Simulation vs. Synthesis

Execution Models for VHDL Programs

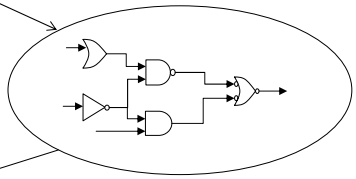
- Two classes of execution models govern the application of VHDL programs
- For Simulation
  - Discrete event simulation
  - Understanding is invaluable in debugging programs
- For Synthesis
  - Hardware inference
  - The resulting circuit is a function of the building blocks used for implementation
    - Primitives: NAND vs. NOR
    - Cost/performance

Simulation vs. Synthesis

```
entity my_ckt is
port (x,y :in bit;
z : out bit)
end entity my_ckt;

architecture behavioral of my_ckt is
-- declarations
begin
-- some code here
end architecture behavioral;
```

Synthesis



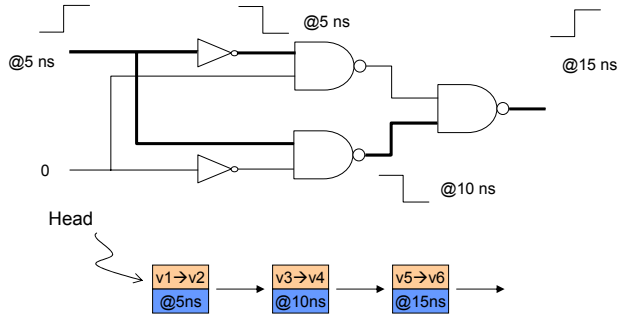
Simulation

```
entity my_ckt is
port (x,y :in bit;
z : out bit)
end entity my_ckt;

architecture behavioral of my_ckt is
-- declarations
begin
-- some code here
end architecture behavioral;
```

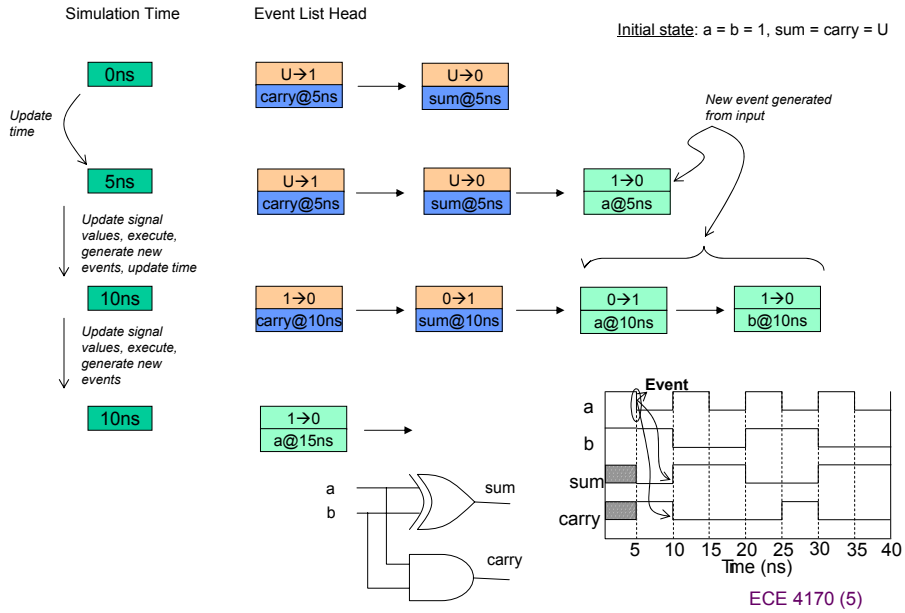
- Simulation and synthesis are complementary processes

Simulation of Digital Systems



- Digital systems are modeled as the generation of events – value transitions – on signals
- Discrete event simulations manage the generation and ordering of events
  - Correct sequencing of event processing
  - Correct sequencing of computations caused by events

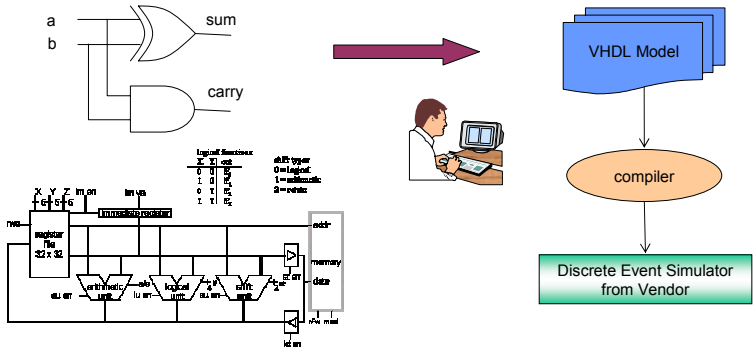
Discrete Event Simulation: Example



Discrete Event Simulation

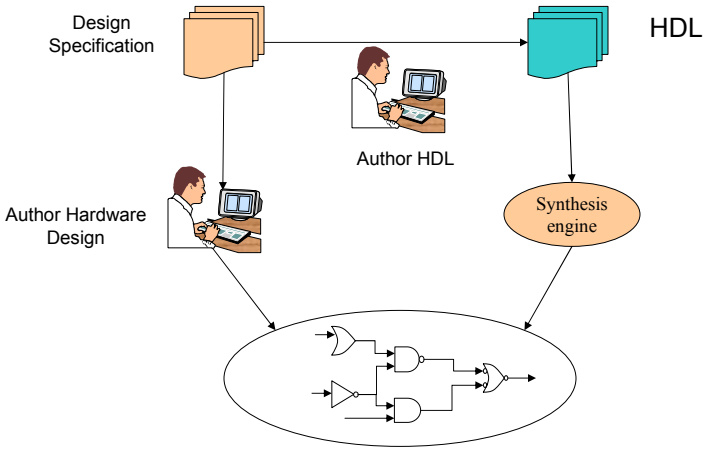
- Management of simulation time: ordering of events
- Two step model of the progression of time
  - Evaluate all affected components at the current time: events on input signals
  - Schedule future events and move to the next time step: the next time at which events take place

Simulation Modeling



- VHDL programs describe the generation of events in digital systems
- Discrete event simulator manages event ordering and progression of time
- Now we can quantitatively understand accuracy vs. time trade-offs
  - Greater detail → more events → greater accuracy
  - Less detail → smaller number of events → faster simulation speed

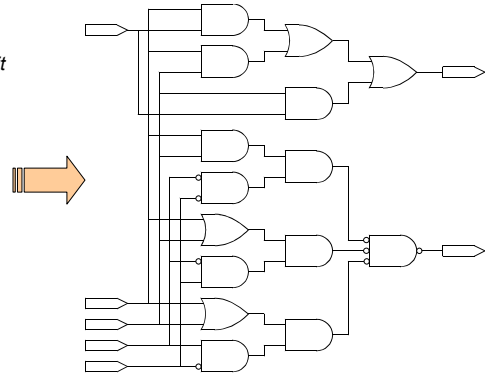
Synthesis and Hardware Inference



- Both processes can produce very different results!

Inferring Combinational Logic

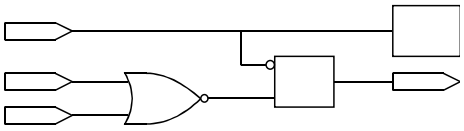
```
--
-- pseudo code for a single bit arithmetic/ logic unit
--
s1 <= in1 and in2;
s2 <= in1 or in2;
s3 <= in1 xor in2; -- perform the sum operation
c_out <=    (in1 and c_in) or (in1 and in2)
             or (in2 and c_in);
out <= s1 when sel = "00" else
       s2 when sel = "01" else
       s3 when sel = "10" else
       '0';
```



- Each statement implies a logic component

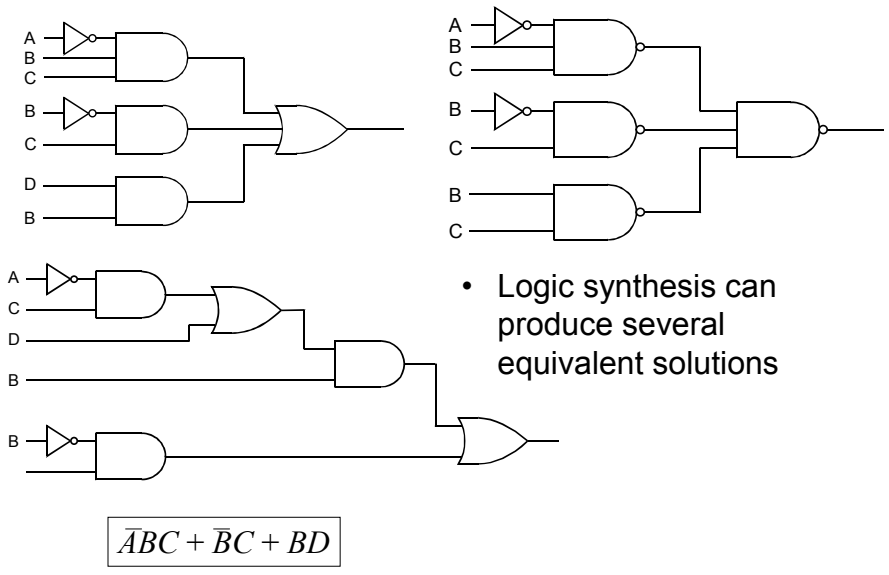
Inferring Sequential Logic

```
--
-- a simple conditional code block
--
        if (sel = '0') then
                z <= in1 nor in2;
        end if;
```



- Inference of sequential components is more subtle
  - All possible execution paths through the code must be accounted for
- Results are very sensitive to the manner in which code is written
- Inferences of latches vs. flip flops

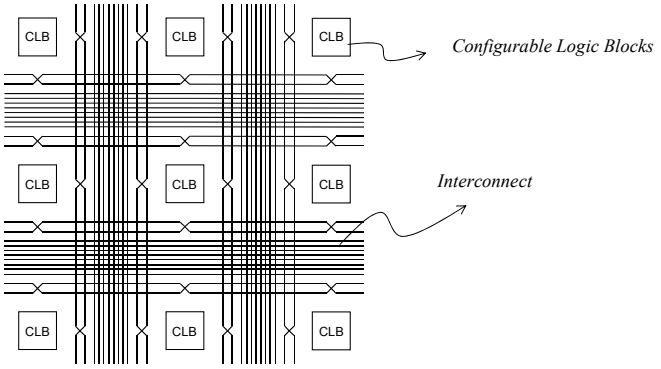
Optimization



- Logic synthesis can produce several equivalent solutions

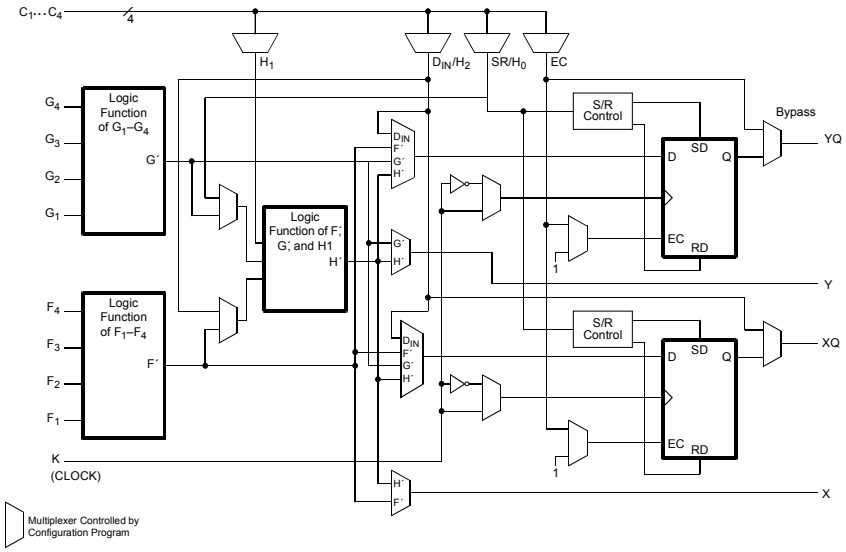
$$\bar{A}BC + \bar{B}C + BD$$

Field Programmable Gate Arrays: Principles

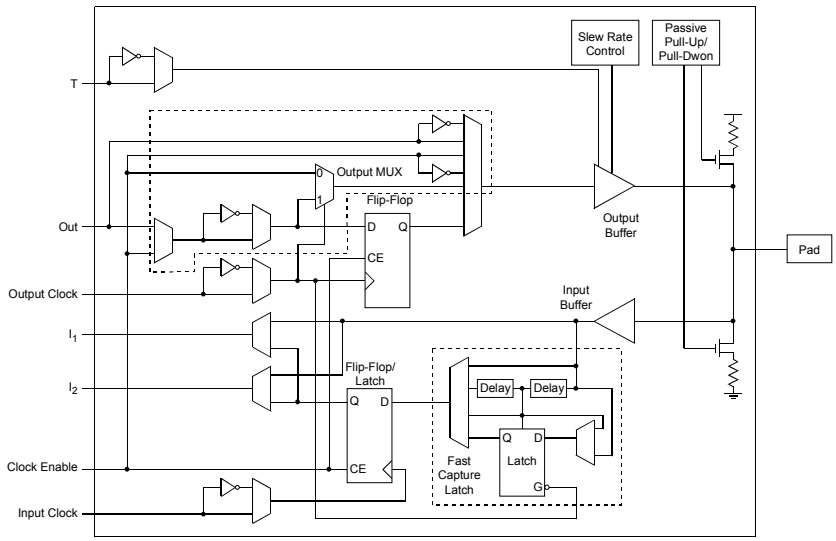


- The chip is tiled with Configurable Logic Blocks (CLBs)
  - each block can "implement" a few gates and flip flops
- An interconnect switching matrix is interleaved with this array of CLBs
- Usage: partition, place, and route a design

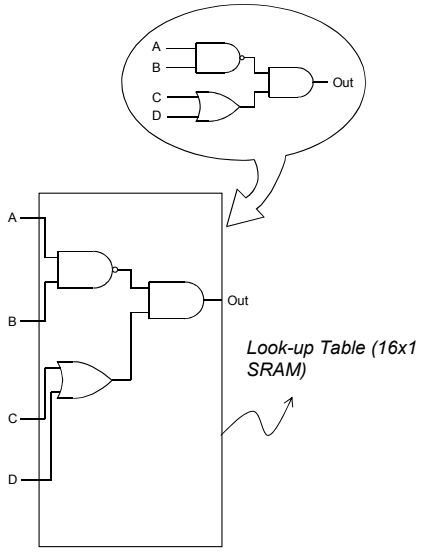
Inside a Xilinx Configurable Logic Block



Inside a Xilinx IO Block

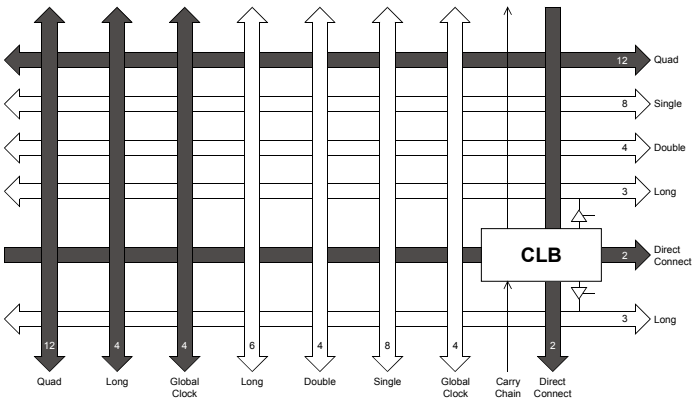


Implementing Digital Circuits in a CLB



- Boolean functions are implemented as look-up tables
- Combinations of lookup tables implement multivariable functions
- Implementation of behavior
- Sequential circuits use CLB latches/flip flops

Wiring Resources



- Optimize signal delay
- “Chip length” signals can be configured as buses
- Global nets for low skew clocks and reset signals

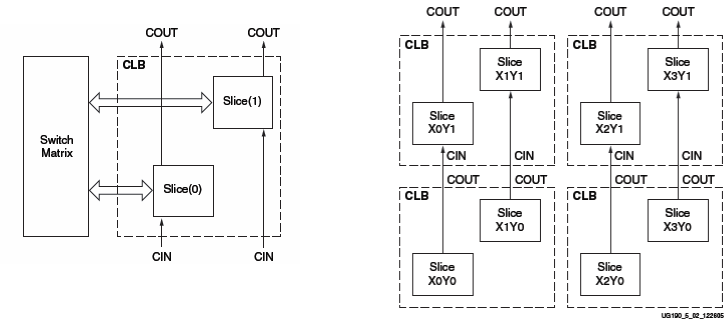
Specialized Resources

- Carry chains between columns of CLBs
- Configuration of CLB RAM as
  - Memories rather than lookup tables
  - Shift registers (Xilinx Virtex 5)
- Core generators
  - Optimized libraries of components
  - Vendor supplied
- Configuration bit stream for configuring a design
  - What about editing the bit stream directly!

Chip Configuration: Xilinx

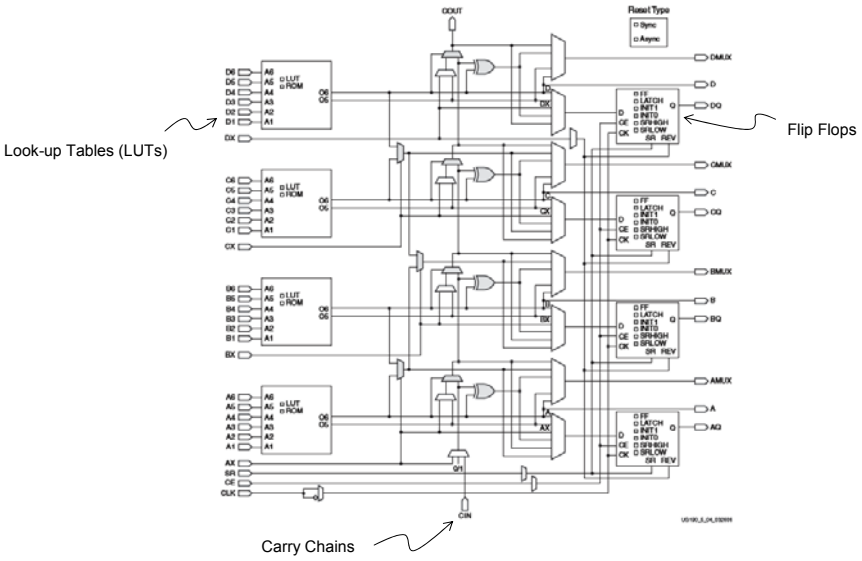
- Configuration bits for CLBs
  - bits for loading the LUTs
  - bits for configuring the flip flops
  - bits for setting multiplexors
- Configuration bits for the switch matrix
  - Connecting horizontal and vertical lines
  - Tri-state devices within the CLBs
- Configuration bits for the IOBs
  - IO clocks
  - storage vs. direct “access” to the pin
- Equivalent concepts for all vendors

Heterogeneous FPGAs: The Xilinx Virtex 5



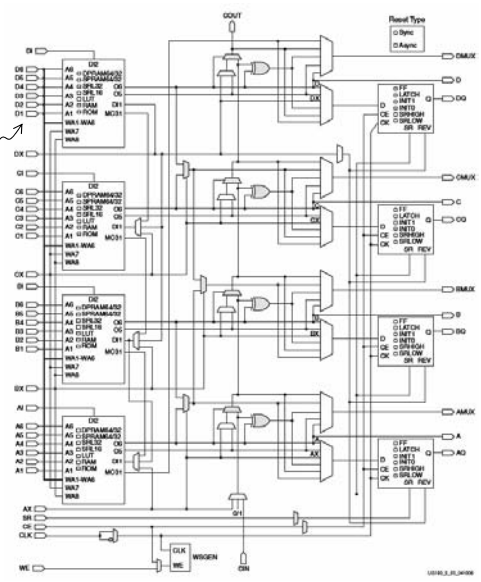
- Basic building block remains the same
- Substrate augmented with “hard cores”

The Basic Virtex 5 Slice

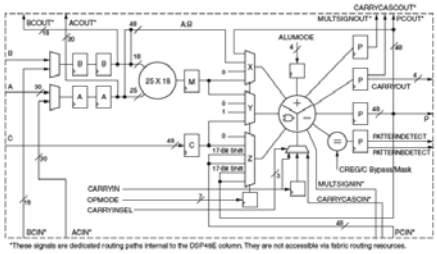


Customized Virtex 5 Slice

Can be configured as Block RAM or Shift register

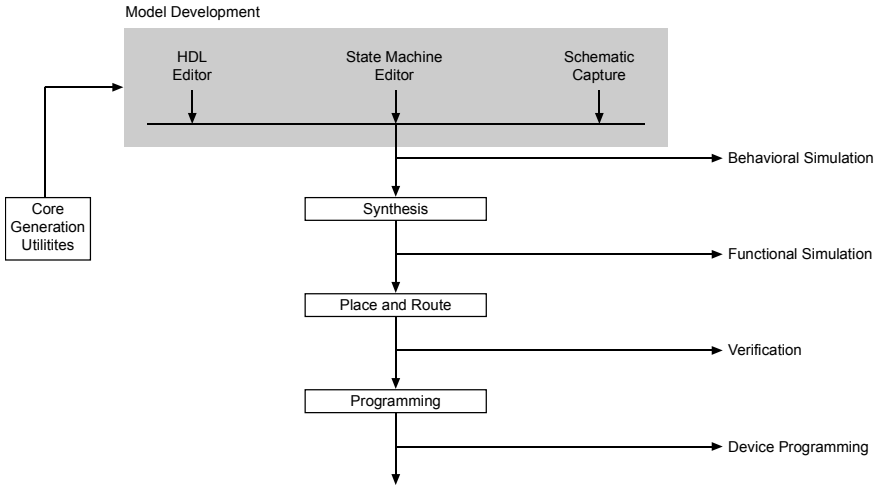


A DSP Slice



- Product families are customized with differing ratios of slice types to address different market segments

A Simplified Design Flow



Summary

- VHDL is used to describe digital systems and hence has language constructs for key attributes
  - Events, propagation delays, and concurrency
  - Timing, and waveforms
  - Signal values and use of multiple drivers for a signal
- VHDL has an underlying discrete event simulation model
  - Model the generation of events on signals
  - Built in mechanisms for managing events and the progression of time
  - Designer simply focuses on writing accurate descriptions