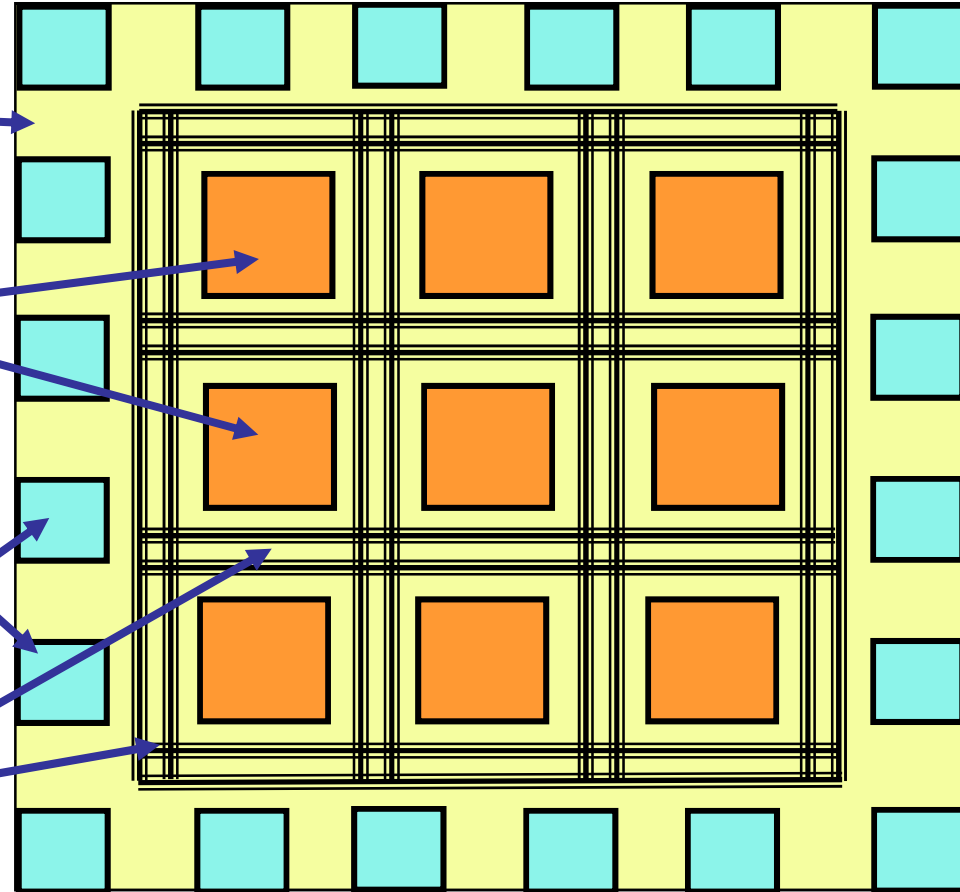


History of FPGAs

- Programmable Logic Arrays ~ 1970
 - Incorporated in VLSI devices
 - Can implement any set of SOP logic equations
 - Outputs can share common product terms
- Programmable Logic Devices ~ 1980
 - MMI Programmable Array Logic (PAL)
 - 16L8 – combinational logic only
 - 8 outputs with 7 programmable PTs of 16 input variables
 - 16R8 – sequential logic only
 - 8 registered outputs with 8 programmable PTs of 16 input variables
 - Lattice 16V8 (AMD 22V10) ~ 1983
 - 8 outputs with 8 programmable PTs of 16 input variables
 - Each output programmable to use or bypass flip-flop
 - Complex PLDs – arrays of PLDs with routing network ~ 1990
- Field Programmable Gate Arrays ~ 1986
 - Xilinx Logic Cell Array (LCA)
- CPLD & FPGA architectures became similar ~ 2000
 - Began to incorporate specialized cores

Field Programmable Gate Arrays

- Configuration Memory
- Programmable Logic Blocks (PLBs)
- Programmable Input/Output Cells
- Programmable Interconnect



Typical Complexity = 5M – 1B transistors

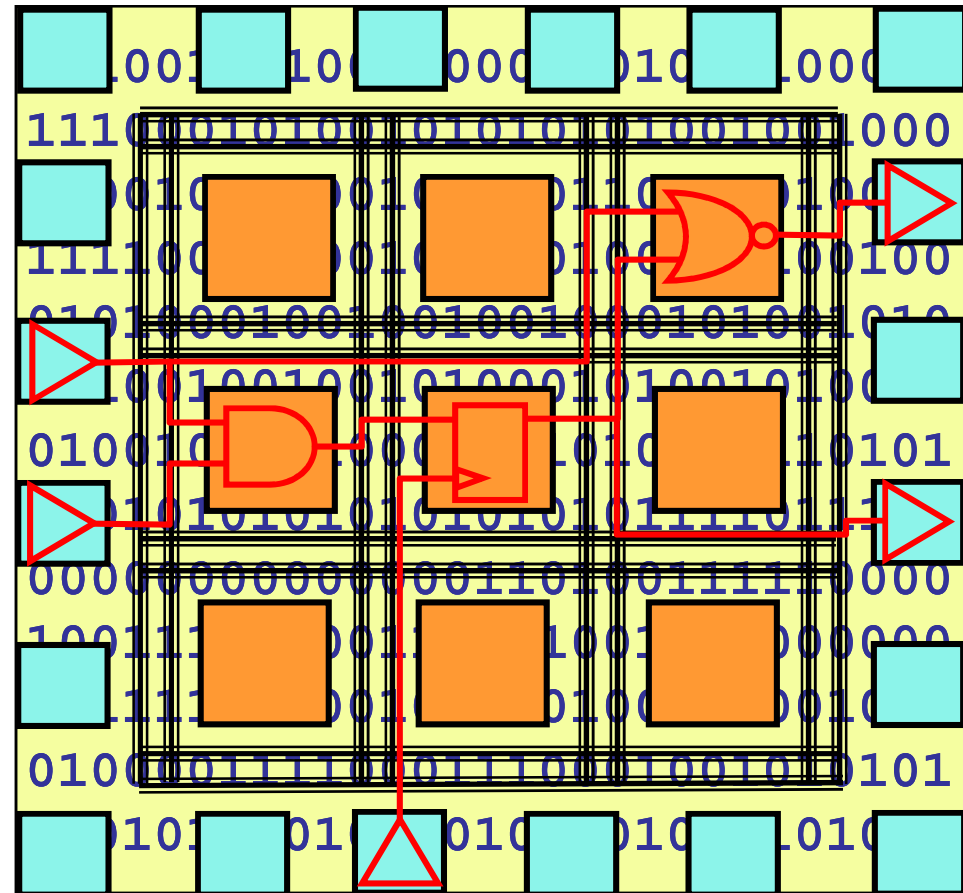
Basic FPGA Operation

Write Configuration Memory

- Defines system function
 - Input/Output Cells
 - Logic in PLBs
 - Connections between PLBs & I/O cells

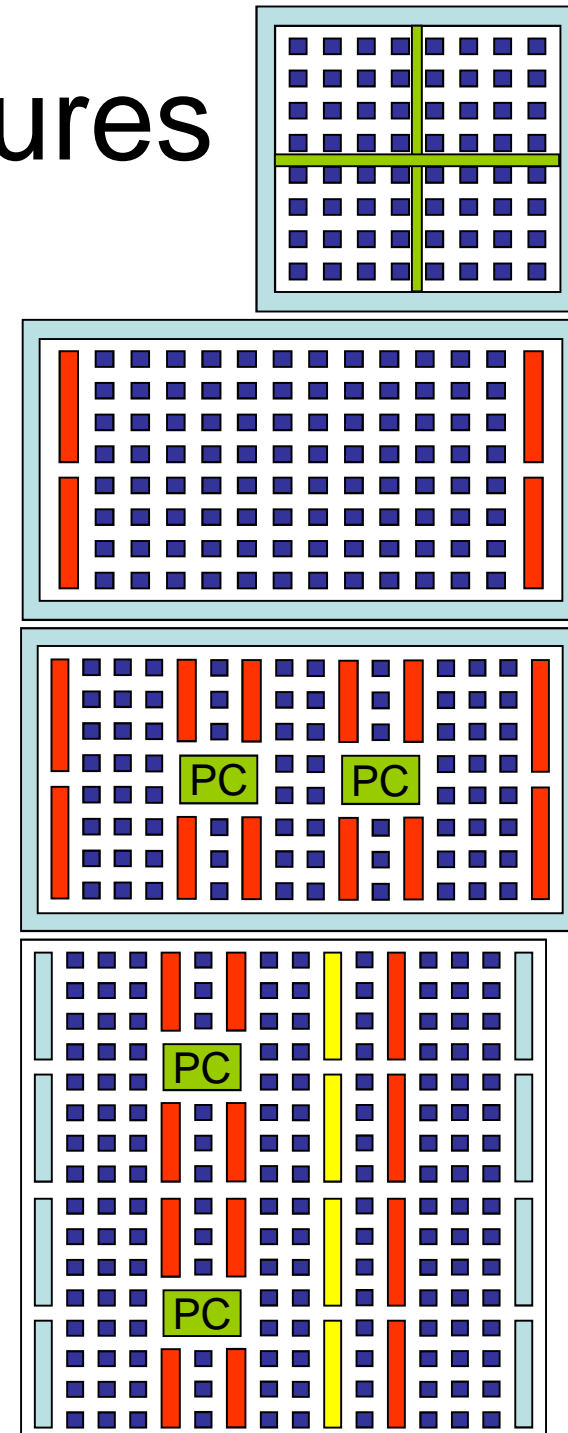
Changing configuration memory data => changes system function

- Can change at anytime
 - Even while system function is in operation
 - Run-time reconfiguration (RTR)



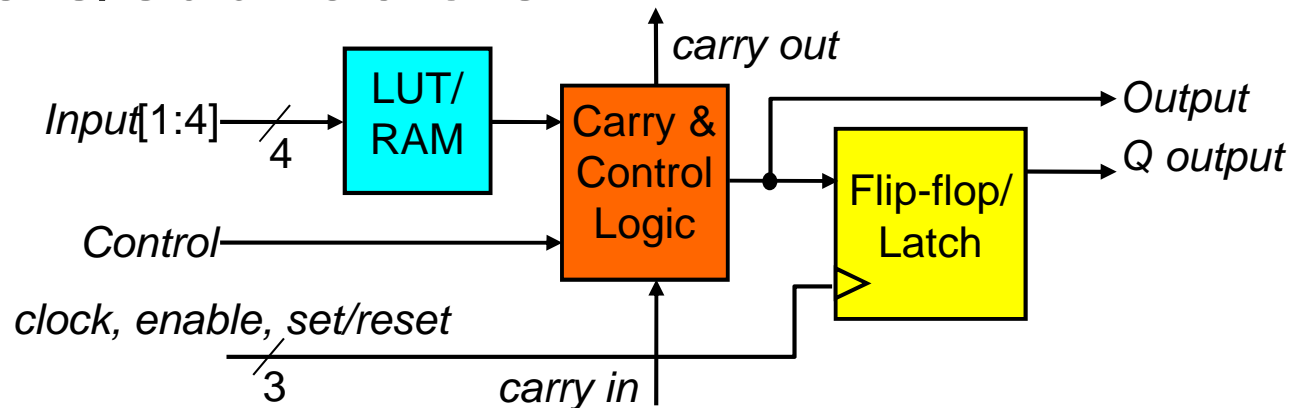
Xilinx FPGA Architectures

- 4000/Spartan ~1995
 - $N \times N$ array of unit cells
 - Unit cell = CLB + routing
 - Special routing along center axes
 - I/O cells around perimeter
- Virtex/Spartan-2 ~2000
 - $M \times N$ array of unit cells
 - Added block 4K RAMs at edges
- Virtex-2/Spartan-3 ~2005
 - Block 18K RAMs in array
 - Added 18x18 multipliers with each RAM
 - Added PowerPCs in Virtex-2 Pro
- Virtex-4/Virtex-5 ~2007
 - Added 48-bit DSP cores w/multipliers
 - I/O cells along columns for BGA



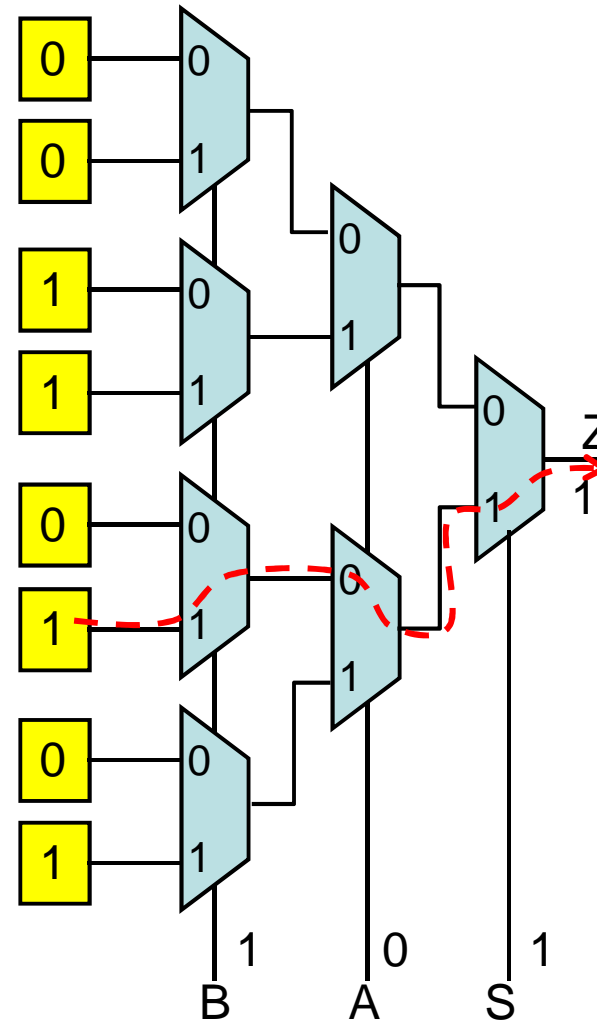
Basic PLB Architecture

- Look-up Table (LUT) implements truth table
- Memory elements:
 - Flip-flop/latch
 - Some FPGAs - LUTs can also implement small RAMs
- Carry & control logic implements fast adders/subtractors

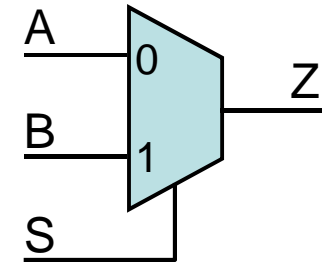


Look-up Tables

- Recall multiplexer example
- Configuration memory holds outputs for truth table
- Internal signals connect to control signals of multiplexers to select value of truth table for any given input value



Multiplexer

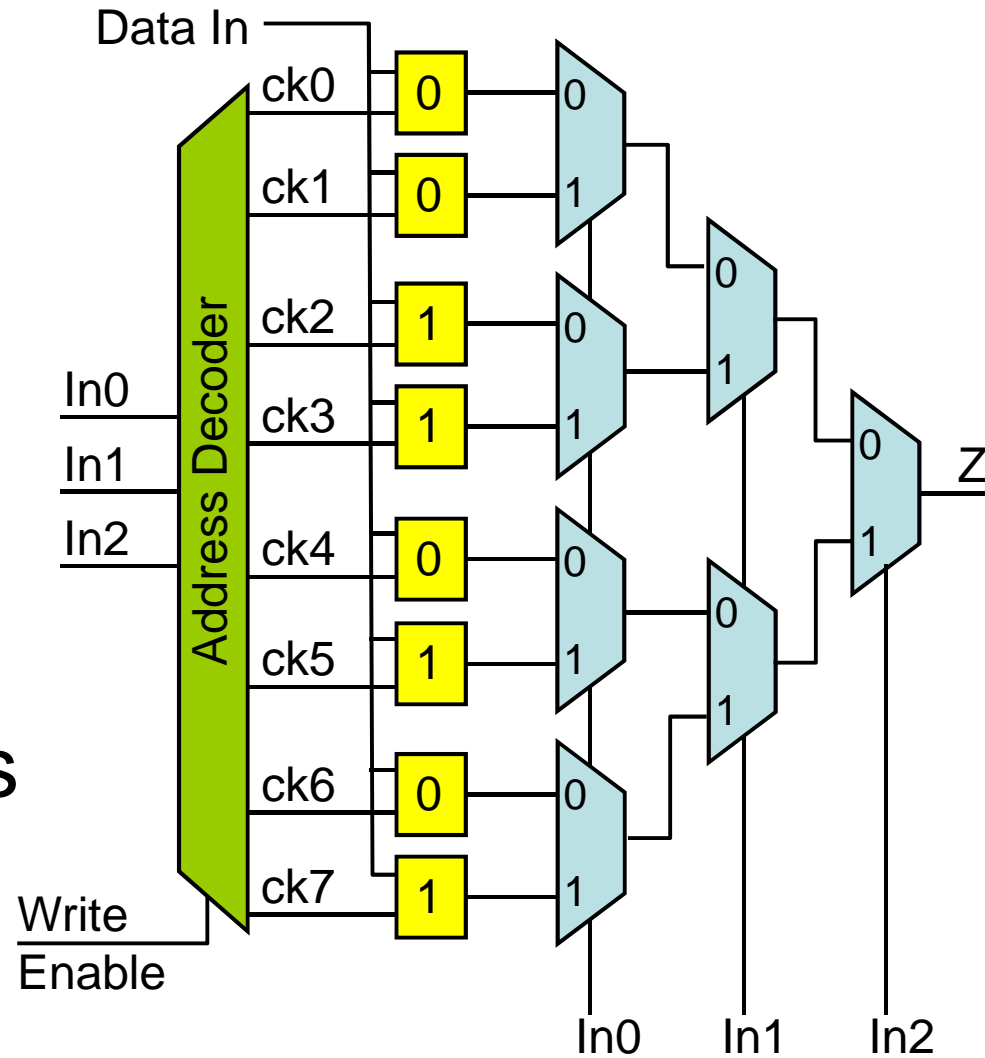


Truth table

S	A	B	Z
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Look-up Table Based RAMs

- Normal LUT mode performs read operations
- Address decoder with write enable generates clock signals to latches for write operations
- Small RAMs but can be combined for larger RAMs



A Simple PLB

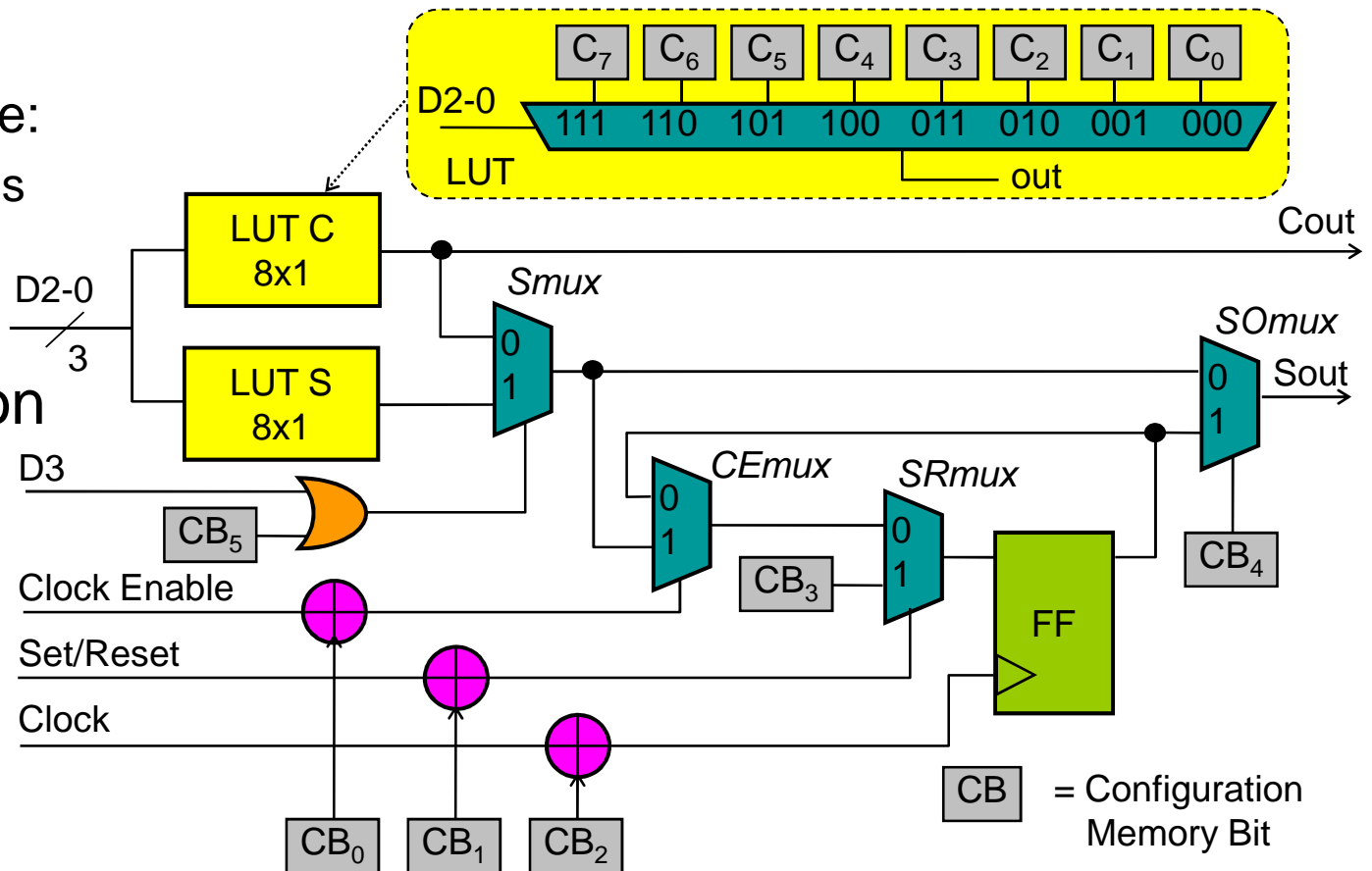
- Two 3-input LUTs
 - Can implement any 4-input combinational logic function

- 1 flip-flop
 - Programmable:
 - Active levels
 - Clock edge
 - Set/reset

- 22 configuration memory bits

- 8 per LUT
 - C0-7
 - S0-7
- 6 controls
 - CB0-7

C. Stroud 10/11

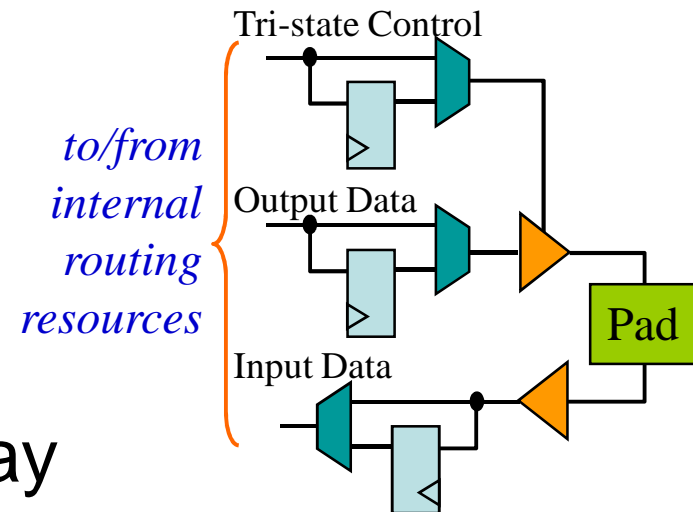


Input/Output Cells

- Bi-directional buffers
 - Programmable for input or output
 - Tri-state control for bi-directional operation
 - Flip-flops/latches for improved timing
 - Set-up and hold times
 - Clock-to-output delay
 - Pull-up/down resistors

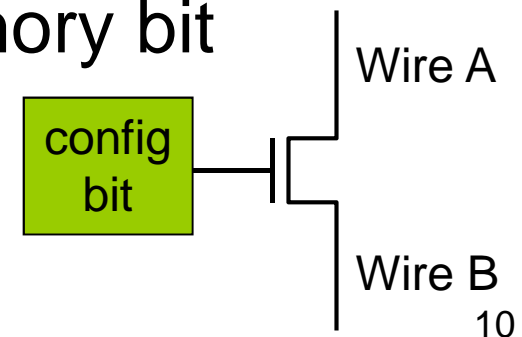
- Routing resources
 - Connections to core of array

- Programmable I/O voltage & current levels



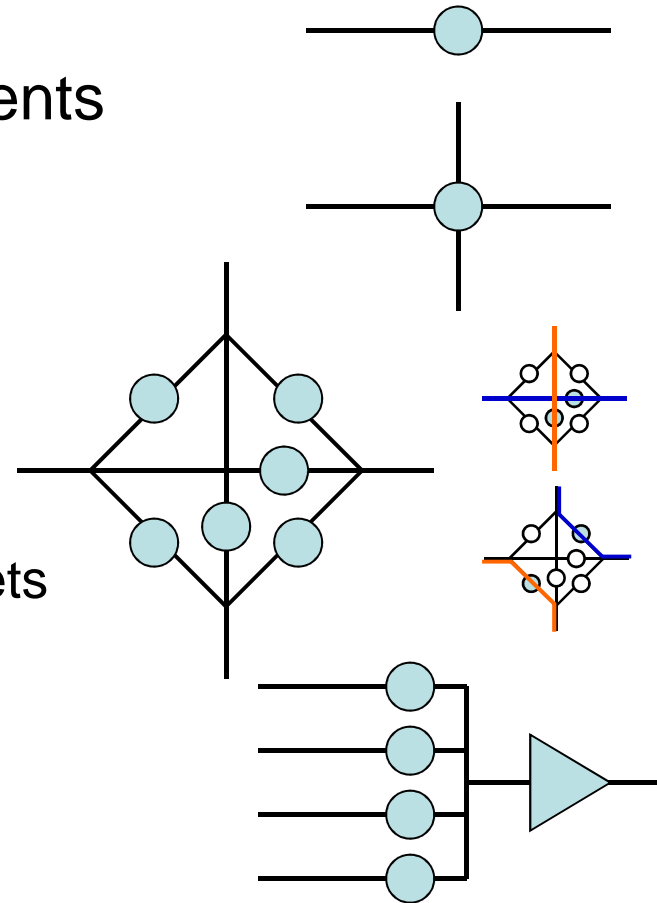
Interconnect Network

- Wire segments of varying length
 - $xN = N$ PLBs in length
 - 1, 2, 4, 6, and 8 are most common
 - $xH =$ half the array in length
 - $xL =$ length of full array
- Programmable Interconnect Points (PIPs)
 - Also known as Configurable Interconnect Points (CIPs)
 - Transmission gate connects to 2 wire segments
 - Controlled by configuration memory bit
 - 0 = wires disconnected
 - 1 = wires connected



PIPs

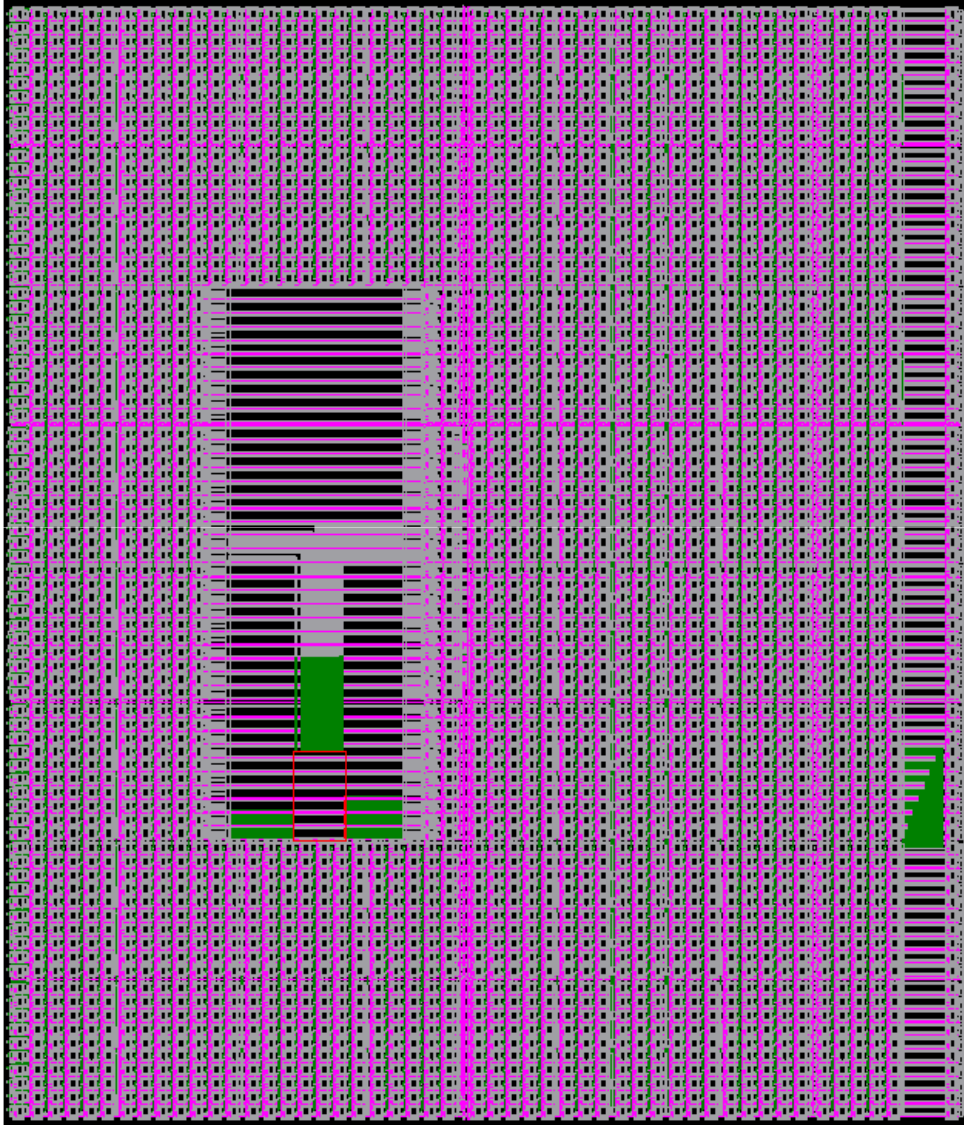
- Break-point PIP
 - Connect or isolate 2 wire segments
- Cross-point PIP
 - Turn corners
- Compound cross-point PIP
 - Collection of 6 break-point PIPs
 - Can route to two isolated signal nets
- Multiplexer PIP
 - Directional and buffered
 - Select 1-of- N inputs for output
 - Decoded MUX PIP – N config bits select from 2^N inputs
 - Non-decoded MUX PIP – 1 config bit per input



Recent Trends

- Incorporate specialized cores
 - RAMs – single-port, dual-port, FIFOs
 - 128 bits to 36K bits per RAM
 - 4 to 575 RAM cores per FPGA
 - DSPs – 18x18-bit multiplier, 48-bit accumulator, etc.
 - up to 512 per FPGA
 - Microprocessors and/or microcontrollers
 - up to 2 per FPGA
 - Hard core processor
 - Support soft core processors
 - Synthesized from HDL into programmable resources

Virtex-5 FX30T



- 5,120 slices
 - 4 FFs & 6-input LUTs
- 68 DPRAMs/FIFOs
 - 36Kbits
- 64 DSPs
 - 24x18 mult & 48-bit ALU
- 1 PowerPC 440
- 1 PCI Express
- 4 Ethernet MACs
- 8 Gigabit Xceivers

Ranges of Resources

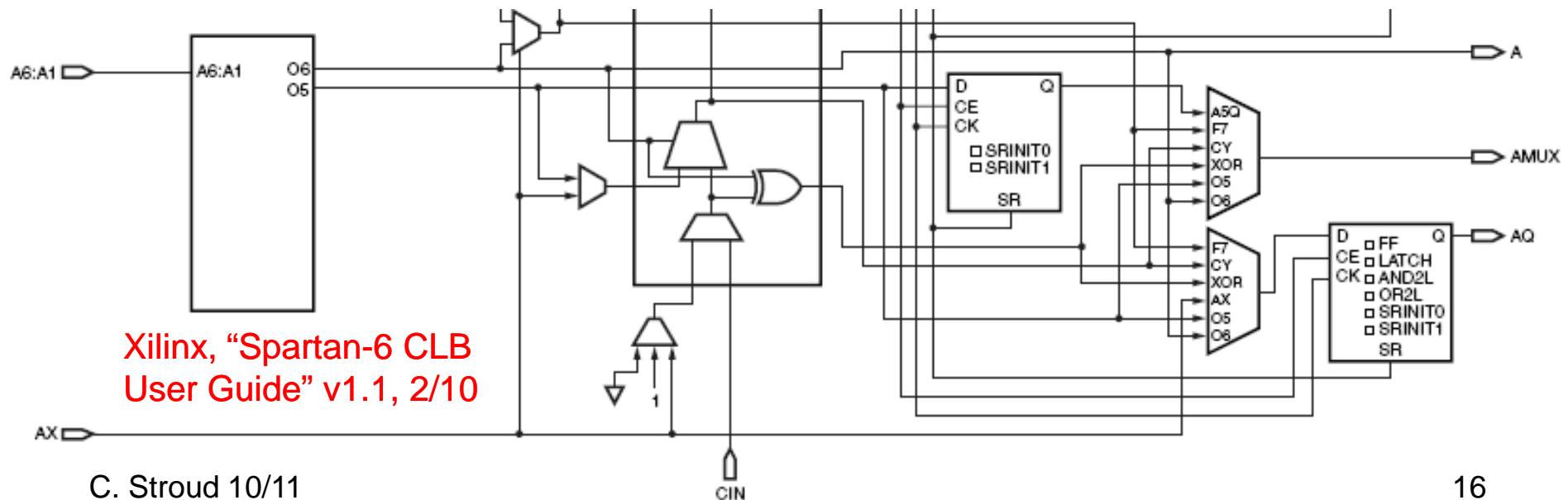
FPGA Resource		Small FPGA	Large FPGA
Logic	PLBs per FPGA	256	25,920
	LUTs and flip-flops per PLB	1	8
Routing	Wire segments per PLB	45	406
	PIPs per PLB	139	3,462
Specialized Cores	Bits per memory core	128	36,864
	Memory cores per FPGA	16	576
	DSP cores	0	512
Other	Input/output cells	62	1,200
	Configuration memory bits	42,104	79,704,832

Programmable RAMs

- 18 Kbit dual-port RAM
- Each port independently configurable as
 - 512 words x 36 bits (32 data bits + 4 parity bits)
 - 1K words x 18 bits (16 data bits + 2 parity bits)
 - 2K words x 9 bits (8 data bits + 1 parity bit)
 - 4K words x 4 bits (no parity)
 - 8K words x 2 bits (no parity)
 - 16K words x 1 bit (no parity)
- Each port has independently programmable
 - clock edge
 - active levels for write enable, RAM enable, reset
- Other modes of operation
 - First-in first-out, error correcting code, FIFO w/ECC

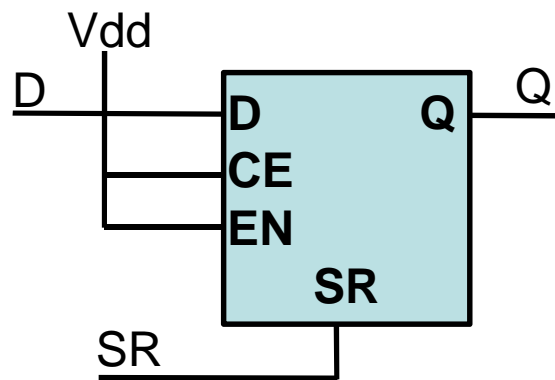
Spartan-6 Slices

- Configurable options
 - LUT contents (truth table)
 - Multiplexer selection (MUXs w/o select leads)
 - FF/latch set/reset values
 - FF/latch initialization values
 - FF/latch operation (AND/OR latch logic)



Latch Logic

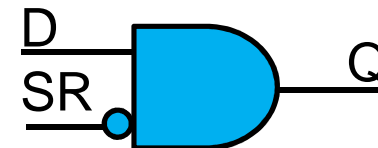
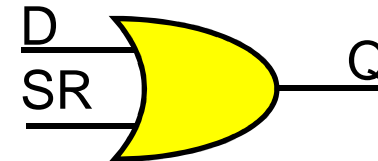
- Latch with CE & CLK (EN) tied active and asynchronous set/reset facilitates logic for:
 - OR obtained when Set/Reset value = 1
 - AND obtained when Set/Reset value = 0
 - AND with SR input inverted
- Other functionality described in application notes



Xilinx, "Spartan-6 CLB User Guide" v1.1, 2/10

D	SR	Q
0	0	0
1	0	1
X	1	1

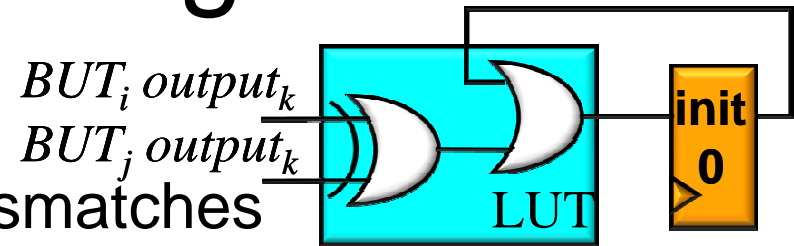
D	SR	Q
0	0	0
1	0	1
0	1	0
1	1	0



Carry-Chain Logic

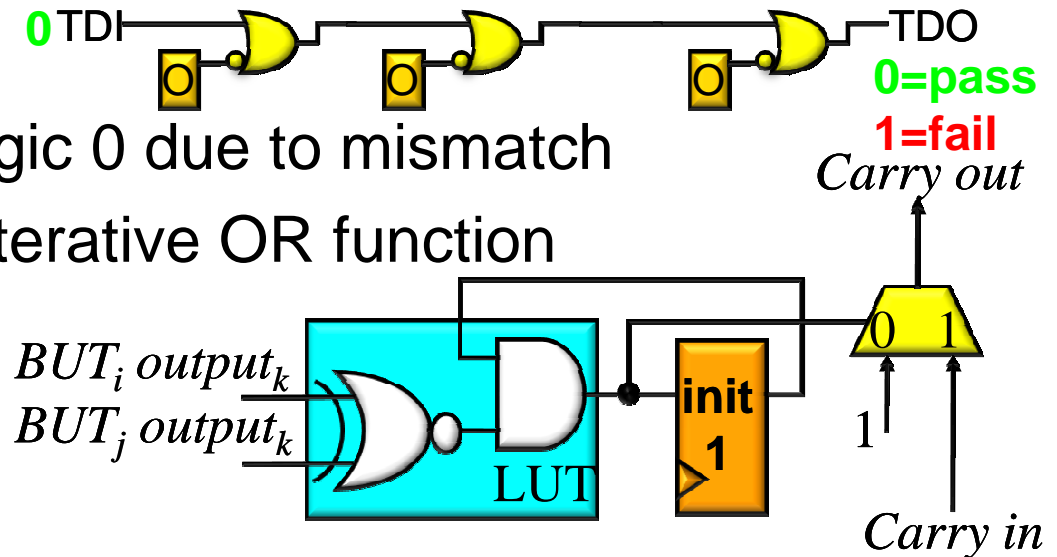
- Output response analyzer

- Logic 1 latched in FF due to mismatches
- Configuration memory readback used to get results



- CLB's have dedicated carry chain for fast adders and counters

- Newer ORA latches logic 0 due to mismatch
- Carry chain performs iterative OR function
- Single pass/fail bit
 - Expected failures require clock enable



- Only read configuration memory to get failing ORA locations for diagnosis

VHDL Initialization & FPGAs

- Initialization via immediate (variable) assignment operator not viable for ASIC
`signal ABC: std_logic_vector (3 downto 0) := "0000";`
- Default initialization in enumeration data types also a problem in ASIC
`type STATES is (IDLE, DECISION, WRITE, READ);`
- Generally not a problem in FPGAs due to FF initialization values
 - Contents of FFs downloaded with configuration data and works like a power-up initialization in an ASIC
 - Power-up initialization in ASIC is difficult to implement
- RAM contents can similarly be initialized
 - Not all FPGAs and/or FPGA functions are initializable

FPGA Synthesis Flow

- “Synthesis” converts VHDL to gate-level netlist in terms of Xilinx basic primitives
- “Implementation” converts the netlist to placed and routed FPGA
 - Translate = convert netlist for behavioral simulation
 - Map = map gate-level netlist to device specific elements
 - LUTs, FF, DSPs, RAMs, etc.
 - Place & route =
 - Place = determine physical location of elements in array
 - Route = determine interconnection of elements with specific routing resources (PIPs and wire segments)
- Configuration bit file generation
- Download configuration and start operation

Specialized Cores/Functions

- FPGA synthesis tools may not recognize some functions in VHDL
 - VHDL maybe technology independent but it is not optimized for a specific technology
- Options
 - Write VHDL model to incorporate functionality of specialized core, or
 - Use technology specific library that calls core
 - Either way you must be knowledgeable in implementation technology and features

VHDL RAM Model

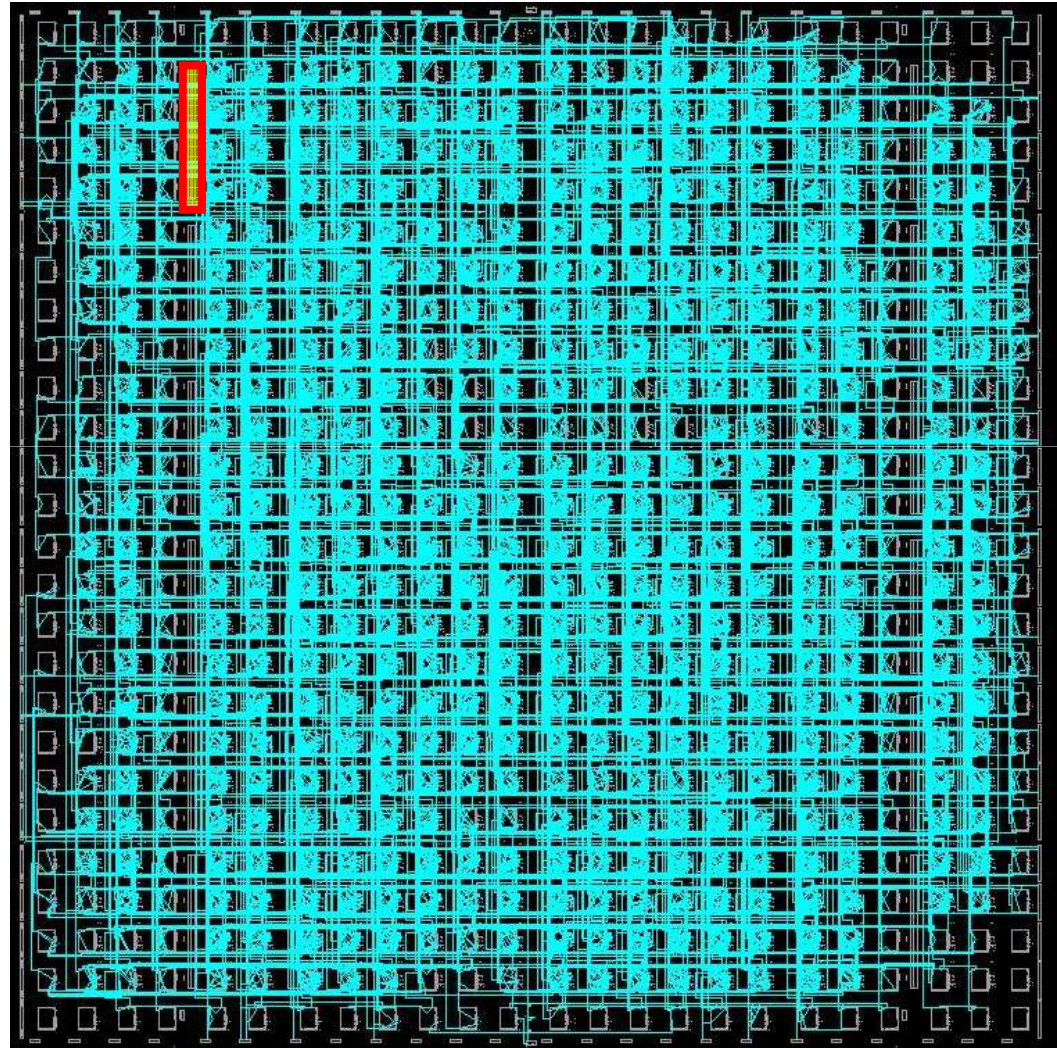
```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
entity RAM is
    generic (K: integer:=8;           -- number of bits per word
            W: integer:=8);         -- number of address bits
    port (WR: in std_logic;          -- active high write enable
          ADDR: in std_logic_vector (W-1 downto 0);  -- RAM address
          DIN: in std_logic_vector (K-1 downto 0);   -- write data
          DOUT: out std_logic_vector (K-1 downto 0)); -- read data
end entity RAM;
architecture RAMBEHAVIOR of RAM is
    subtype WORD is std_logic_vector ( K-1 downto 0);  -- define size of WORD
    type MEMORY is array (0 to 2**W-1) of WORD;       -- define size of MEMORY
    signal RAMKXN: MEMORY;  -- define RAMKXN as signal of type MEMORY
begin
    process (WR, DIN, ADDR)
        variable RAM_ADDR_IN: integer range 0 to 2**W-1; -- to translate address to integer
        begin
            RAM_ADDR_IN := conv_integer (ADDR); -- converts address to integer
            if (WR='1') then -- write operation to RAM
                RAMKXN (RAM_ADDR_IN) <= DIN ;
            end if;
            DOUT <= RAMKXN (RAM_ADDR_IN); -- always does read operation
        end process;
end architecture RAMBEHAVIOR;
```

C. Stroud 10/11

Synthesized RAM

In Spartan-3 200
RAM on previous
page required
434 CLBs

But it can easily
fit into 1 18K-bit
RAM (only
need 2K of 18K
RAM)



LUT DPRAM Instantiation

```
library UNISIM;
use UNISIM.vcomponents.all;          -- no component declaration in architecture
-- RAM16X1D: 16 x 1 positive edge write, asynchronous read dual-port distributed
   RAM
RAM16X1D_inst : RAM16X1D  generic map (INIT => X"0000")
port map (
    DPO => DPO,                      -- Port A 1-bit data output
    SPO => SPO,                      -- Port B 1-bit data output
    A0 => A0,                        -- Port A address[0] input bit
    A1 => A1,                        -- Port A address[1] input bit
    A2 => A2,                        -- Port A address[2] input bit
    A3 => A3,                        -- Port A address[3] input bit
    D => D,                          -- Port A 1-bit data input
    DPRA0 => DPRA0,                  -- Port B address[0] input bit
    DPRA1 => DPRA1,                  -- Port B address[1] input bit
    DPRA2 => DPRA2,                  -- Port B address[2] input bit
    DPRA3 => DPRA3,                  -- Port B address[3] input bit
    WCLK => WCLK,                   -- Port A write clock input
    WE => WE                          -- Port A write enable input
);
```

Xilinx, Inc., "Virtex-4 Libraries Guide for HDL Designs"