

# FPGAs: Here we go

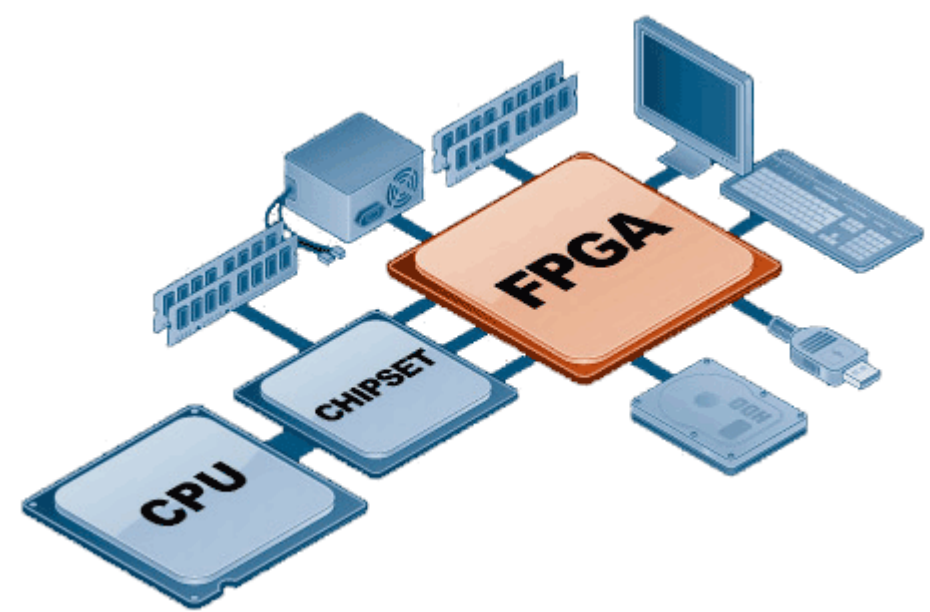
Chris Roszbach

cs378 Fall 2018

10/29/2018

# Today

- Questions?
- Administrivia
  - Exam Wednesday
  - Start thinking about projects!
- Agenda
  - FPGAs: ~45 minutes
  - Exam Review: ~30 minutes

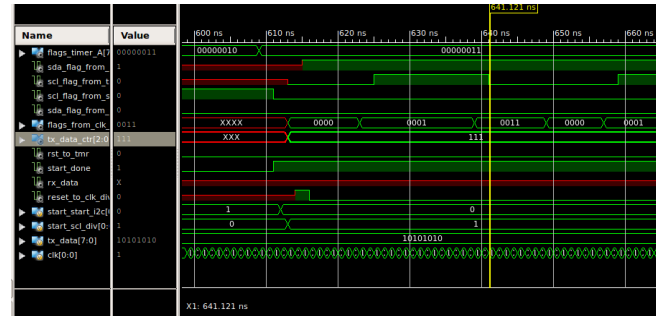


## Acknowledgements:

- [http://www.ee.unlv.edu/~yingtao/2012\\_Spring/ECE720/student%20presentations/Introduction\\_to\\_Field\\_Programmable\\_Gate\\_Arrays.pptx](http://www.ee.unlv.edu/~yingtao/2012_Spring/ECE720/student%20presentations/Introduction_to_Field_Programmable_Gate_Arrays.pptx)
- [http://rtds.cse.tamu.edu/wp-content/uploads/2013/03/fpga\\_intro.pptx](http://rtds.cse.tamu.edu/wp-content/uploads/2013/03/fpga_intro.pptx)
- [ftp://ftp.altera.com/up/pub/Courses/Korea\\_2016/t0\\_intro.pptx](ftp://ftp.altera.com/up/pub/Courses/Korea_2016/t0_intro.pptx)
- <https://www.slideshare.net/mobile/TaylorRiggan/a-primer-on-fpgas-field-programmable-gate-arrays>

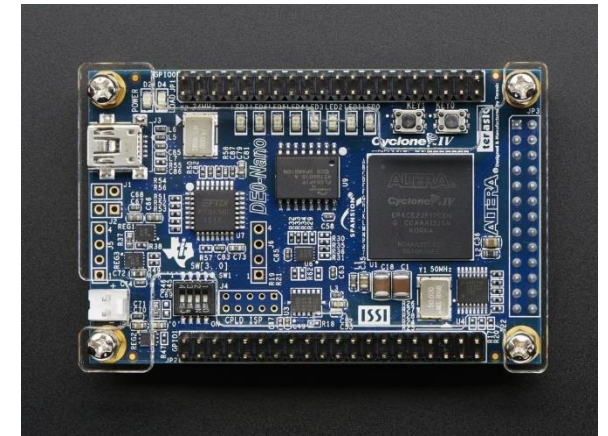
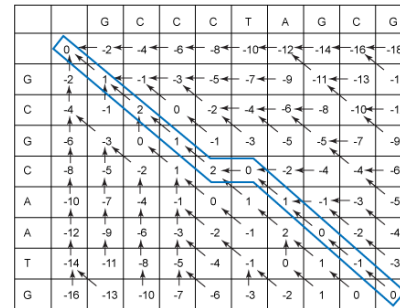
# Faux Quiz Questions

- Why/when might one prefer an FPGA over an ASIC, CPU, or GPU?
- Define CLB, BRAM, and LUT. What role do these things play in FPGA programming?
- Describe the FPGA build process; which phases are relatively short vs. relatively long?



# FPGA Unit Overview

- The Hardware – FPGAs
- The Software – HDL + Verilog
- The Project – Needleman-Wunsch
- The Research – Cascade + FPGA Programmability

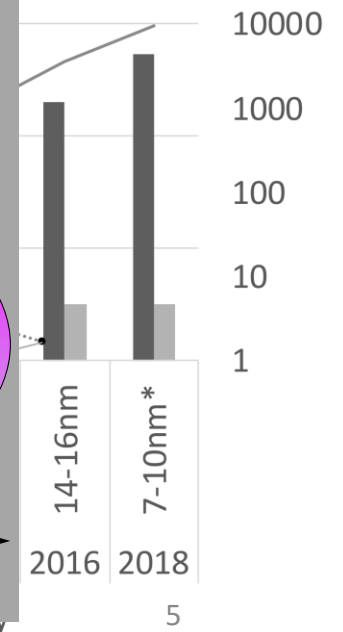
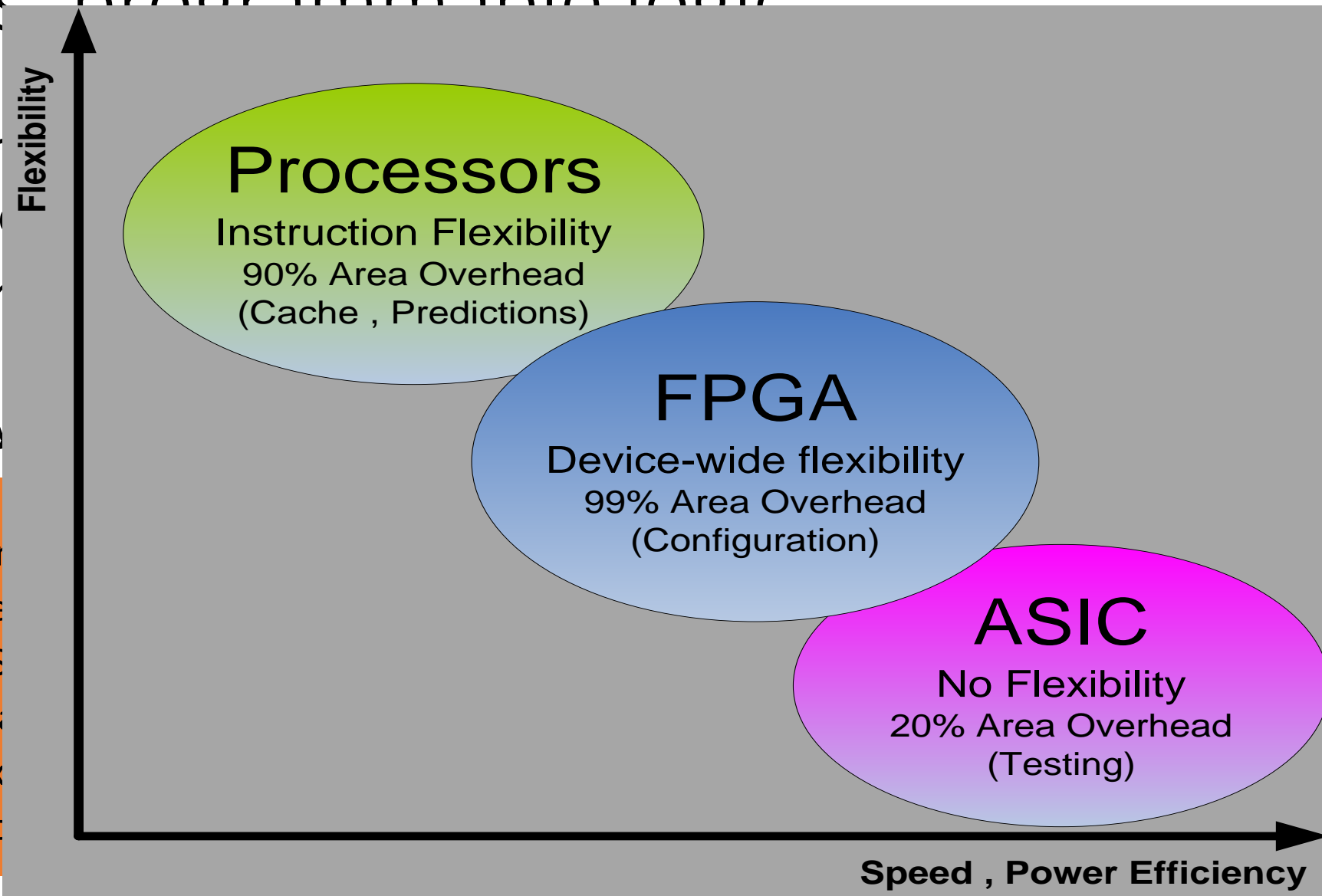


# FPGAs: programmable logic

- Field Programmable Gate Array (FPGA)
  - “Field Programmable”
- Gate Array
  - Gate Array
  - Array

## Pros:

- Short Development Time
- Reconfigurable
- No need for ASIC
- Fast time to market
- Bugs can be fixed
- Off the shelf solutions



# Why Study FPGAs in Concurrency?

*(Shouldn't this be in architecture class?)*

- Programmable hardware
  - Can implement arbitrary accelerators
  - Accelerators can be parallel/concurrent
  - Host+Accelerator programming involves parallelism/concurrency
  - *Extreme Heterogeneity*
- FPGAs are everywhere
  - Consumer electronics, networking, telecom
  - Cars, airplanes, trains
  - Medical equipment, and industrial control
- ***FPGAs are highly concurrent and parallel***

# History: PLA

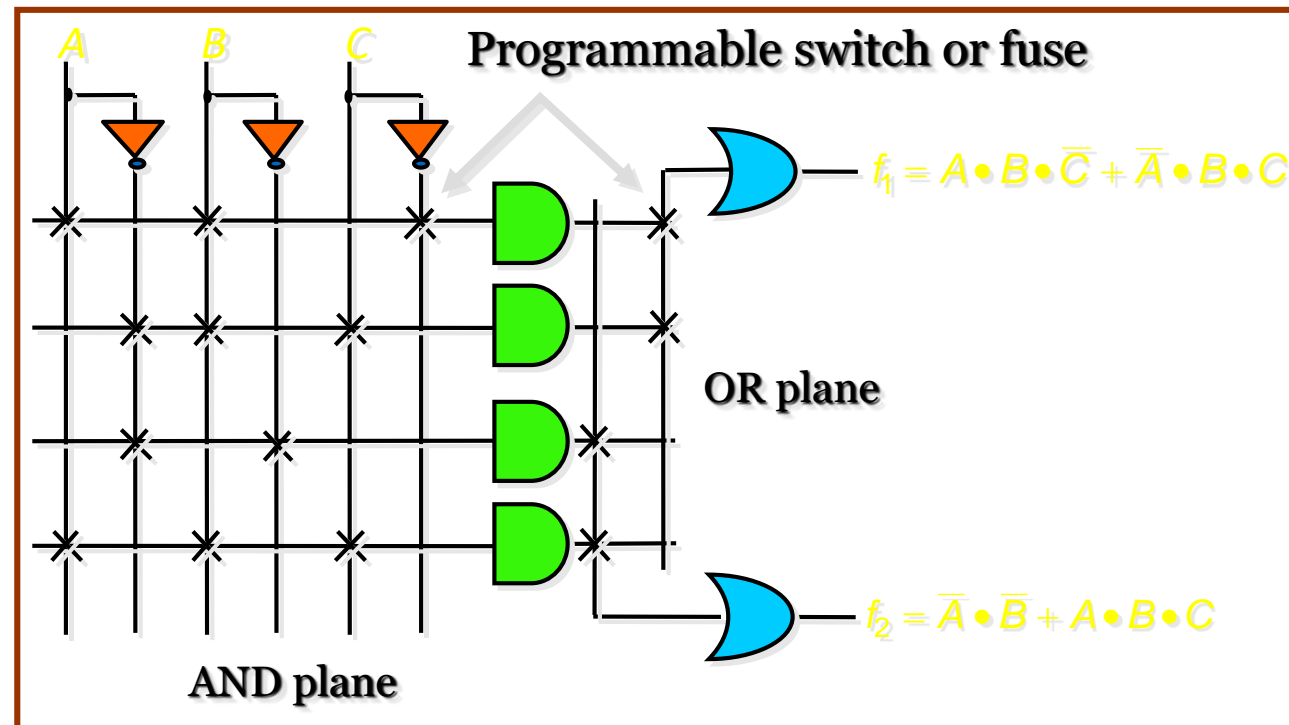
- Programmable Logic Array
- First programmable device
- 2-level and-or structure
- One-time programmable

## Pros

- Simple
- Captures arbitrary combinational logic

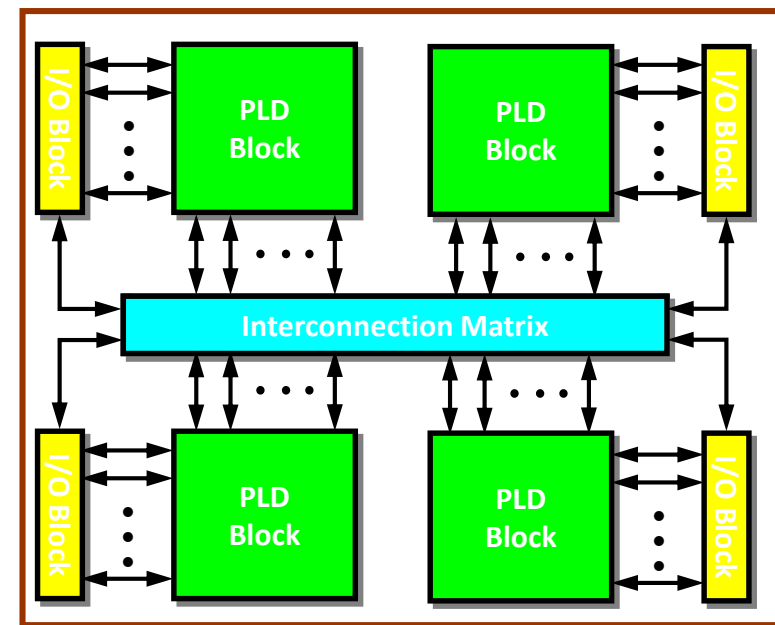
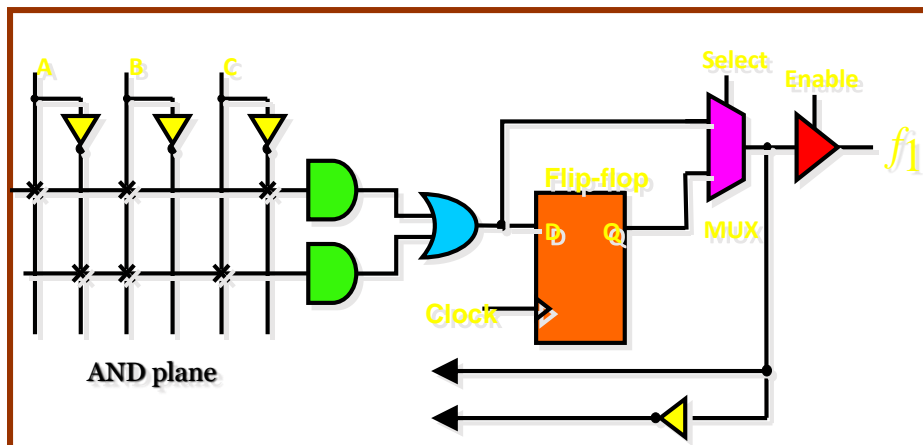
## Cons

- Low level
- stateless



# SPLD - CPLD

- Simple Programmable logic device
  - Single AND Level
  - Flip-Flops and feedbacks
- Complex Programmable logic device
  - Several PLDs Stacked together





# FPGA - Field Programmable Gate Array

- Programmable logic blocks (Logic Element “LE”)

Implement combinatorial and sequential logic. Based on LUT and DFF.

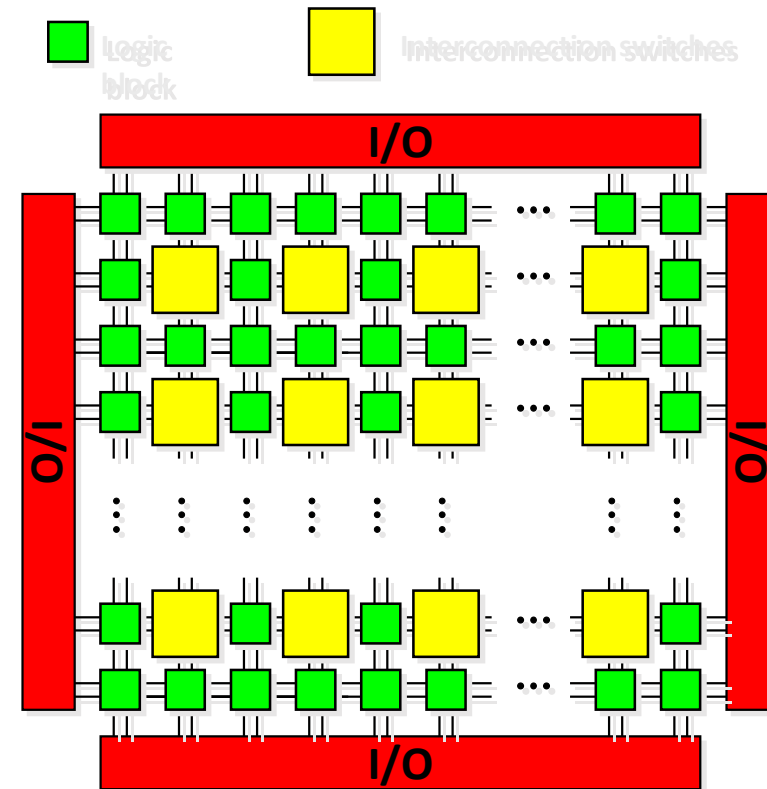
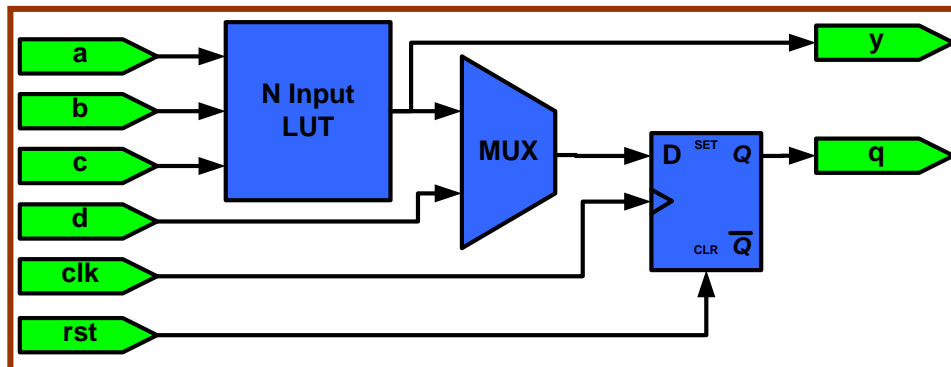
- Programmable I/O blocks

Configurable I/Os for external connections, various voltages, tri-states.

- Programmable interconnect

Wires to connect inputs , outputs and logic blocks.

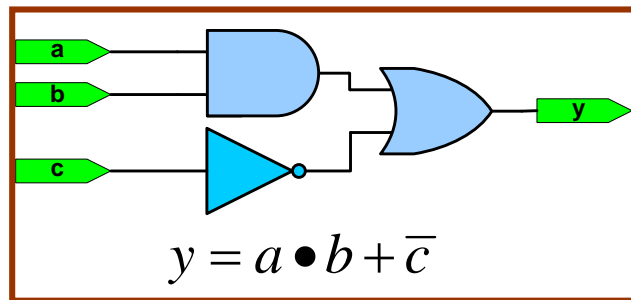
- clocks
- short distance local connections
- long distance connections across chip



# Configuring a Lookup Table

- LUT: RAM with data width of 1bit.
- Contents programmed at power up

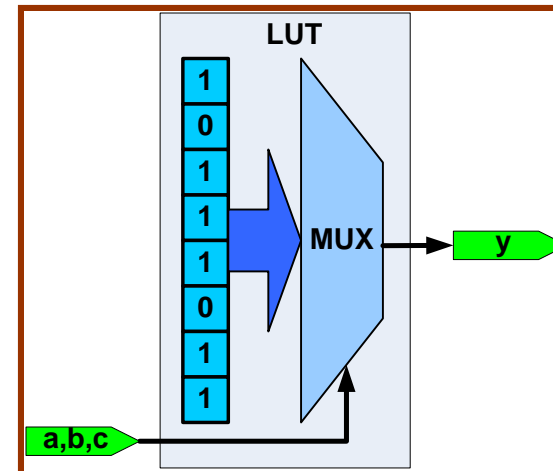
Required Function



Truth Table

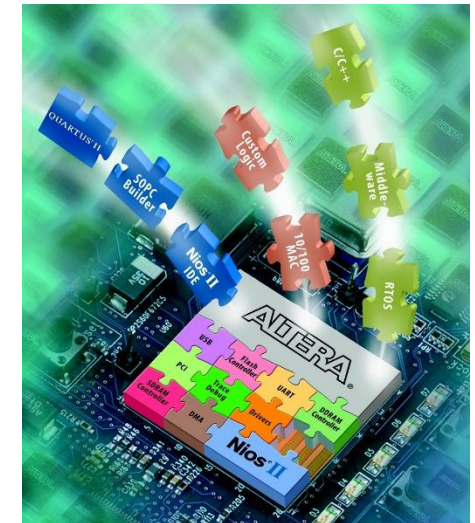
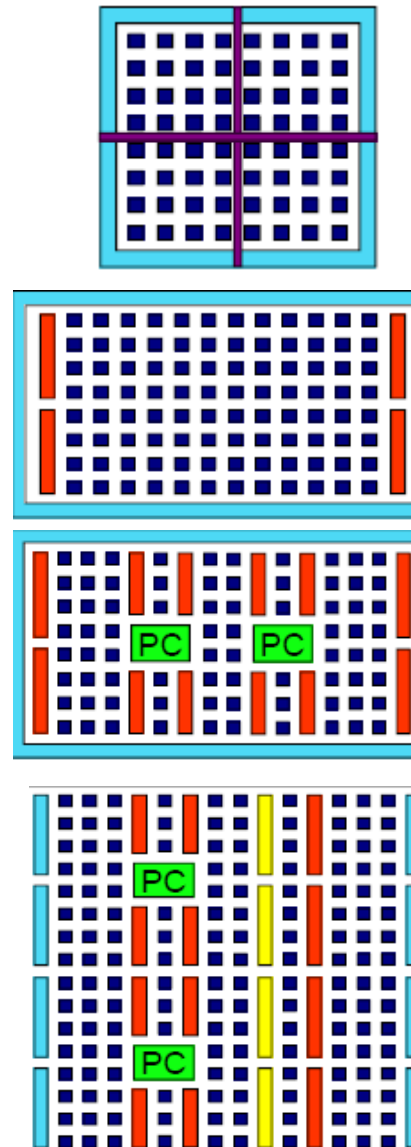
a	b	c	y
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Programmed LUT



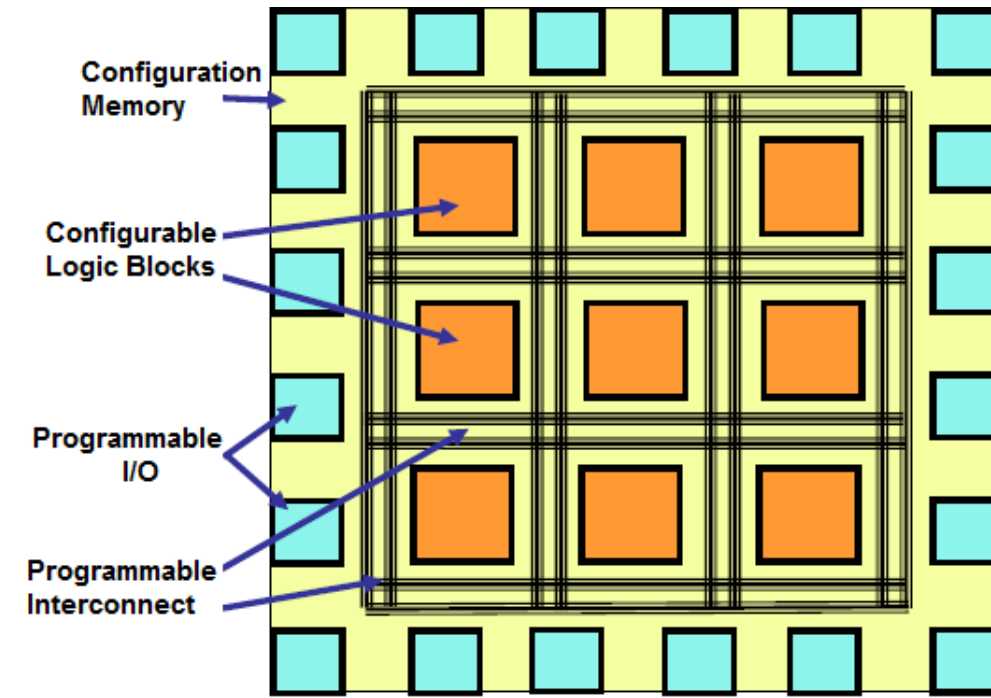
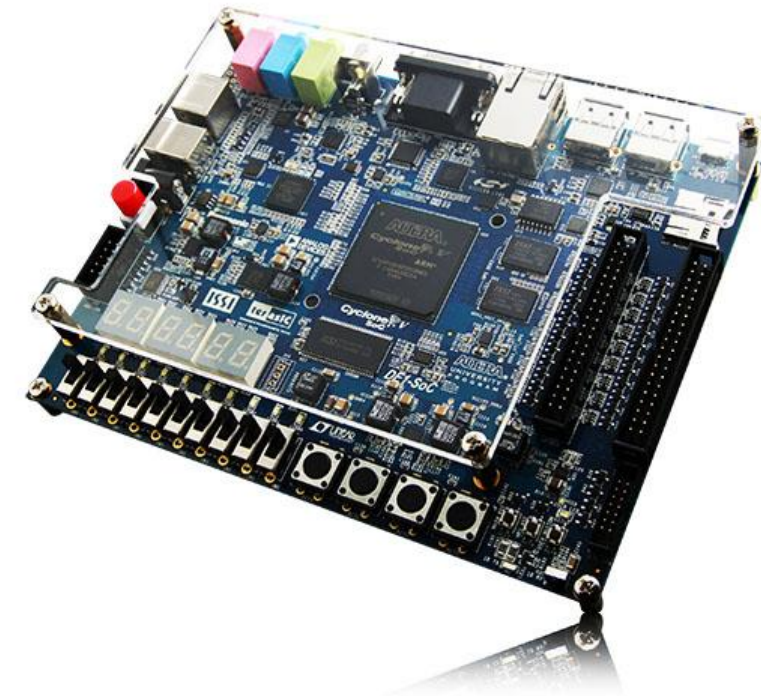
# FPGA Architectures

- Early FPGAs
  - N x N array of unit cells (CLB + routing)
    - Special routing along center axis
- Next Generation FPGAs
  - M x N unit cells
  - Small block RAMs around edges
- More recent FPGAs
  - Added block RAM arrays
  - Added multiplier cores
  - Added processor cores
- Special Functions
  - Internal SRAM
  - Embedded Multipliers, DSP blocks, logic analyzer, CPUs
  - High speed I/O (~10GHz)
  - DDR/DDRII/DDRIII SDRAM interfaces

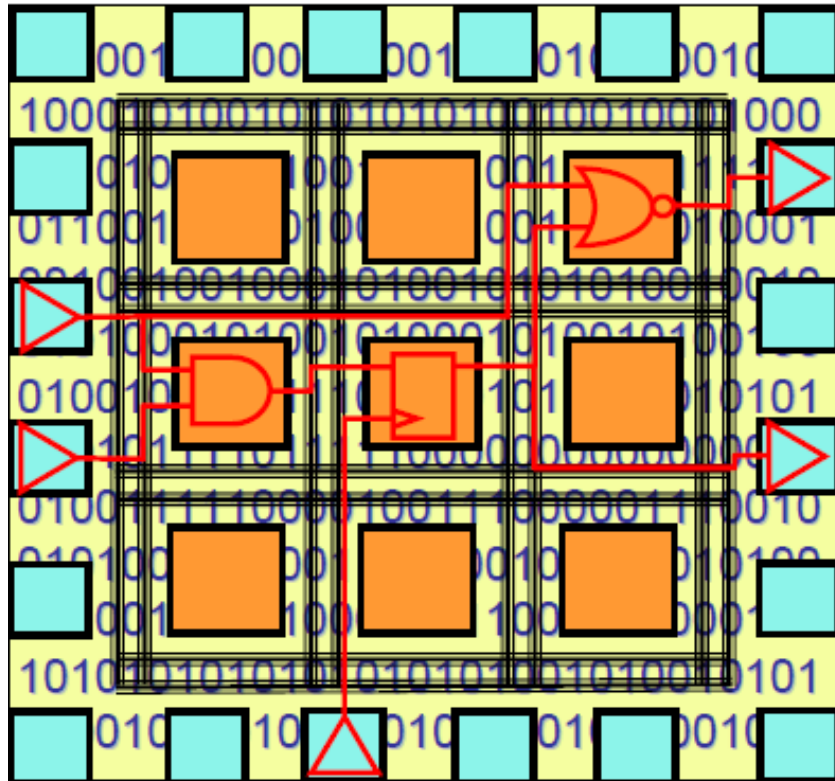


# Basic FPGAs vs DE1\*-SoC

- ~\$100 USD
- Cyclone V SoC FPGA
- Dual-core ARM Cortex-A9
  - 1GB DDR 3 SDRAM, MicroSD
  - USB, Triple-speed Ethernet
  - ADC, Accelerometer, LED, Pushbutton
- FPGA
  - 85K Programmable Logic Elements
  - 64 MB SDRAM
  - DVD-quality audio in/out, Video in/VGA out
  - PS/2, IrDA
  - 4 de-bounced pushbuttons, 10 slider switches, 10 red LEDs, six 7-segment displays
  - Expansion headers
- Built-in USB “Blaster” for FPGA programming



# FPGA Operation



User writes configuration:

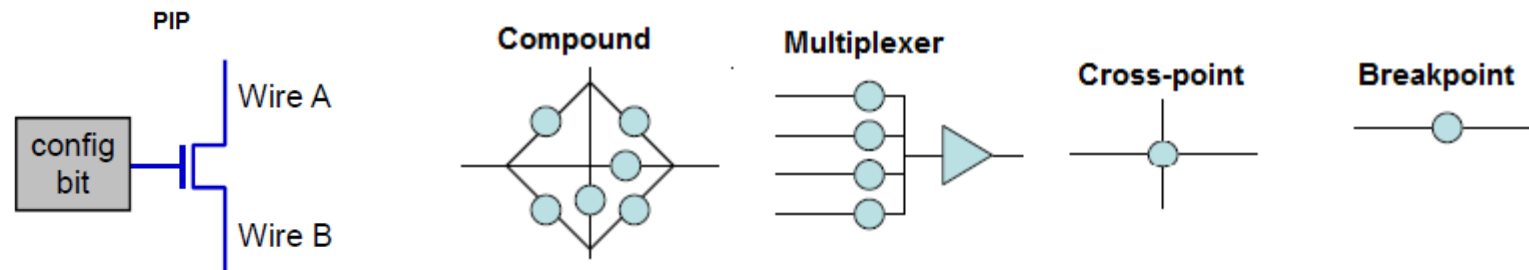
- defines the function of the system
- connectivity between the CLBs and the I/O
- logic to be implemented by CLBs
- I/O blocks.

Changing data in the configuration memory →

- function of the system changes
- can happen at anytime during FPGA operation
- (run-time configuration).

# Programmable Interconnect

- Horizontal and vertical mesh of wire segments
- Interconnected by programmable switches
  - programmable interconnect points (PIPs).
  - PIPs implemented with transmission gate & memory bits from configuration memory.
- Global routing: connect PLBs to I/O buffers, non-adjacent PLBs, etc
- Local routing: connects PLBs to adjacent PLBs and PLBs to global routing



- Types of PIPs
  - Cross-point = connects vertical or horizontal wire segments allowing turns
  - Breakpoint = connects or isolates 2 wire segments
  - Decoded MUX = group of  $2^n$  cross-points connected to a single output configure by n configuration bits
  - Non-decoded MUX = n wire segments each with a configuration bit (n segments)
  - Compound cross-point = 6 Break-point PIPS (can isolate two isolated signal nets)

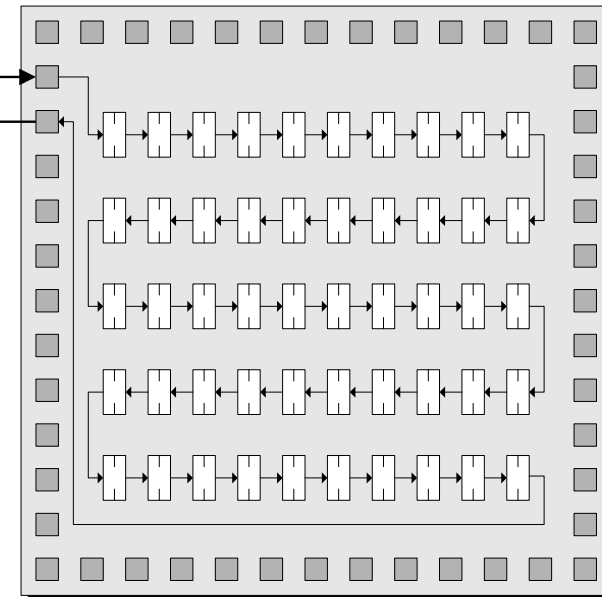
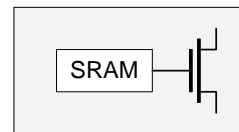
# Programming vs Configuring an FPGA

- SRAM cells holding configuration are Volatile Memory
- Lose configuration when board power is turned off.
- Keep Bit Pattern describes the Logic Functions in non-Volatile Memory e.g. ROM or Compact Flash card
- Reprogramming takes ~ secs
- Uses JTAG Boundary Scan



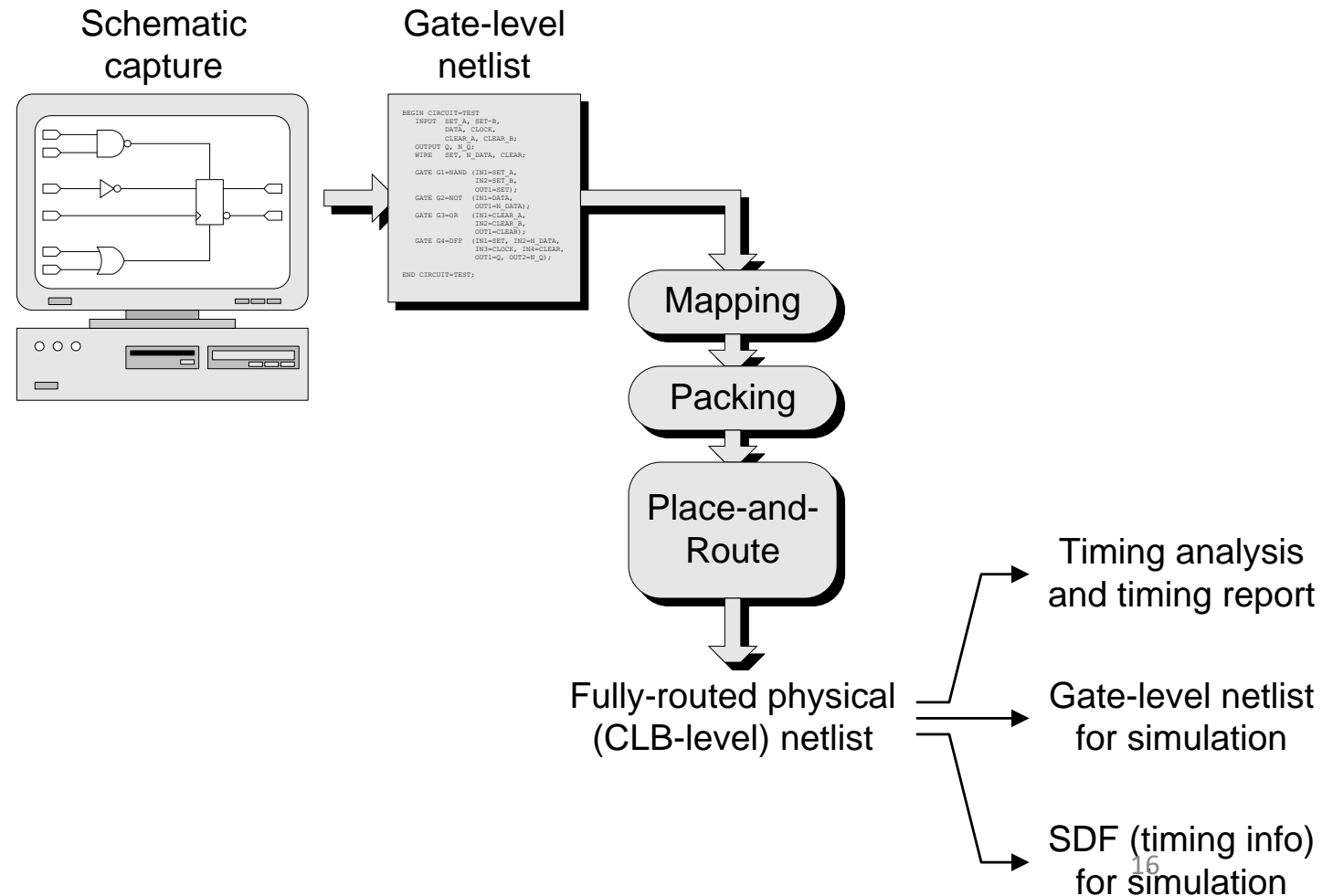
Configuration data in  
Configuration data out

■ = I/O pin/pad  
□ = SRAM cell



# Design/Build Flows

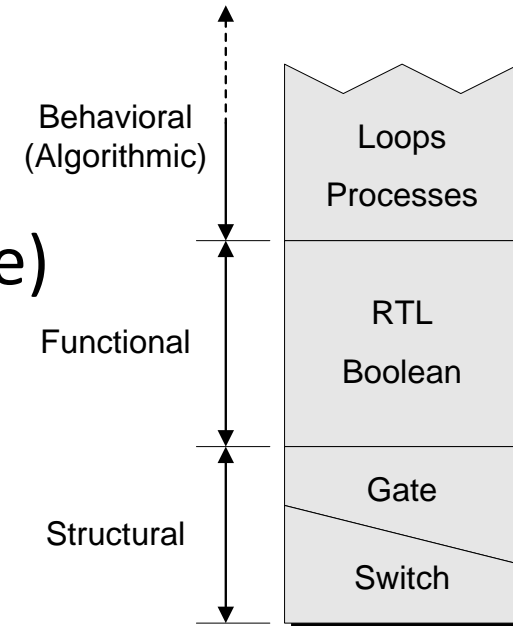
- High level Description of Logic Design
  - Schematic
  - Hardware Description Language
- Compile to netlist
  - Low (Logic Gates) level description.
- Target Netlist to FPGA Fabric
  - Mapping and Packing
  - Placing and Routing
- Tools Generate the Bit File
- Simulation
- Timing Analysis





# Hardware Description Languages

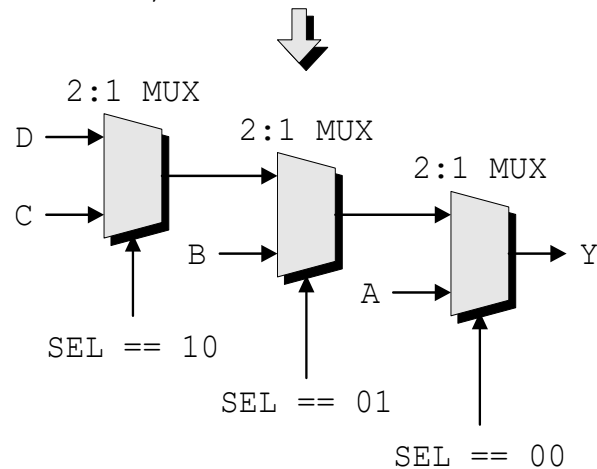
- Behavioural / Register Transfer Level Description
  - Program Statements. Loops. If Statements ...etc
- Describing mixed Combinatorial and Sequential Logic and Signals between.
- VHDL (VHSIC Hardware Description Language)
  - Very High Speed Integrated Circuit
- VERILOG (US)



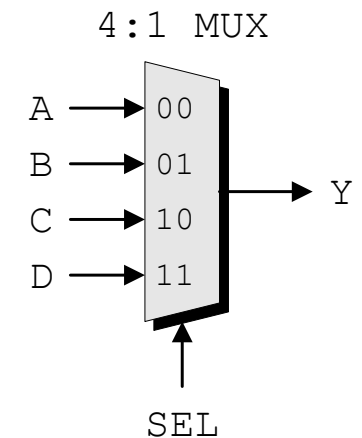
# Hardware Description Languages

- **Logic** → **collection of Processes operating in Parallel**
- Language Constructs for Multiplexers, FlipFlops ...etc
- Restrictive set of RTL for Synthesis
- Synthesis Tools recognise constructs, generate logic

```
if      SEL == "00" then Y = A;  
elseif SEL == "01" then Y = B;  
elseif SEL == "10" then Y = C;  
else  
      Y = D;  
end if;
```

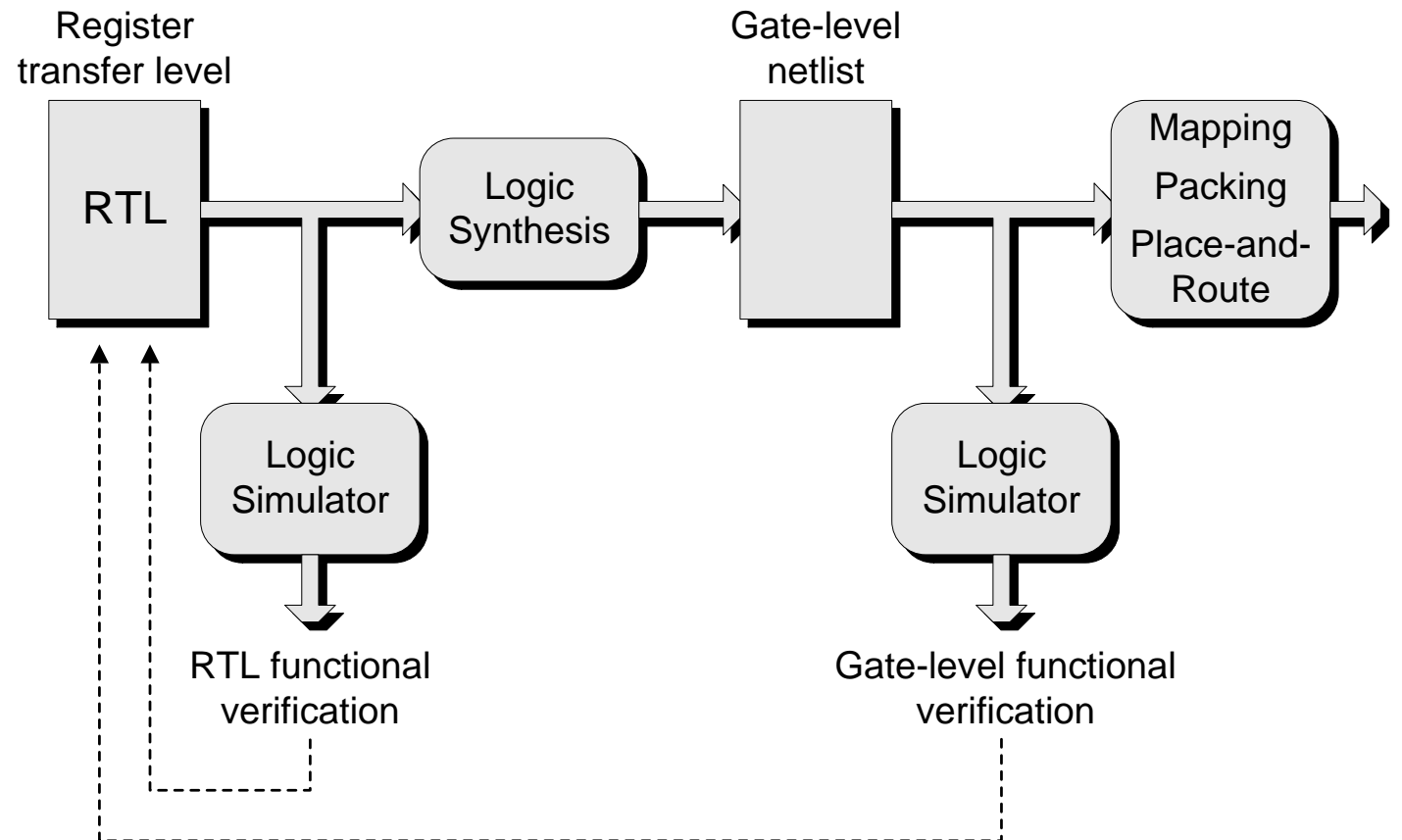
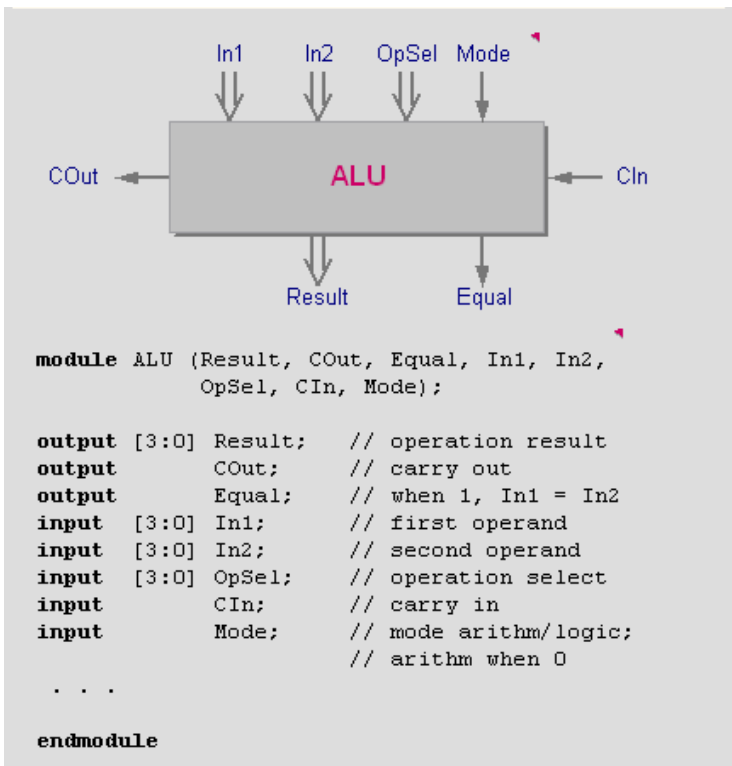


```
case SEL of;  
  "00": Y = A;  
  "01": Y = B;  
  "10": Y = C;  
otherwise: Y = D;  
end case;
```



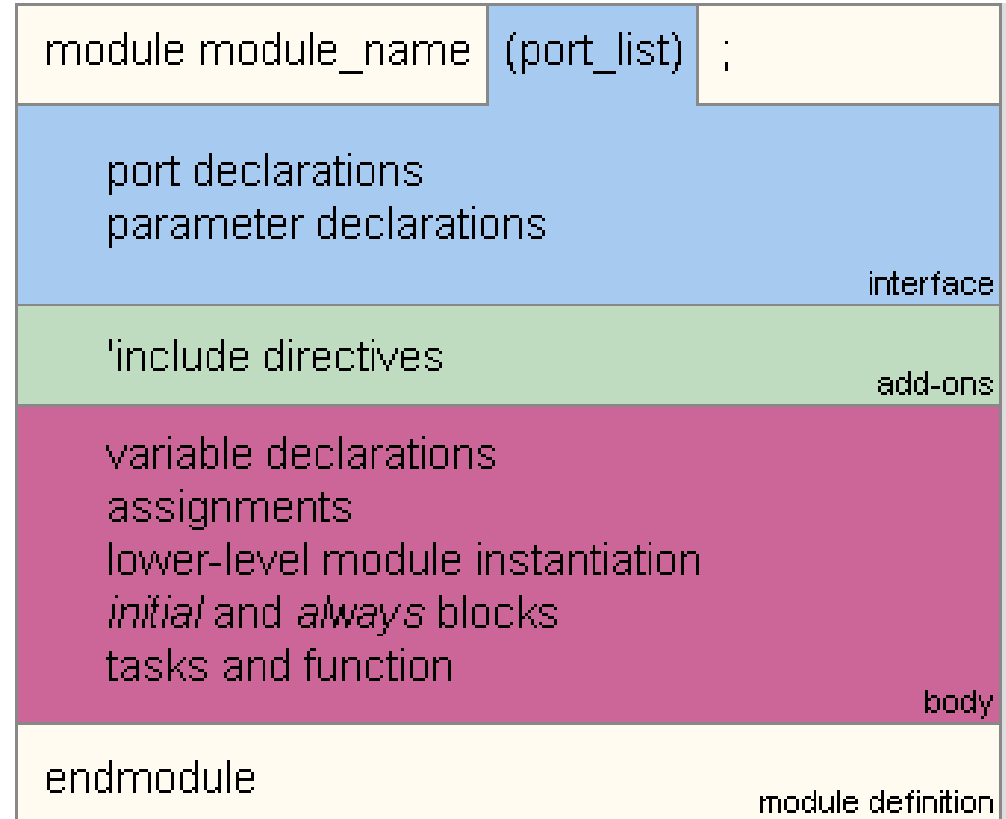
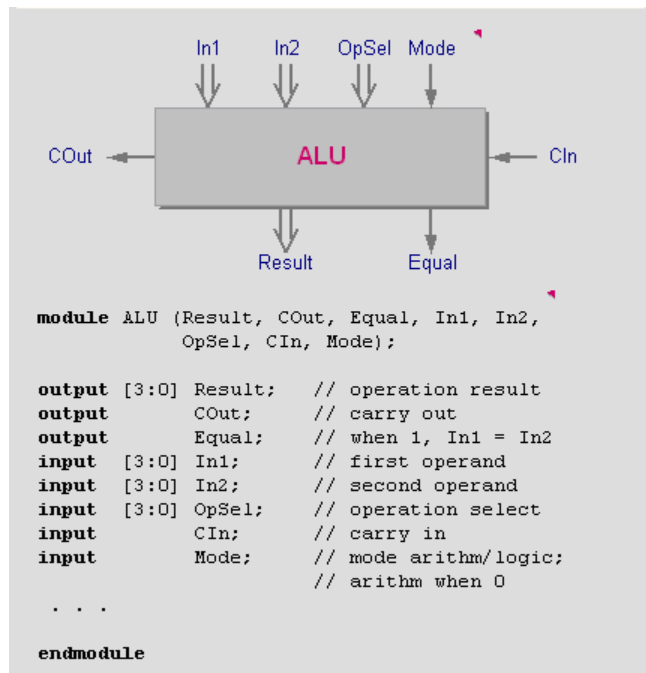
# Hardware Description Languages

- Synthesis (Compilation)
- Generate Netlist



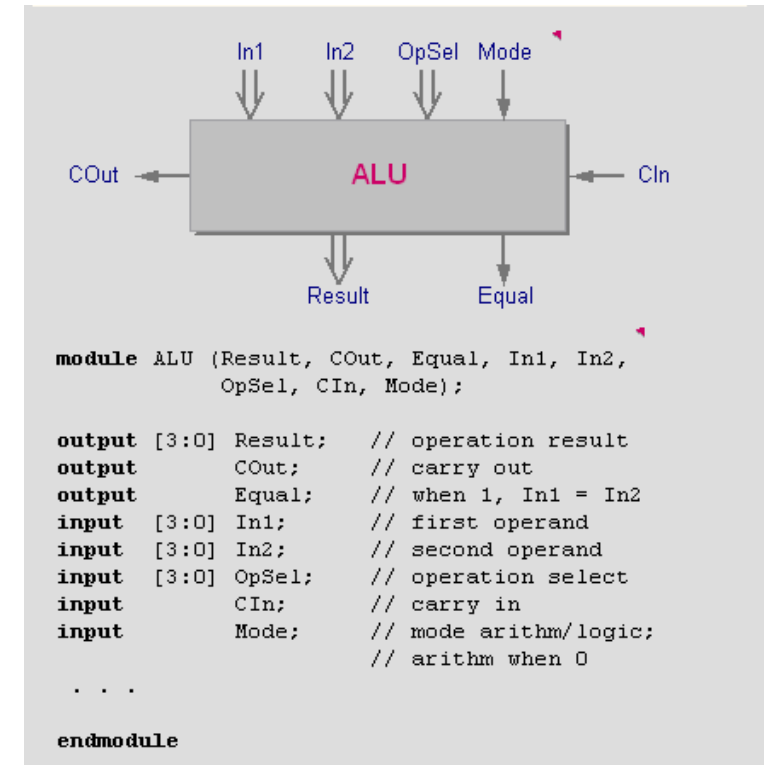
# Definition of Module

- Interface
  - port and parameter declaration
- Body: Internal part of module
- Add-ons (optional)



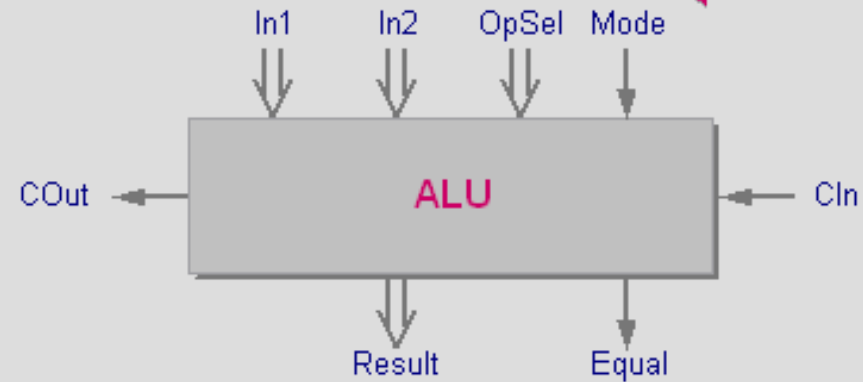
# Basic Structure

- The name of Module
- Comments in Verilog
  - One line comment (// .....)
  - Block Comment (/\* .....\*/ )
- Description of Module (optional but suggested)



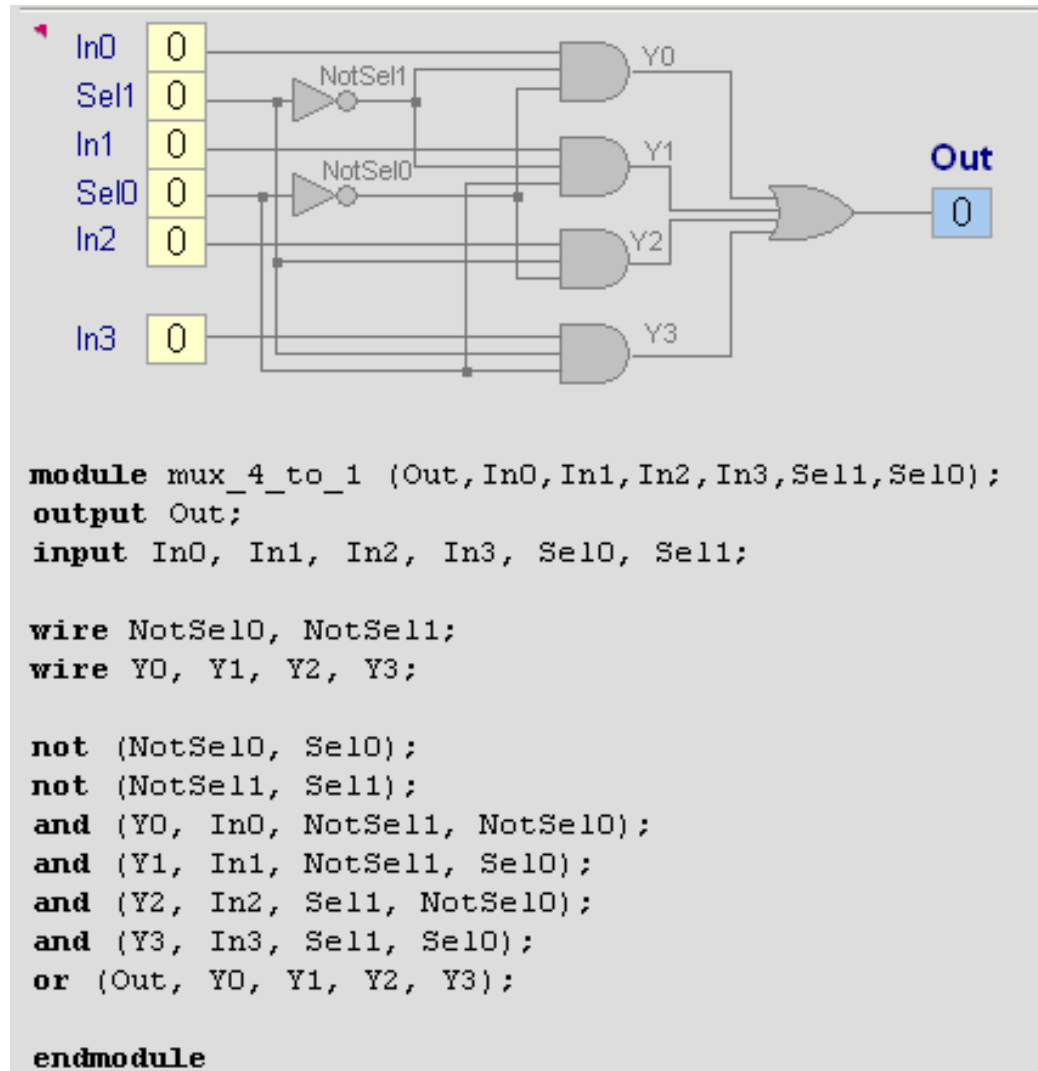
# The Module Interface

- Port List
- Port Declaration

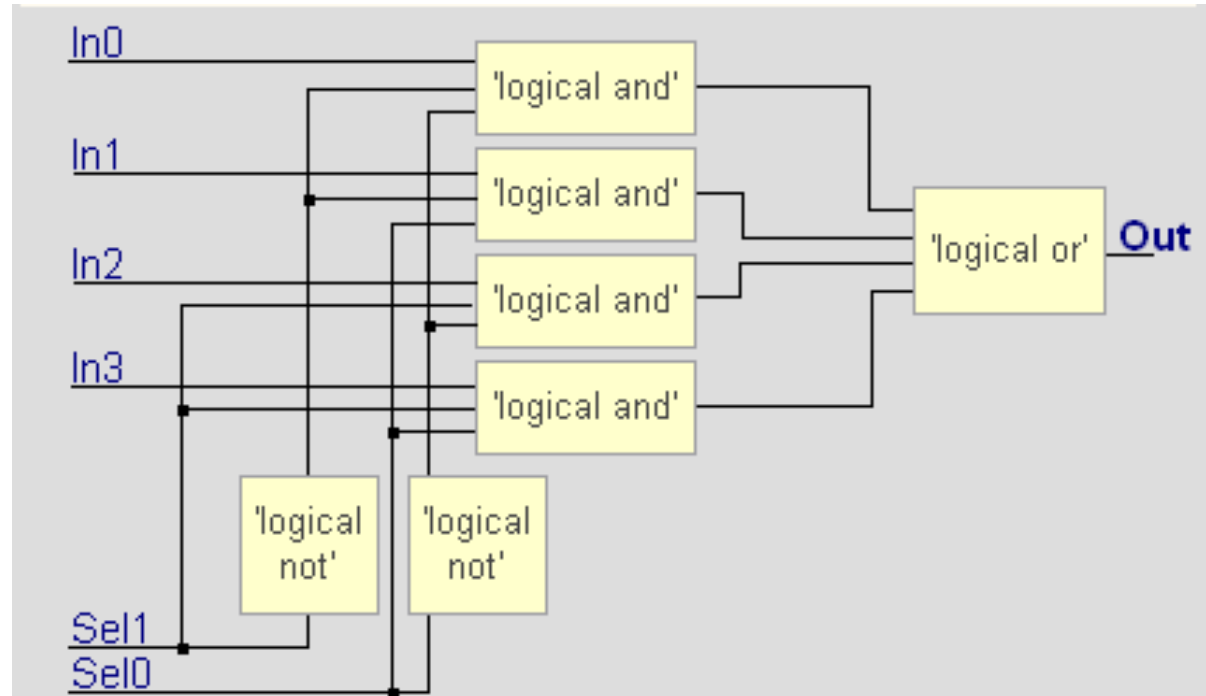


```
module ALU (Result, COut, Equal, In1, In2,  
            OpSel, CIn, Mode);  
  
output [3:0] Result;    // operation result  
output      COut;      // carry out  
output      Equal;     // when 1, In1 = In2  
input  [3:0] In1;      // first operand  
input  [3:0] In2;      // second operand  
input  [3:0] OpSel;    // operation select  
input      CIn;       // carry in  
input      Mode;      // mode arithm/logic;  
                        // arithm when 0  
  
    . . .  
  
endmodule
```

# Structural style: Verilog Code



# Dataflow style: Verilog Code



```
module mux_4_to_1 (Out, In0, In1, In2, In3, Sel1, Sel0);
```

```
output Out;
```

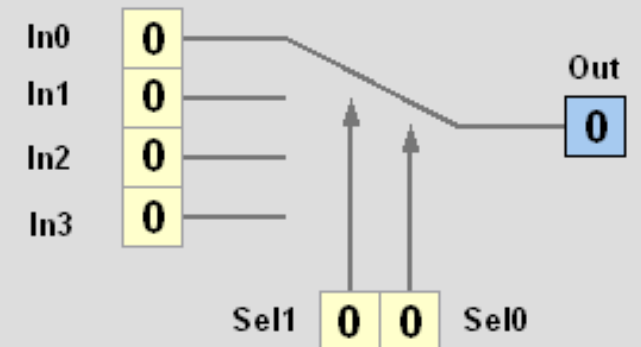
```
input In0, In1, In2, In3, Sel0, Sel1;
```

```
assign Out = (~Sel1 & ~Sel0 & In0) | (~Sel1 & Sel0 & In1)  
             | (Sel1 & ~Sel0 & In2) | (Sel1 & Sel0 & In3);
```

```
endmodule
```



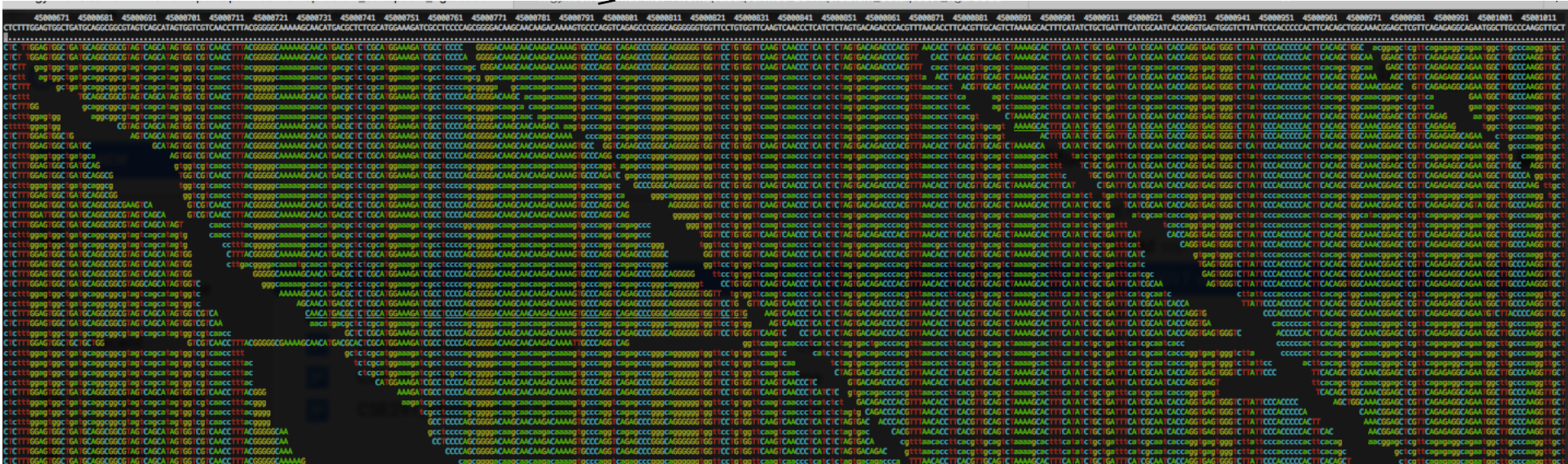
# Behavioral style: Verilog Code



```
module mux_4_to_1 (Out, In0, In1, In2, In3, Sel1, Sel0);  
  output Out;  
  input In0, In1, In2, In3, Sel0, Sel1;  
  reg Out;  
  
  always @(Sel1 or Sel0 or In0 or In1 or In2 or In3)  
  begin  
    case ((Sel1, Sel0))  
      2'b00 : Out = In0;  
      2'b01 : Out = In1;  
      2'b10 : Out = In2;  
      2'b11 : Out = In3;  
      default : Out = 1'bx;  
    endcase  
  end  
  
endmodule
```

# Genomics: Alignment

Reference genome



- Reveal structural, functional and evolutionary relationships biological sequences
- Similar sequences may have similar structure and function
- Similar sequences are likely to have common ancestral sequence
- Modelling of protein structures
- Design and analysis of gene expression experiments

# Sequence alignment: Types

- Global alignment
  - Aligns each residue in each sequence by introducing gaps
  - Example: Needleman-Wunsch algorithm

```
FTFTALILLAVAV  
F--TAL-LLA-AV
```

```
L G P S S K Q T G K G S - S R I W D N  
L N - I T K S A G K G A I M R L G D A
```

# Sequence alignment: Types

- Local alignment
  - Finds regions with the highest density of matches locally
  - Example: Smith-Waterman algorithm

```
FTFTALILL-AVAV  
--FTAL-LLAAV--
```

```
- - - - - T G K G - - - - -  
- - - - - A G K G - - - - -
```

# Sequence alignment: Scoring

T A C G G G C A G  
- A C - G G C - G

Option 1

T A C G G G C A G  
- A C G G - C - G

Option 2

T A C G G G C A G  
- A C G - G C - G

Option 3

- Scoring matrices are used to assign scores to each comparison of a pair of characters
- Identities and substitutions by similar amino acids are assigned positive scores
- Mismatches, or matches that are unlikely to have been a result of evolution, are given negative scores

A	C	D	E	F	G	H	I	K
A	C	Y	E	F	G	R	I	K
+5	+5	-5	+5	+5	+5	-5	+5	+5

# Pairwise alignment: the problem

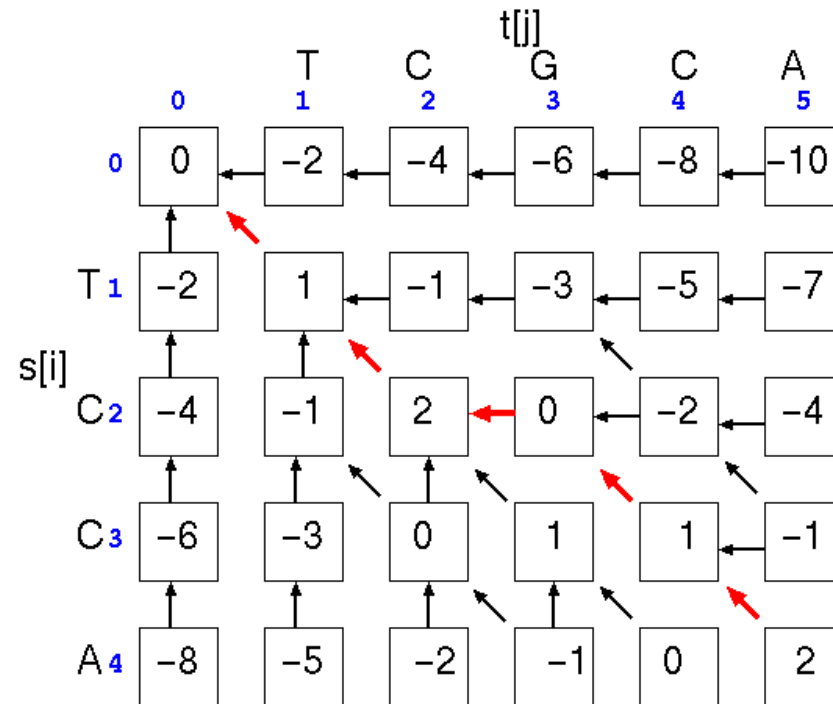
Number of possible pairwise alignments explodes with sequence length  
2 protein sequences of length 100 amino acids can be aligned in  $10^{60}$  ways



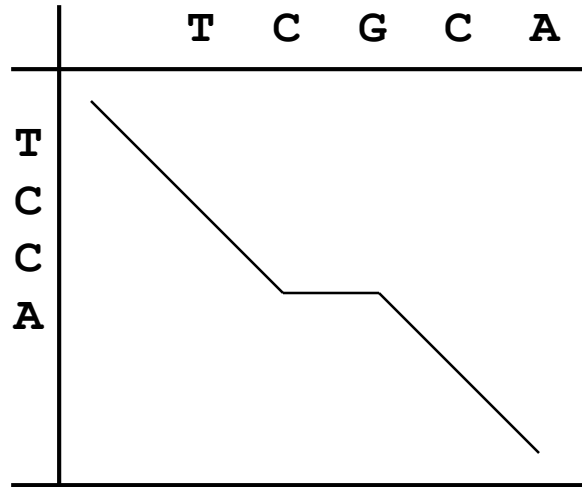
Time needed to test all possibilities is same order of magnitude as the entire lifetime of the universe.

# Pairwise alignment: the solution

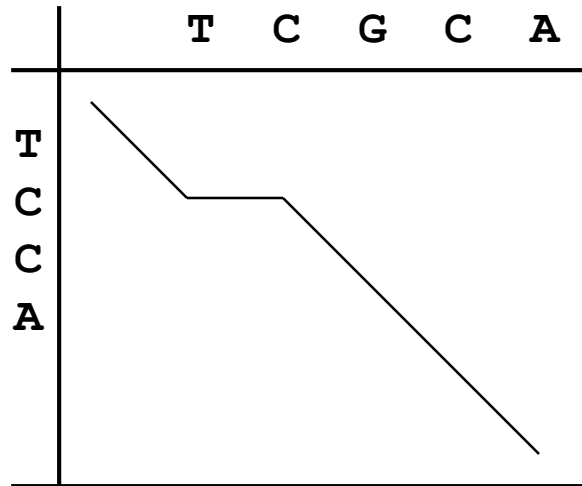
**"Dynamic programming"**  
(the Needleman-Wunsch algorithm)



# Alignment depicted as path in matrix



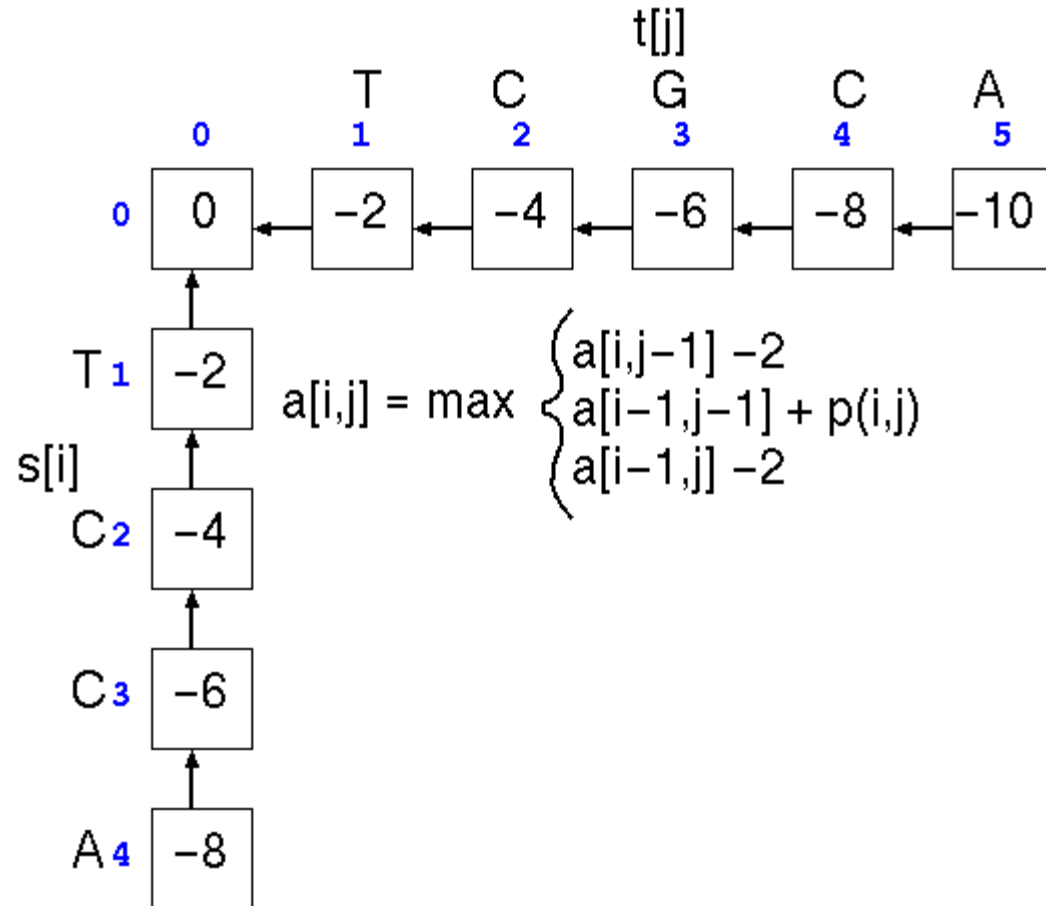
**TCGCA**  
**TC-CA**



**TCGCA**  
**T-CCA**



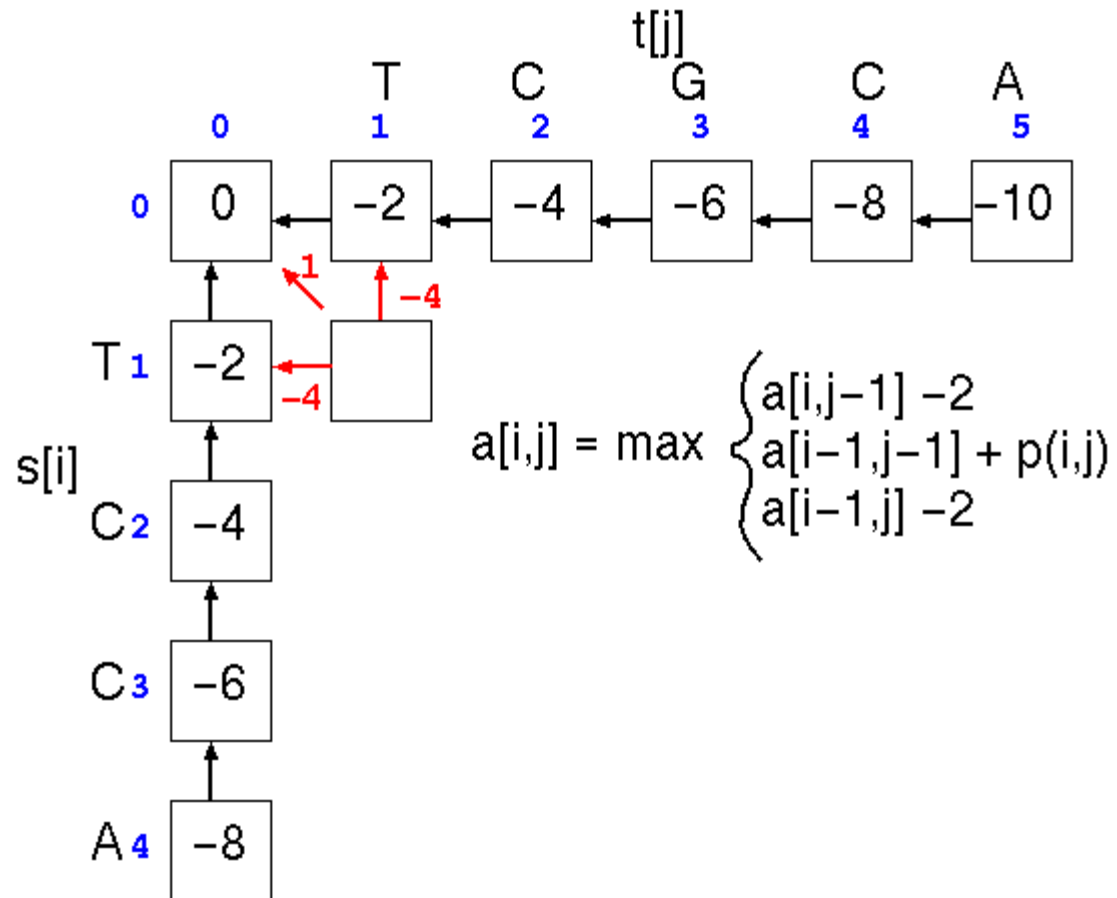
# Dynamic programming: example



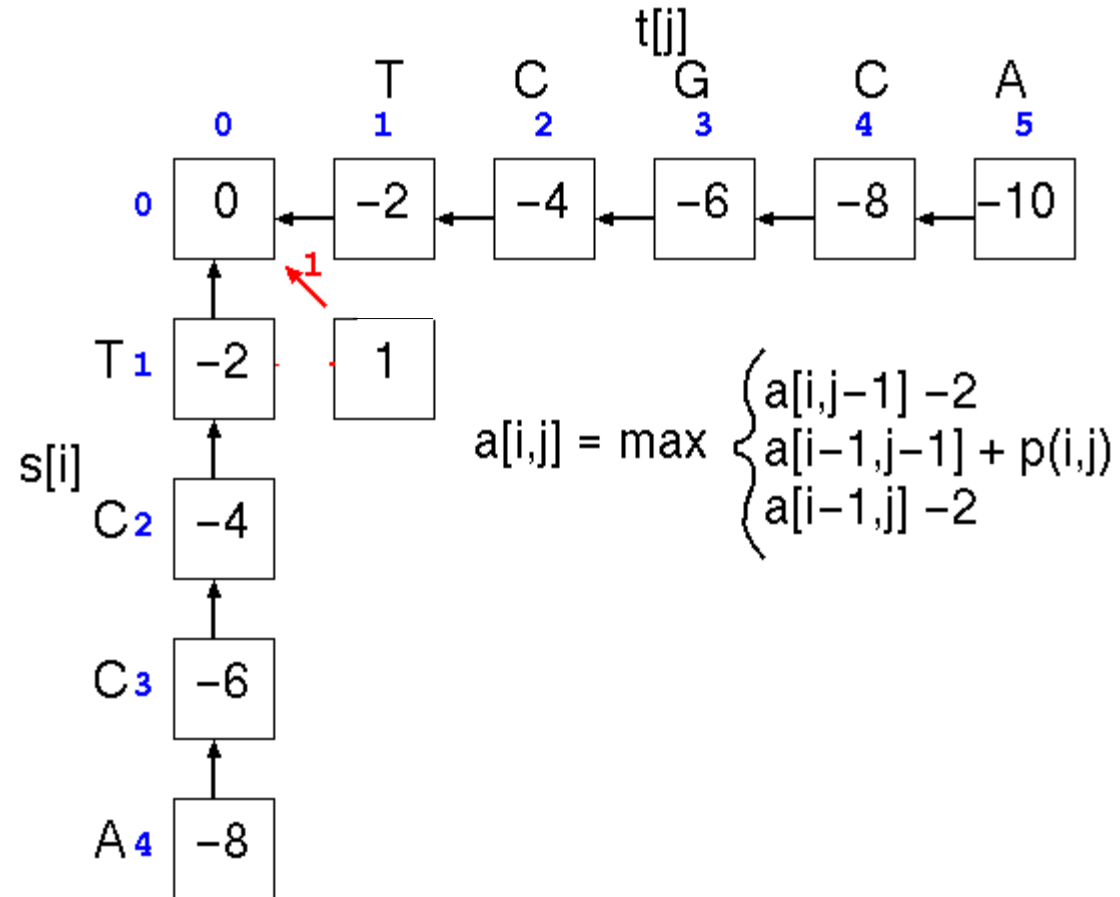
	A	C	G	T
A	1	-1	-1	-1
C	-1	1	-1	-1
G	-1	-1	1	-1
T	-1	-1	-1	1

Gaps: -2

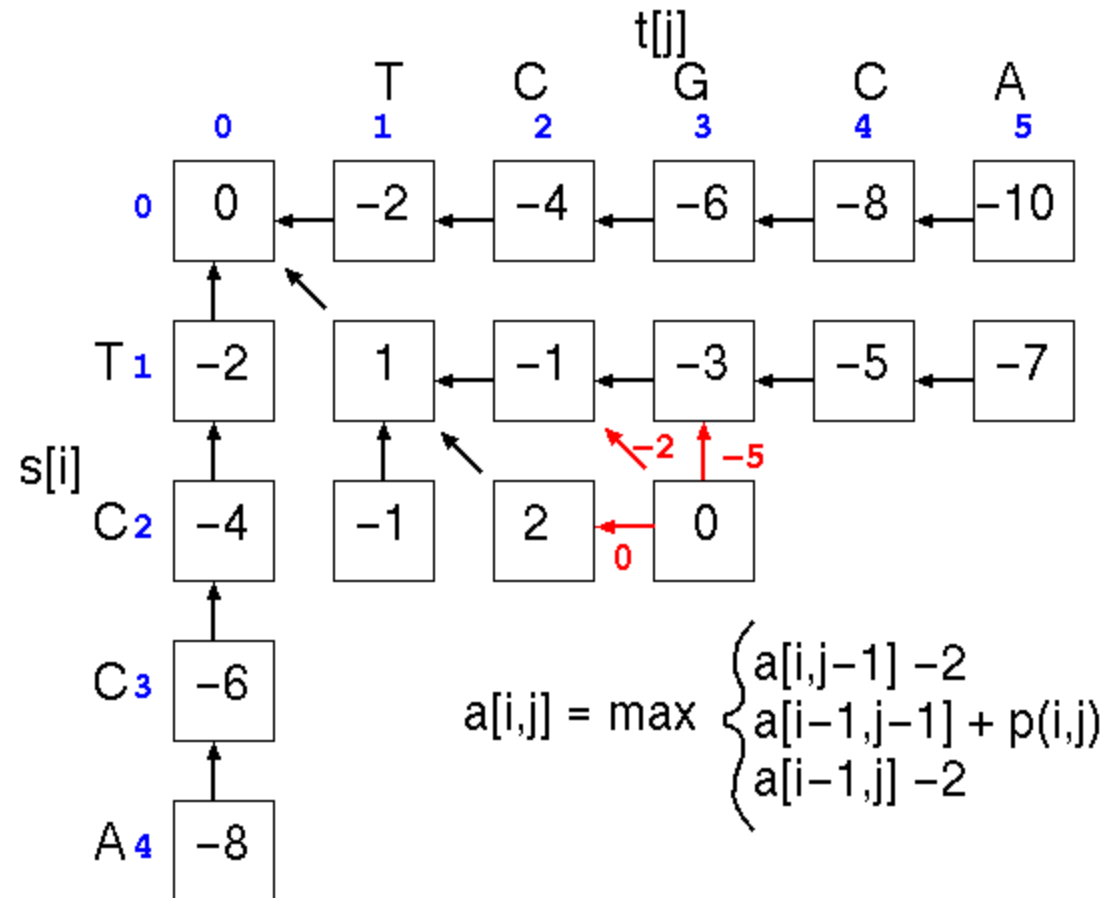
# Dynamic programming: example



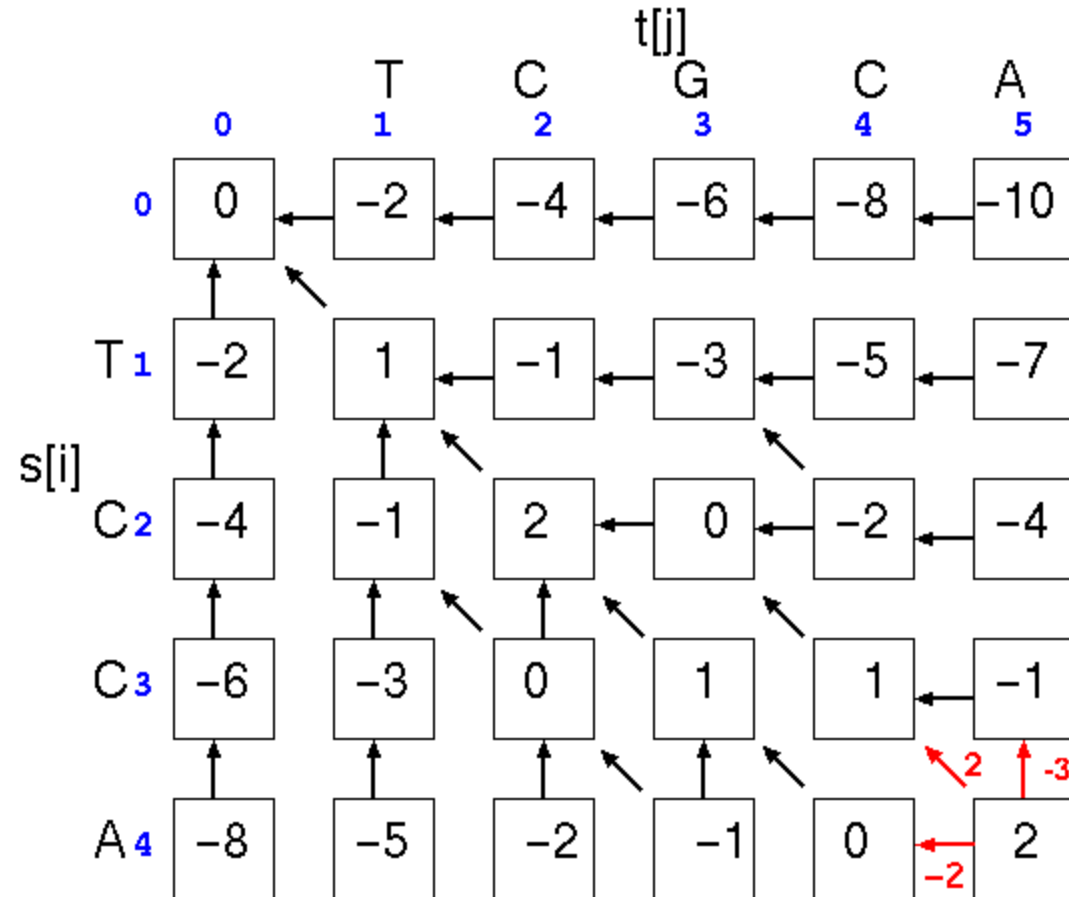
# Dynamic programming: example



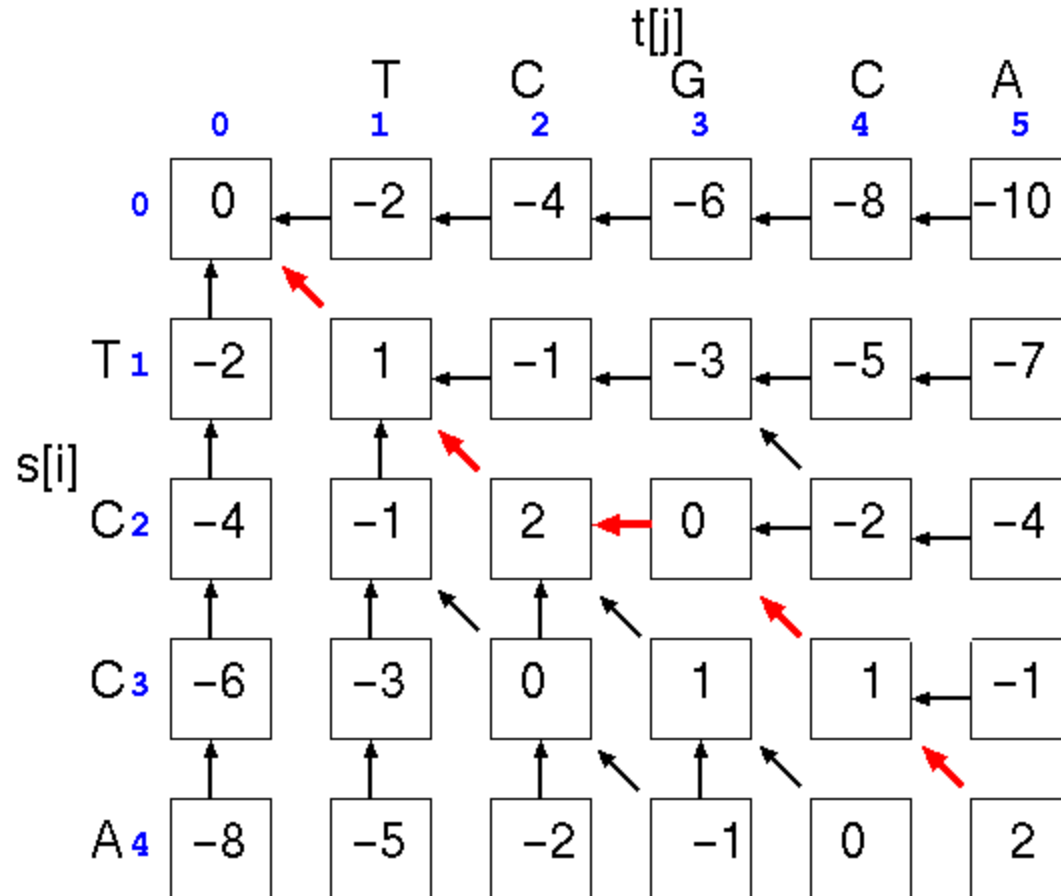
# Dynamic programming: example



# Dynamic programming: example



# Dynamic programming: example



$$\begin{array}{cccccc}
 \text{T} & \text{C} & \text{G} & \text{C} & \text{A} & \\
 : & : & & : & : & \\
 \text{T} & \text{C} & - & \text{C} & \text{A} & \\
 \hline
 1 & + & 1 & - & 2 & + & 1 & + & 1 & = & 2
 \end{array}$$

# Thoughts On Cascade

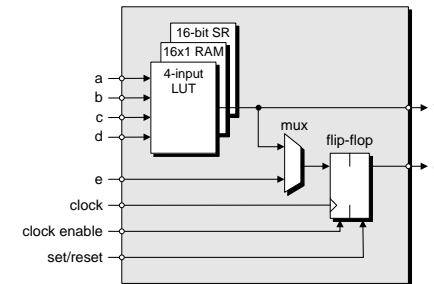
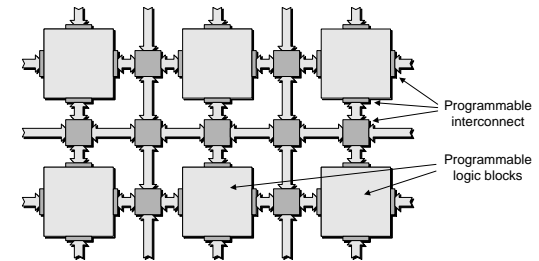
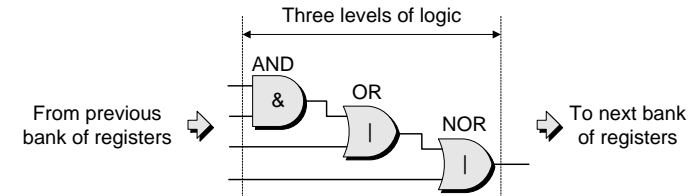
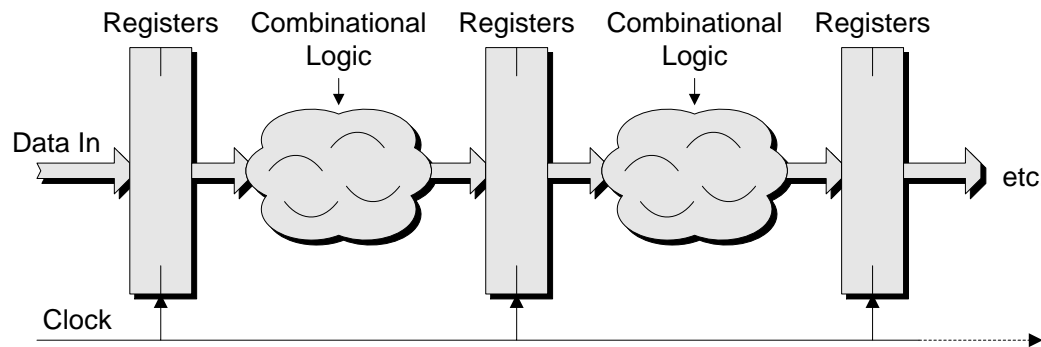
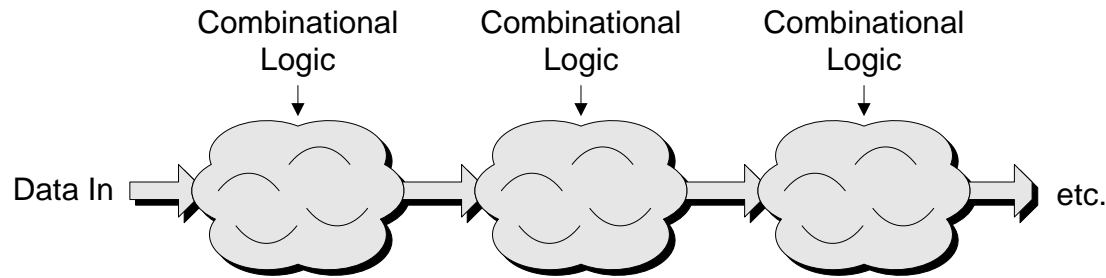


# Exam Review



# FPGA Design Synchronous Logic

- Pipelined. Clocked Logic.
- Combinational and Sequential Logic.
- Register Transfer Level Logic.

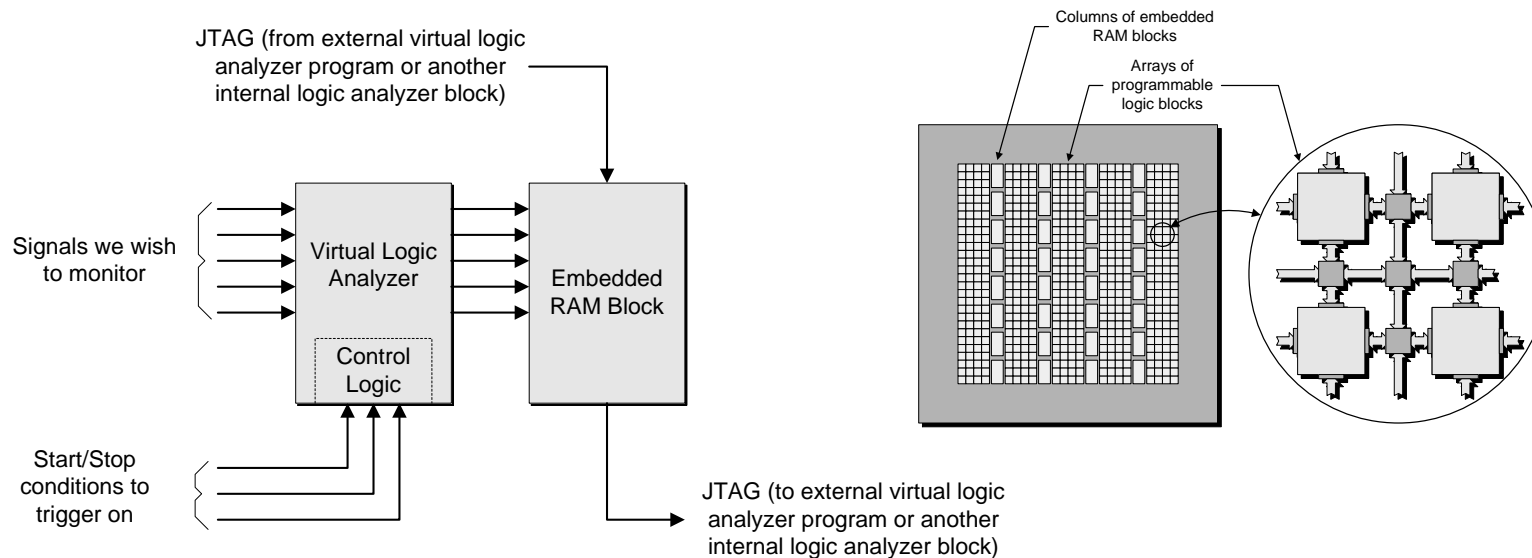


# Firmware Libraries

- Libraries of Firmware aka IP (Intellectual Property), Cores
  - Buy from FPGA Vendor
  - Buy from Third Parties
  - Open Source
- Libraries
  - VHDL code
  - Black Box NetList
  - Hardwired in Silicon
- Large User Community

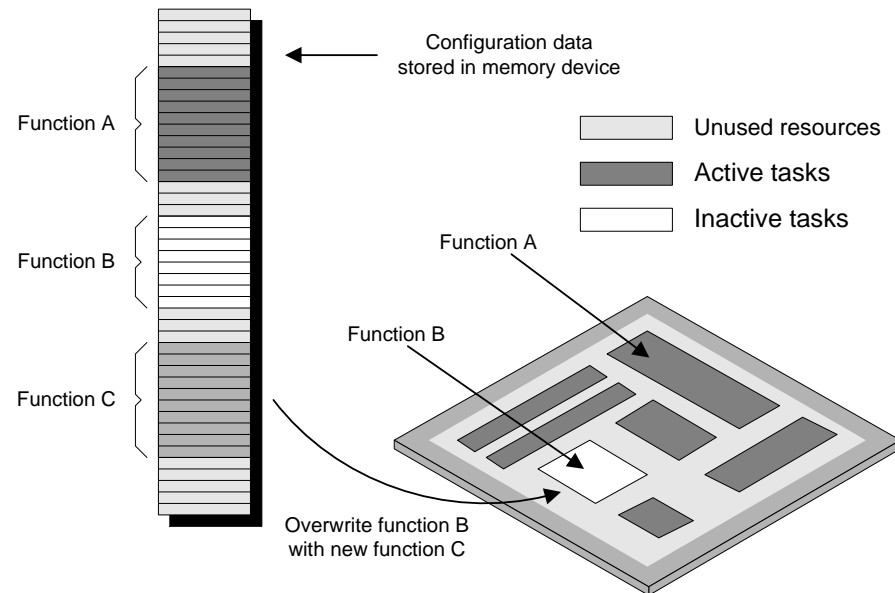
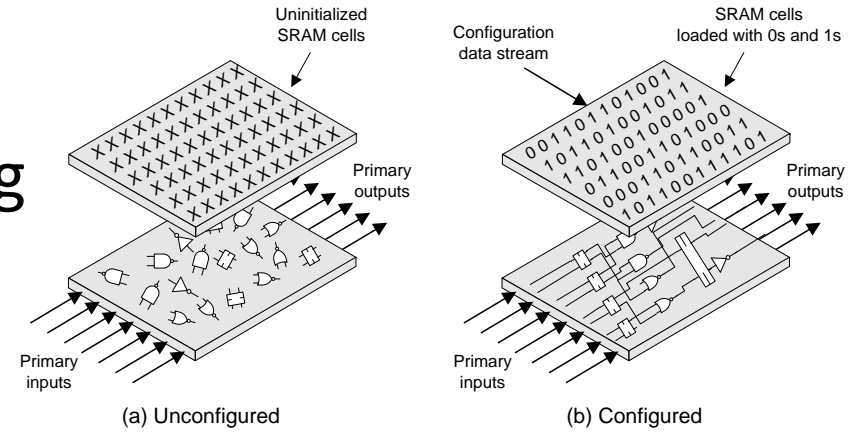
# Debugging Designs

- Logic Simulation Tools
  - Model of Logic
  - Input: Test Vector signals
  - Compare output with expected pattern
- Virtual Logic Analysers
  - Capture signals in real time while FPGA is running



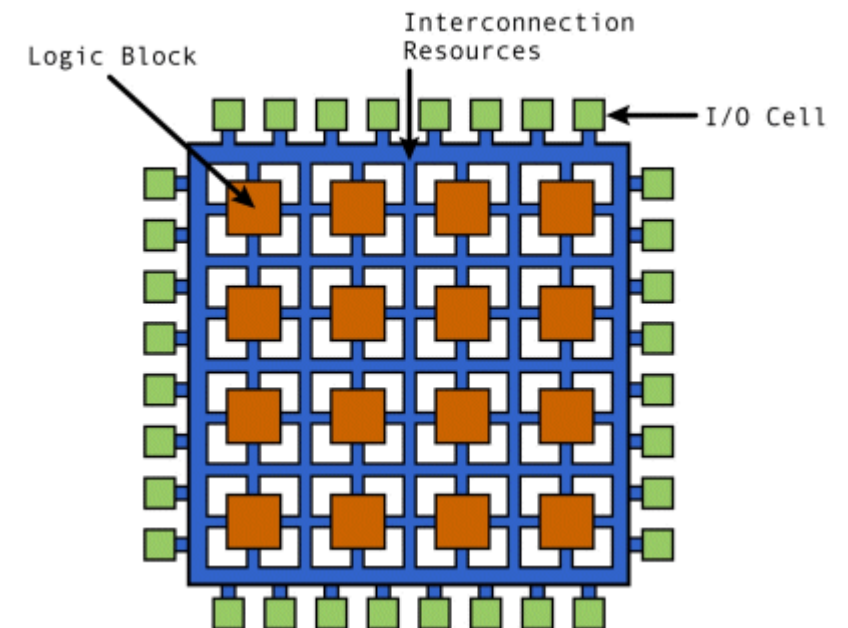
# FPGA Research Developments

- Reconfigurable Computing
- Virtual Hardware



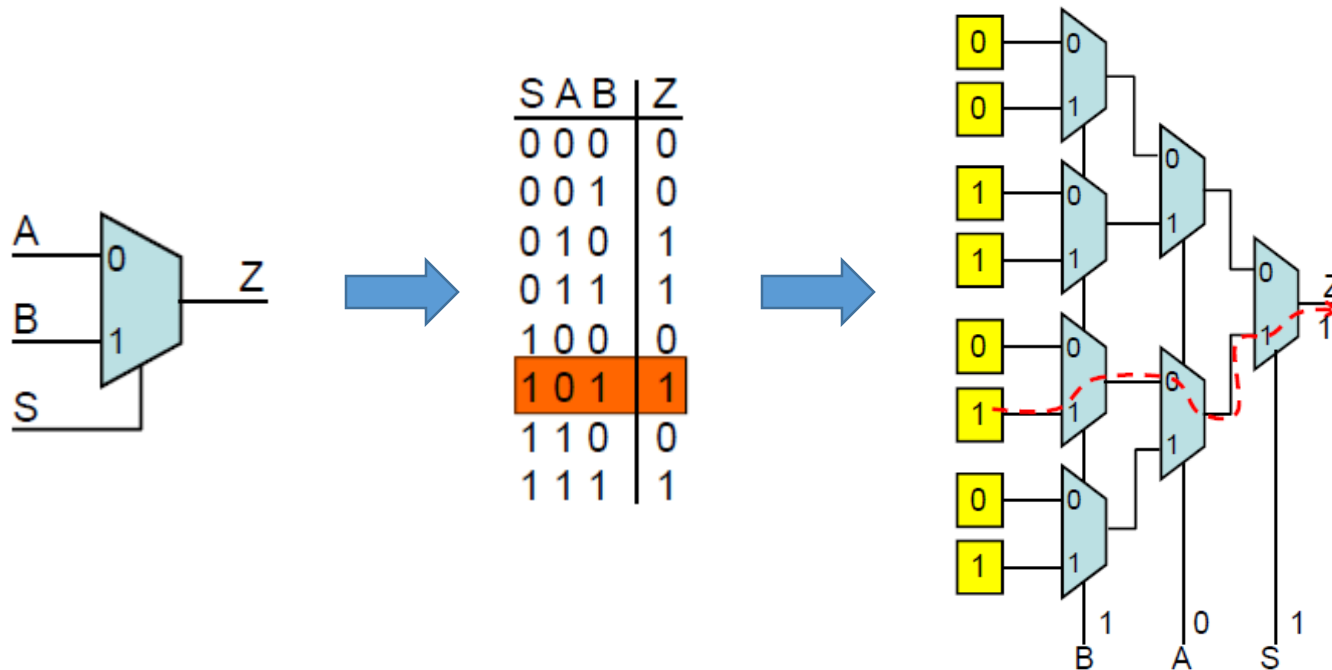
# FPGAs

- Field Programmable Gate Array
  - “Field” → architecture can be changed after deployment
- Gate Array
  - Gate – Short for transistor logic gate (e.g. NAND)
  - Array – Lots of them
- An integrated circuit (“chip”)
- Programmable logic
  - Not just gates
  - Lookup tables, DSPs, other components



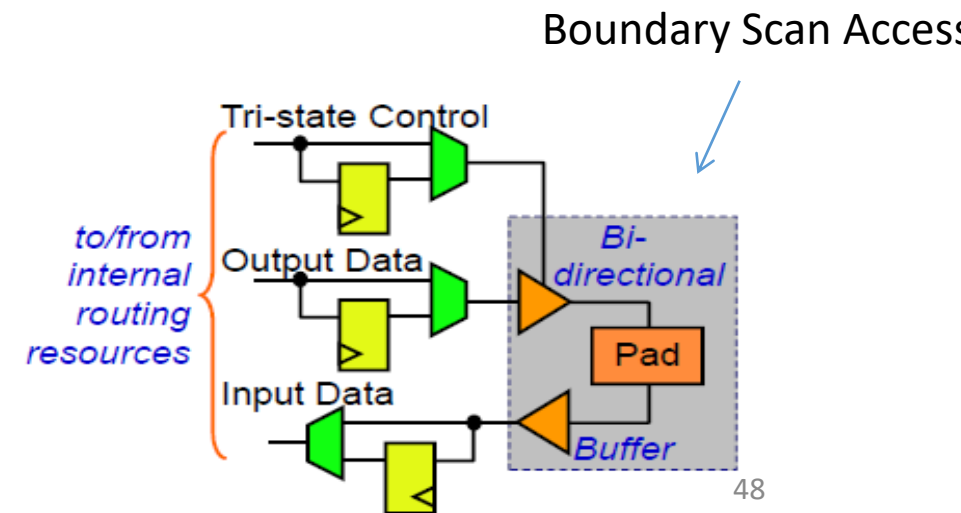
# Look-up Tables (2:1 MUX Example)

- Configuration memory holds output of truth table entries
- Internal signals connect to control signals of MUXs
  - select values of the truth tables for any given input signals



# Programmable Input/Output

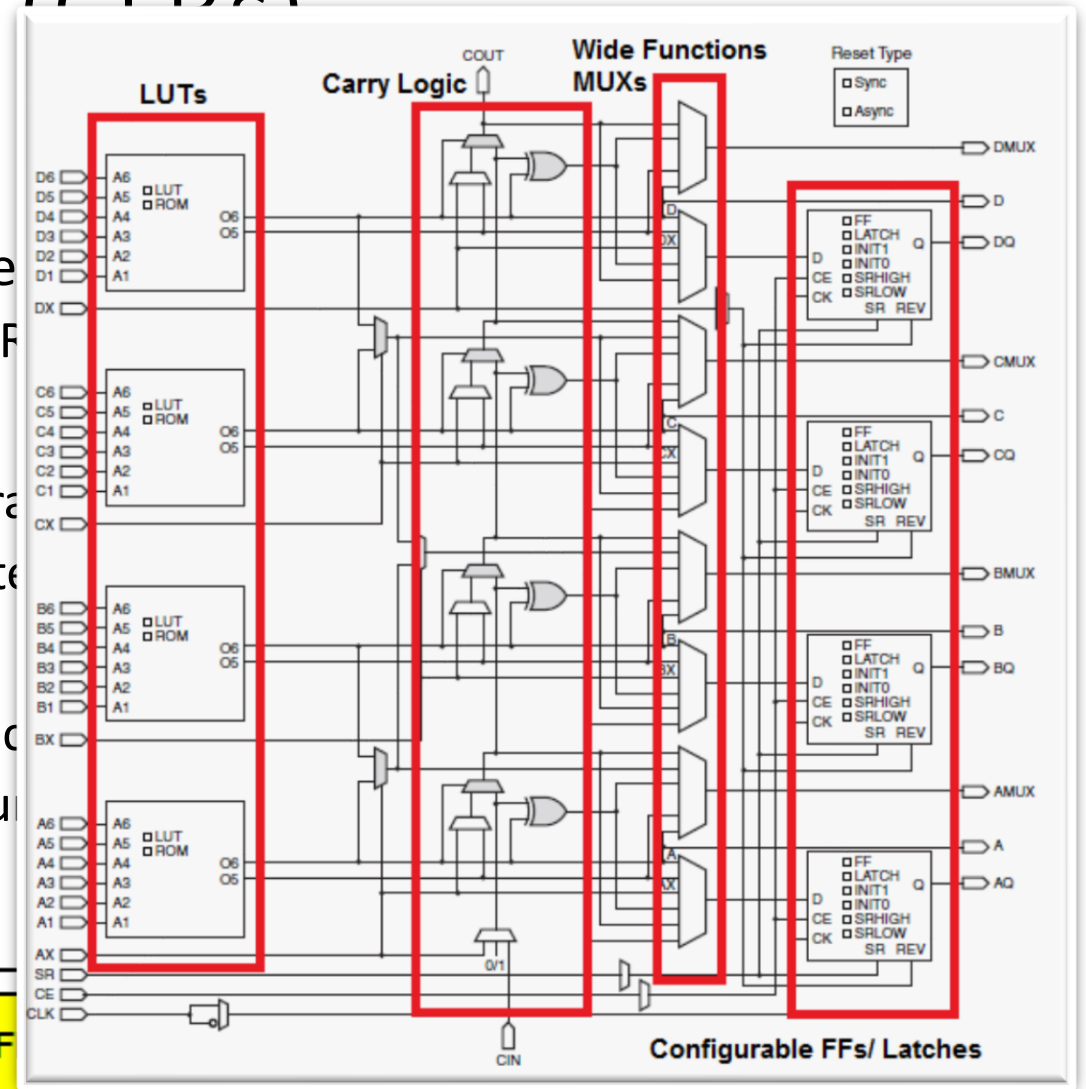
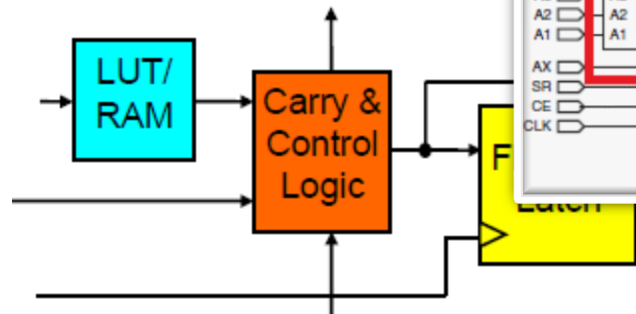
- Bi-directional Buffers
  - Programmable for inputs or outputs
  - Tri-state controls bi-directional operation
  - Pull-up/down resistors
  - FFs/ Latches are used to improve timing issues
    - Set-up and hold times
    - Clock-to-out delay
- Routing Resources
  - Connections to core of array
- Programmable I/O voltage and current levels



# Configurable Logic Blocks (CLBs)

CLBs consist of:

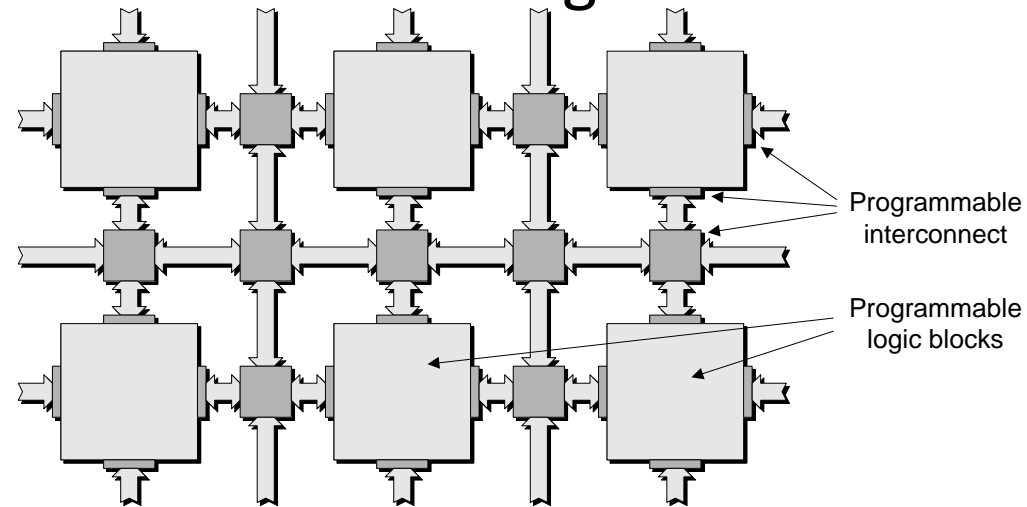
- Look-up Tables (LUTs) → implement the e
  - Some FPGAs can use LUTs to implement F
- Carry and Control Logic
  - fast arithmetic operations (adders/ subtractors)
  - additional operations (Built-in-Self Test ite
- Memory Elements
  - Configurable Flip Flops (FFs)/ Latches(cloc
  - Memory elements usually can be configu





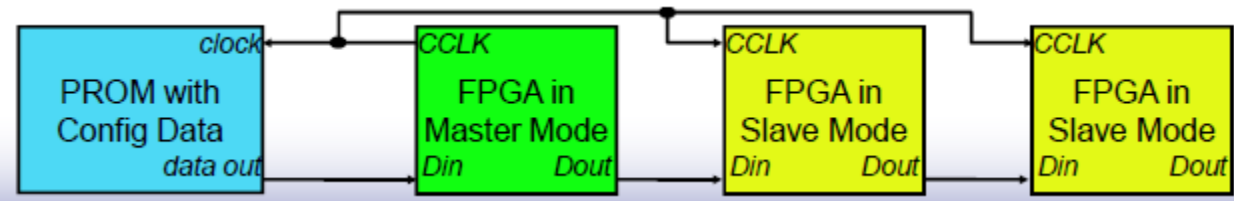
# Programming an FPGA

- Field Programmable Gate Array
  - Configurable (Programmable) General Logic Blocks
  - Configurable Interconnects
- Bit File contains the Configuration Information



# FPGA Configuration Interfaces

- Master (Serial or Parallel)
  - FPGA retrieves configuration from ROM at initial power-up
- Slave (Serial or Parallel)
  - FPGA configured by an external source (i.e microprocessor/ other FPGA)
  - Used for dynamic partial re-configuration
- Boundary Scan
  - 4-wire IEEE standard serial interface used for testing
  - Write and read access to configuration memory
  - Interfaces to FPGA core internal routing network

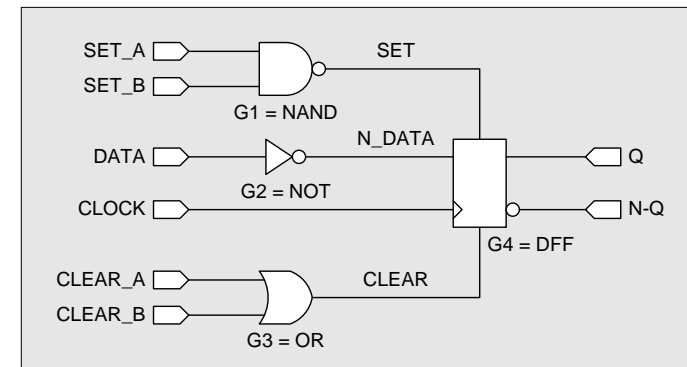
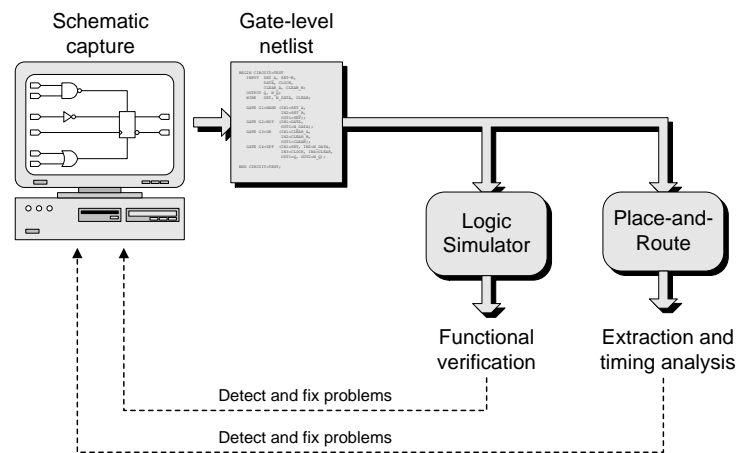


# FPGA Configuration Techniques

- Full configuration and readback
  - Simple configuration interface
    - Automatic internal calculation of frame address
  - Larger FPGAs have a longer download time
- Compressed configuration
  - Requires multiple frame write capability
    - Identical frames of configuration data are written to multiple frame addresses
  - Extension of partial re-configuration interface capabilities
    - Frame address is much smaller than frame of configuration data
  - Reduces download time for initial configuration
    - depends on “regularity” of system function and array utilization
- Partial re-configuration and readback
  - Change portions of configuration memory
    - Reduces download time for re-configuration

# Design Flows

- Schematic Capture of Logic Design.
- Create Netlist. Text file with signal connections.



```
BEGIN CIRCUIT=TEST
  INPUT  SET_A, SET_B, DATA, CLOCK, CLEAR_A, CLEAR_B;
  OUTPUT Q, N_Q;
  WIRE   SET, N_DATA, CLEAR;

  GATE G1=NAND (IN1=SET_A, IN2=SET_B, OUT1=SET);
  GATE G2=NOT  (IN1=DATA, OUT1=N_DATA);
  GATE G3=OR   (IN1=CLEAR_A, IN2=CLEAR_B, OUT1=CLEAR);
  GATE G4=DFF  (IN1=SET, IN2=N_DATA, IN3=CLOCK,
               IN4=CLEAR, OUT1=Q, OUT2=N_Q);

END CIRCUIT=TEST;
```

# Software Languages?

- Can Logic be expressed at a higher level of Abstraction?
- Familiar to Software Programmer?
- System C
  - C/C++ Representation of Algorithms
  - Class based
  - Faster simulation
  - Auto translation to HDL
  - Lacks support by Tools
- Augmented C++
  - Special Statements to support
  - Concurrency, clocks, pins ..etc
- Digital Signal Processing Functions

