

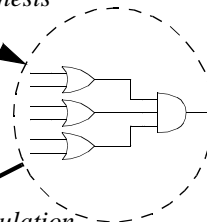
## Execution Models for VHDL Programs

- Two classes of execution models govern the application of VHDL programs
- Simulation
  - discrete event simulation
  - understanding is invaluable in debugging programs
- Synthesis
  - inference of hardware
  - a function of the building blocks used for implementation

## 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
begin
-- some code here
--
end architecture behavioral;
```

*synthesis*

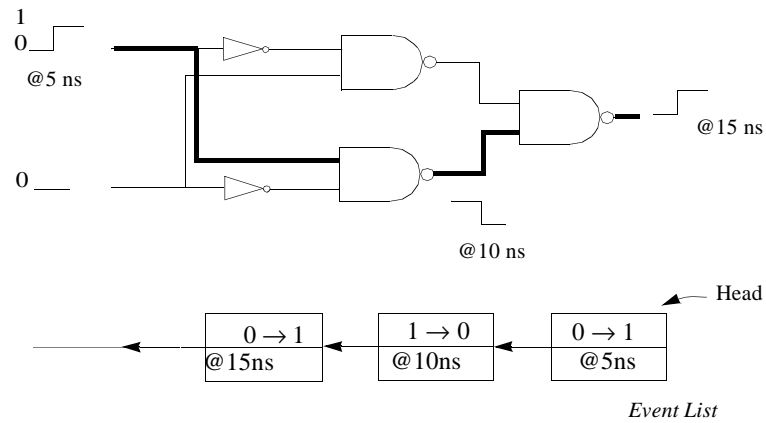


```
entity my_ckt is
port(x, y :in bit;
      z : out bit)
end entity my_ckt;
architecture behavioral of
my_ckt is
begin
-- some code here
--
end architecture behavioral;
```

*simulation*

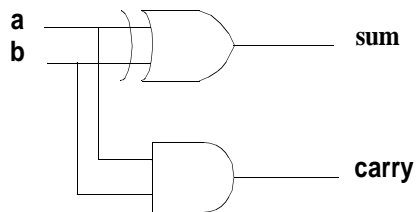
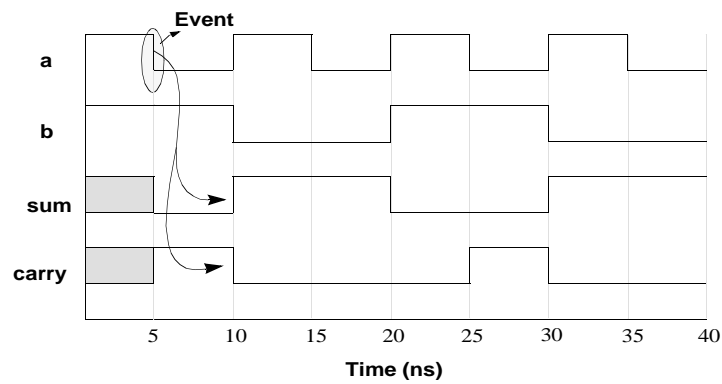
- Simulation and synthesis are complementary processes

## Simulation of Digital Systems

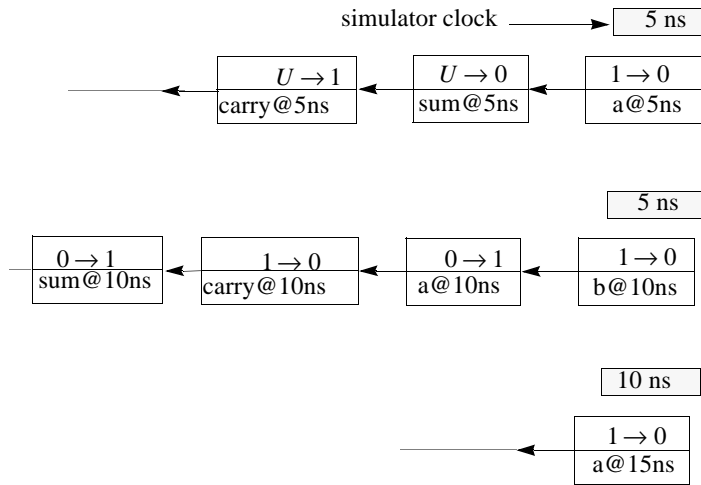


- Digital systems propagate events
- Discrete event simulations manage the generation and recording of events

## Discrete Event Simulation: An Example

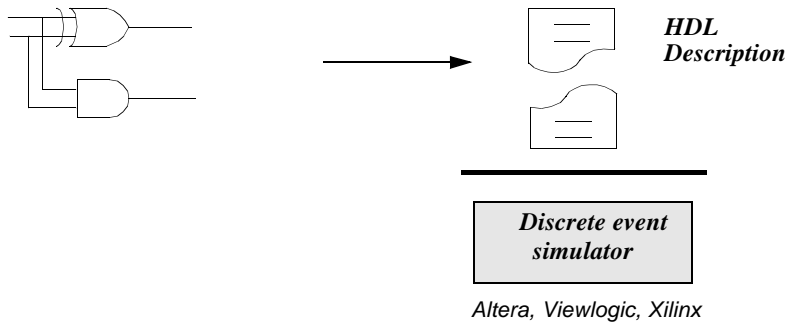


## Discrete Event Simulation: Data Structures



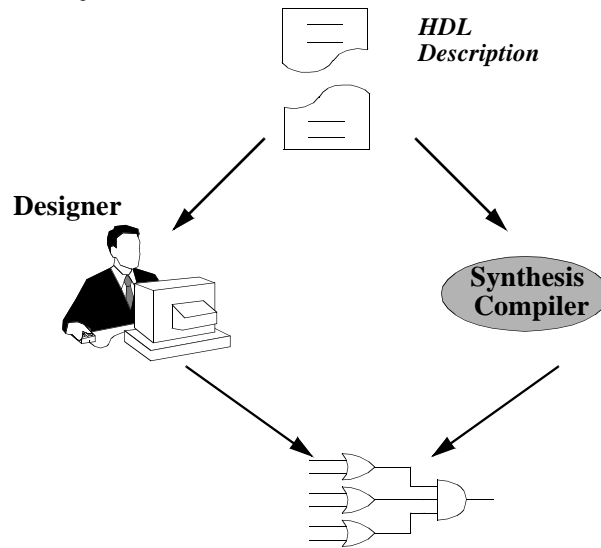
- Management of simulation time: ordering of events
- Two step model of the progression of time

## 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

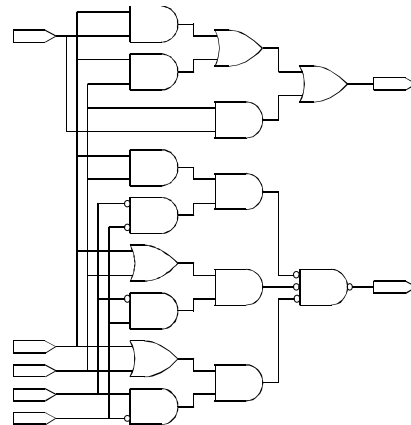
## Synthesis and Hardware Inference



- Both processes can produce very different results!

## Inferring Combinational Logic

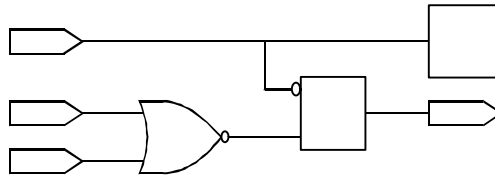
```
--  
-- pseudo code for a single bit arithmetic/  
logic unit  
--  
s1 <= in1 and in2;  
s2 <= in1 or in2;  
s3 <= in1 xor in2; -- perform the sum operation  
c_out <= (in1 and c_in) or (in1 and in2)  
         or (in2 and c_in);  
out <= s1 when sel = "00" else  
      s2 when sel = "01" else 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



- Each statement implies a logic component

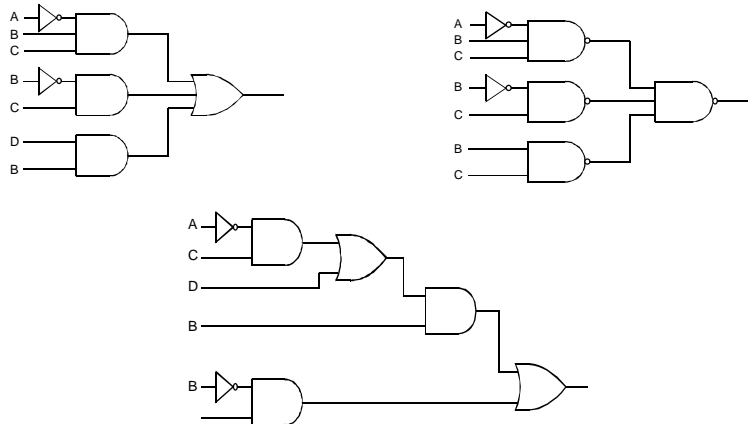
## Inferring Sequential Logic

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



- Inference of sequential components is more subtle
- Results are very sensitive to the manner in which code is written
- Inferences of latches vs. flip flops

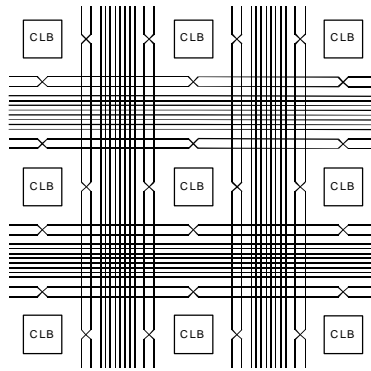
## Target Primitives



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

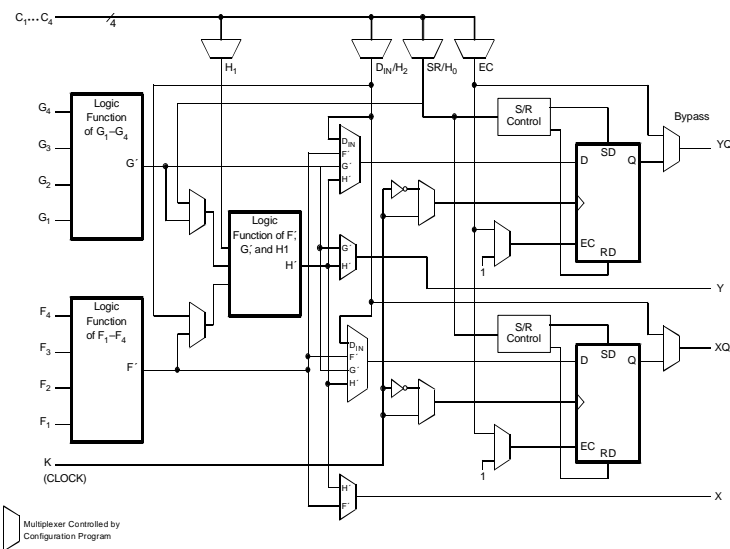
- The resulting circuit depends on the available building blocks

## Field Programmable Gate Arrays: Principles



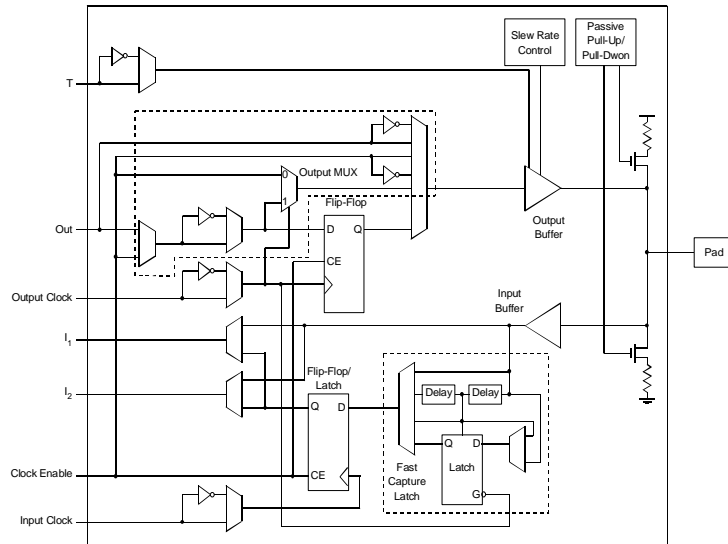
- The chip is tiled with Configurable Logic Blocks (CLBs)
  - each block can “implement” a few gates and flip flops
- A switching matrix is interleaved with this array of CLBs
- Usage: partition, place, and route a design

## Inside a Configurable Logic Block (CLB)



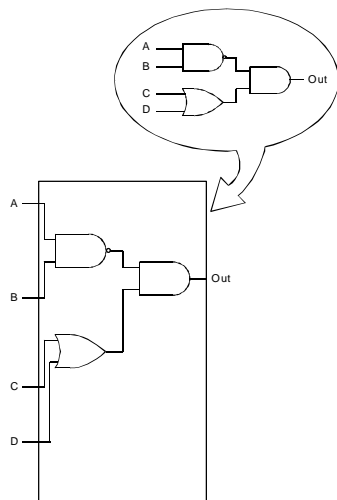
- Combinational and sequential components

## Inside a I/O Block (IOB)



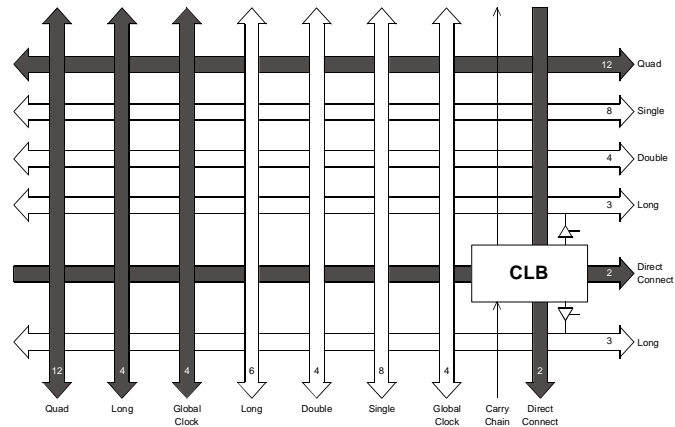
- Pin multiplexing and distinct clocks

## Implementing Combinational Logic in a CLB



- Boolean functions implemented as look-up tables
- Combinations of look-up tables for 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

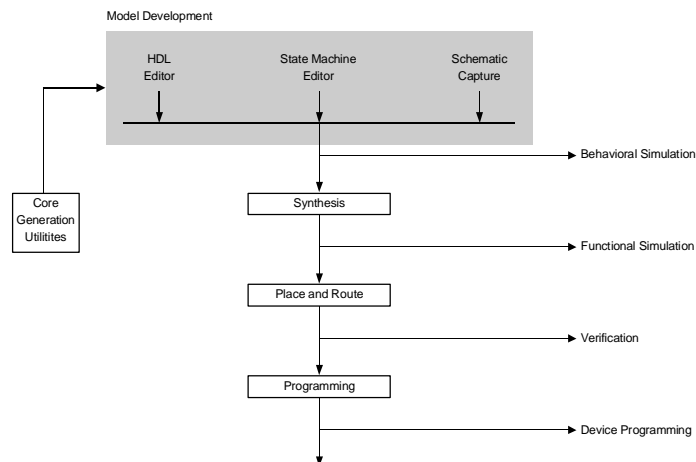
## Specialized Resources

- Carry chains between columns of CLBs
- Configuration of CLB RAM as memories rather than lookup tables
- Core generators
  - optimized libraries of components
  - vendor supplied
- 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
  - tristate devices within the CLBs
- Configuration bits for the IOBs
  - IO clocks
  - storage vs. direct “access” to the pin
- Equivalent concepts for all vendors

## Design Flow



- Design flow is a function of the target implementation, for example, FPGAs

## **Summary**

- Simulation Model
  - discrete event execution model for VHDL programs
  - accuracy vs. time
- Synthesis Model
  - need for inference from language constructs
  - basic principles behind the use of FPGAs
  - challenges in the design flow