

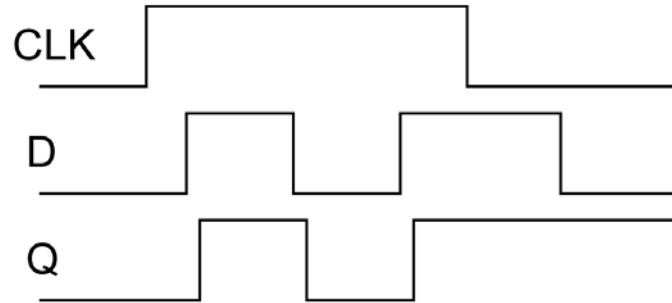
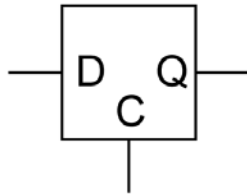
EECS150 - Digital Design
Lecture 3 - Field Programmable Gate
Arrays (FPGAs)

January 28, 2003

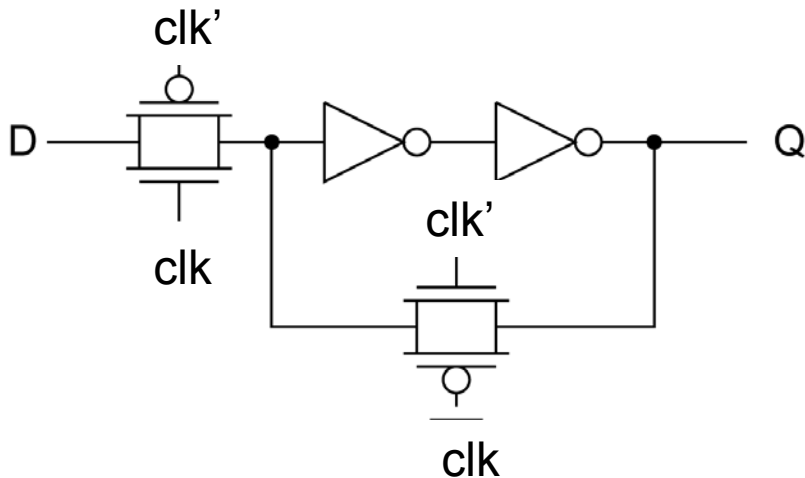
John Wawrzynek

Transistor-level Logic Circuits

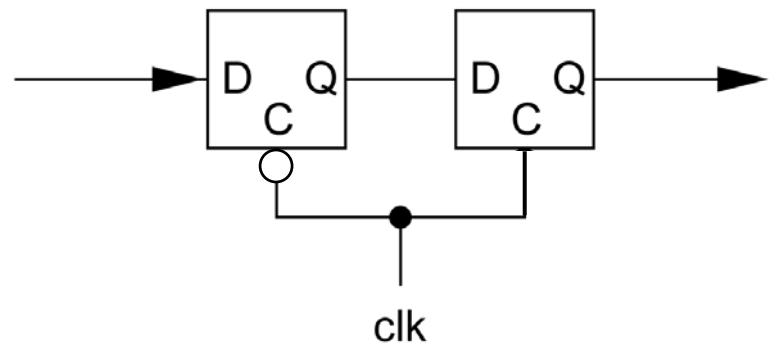
- Positive Level-sensitive latch



- Transistor Level

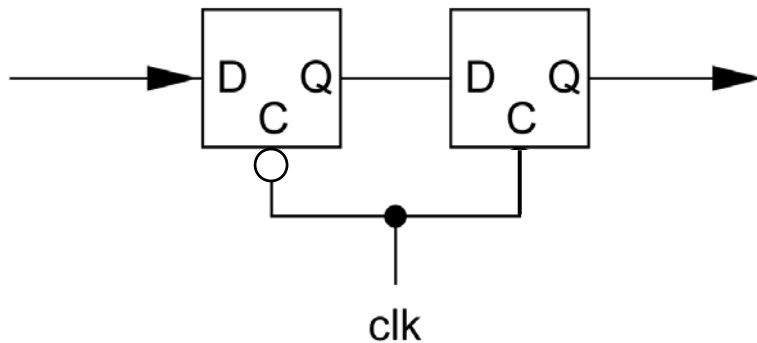


- Positive Edge-triggered flip-flop built from two level-sensitive latches:

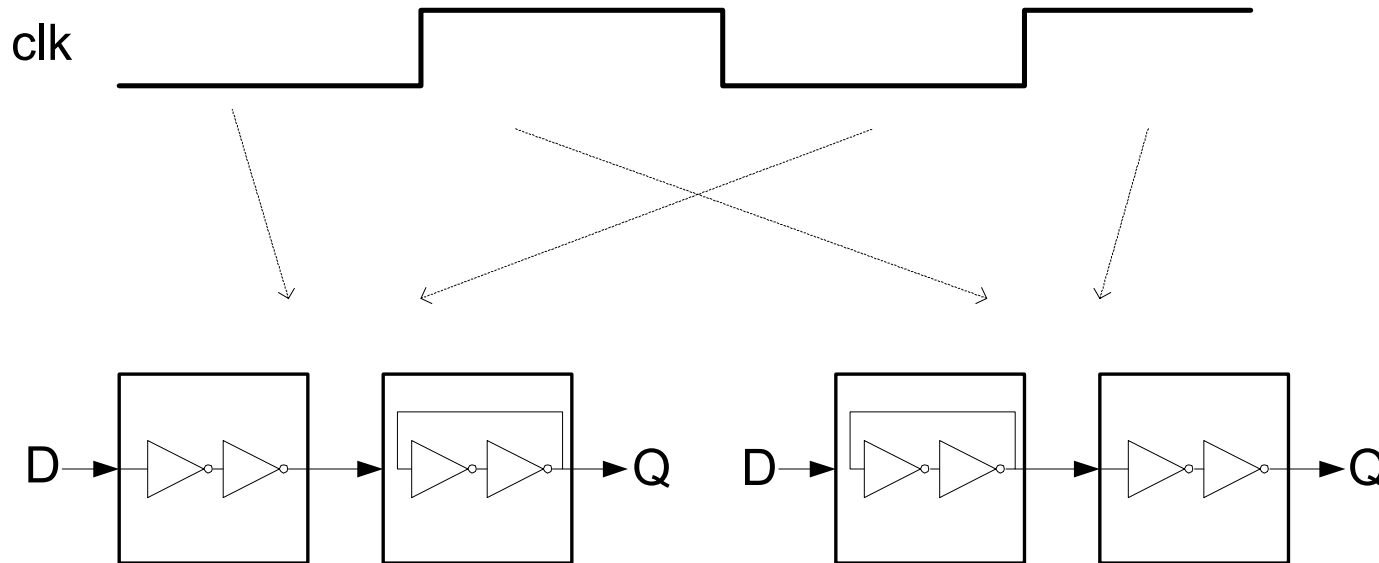


Positive Edge-triggered Flip-flop

- Flip-flop built from two latches:



- When clk low, left latch acts as feedthrough, and Q is stored value of right latch.
- When clk high left latch stores values and right latch acts as feedthrough.

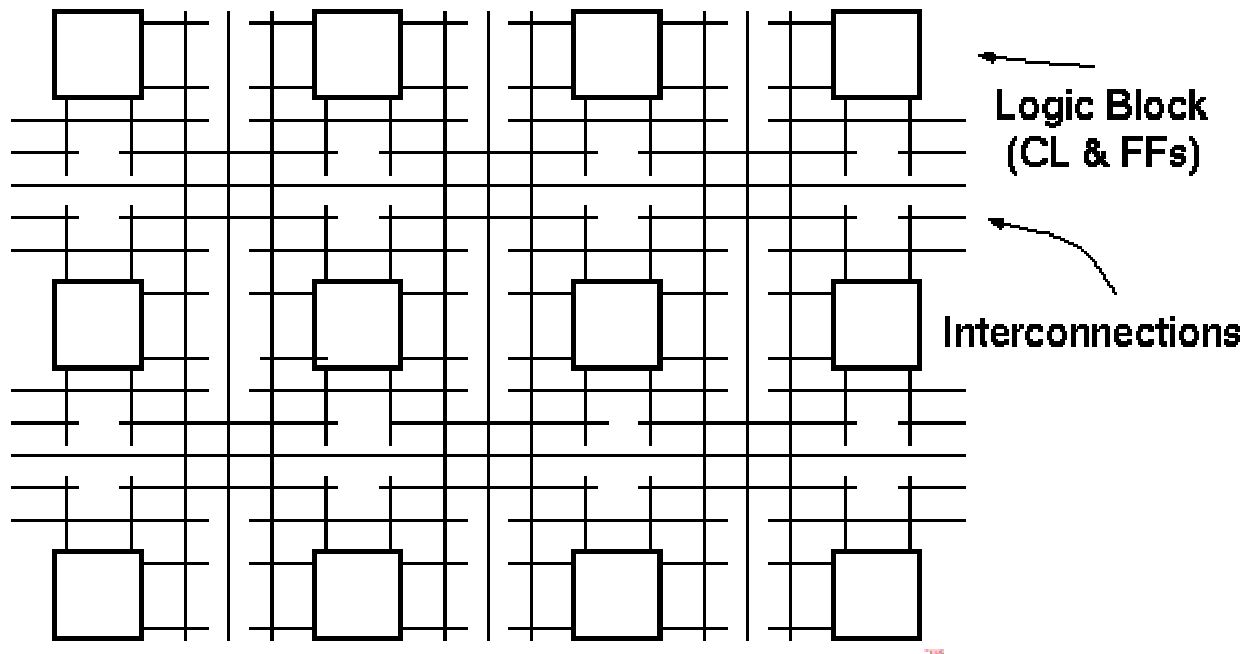


Outline

- What are FPGAs?
- Why use FPGAs (a short history lesson).
- FPGA variations
- Internal logic blocks.
- Designing with FPGAs.
- Specifics of Xilinx Virtex-E series.

FPGA Overview

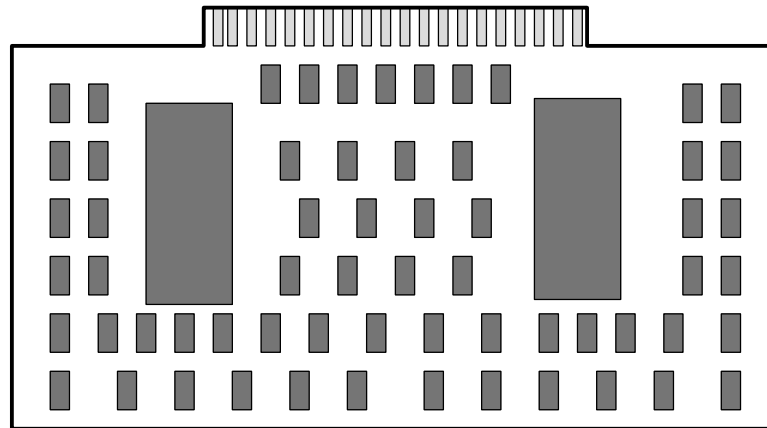
- Basic idea: two-dimensional array of logic blocks and flip-flops with a means for the user to configure:
 1. the interconnection between the logic blocks,
 2. the function of each block.



Simplified version of FPGA internal architecture:

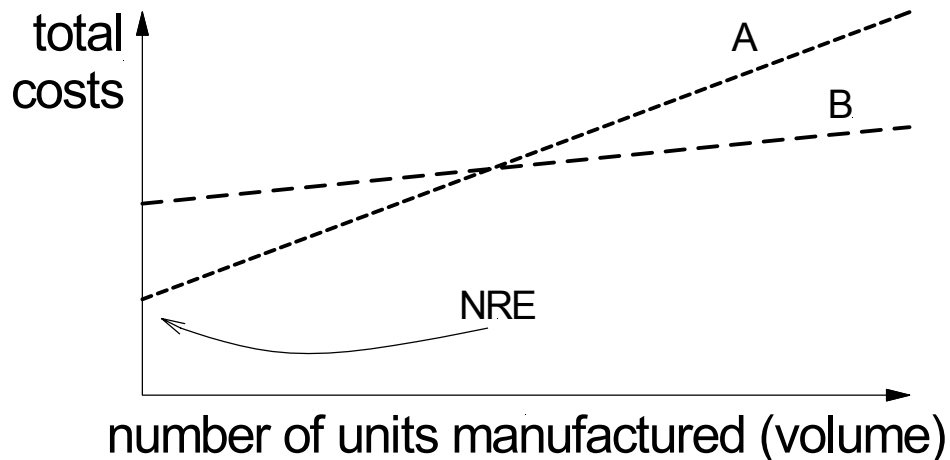
Why FPGAs?

- By the early 1980's most of the logic circuits in typical systems were absorbed by a handful of standard large scale integrated circuits (LSI).
 - Microprocessors, bus/IO controllers, system timers, ...
- Every system still had the need for random “glue logic” to help connect the large ICs:
 - generating global control signals (for resets etc.)
 - data formatting (serial to parallel, multiplexing, etc.)
- Systems had a few LSI components and lots of small low density SSI (small scale IC) and MSI (medium scale IC) components.



Why FPGAs?

- Custom ICs where sometimes designed to replace the large amount of glue logic:
 - reduced system complexity and manufacturing cost, improved performance.
 - However, custom ICs are relatively 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?

- Therefore the custom IC approach was only viable for products with very high volume (where NRE could be amortized), and which were not time to market sensitive.
- FPGAs were 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)
 - 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.

Why FPGAs?

- FPGAs continue to compete with custom ICs for special processing functions (and glue logic) but now also compete with microprocessors in dedicated and embedded applications.
 - Performance advantage over microprocessors because circuits can be customized for the task at hand. Microprocessors must provide special functions in software (many cycles).

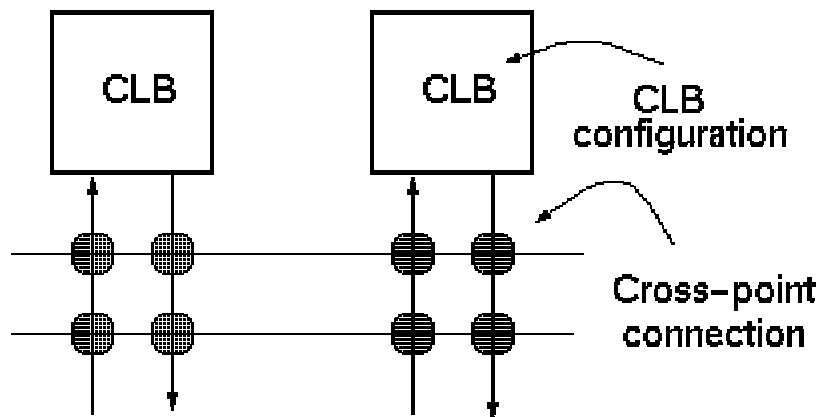
- Summary:

	performance	NREs	Unit cost	TTM
↑	ASIC	ASIC	FPGA	ASIC
	FPGA	FPGA	MICRO	FPGA
	MICRO	MICRO	ASIC	MICRO

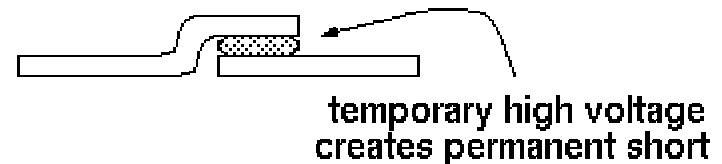
ASIC = custom IC, MICRO = microprocessor

FPGA Variations

- Families of FPGA's differ in:
 - physical means of implementing user programmability,
 - arrangement of interconnection wires, and
 - the basic functionality of the logic blocks.
- Most significant difference is in the method for providing flexible blocks and connections:



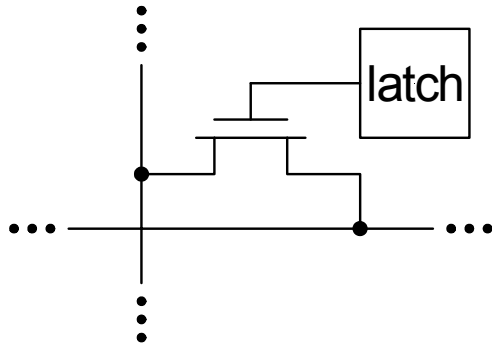
- Anti-fuse based (ex: Actel)



- + Non-volatile, relatively small
- fixed (non-reprogrammable)

User Programmability

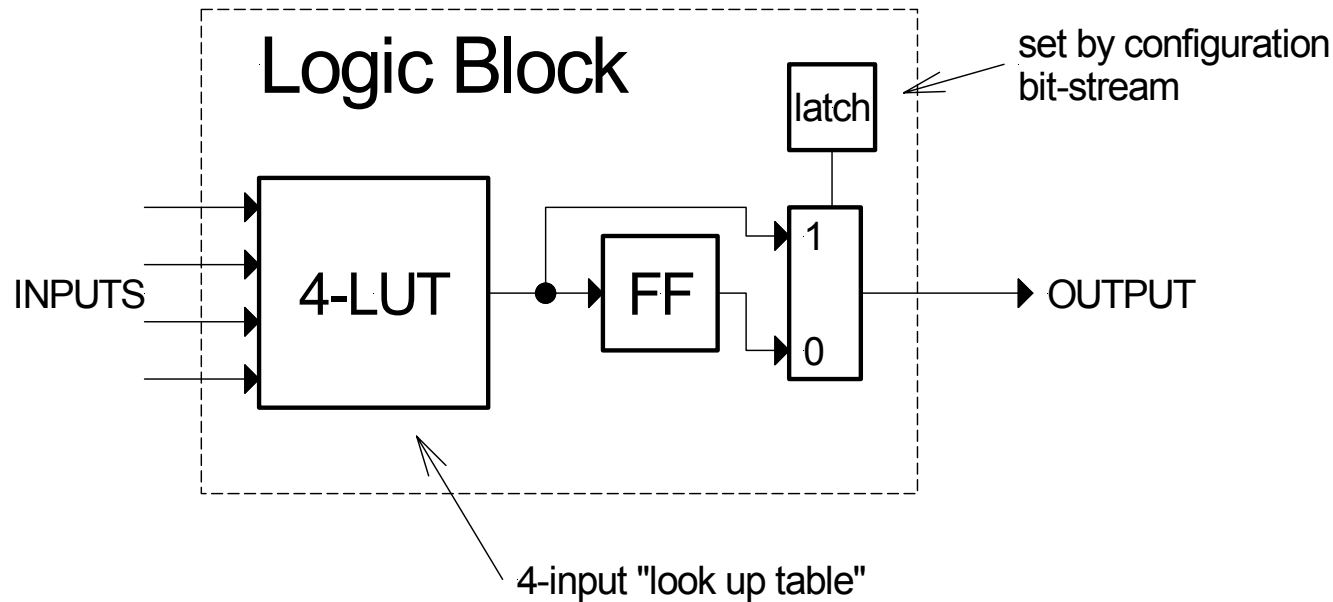
- Latch-based (Xilinx, Altera, ...)



- + reconfigurable
- volatile
- relatively large.

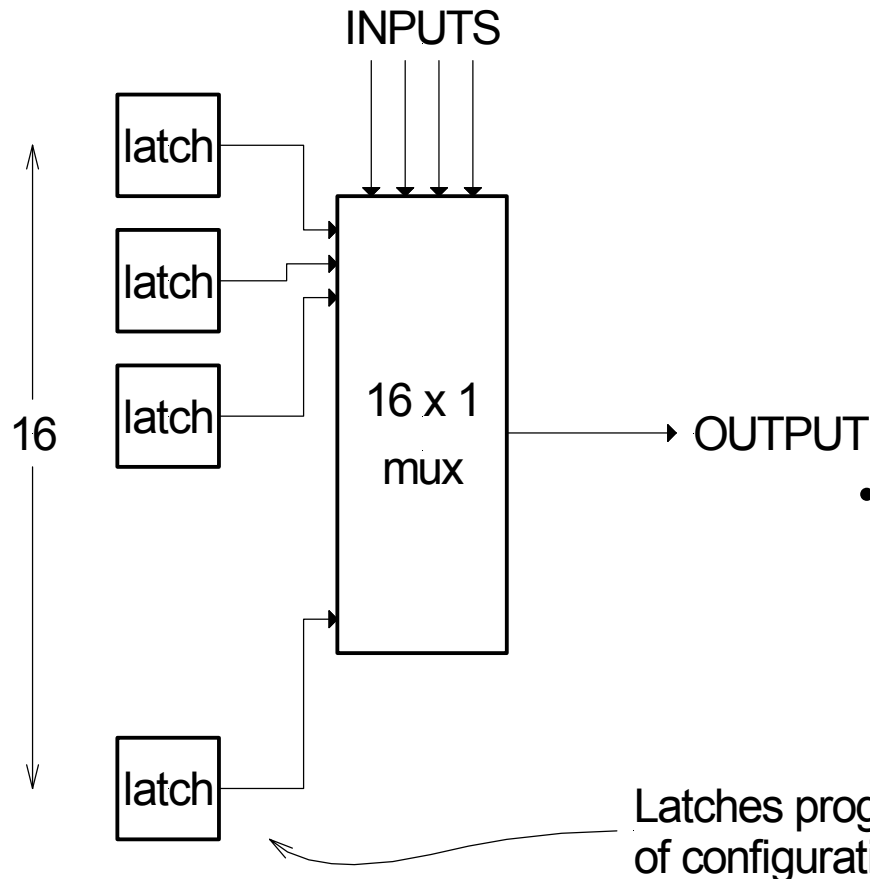
- Latches are used to:
 1. make or break cross-point connections in the interconnect
 2. define the function of the logic blocks
 3. set user options:
 - within the logic blocks
 - in the input/output blocks
 - global reset/clock
- “Configuration bit stream” can be loaded under user control:
 - All latches are strung together in a shift chain:

Idealized FPGA Logic Block



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

4-LUT Implementation



- n-bit LUT is implemented as a $2^n \times 1$ 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!

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	

Implements *any* function of 2 inputs.

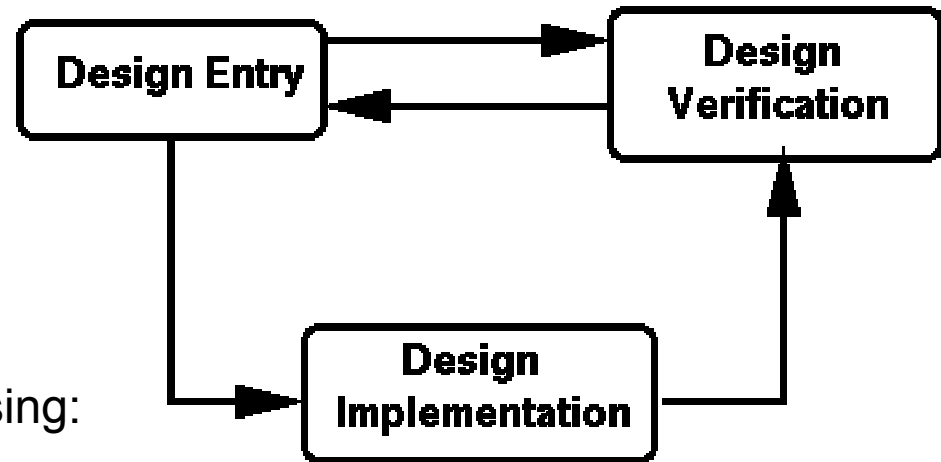
How many of these are there?

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		

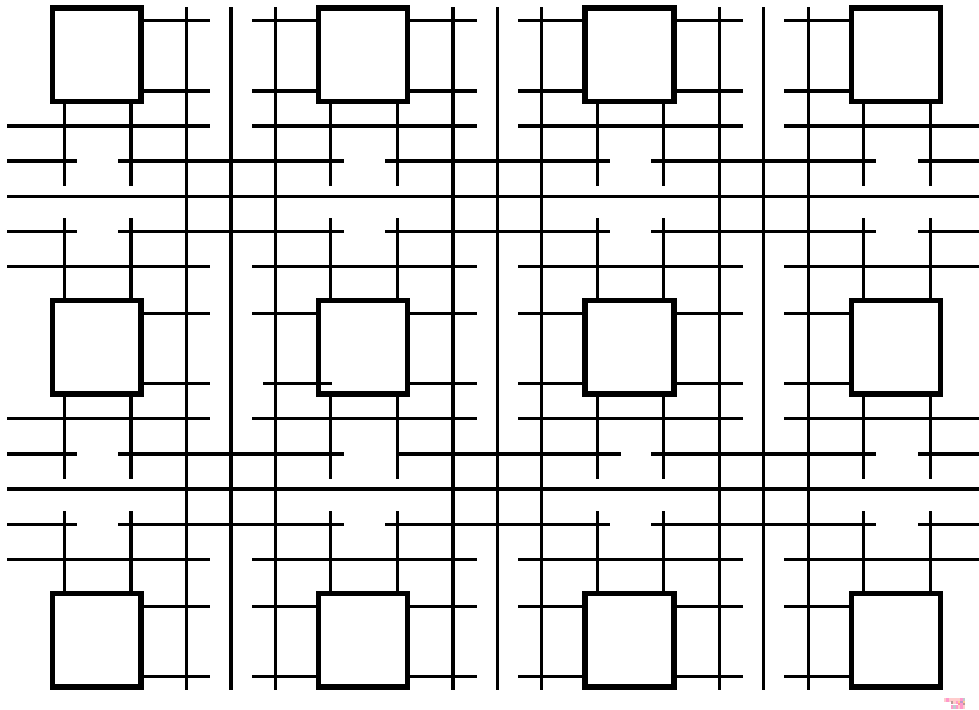
FPGA Generic Design Flow



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

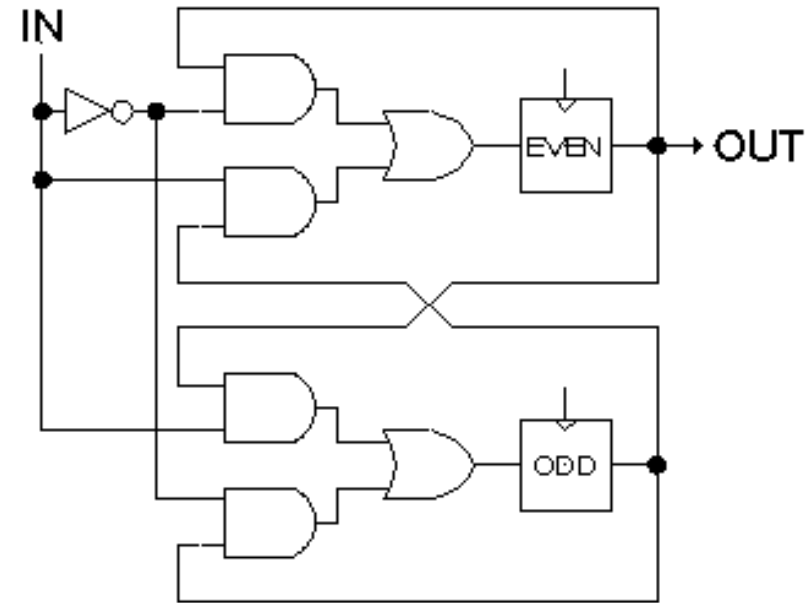
Example Partition, Placement, and Route

- Idealized FPGA structure:



- Example Circuit:

- collection of gates and flip-flops



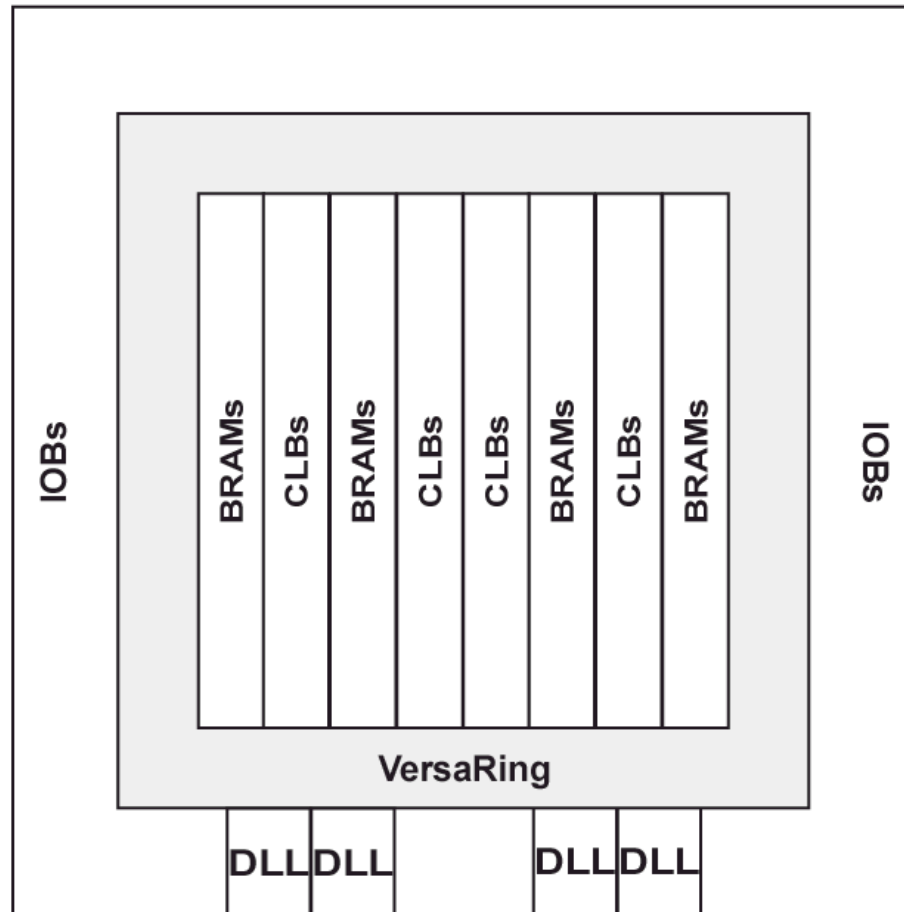
Circuit combinational logic must be “covered” by 4-input 1-output “gates”.

Flip-flops from circuit must map to FPGA flip-flops.

(Best to preserve “closeness” to CL to minimize wiring.)

Placement in general attempts to minimize wiring.

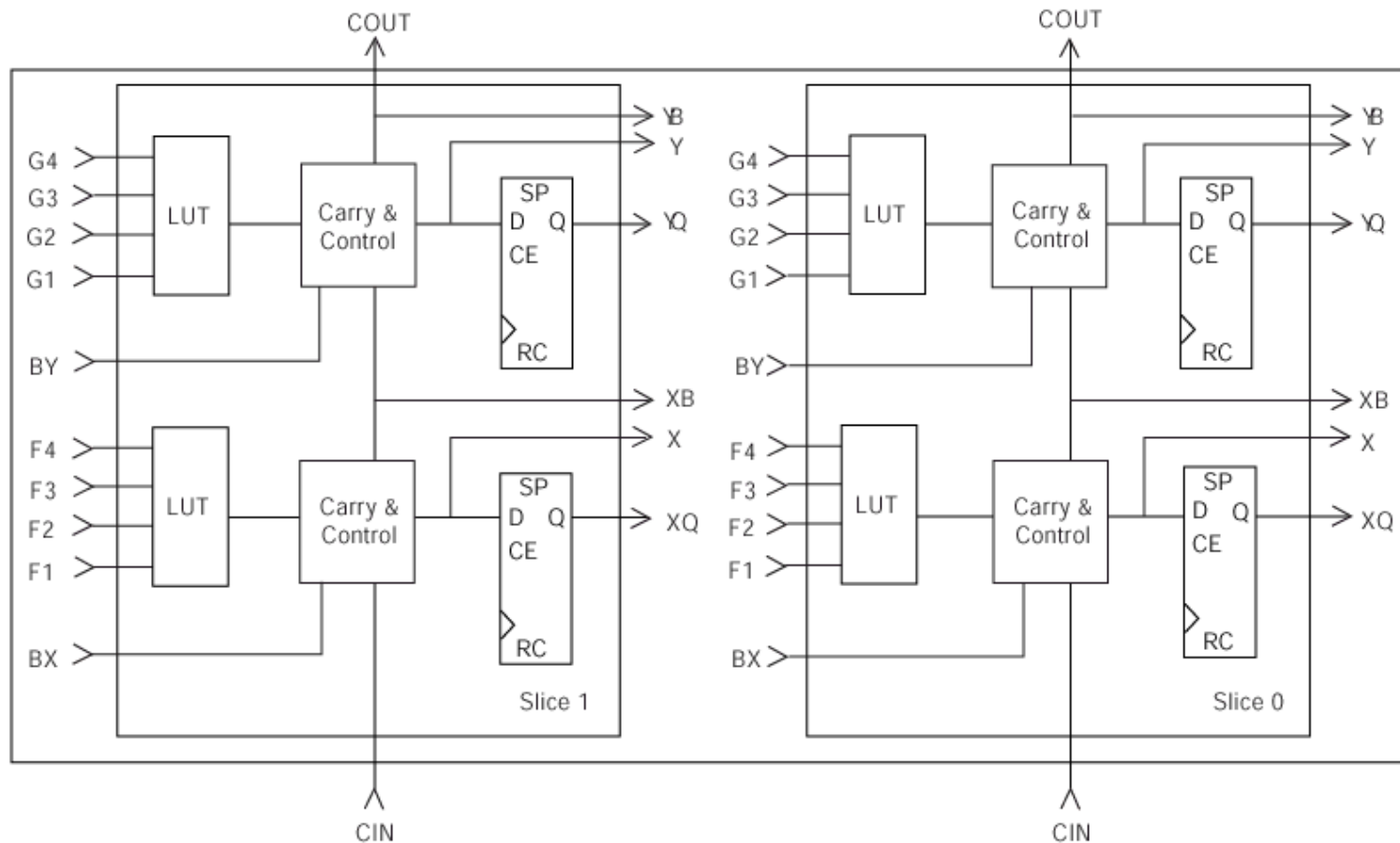
Xilinx Virtex-E Floorplan



ds022_01_121099

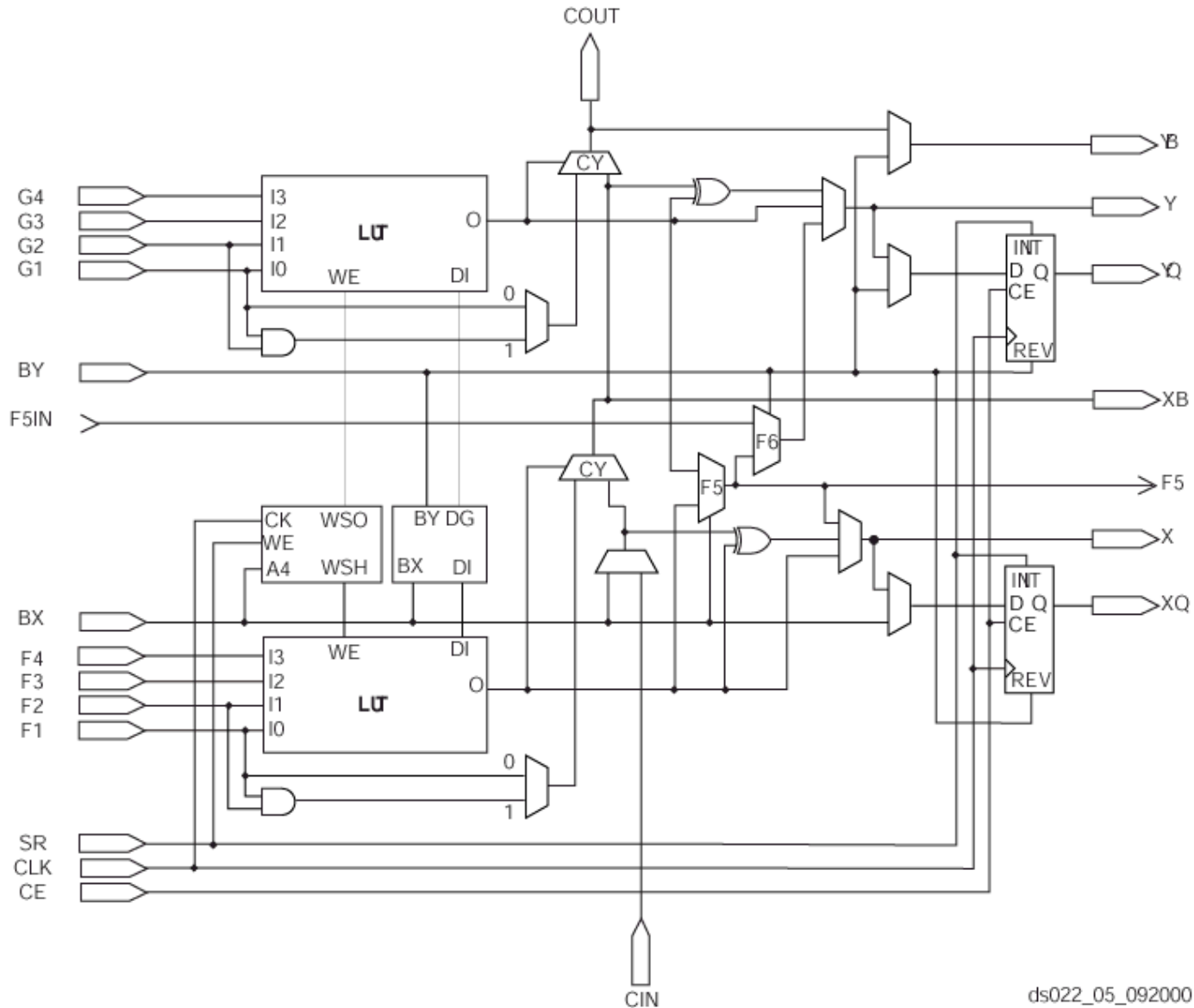
Virtex-E Configurable Logic Block (CLB)

2 “logic slices”



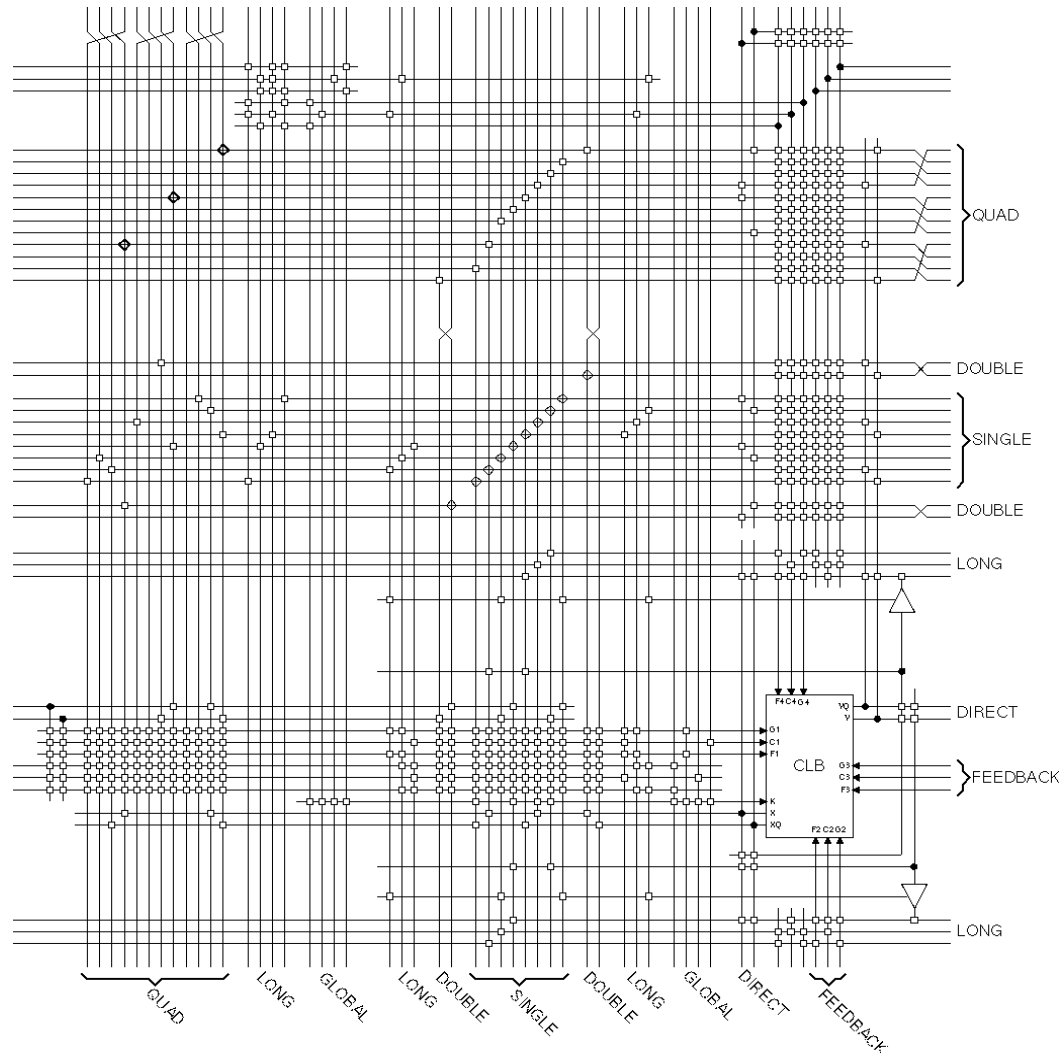
ds022_04_121799

Details of Virtex-E Slice

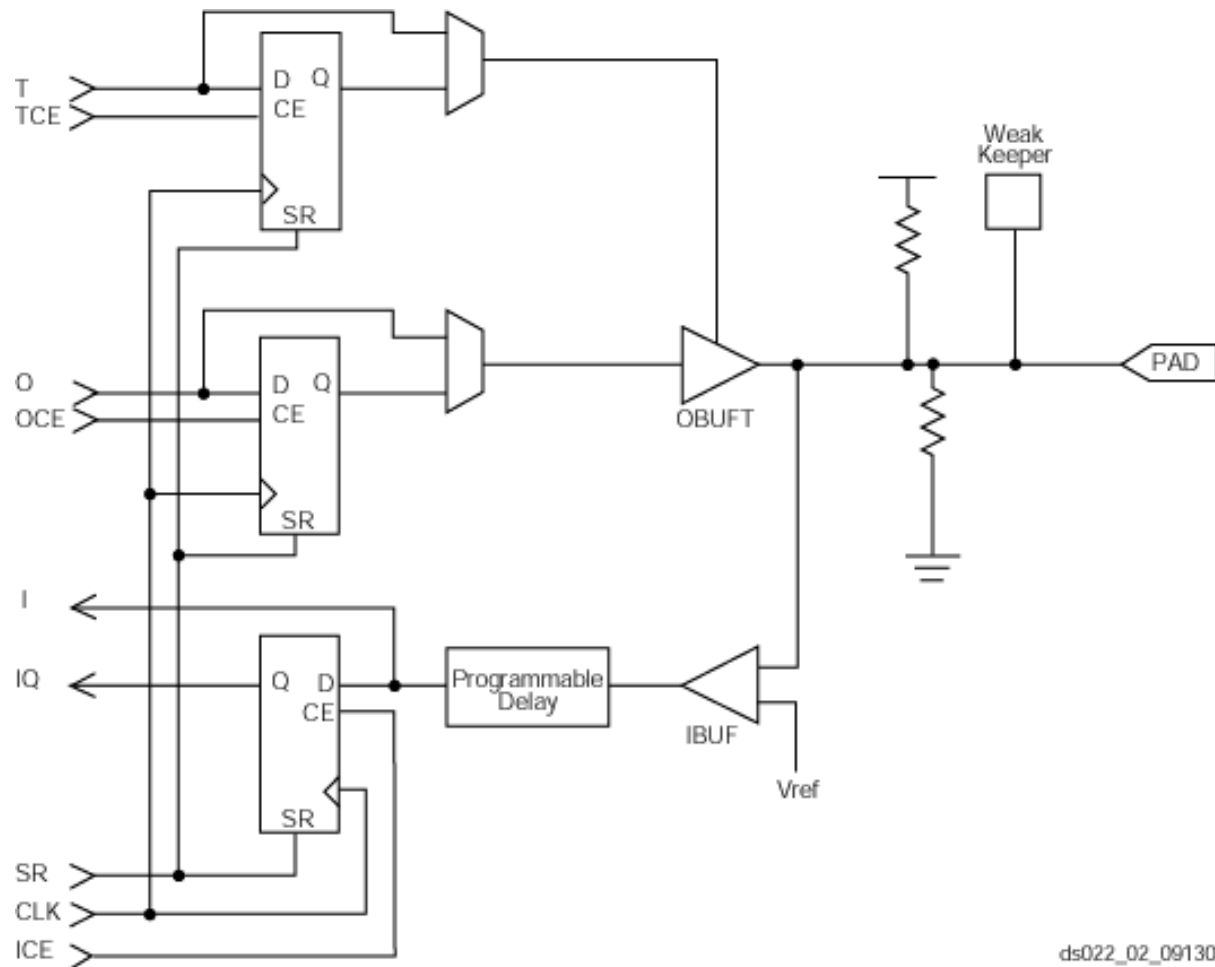


ds022_05_092000

Xilinx FPGAs (interconnect detail)



Virtex-E Input/Output block (IOB) detail



Virtex-E Family of Parts

Table 1: Virtex-E Field-Programmable Gate Array Family Members

Device	System Gates	Logic Gates	CLB Array	Logic Cells	Differential I/O Pairs	User I/O	BlockRAM Bits	Distributed RAM Bits
XCV50E	71,693	20,736	16 x 24	1,728	83	176	65,536	24,576
XCV100E	128,236	32,400	20 x 30	2,700	83	196	81,920	38,400
XCV200E	306,393	63,504	28 x 42	5,292	119	284	114,688	75,264
XCV300E	411,955	82,944	32 x 48	6,912	137	316	131,072	98,304
XCV400E	569,952	129,600	40 x 60	10,800	183	404	163,840	153,600
XCV600E	985,882	186,624	48 x 72	15,552	247	512	294,912	221,184
XCV1000E	1,569,178	331,776	64 x 96	27,648	281	660	393,216	393,216
XCV1600E	2,188,742	419,904	72 x 108	34,992	344	724	589,824	497,664
XCV2000E	2,541,952	518,400	80 x 120	43,200	344	804	655,360	614,400
XCV2600E	3,263,755	685,584	92 x 138	57,132	344	804	753,664	812,544
XCV3200E	4,074,387	876,096	104 x 156	73,008	344	804	851,968	1,038,336

Xilinx FPGAs

- How they differ from idealized array:
 - In addition to their use as general logic “gates”, LUTs can alternatively be used as general purpose RAM.
 - Each 4-lut can become a 16x1-bit RAM array.
 - Special circuitry to speed up “ripple carry” in adders and counters.
 - Therefore adders assembled by the CAD tools operate much faster than adders built from gates and luts alone.
 - Many more wires, including tri-state capabilities.