (or more) interconnects between 2 cells
 - Also, different levels of interconnect (e.g. global for clock distribution)
- Physical Pins
 - Used for inputs to programmable device (drives CLB's)
 - Used for outputs from programmable device (driven by CLB's)
 - May incorporate additional features (e.g. Tri-State etc.)

Designing a Device- Introduction

- Our goal is to design a *fine-grained* programmable logic device
- We'll consider gate-level connections
 - Not really interested in fabrication details
- Mainly concentrating on architecture and implementation details
 - How do we do the routing and how do we build the CLB ?

Background Topics- I

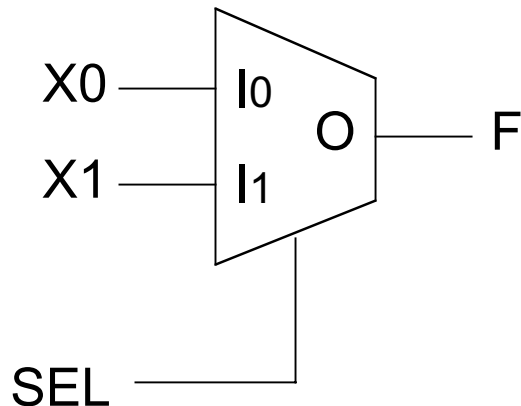
- Multiplexers
 - Used as data selector (4 ->1)



S1	S0	Z
0	0	A
0	1	B
1	0	C
1	1	D

Background Topics- II

- Universal Logic Module (ULM)
 - Based on 2->1 multiplexer
 - Permits many different logic functions of 2 variables by selection of X0, X1 & SEL
 - Choice of connection assignment dictates logic function



From truth-table,
$$F = \overline{SEL} \cdot X0 + SEL \cdot X1$$

Background Topics- III

- ULM Examples

Implement, $F=A \cdot B$

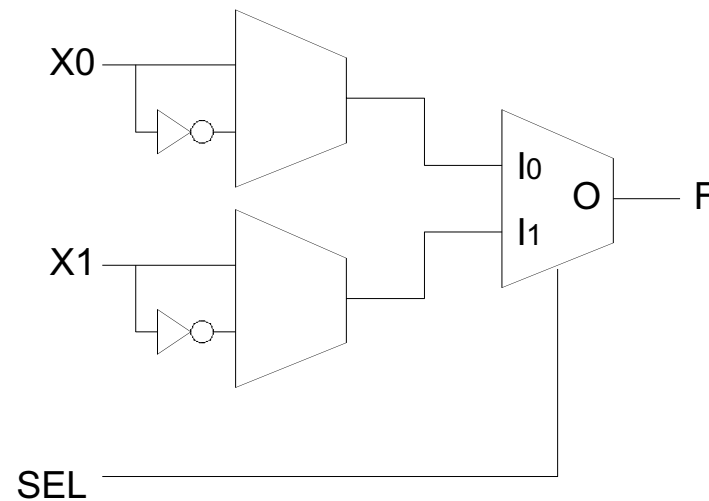
$$\left. \begin{array}{l} \text{SEL} = A \\ X0 = A \\ X1 = B \end{array} \right\} \begin{array}{l} F = \overline{\text{SEL}} \cdot X0 + \text{SEL} \cdot X1 \\ = \overline{A} \cdot A + A \cdot B \\ = 0 + A \cdot B \\ = A \cdot B \end{array}$$

Implement $F=A \oplus B$

$$\left. \begin{array}{l} \text{SEL} = A \\ X0 = B \\ X1 = \text{not}(B) \end{array} \right\} \begin{array}{l} F = \overline{\text{SEL}} \cdot X0 + \text{SEL} \cdot X1 \\ = \overline{A} \cdot B + A \cdot \overline{B} \\ = A \oplus B \end{array}$$

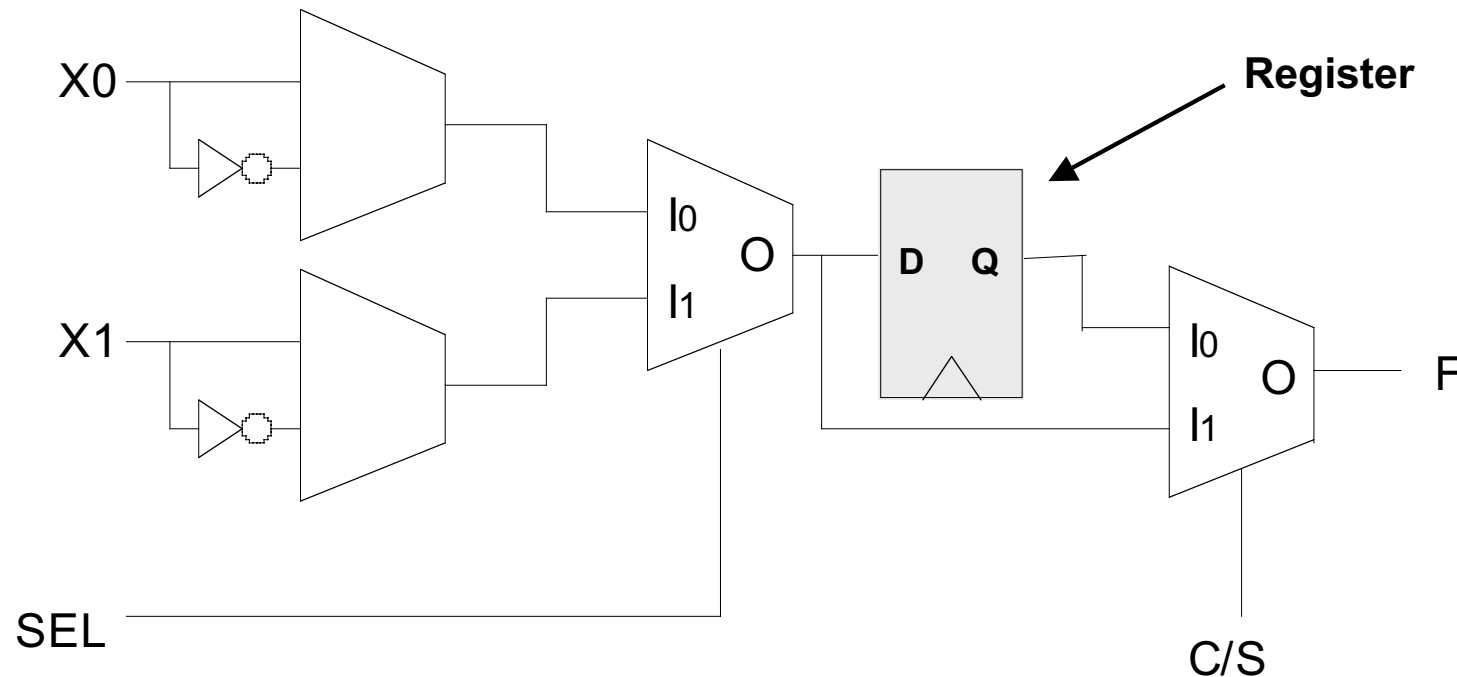
Design of Config'ble Logic Block (CLB)

- We will use a universal logic module as the basis of our CLB
 - Add a small amount of peripheral logic around the ULM to provide inverts etc.
 - e.g. in XOR example above, needed \overline{B}



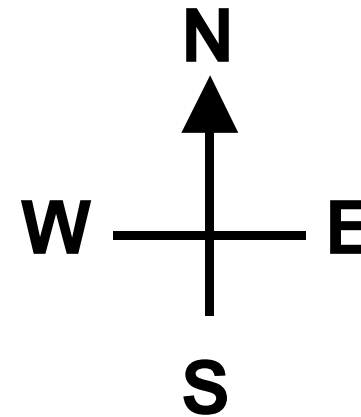
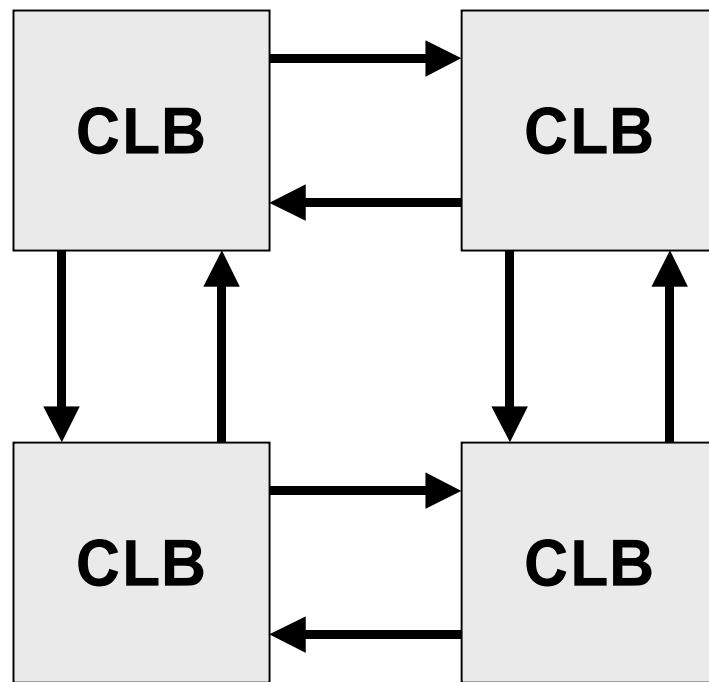
Final Configurable Logic Block (CLB)

- Also, require provision for registers in the system
 - C/S controls either combinational or synchronous output



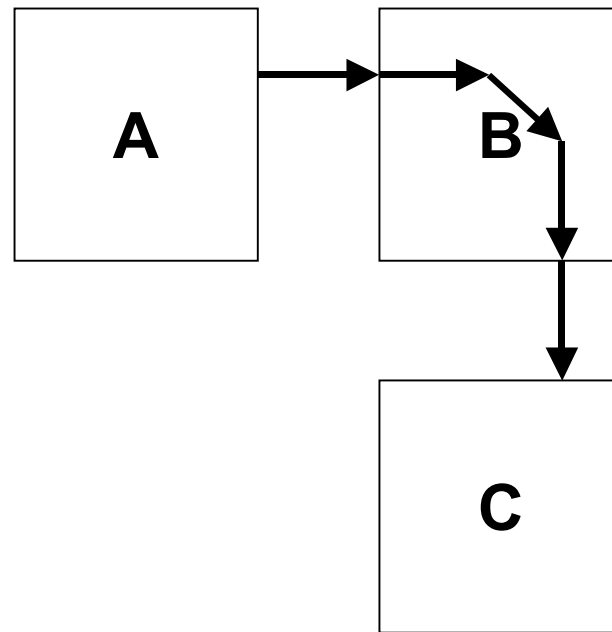
Routing- Introduction

- Our basic device will offer only local routing
 - So can input and output from adjacent cells only
- Example



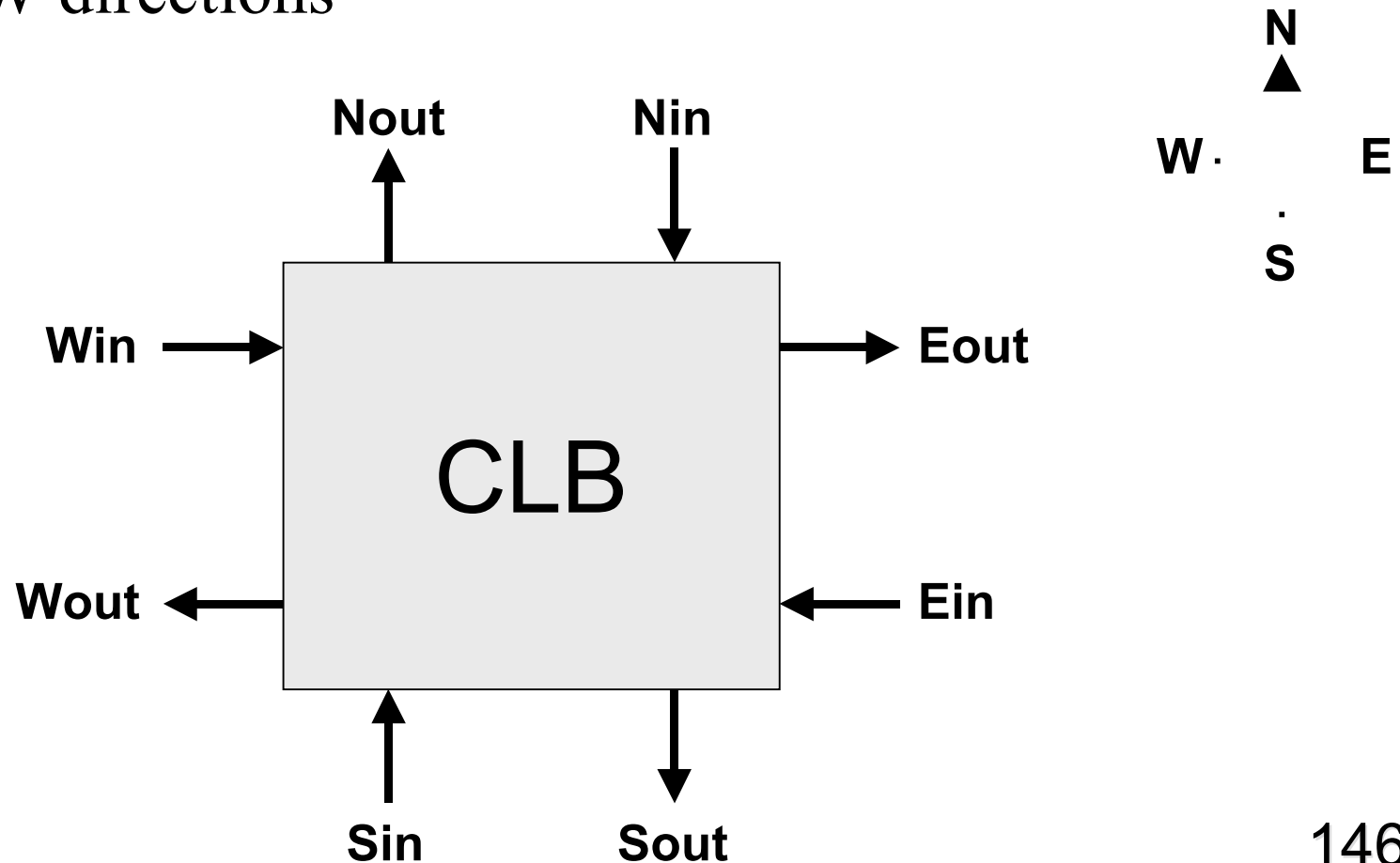
Routing Requirements

- Require routing resources to permit routing between non-adjacent cells
- Example- want to route from cell 'A' to cell 'C'
 - This is achieved by routing through cell 'B'



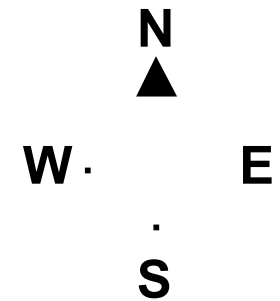
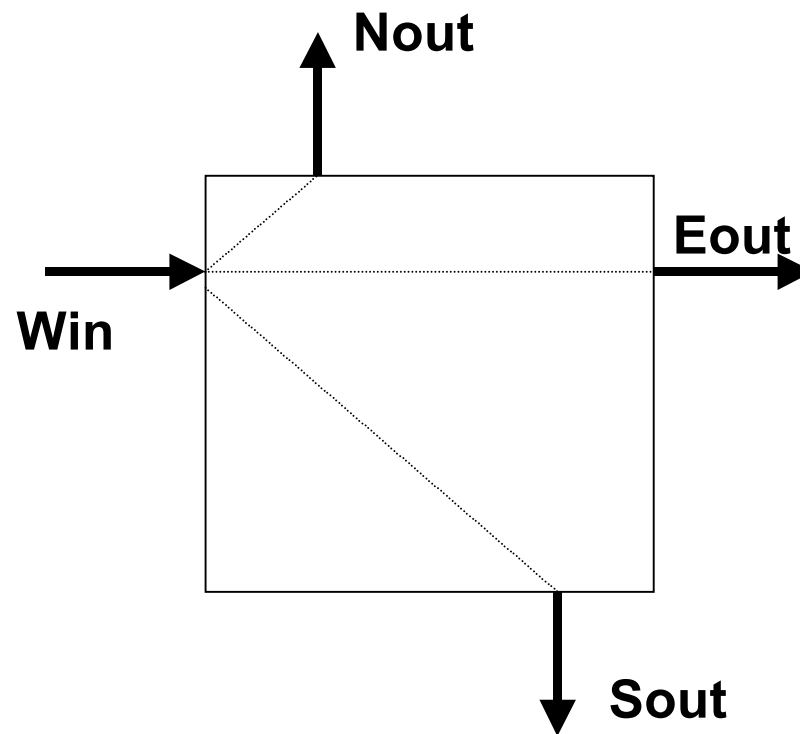
Routing Detail- I

- So, each cell has inputs and outputs in each of the N, S, E, W directions



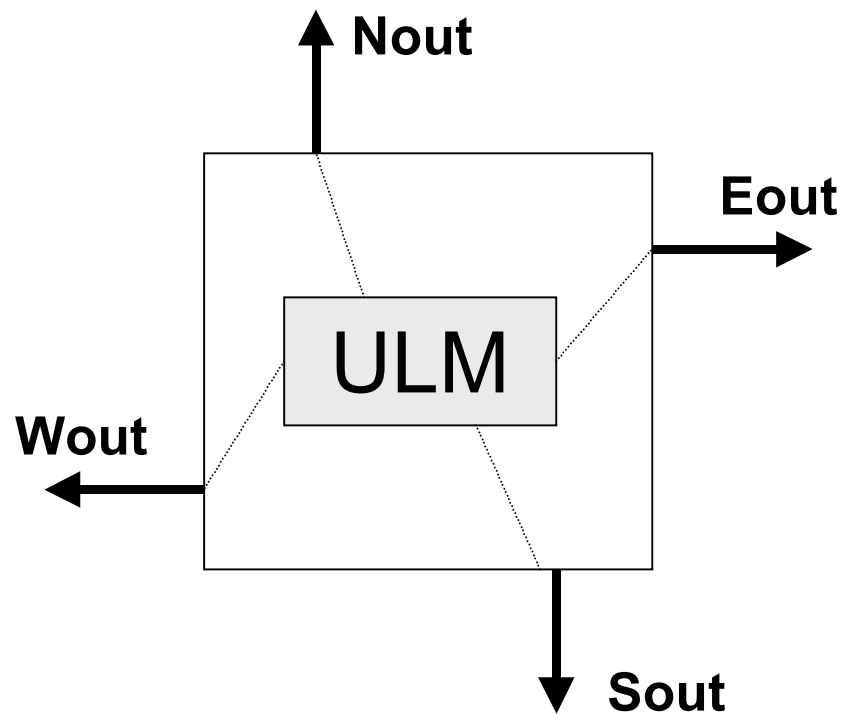
Routing Detail- II

- Require routing to be able to route from any adjacent cell to any other locally adjacent cell
 - This permits routing between non-adjacent cells



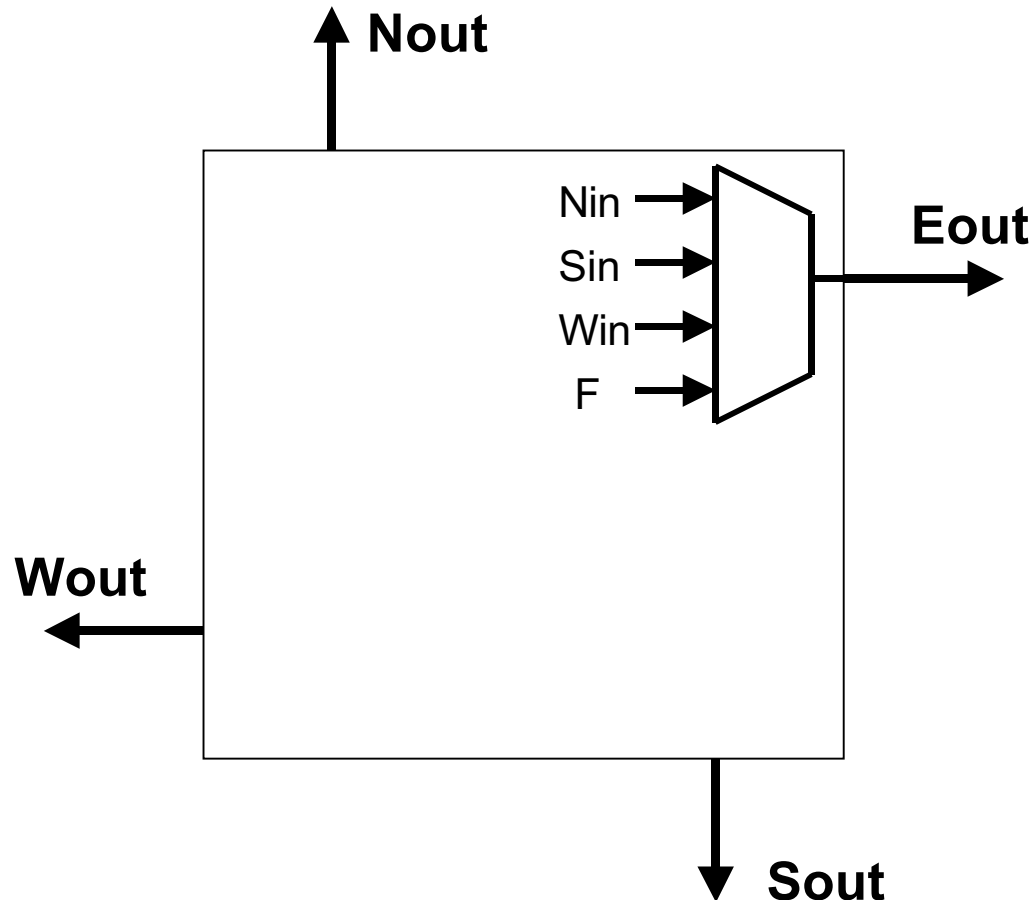
Routing Detail- III

- Also, drive any combination of the N, S, E, W outputs from the output of the ULM (F)



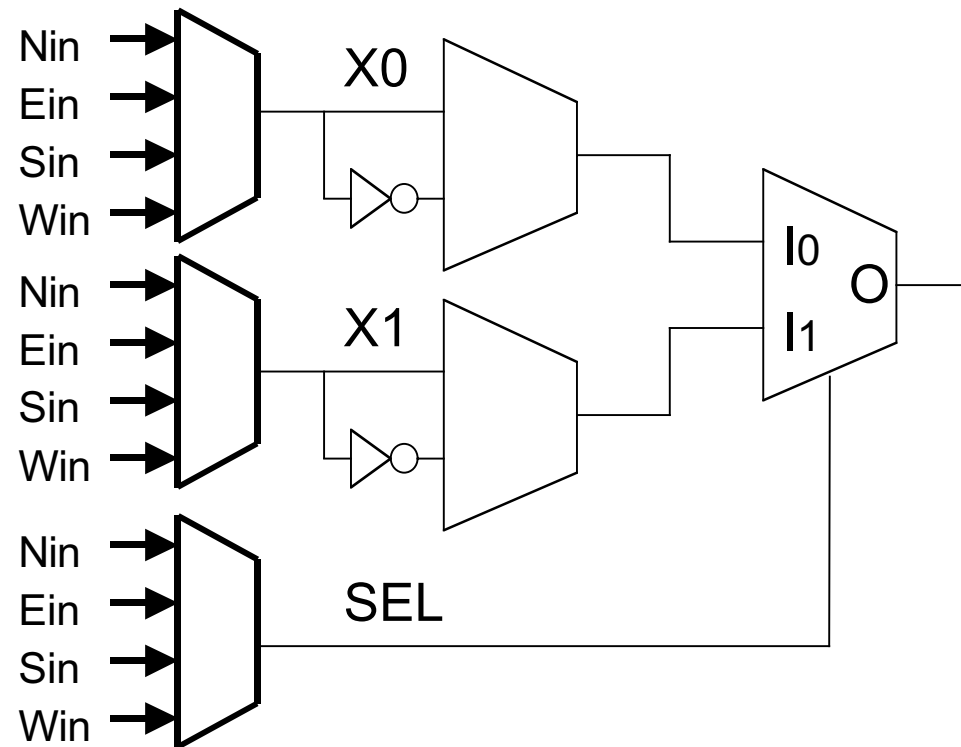
Routing Implementation

- Implementation is fairly trivial using multiplexers
 - 4->1 mux on each of the N, E, S, W outputs

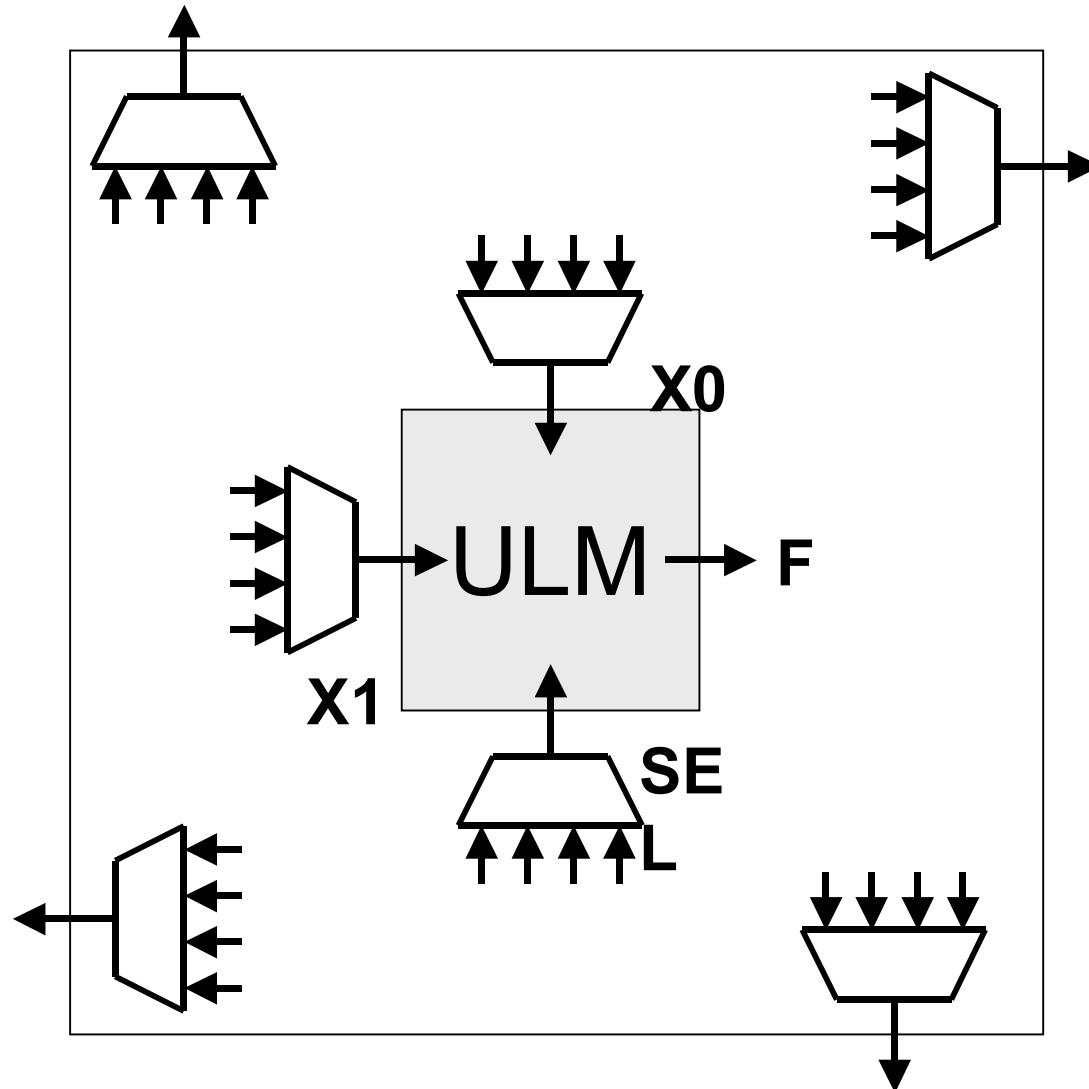


Routing to ULM

- So, how are the inputs to the ULM controlled ?
 - Again with multiplexers
 - X0, X1, SEL driven by individual mux selecting N, S, E, W

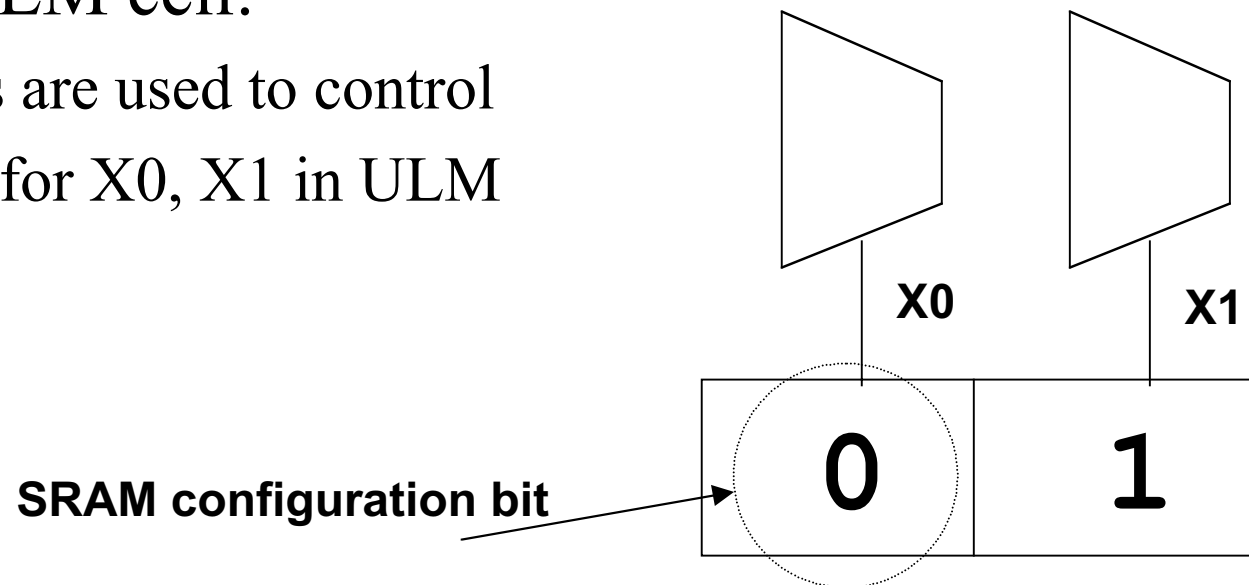


Final CLB Configuration



Configuring the Device- I

- How is the device configured ?
- Each of the multiplexers are controlled by SRAM configuration bits
- Example, ULM cell:
 - Here 2 bits are used to control the mux's for X0, X1 in ULM



Configuring the Device- II

- For each CLB, there are 17 programming (configuration) bits
- 2 for each of the CLB output multiplexers (8)
- 1 each for X0,X1 selection multiplexers (2)
- 2 for each of the ULM input multiplexers (6)
- 1 for combinatorial/ sequential multiplexer (1)
- So, entire device can be configured simply by programming the desired bitstream (generated automatically by software) in ‘The Field’
 - Field Programmable Gate Array

Summary of Device

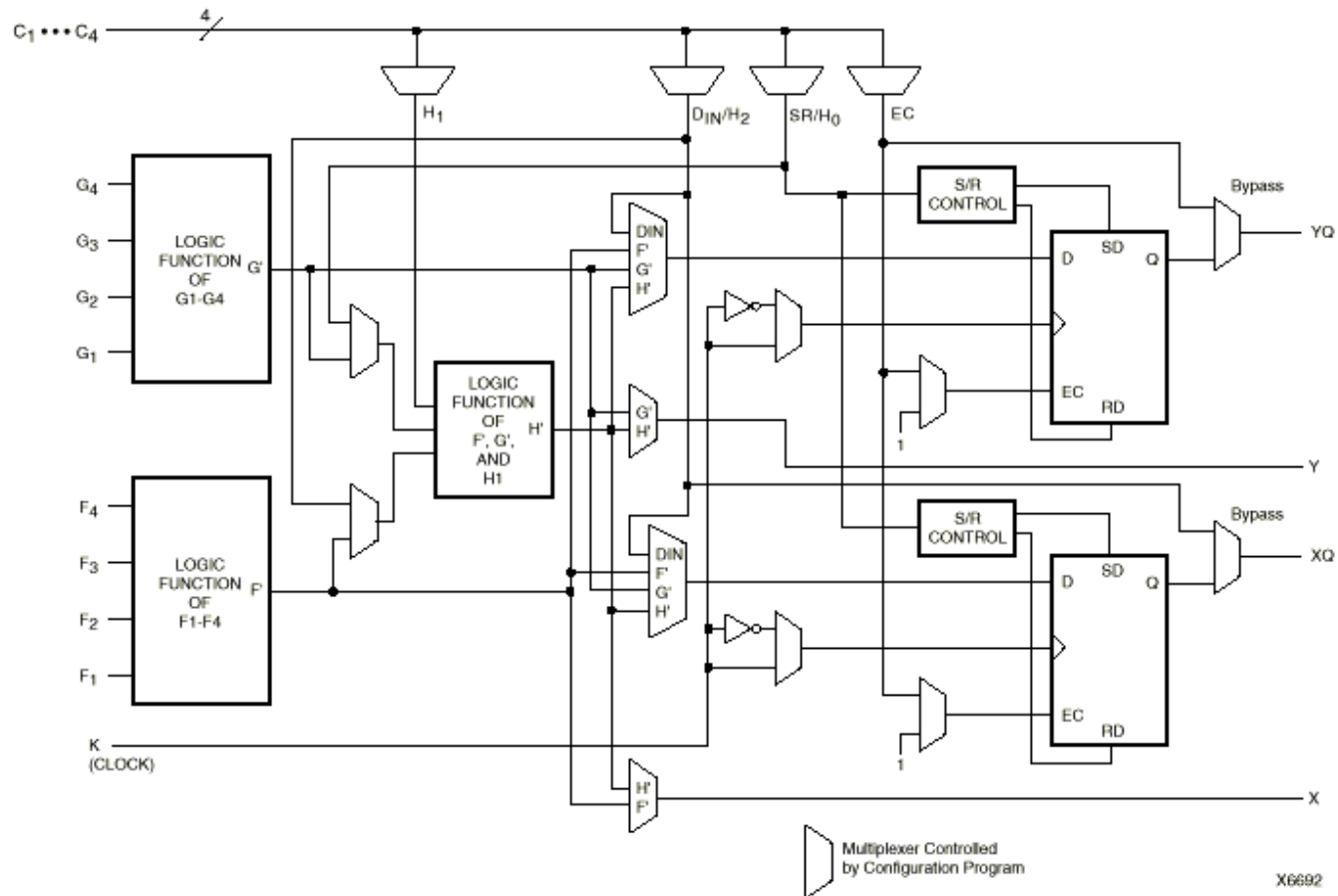
- We have developed a simple fine-grained device
- Note the regular structure
 - We can have any number of CLB's on a physical device- simply add more cells onto the mask
- So, does it work ?
- The above is really a derivation of a commercial device called a Xilinx XC6200
 - This device has a number of additional features (e.g. more routing resources etc.)
 - But the CLB structure and local routing etc. is very similar

Commercial Devices- I

- So, why are there fine and course grained devices ?
- Depends on the target application
- Our fine-grained device has a high overhead in terms of routing
 - CLB's only have 1 logic function and require lots of routing between CLB's
- Often common to include lots of combinational logic within a single CLB
 - Use RAM based look-up-tables (LUT's) to implement logic

Commercial Devices- II

- Course grained device example- Xilinx 4000 series



Conclusion

- Provided introduction to advanced programmable logic devices
- Derived a fine-grained device from first principles
- Touched on some commercial devices
- The future
 - Devices already in design/ production offering > 1 Million equivalent gates
 - Reconfigurable computing- custom processor using FPGA connected to normal CPU to accelerate computationally intensive tasks

