EE 459/500 – HDL Based Digital Design with Programmable Logic

Lecture 9

Field Programmable Gate Arrays (FPGAs)

Read before class:

Chapter 3 from textbook

Overview

- FPGA Devices
  - ASIC vs. FPGA
  - FPGA architecture
    - CLB, RAM
    - IO, Interconnects
- FPGA Design Flow
  - Synthesis
  - Place
  - Route

Evolution of implementation technologies

- Logic gates (1950s-60s)
- Regular structures for two-level logic (1960s-70s)
  - muxes and decoders, PLAs
- Programmable sum-of-products arrays (1970s-80s)
  - PLDs, complex PLDs
- Programmable gate arrays (1980s-90s)
  - densities high enough to permit entirely new class of application, e.g., prototyping, emulation, acceleration

$\downarrow$ **trend toward higher levels of integration**

ASIC vs. FPGA

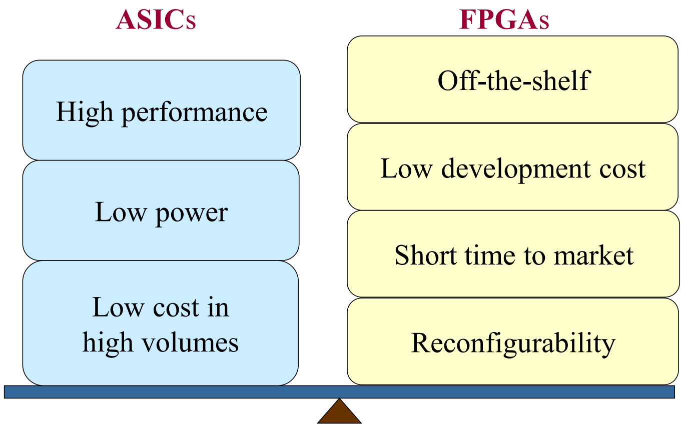
ASIC
Application **S**pecific **I**ntegrated **C**ircuit

- designed all the way from behavioral description to physical layout
- designs must be sent for expensive and time consuming fabrication in semiconductor foundry

FPGA
Field **P**rogrammable **G**ate **A**rray

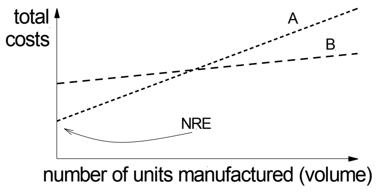
- no physical layout design; design ends with a bitstream used to configure a device
- bought off the shelf and reconfigured by designers themselves

Which way to go?



Why FPGAs?

- Custom ICs sometimes designed to replace large amount of glue logic:
  - Reduced system complexity and manufacturing cost, improved performance.
  - However, custom ICs are very expensive to develop, and delay introduction of product to market (time to market) because of increased design time.
- Note: need to worry about two kinds of costs:
  1. cost of development, sometimes called non-recurring engineering (NRE)
  2. cost of manufacture
  - A tradeoff usually exists between NRE cost and manufacturing costs



Why FPGAs?

- Custom IC approach viable for products that are …
  - very high volume (where NRE could be amortized),
  - not time-to-market sensitive.
- FPGAs introduced as an alternative to custom ICs for implementing glue logic:
  - improved density relative to discrete SSI/MSI components (within around 10x of custom ICs)
  - with the aid of computer aided design (CAD) tools circuits could be implemented in a short amount of time (no physical layout process, no mask making, no IC manufacturing), relative to ASICs.
    - lowers NREs
    - shortens TTM
- Because of Moore’s law the density (gates/area) of FPGAs continued to grow through the 80’s and 90’s to the point where major data processing functions can be implemented on a single FPGA.

Applications of FPGAs

- Implementation of random logic
  - easier changes at system-level (one device is modified)
  - can eliminate need for full-custom chips
- Prototyping
  - ensemble of gate arrays used to emulate a circuit to be manufactured
  - get more/better/faster debugging done than possible with simulation
- Reconfigurable hardware
  - one hardware block used to implement more than one function
  - functions must be mutually-exclusive in time
  - can greatly reduce cost while enhancing flexibility
  - RAM-based only option
- Special-purpose computation engines
  - hardware dedicated to solving one problem (or class of problems)
  - accelerators attached to general-purpose computers

Major FPGA Vendors

SRAM-based FPGAs

- **Xilinx, Inc.**
- **Altera Corp.**
- Atmel
- Lattice Semiconductor

Share about 90% of the market

Flash & antifuse FPGAs

- Actel Corp.
- Quick Logic Corp.

Xilinx FPGA Families

- **Old families**
  - XC3000, XC4000, XC5200
  - Old 0.5µm, 0.35µm and 0.25µm technology. Not recommended for modern designs.
- **High-performance families**
  - Virtex (220 nm)
  - Virtex-E, Virtex-EM (180 nm)
  - Virtex-II, Virtex-II PRO (130 nm)
  - Virtex-4 (90 nm)
  - Virtex-5 (65 nm)
  - Virtex-6
- **Low Cost Family**
  - Spartan/XL – derived from XC4000
  - Spartan-II – derived from Virtex
  - Spartan-IIE – derived from Virtex-E
  - Spartan-3 (90 nm)
  - Spartan-3E (90 nm) – logic optimized
  - Spartan-3A (90 nm) – I/O optimized
  - Spartan-3AN (90 nm) – non-volatile
  - Spartan-3A DSP (90 nm) – DSP optimized
  - Spartan-6

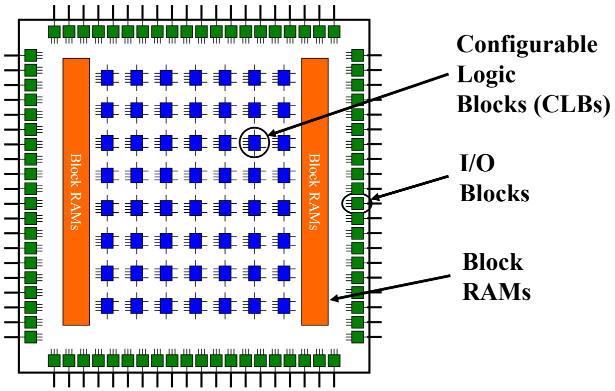
Altera FPGA Families

- High & Medium Density FPGAs
  - Stratix™ II, Stratix, APEX™ II, APEX 20K, & FLEX® 10K
- Low-Cost FPGAs
  - Cyclone™ & ACEX® 1K
- FPGAs with Clock Data Recovery
  - Stratix GX & Mercury™
- CPLDs
  - MAX® 7000 & MAX 3000
- Embedded Processor Solutions
  - Nios™, Excalibur™
- Configuration Devices
  - EPC

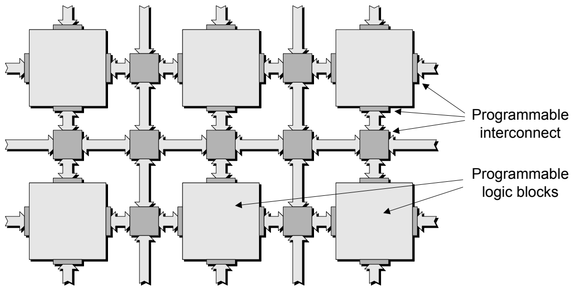
Overview

- FPGA Devices
  - ASIC vs. FPGA
  - FPGA architecture
    - CLB, RAM
    - IO, Interconnects
- FPGA Design Flow
  - Synthesis
  - Place
  - Route

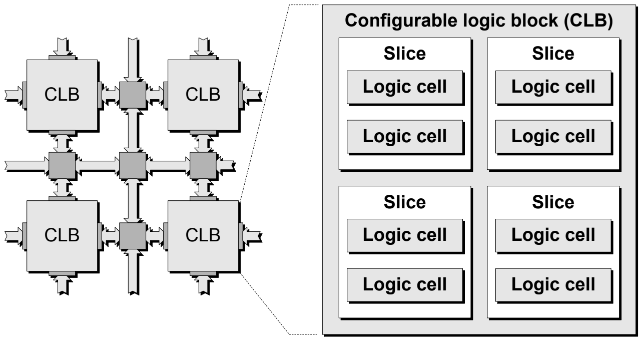
What is an FPGA?



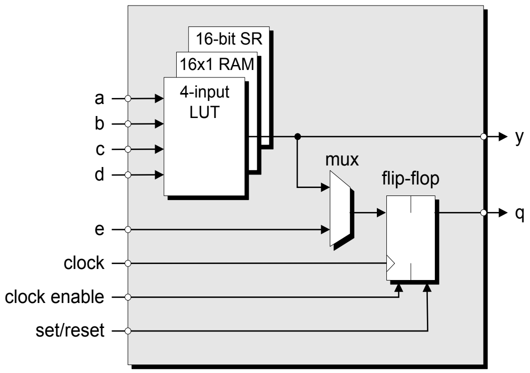
What is an FPGA?



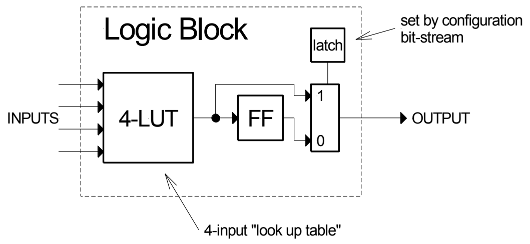
Example of Xilinx CLB



Simplified view of a Xilinx Logic Cell

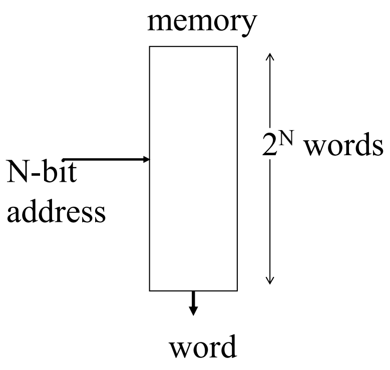


Idealized Configurable Logic Block (CLB)



- 4-input **look-up table (LUT)**
  - implements combinational logic functions
- Register
  - optionally stores output of LUT

How could you build a generic Boolean logic circuit? **Memories as LUTs**



- 1-bit memory to hold boolean value
- Address is vector of boolean input values
- Contents encode a boolean function
- Read out logical value (col) for associated row

LUT as general logic gate

- An n-lut as a direct implementation of a function truth-table.
- Each latch location holds the value of the function corresponding to one input combination.

Example: 2-lut

INPUTS	AND	OR
00	0	0
01	0	1
10	0	1
11	1	1

• • •

Can be used to implement *any* function of 2 inputs.

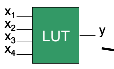
How many of these are there?

How many functions of n inputs?

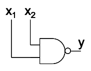
Example: 4-lut

INPUTS		
0000	F(0,0,0,0)	⟵ store in 1st latch
0001	F(0,0,0,1)	⟵ store in 2nd latch
0010	F(0,0,1,0)	⟵
0011	F(0,0,1,1)	⟵
0011		
0100	•	
0101	•	
0110	•	
0111		
1000		
1001		
1010		
1011		
1100		
1101		
1110		
1111		

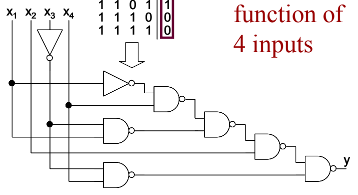
LUT as general logic gate



x_1	x_2	x_3	x_4	y
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	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

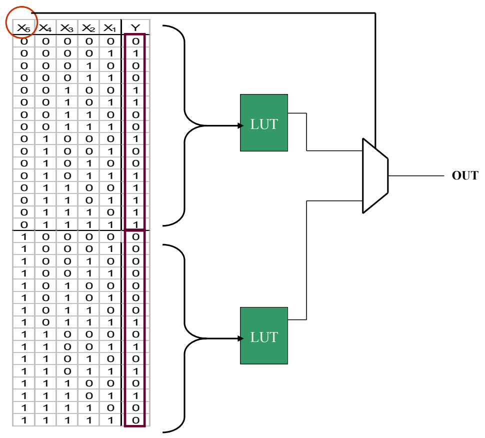


x_1	x_2	x_3	x_4	y
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0



- Look-Up Tables are primary elements for logic implementation
- Each LUT can implement any function of 4 inputs

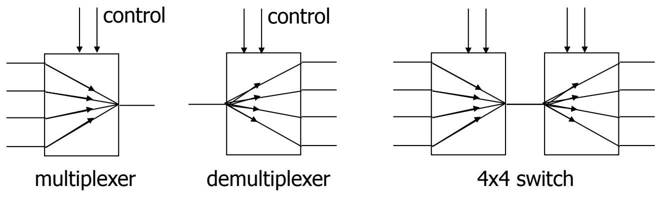
5-Input functions implemented using two LUTs



X_5	X_4	X_3	X_2	X_1	Y
0	0	0	0	0	0
0	0	0	0	1	1
0	0	0	1	0	0
0	0	0	1	1	0
0	0	1	0	0	1
0	0	1	0	1	1
0	0	1	1	0	0
0	0	1	1	1	0
0	1	0	0	0	1
0	1	0	0	1	0
0	1	0	1	0	0
0	1	0	1	1	1
0	1	1	0	0	1
0	1	1	0	1	1
0	1	1	1	0	1
0	1	1	1	1	1
1	0	0	0	0	0
1	0	0	0	1	0
1	0	0	1	0	0
1	0	0	1	1	0
1	0	1	0	0	0
1	0	1	0	1	0
1	0	1	1	0	0
1	0	1	1	1	1
1	1	0	0	0	0
1	1	0	0	1	1
1	1	0	1	0	0
1	1	0	1	1	1
1	1	1	0	0	0
1	1	1	0	1	1
1	1	1	1	0	0
1	1	1	1	1	0

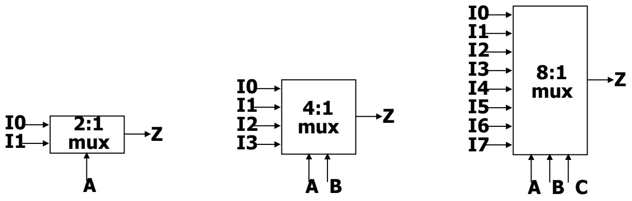
Recall: Multiplexer/Demultiplexer

- *Multiplexer:* route one of many inputs to a single output
- *Demultiplexer:* route single input to one of many outputs



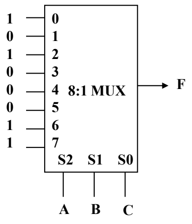
Multiplexers/Selectors: to implement logic

- 2:1 mux: Z = A' I0 + A I1
- 4:1 mux: Z = A' B' I0 + A' B I1 + A B' I2 + A B I3
- 8:1 mux: Z = A'B'C'I0 + A'B'CI1 + A'BC'I2 + A'BCI3 + AB'C'I4 + AB'CI5 + ABC'I6 + ABCI7



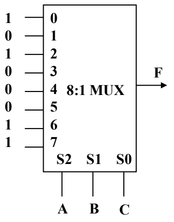
Multiplexers as LUTs

- 2^n:1 multiplexer implements any function of n variables
  - With the variables used as control inputs and
  - Data inputs tied to 0 or 1
  - In essence, *a look-up table*
- Example:
  - F(A,B,C) = m0 + m2 + m6 + m7
    = A'B'C' + A'BC' + ABC' + ABC
    = A'B'(C') + A'B(C') + AB'(0) + AB(1)

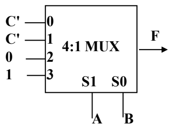


Multiplexers as LUTs (cont'd)

- 2^{n-1}:1 mux can implement any function of n variables
  - With n-1 variables used as control inputs and
  - Data inputs tied to the last variable or its complement
- Example:
  - F(A,B,C) = m0 + m2 + m6 + m7
    = A'B'C' + A'BC' + ABC' + ABC
    = A'B'(C') + A'B(C') + AB'(0) + AB(1)

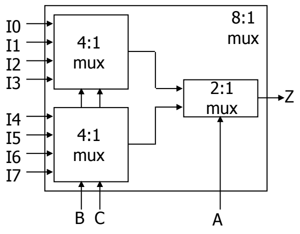


A	B	C	F	
0	0	0	1	C'
0	0	1	0	
0	1	0	1	C'
0	1	1	0	
1	0	0	0	0
1	0	1	0	
1	1	0	1	1
1	1	1	1	



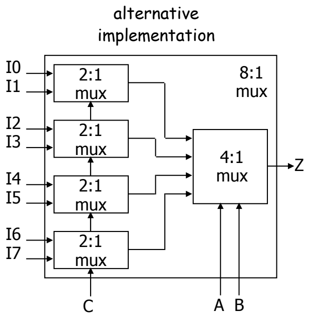
Cascading Multiplexers

- Large multiplexers implemented by cascading smaller ones

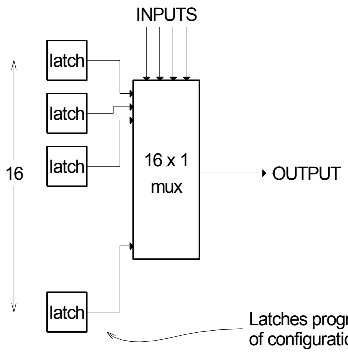


control signals B and C simultaneously choose one of I0, I1, I2, I3 and one of I4, I5, I6, I7

control signal A chooses which of the upper or lower mux's output to gate to Z



4-LUT Implementation

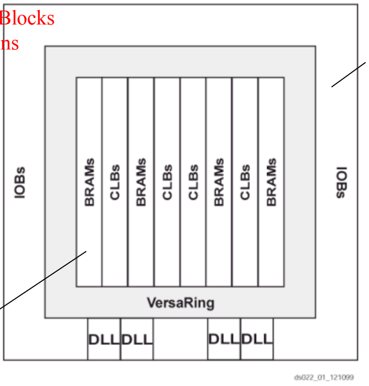


- n-bit LUT is implemented as a 2^n x 1 memory:
  - Inputs choose one of 2^n memory locations.
  - Memory locations (latches) are normally loaded with values from user's configuration bit stream.
  - Inputs to mux control are the CLB inputs.
- Result is a general purpose "logic gate".
  - n-LUT can implement *any* function of n inputs!

Latches programmed as part of configuration bit-stream

Example: Xilinx Virtex-E Floorplan

Configurable Logic Blocks
- 4-input function gens
- buffers
- flipflop



Input/Output Blocks
- combinational, latch, and flipflop output
- sampled inputs

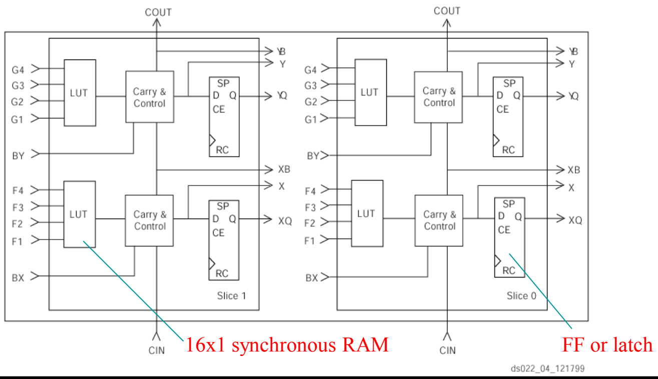
Block RAM
- 4096 bits each
- every 12 CLB columns

Virtex-E Configurable Logic Block (CLB)

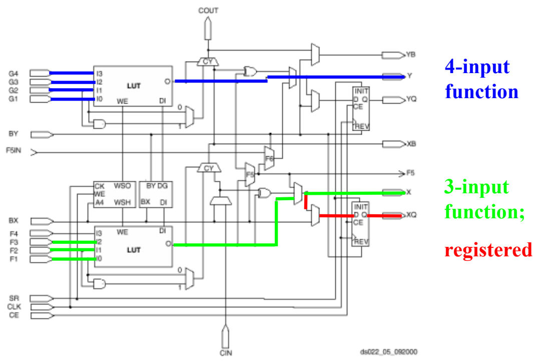
CLB = 4 logic cells (LC) in two slices

LC: 4-input function generator, carry logic, storage element

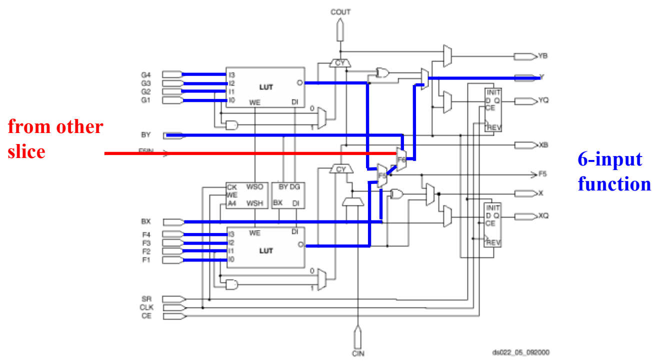
80 x 120 CLB array on 2000E



Details of Virtex-E Slice – implements any two 4-input functions

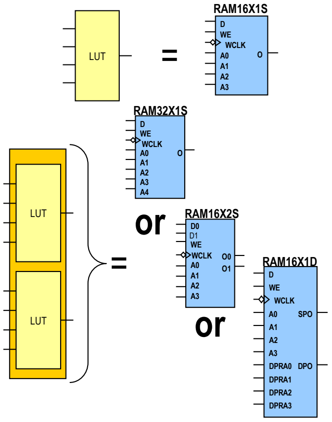


Details of Virtex-E Slice – any two 6-input function



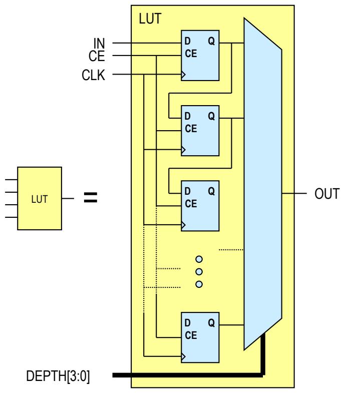
Distributed RAM

- CLB LUT configurable as Distributed RAM
  - A single LUT equals 16x1 RAM
  - Two LUTs Implement Single and Dual-Port RAMs
  - Cascade LUTs to increase RAM size
- Synchronous write
- Synchronous/Asynchronous read
  - Accompanying flip-flops used for synchronous read

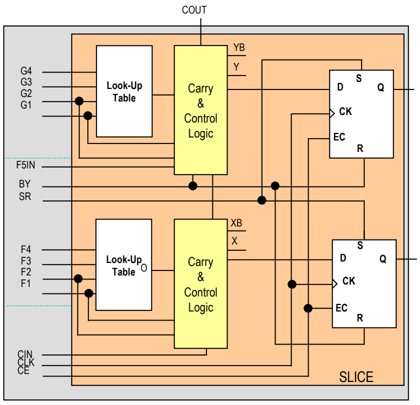


Shift Register

- Each LUT can be configured as shift register
  - Serial in, serial out
- Dynamically addressable delay up to 16 cycles
- For programmable pipeline
- Cascade for greater cycle delays
- Use CLB flip-flops to add depth

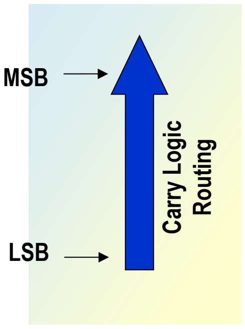


Carry & Control Logic



Fast Carry Logic

- Each CLB contains separate logic and routing for the fast generation of sum & carry signals
  - Increases efficiency and performance of adders, subtractors, accumulators, comparators, and counters
- Carry logic is independent of normal logic and routing resources



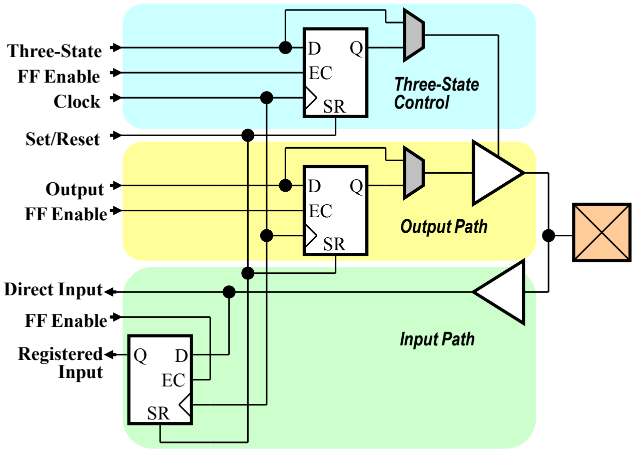
Accessing Carry Logic

- All major synthesis tools can infer carry logic for arithmetic functions
  - Addition (SUM <= A + B)
  - Subtraction (DIFF <= A - B)
  - Comparators (if A < B then…)
  - Counters (count <= count +1)

Overview

- FPGA Devices
  - ASIC vs. FPGA
  - FPGA architecture
    - CLB, RAM
    - IO, Interconnects
- FPGA Design Flow
  - Synthesis
  - Place
  - Route

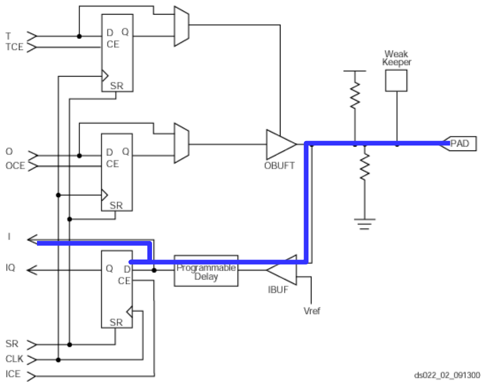
Basic I/O Block (IOB) Structure



IOB Functionality

- IOB provides interface between the package pins and CLBs
- Each IOB can work as uni- or bi-directional I/O
- Outputs can be forced into High Impedance
- Inputs and outputs can be registered
  - advised for high-performance I/O
- Inputs can be delayed

Example: Virtex-E IOB detail



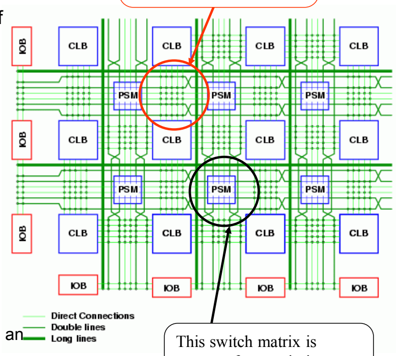
Interconnects: Routing

- Logic blocks embedded in a 'sea' of connection resources
- CLB = logic block
  IOB = I/O buffer
  PSM = programmable
  switch matrix
- Interconnections critical
  - Transmission gates on paths
    - ⇨ Flexibility
    - ⇨ Connect any LB to any other

  but

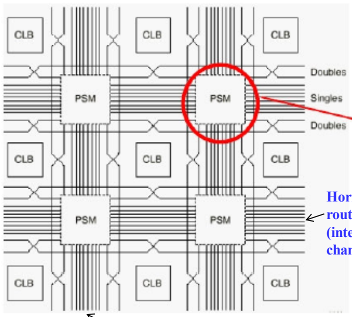
    - ✕ Much slower than connections within a logic block
    - ✕ Much slower than long lines on an ASIC

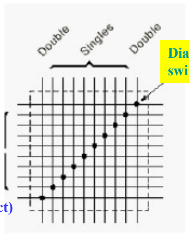
Every one of these connection points is a transmission gate



This switch matrix is a mass of transmission gates too!

Programmable switch matrix





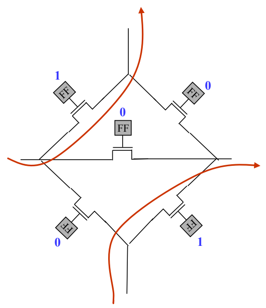
Diamond switch

Horizontal routing (interconnect) channel

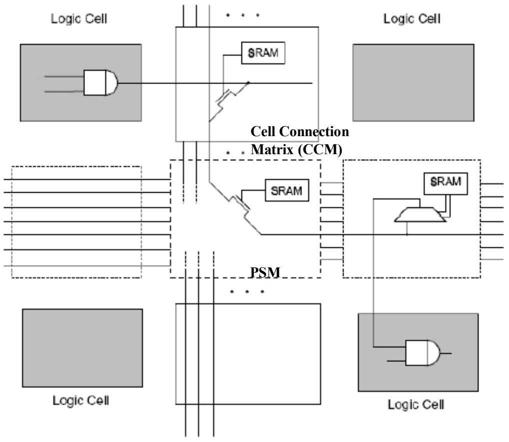
Vertical routing channels

PSM: Programmable Switch Matrix (for making connections between interconnects of different channels). The structure shown only allows *i-to-i* connections

Diamond switch

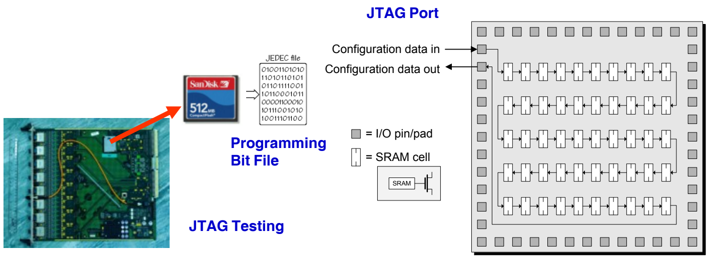


Example: SRAM-type FPGA Interconnection



Configuring an FPGA

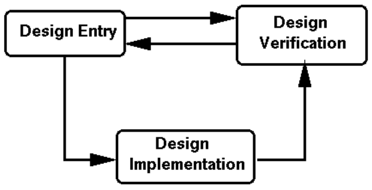
- Millions of SRAM cells holding LUTs and Interconnect Routing info
- Volatile Memory. Loses configuration when board power is turned off.
- Keep Bit Pattern describing the SRAM cells in non-Volatile Memory e.g. ROM or Digital Camera card
- Configuration takes ~ secs



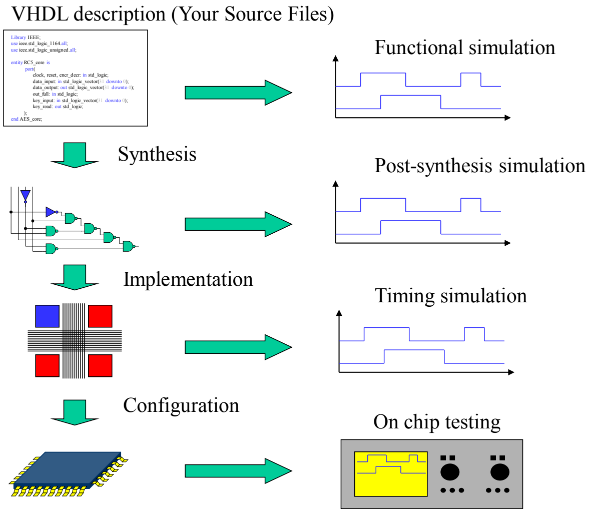
Overview

- FPGA Devices
  - ASIC vs. FPGA
  - FPGA architecture
    - CLB, RAM
    - IO, Interconnects
- FPGA Design Flow
  - Synthesis
  - Place
  - Route

FPGA Generic Design Flow



- Design Entry:
  - Create your design files using:
    - schematic editor or
    - hardware description language (VHDL, Verilog)
- Design implementation on FPGA:
  - *Partition, place, and route* to create bit-stream file
- Design verification:
  - Use Simulator to check function.
  - Load onto FPGA device (cable connects PC to development board)
  - Check operation at full speed in real environment



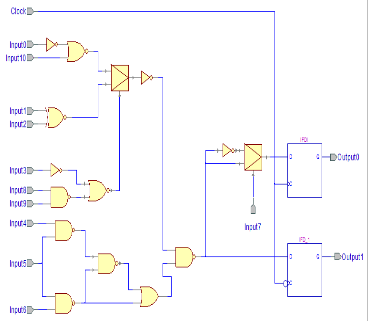
```
Library IEEE;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity RC5_core is
        port(
            clock, reset, encr_decr: in std_logic;
            data_input: in std_logic_vector(31 downto 0);
            data_output: out std_logic_vector(31 downto 0);
            out_full: in std_logic;
            key_input: in std_logic_vector(31 downto 0);
            key_read: out std_logic;
        );
end AES_core;
```

Logic Synthesis

VHDL description

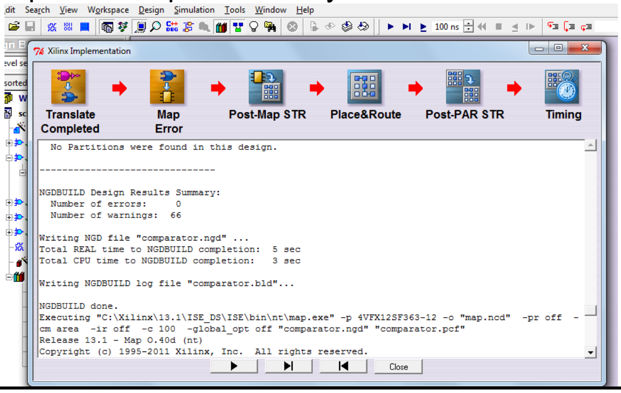
```
architecture MLU_DATAFLOW of MLU is

signal A1:STD_LOGIC;
signal B1:STD_LOGIC;
signal Y1:STD_LOGIC;
signal MUX_0, MUX_1, MUX_2, MUX_3: STD_LOGIC;

begin
            A1<=A when (NEG_A='0') else
                        not A;
            B1<=B when (NEG_B='0') else
                        not B;
            Y<=Y1 when (NEG_Y='0') else
                        not Y1;

            MUX_0<=A1 and B1;
            MUX_1<=A1 or B1;
            MUX_2<=A1 xor B1;
            MUX_3<=A1 xnor B1;

            with (L1 & L0) select
                        Y1<=MUX_0 when "00",
                                    MUX_1 when "01",
                                    MUX_2 when "10",
                                    MUX_3 when others;

end MLU_DATAFLOW;
```

Circuit netlist



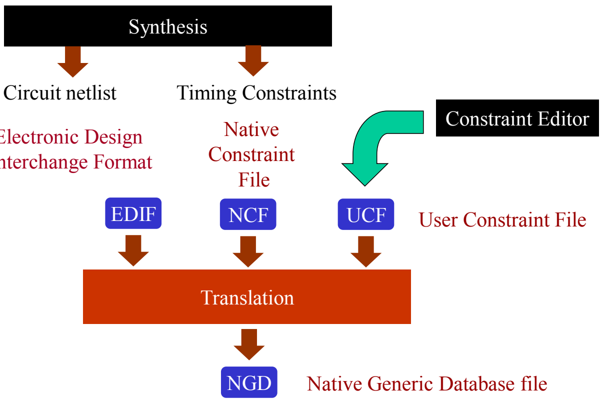
Implementation

- After synthesis the entire implementation process is performed by FPGA vendor tools

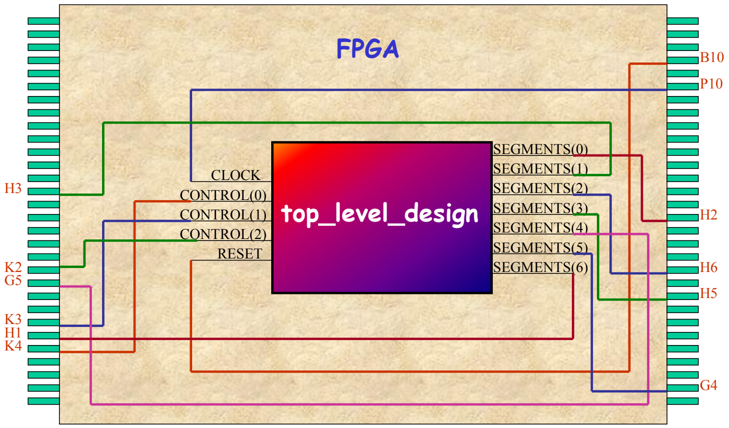


```
 No Partitions were found in this design.

-------------------------------

NGDBUILD Design Results Summary:
  Number of errors:     0
  Number of warnings:  66

Writing NGD file "comparator.ngd" ...
Total REAL time to NGDBUILD completion:  5 sec
Total CPU time to NGDBUILD completion:   3 sec

Writing NGDBUILD log file "comparator.bld"...

NGDBUILD done.
Executing "C:\Xilinx\13.1\ISE_DS\ISE\bin\nt\map.exe" -p 4VFX12SF363-12 -o "map.ncd"  -pr off  -
cm area  -ir off  -c 100  -global_opt off "comparator.ngd" "comparator.pcf"
Release 13.1 - Map O.40d (nt)
Copyright (c) 1995-2011 Xilinx, Inc.  All rights reserved.
```

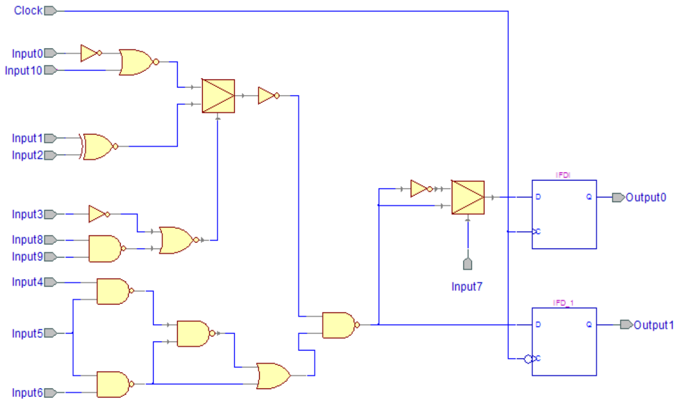
Translation



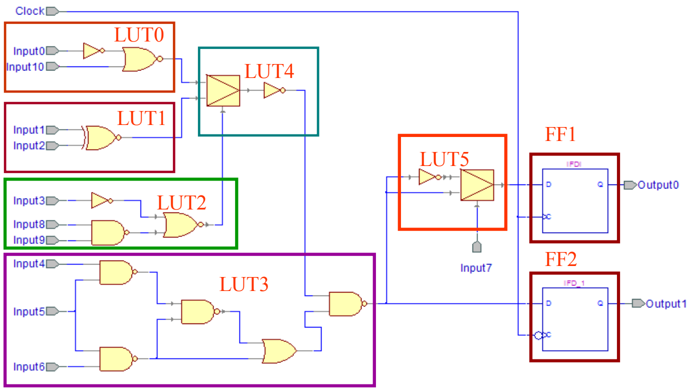
Pin Assignment



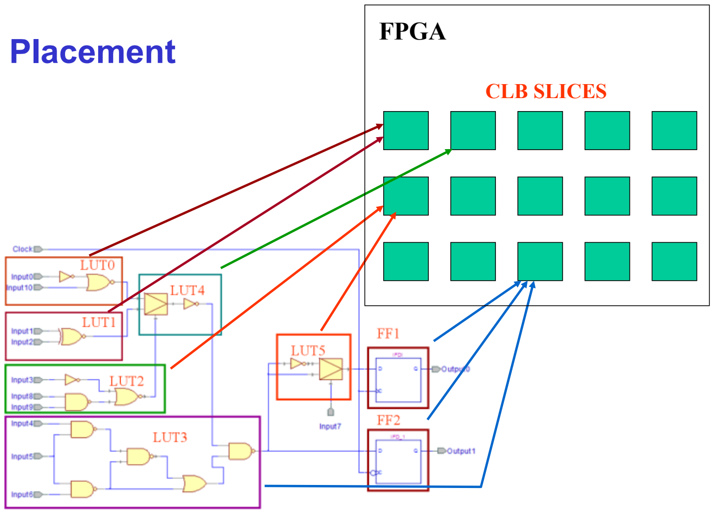
Circuit netlist



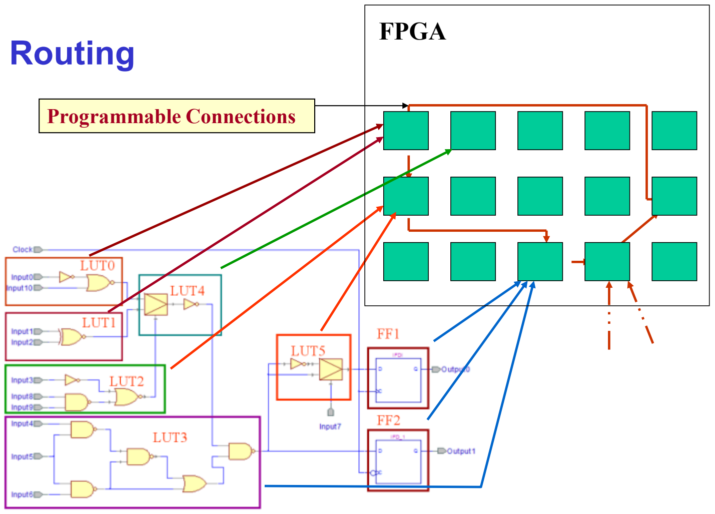
Mapping



Placement



Routing



Configuration

- Once a design is implemented, you must create a file that the FPGA can understand
  - This file is called a **bitstream**: a BIT file (.bit extension)
- The BIT file can be downloaded directly to the FPGA, or can be converted into a PROM file which stores the programming information

Map report

```
Design Summary
--------------
Number of errors:        0
Number of warnings:      0
Logic Utilization:
  Number of Slice Flip Flops:          30 out of  26,624    1%
  Number of 4 input LUTs:              38 out of  26,624    1%
Logic Distribution:
  Number of occupied Slices:                     33 out of  13,312    1%
  Number of Slices containing only related logic:      33 out of      33  100%
  Number of Slices containing unrelated logic:          0 out of      33    0%
  *See NOTES below for an explanation of the effects of unrelated logic
Total Number 4 input LUTs:              62 out of  26,624    1%
  Number used as logic:                38
  Number used as a route-thru:         24
  Number of bonded IOBs:               10 out of     221    4%
    IOB Flip Flops:                 7
  Number of GCLKs:                  1 out of       8   12%
```

Place & route report

```
Asterisk (*) preceding a constraint indicates it was not met.
   This may be due to a setup or hold violation.

------------------------------------------------------------------------------------------------------
  Constraint                                  | Requested  | Actual      | Logic  | Absolute    |Number of
                                              |            |             | Levels | Slack       |errors
------------------------------------------------------------------------------------------------------
* TS_CLOCK = PERIOD TIMEGRP "CLOCK" 5 ns      | 5.000ns    | 5.140ns     | 4      | -0.140ns    | 5
  HIGH 50%                                    |            |             |        |             |
------------------------------------------------------------------------------------------------------
  TS_gen1Hz_Clock1Hz = PERIOD TIMEGRP "gen1   | 5.000ns    | 4.137ns     | 2      | 0.863ns     | 0
  "gen1Hz_Clock1Hz" 5 ns HIGH 50%             |            |             |        |             |
------------------------------------------------------------------------------------------------------
```

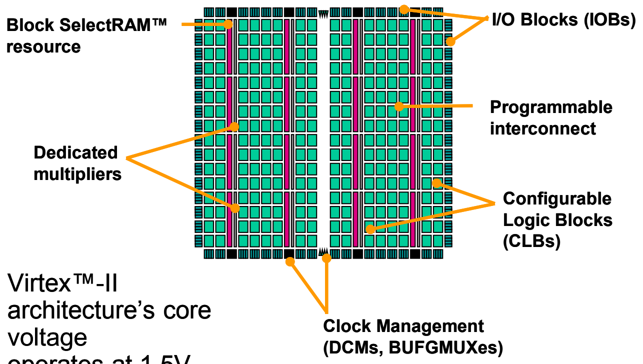
Post layout timing report

```
Clock to Setup on destination clock CLOCK
---------------+---------+---------+---------+---------+
               | Src:Rise| Src:Fall| Src:Rise| Src:Fall|
Source Clock   |Dest:Rise|Dest:Rise|Dest:Fall|Dest:Fall|
---------------+---------+---------+---------+---------+
CLOCK          |    5.140|         |         |         |
---------------+---------+---------+---------+---------+


Timing summary:
---------------

Timing errors: 9  Score: 543

Constraints cover 574 paths, 0 nets, and 187 connections

Design statistics:
   Minimum period:   5.140ns  (Maximum frequency: 194.553MHz)
```

Summary

- FPGAs are more and more prevalent!
- They offer a flexible platform for increasingly complex systems
- Design automation tools take care of the entire design process from VHDL → configuration bitstream file

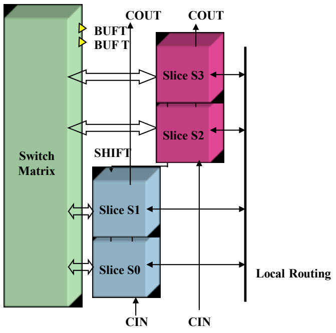
Appendix A: other FPGA architectures Virtex-II


Block SelectRAM™ resource
Dedicated multipliers
I/O Blocks (IOBs)
Programmable interconnect
Configurable Logic Blocks (CLBs)
Clock Management (DCMs, BUFGMUXes)

Virtex™-II architecture's core voltage operates at 1.5V

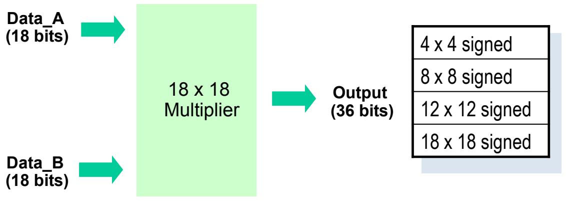
Slices and CLBs

- Each Virtex-II CLB contains four slices
  - Local routing provides feedback between slices in the same CLB, and it provides routing to neighboring CLBs
  - A switch matrix provides access to general routing resources

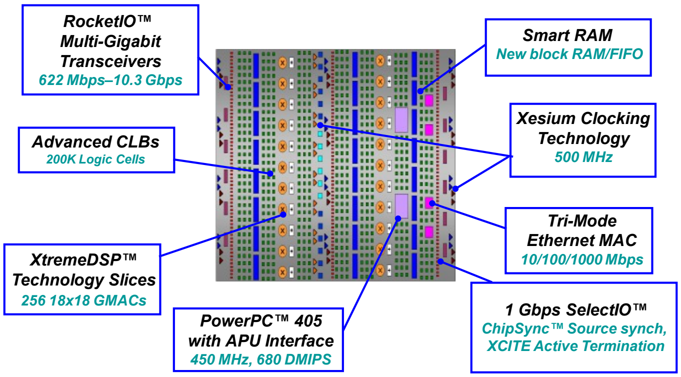


Dedicated Multiplier Blocks

- 18-bit twos complement signed operation
- Optimized to implement Multiply and Accumulate functions
- Multipliers are physically located next to block SelectRAM™ memory



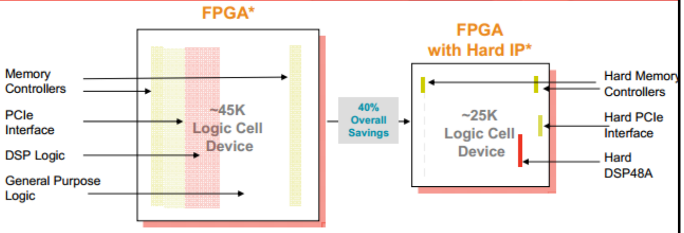
Virtex-4 Architecture


RocketIO™ Multi-Gigabit Transceivers
622 Mbps–10.3 Gbps
Advanced CLBs
200K Logic Cells
XtremeDSP™ Technology Slices
256 18x18 GMACs
PowerPC™ 405 with APU Interface
450 MHz, 680 DMIPS
Smart RAM
New block RAM/FIFO
Xesium Clocking Technology
500 MHz
Tri-Mode Ethernet MAC
10/100/1000 Mbps
1 Gbps SelectIO™
ChipSync™ Source synch, XCITE Active Termination

Choose the Platform that Best Fits the Application!

Resource	LX	FX	SX
Logic	14K–200K LCs	12K–140K LCs	23K–55K LCs
Memory	0.9–6 Mb	0.6–10 Mb	2.3–5.7 Mb
DCMs	4–12	4–20	4–8
DSP Slices	32–96	32–192	128–512
SelectIO	240–960	240–896	320–640
RocketIO	N/A	0–24 Channels	N/A
PowerPC	N/A	1 or 2 Cores	N/A
Ethernet MAC	N/A	2 or 4 Cores	N/A

Spartan-6 FPGA Big Cost Savings:
Hard Memory, DSP, PCIe Blocks



▪ 2 SDRAM Controllers	6K Logic Cells	**Hard Block**
▪ PCIe Interface	6K Logic Cells	**Hard Block**
▪ General Purpose Logic	22K Logic Cells	~24K Logic Cells**
▪ DSP (FFT, FIR, Symmetric TAP etc.)	11K Logic Cells	**Hard Block**
▪ Total	45K Logic Cells	24K Logic Cells

Hard IP Blocks Provide 80%+ Die Area Savings