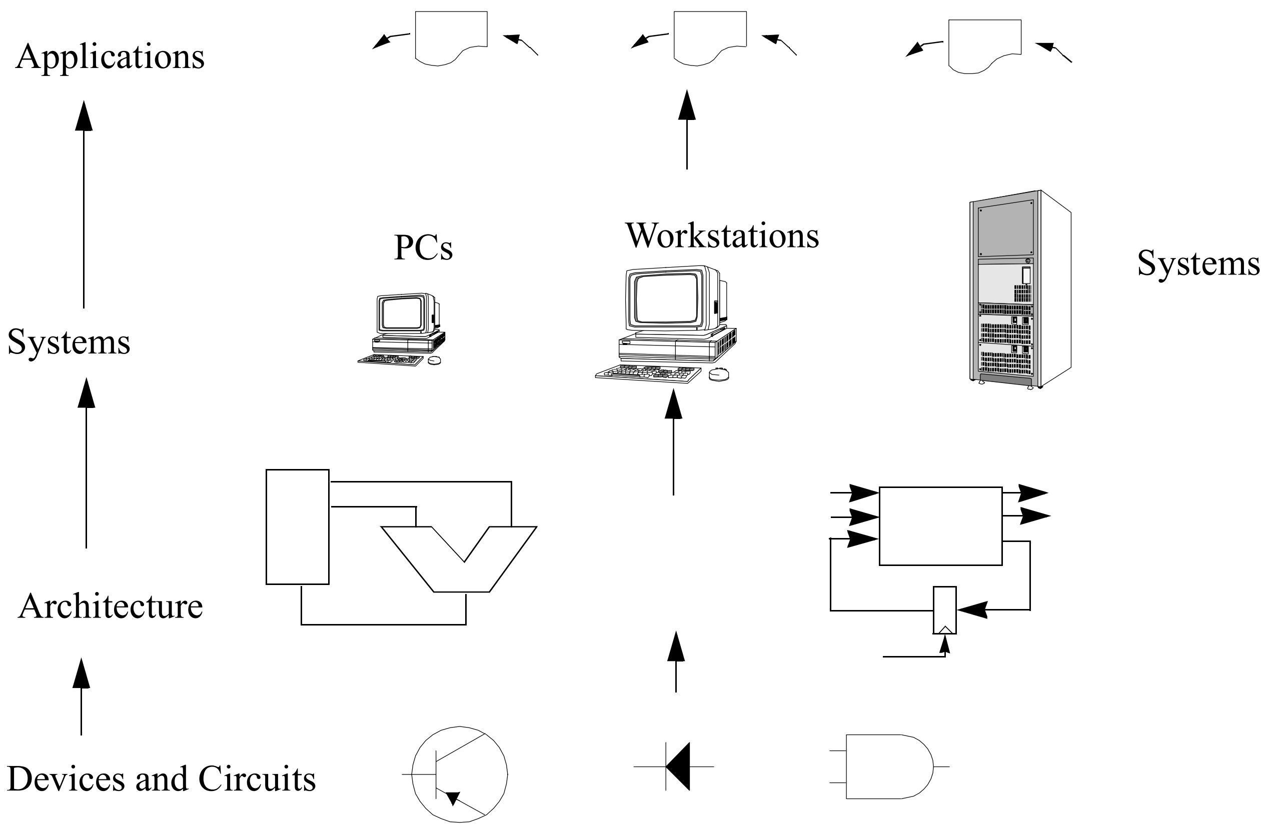
COMP3222/9222 Digital Circuits & Systems

8. VHDL for Simulation

Describing Digital Systems



Applications

Systems

Architecture

Devices and Circuits

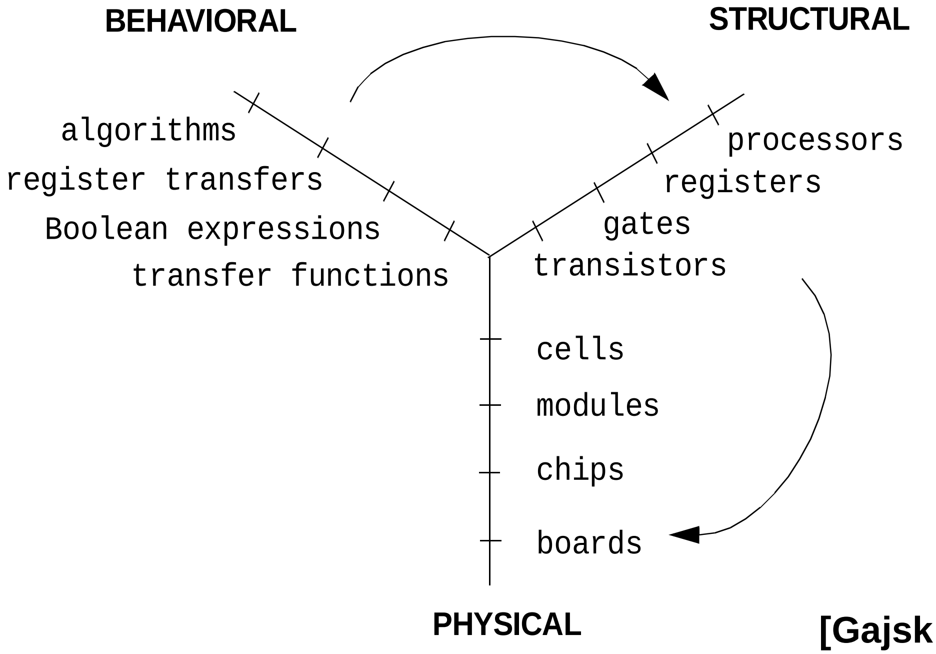
PCs

Workstations

Systems

- Systems are described at multiple levels of abstraction

Hardware Description Languages



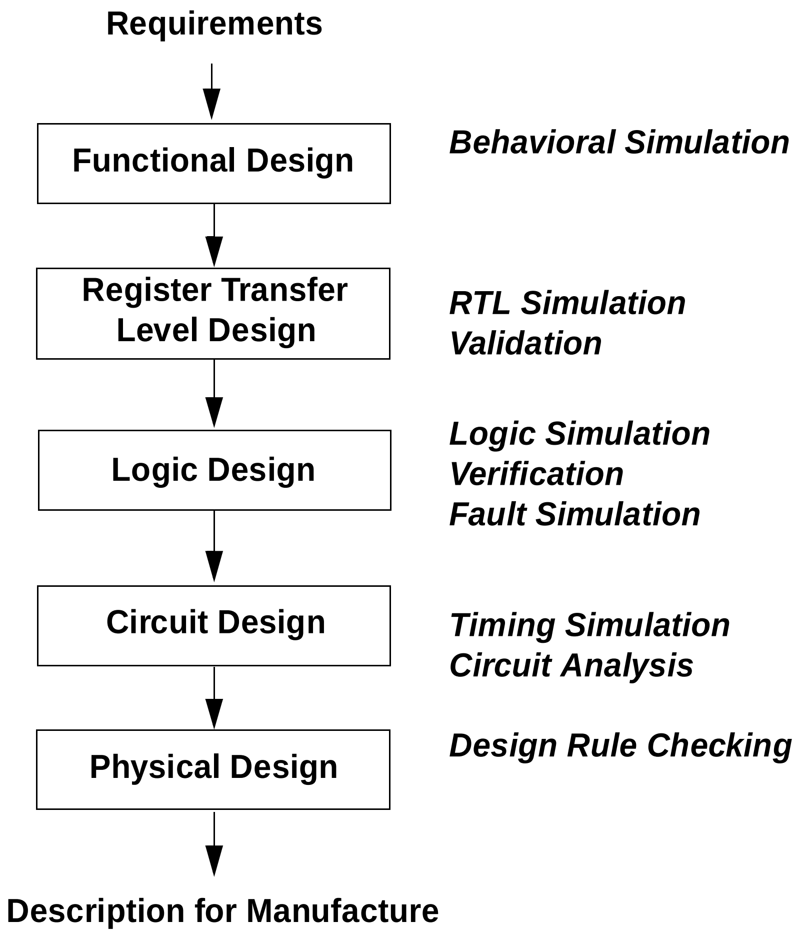
[Gajski and Kuhn]

- And design is structured around a hierarchy of representations
- HDLs can describe distinct aspects of a design at multiple levels of abstraction

Design description has multiple purposes

- Design Specification
    - unambiguous definition of components and interfaces in a large design
- Design Simulation
    - verify system/subsystem/chip performance prior to design implementation
- Design Synthesis
    - automated generation of hardware

Simulation in the digital system design process



Requirements

Functional Design — *Behavioral Simulation*

Register Transfer Level Design — *RTL Simulation*, *Validation*

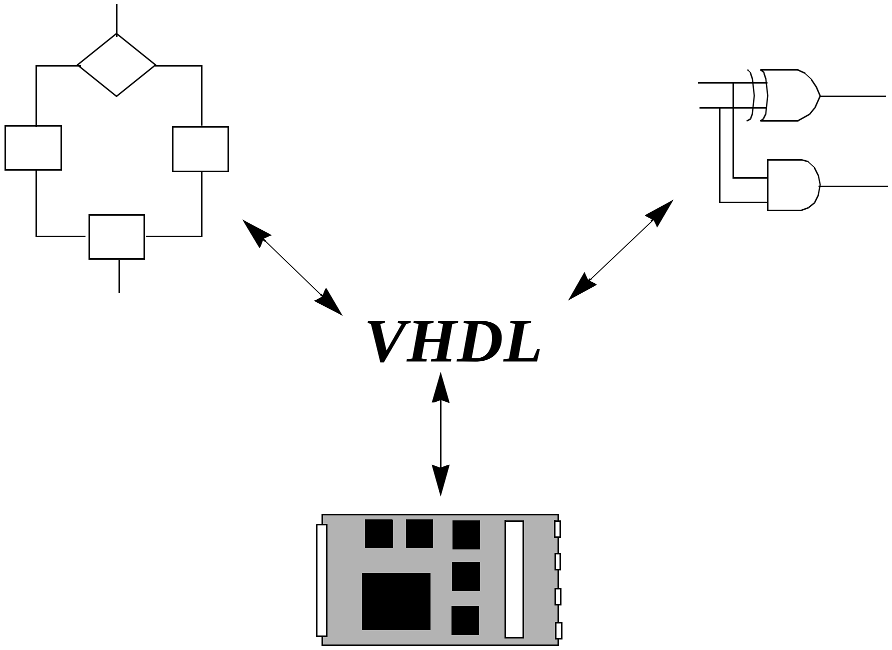
Logic Design — *Logic Simulation*, *Verification*, *Fault Simulation*

Circuit Design — *Timing Simulation*, *Circuit Analysis*

Physical Design — *Design Rule Checking*

Description for Manufacture

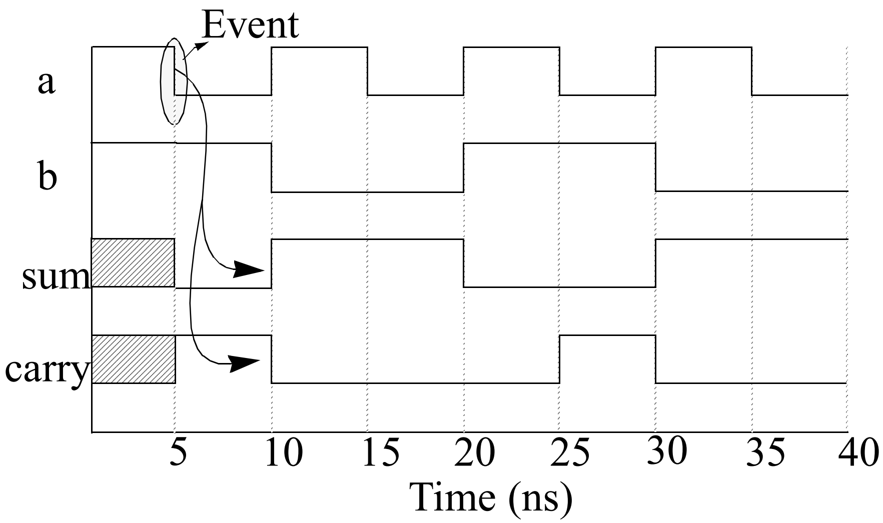
Role of VHDL

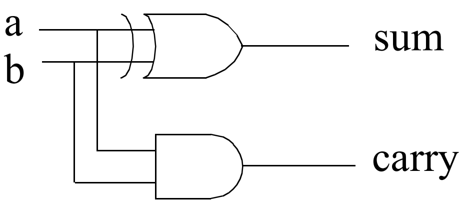


VHDL

- System description and documentation
- System simulation <= focus of this lecture
- System synthesis <= overall concern of this course!

Behavioral Attributes of Digital Systems

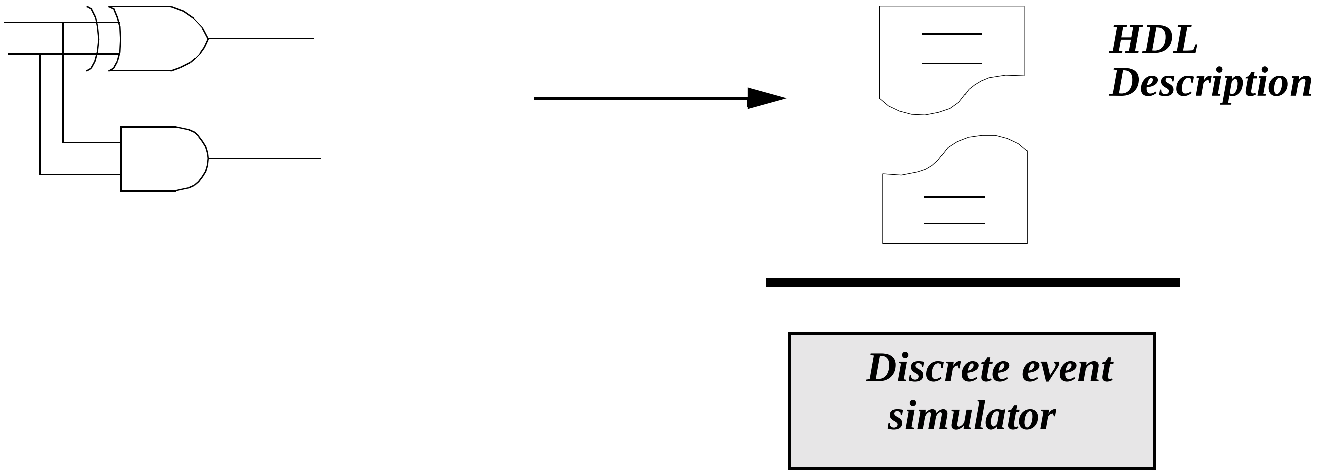




Assuming 5ns propagation delay per gate

- Digital systems are concerned with *signals* and their *values*
- *Model events*, *propagation delays*, *concurrency*
- The time-ordered sequence of events produces a *waveform*

Simulation Modeling



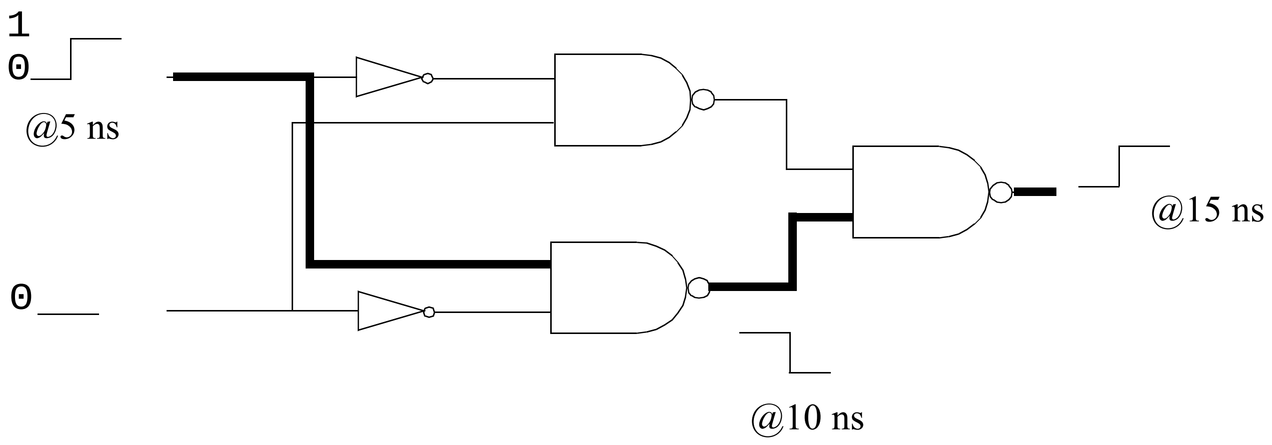
HDL Description

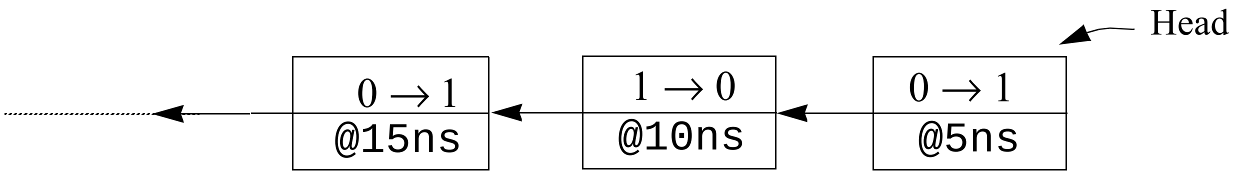
Discrete event simulator

Quartus, ModelSim, Xilinx

- VHDL designs describe the generation of events in digital systems
- A discrete event simulator manages event ordering and the progression of time

Simulation of Digital Systems

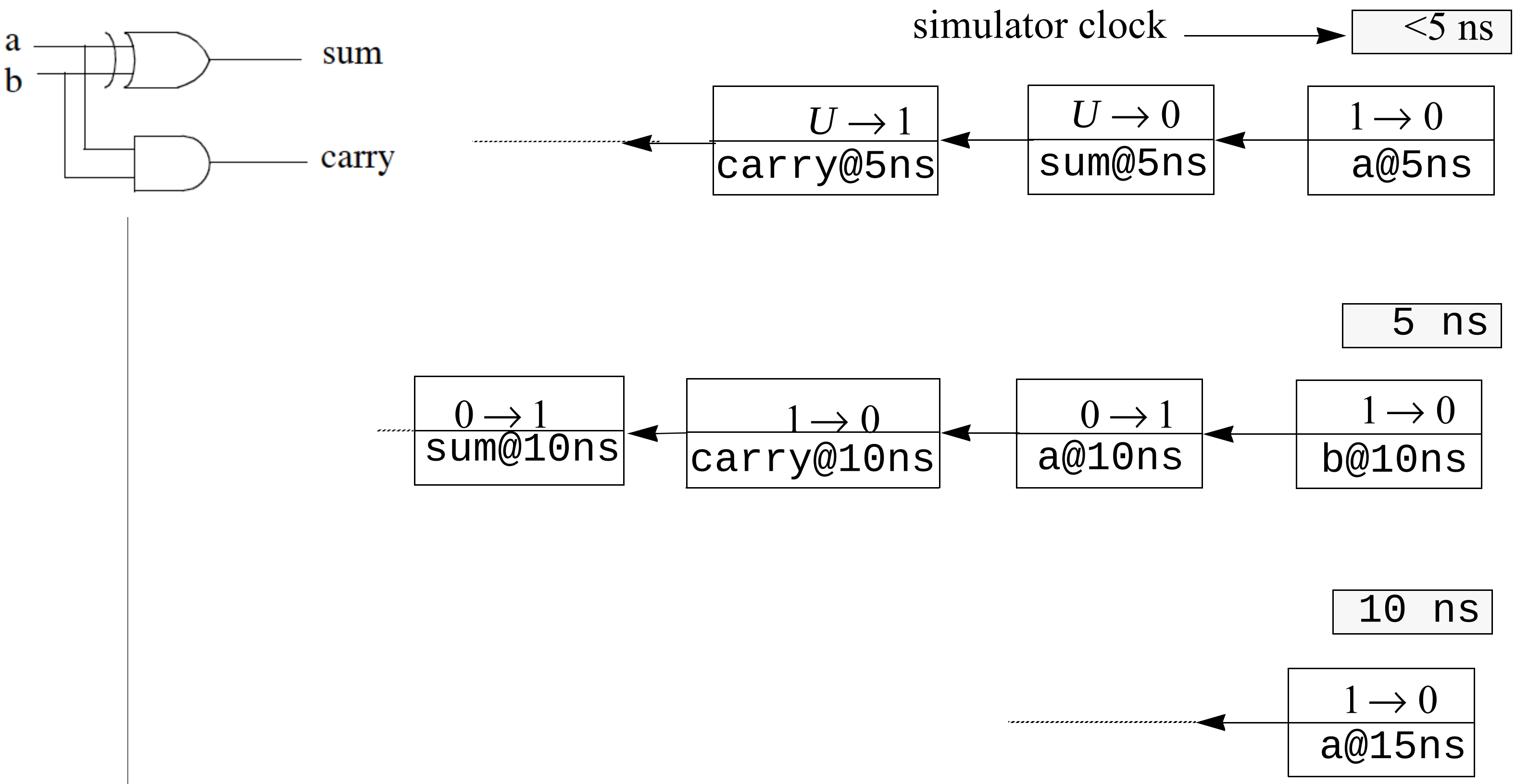




Event List

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

Discrete Event Simulation: Half Adder Model:



simulator clock → <5 ns

carry@5ns $U \rightarrow 1$ ← sum@5ns $U \rightarrow 0$ ← a@5ns $1 \rightarrow 0$

5 ns

sum@10ns $0 \rightarrow 1$ ← carry@10ns $1 \rightarrow 0$ ← a@10ns $0 \rightarrow 1$ ← b@10ns $1 \rightarrow 0$

10 ns

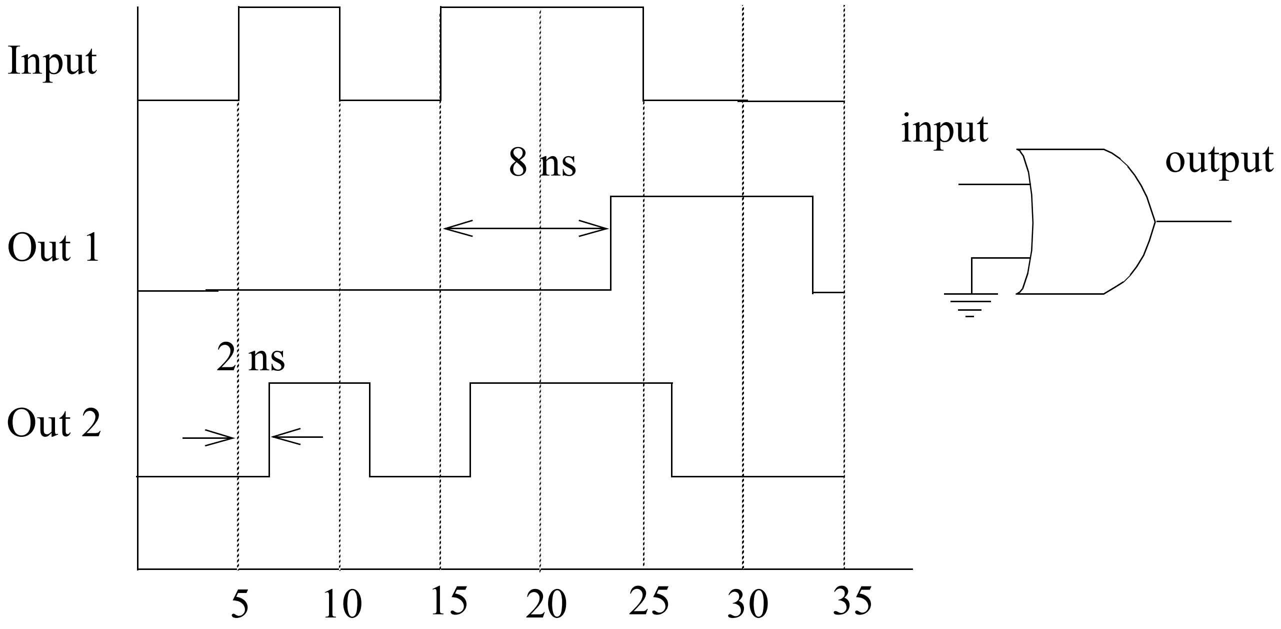
a@15ns $1 \rightarrow 0$

- Assume a & b are set to 1 initially and the gates have 5ns propagation delay
- Management of simulation time and ordering of events
- Two step model of the progression of time

Modeling Digital Systems

- VHDL programs describe behavioral attributes of digital systems
  - events, propagation delays, concurrency
  - waveforms and timing
  - signal values
- Discrete event simulators mimic the operation of the system being described using a two step model of time:
  1. assign values scheduled for signals at the current time
  2. process affected components and schedule future signal values
  - This ensure events are generated and processed in the correct order
  - And that *only and all* those events that would have been generated in the physical system are generated

Understanding delays



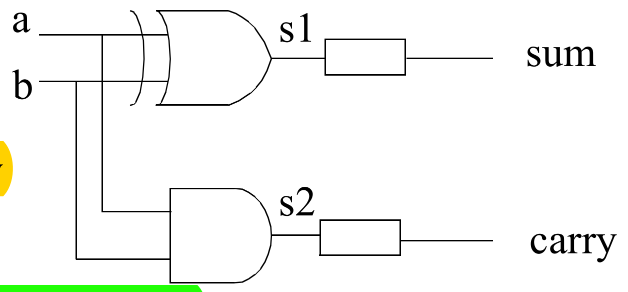
- Inertial delay model
- Transport delay model
- Delta delay model

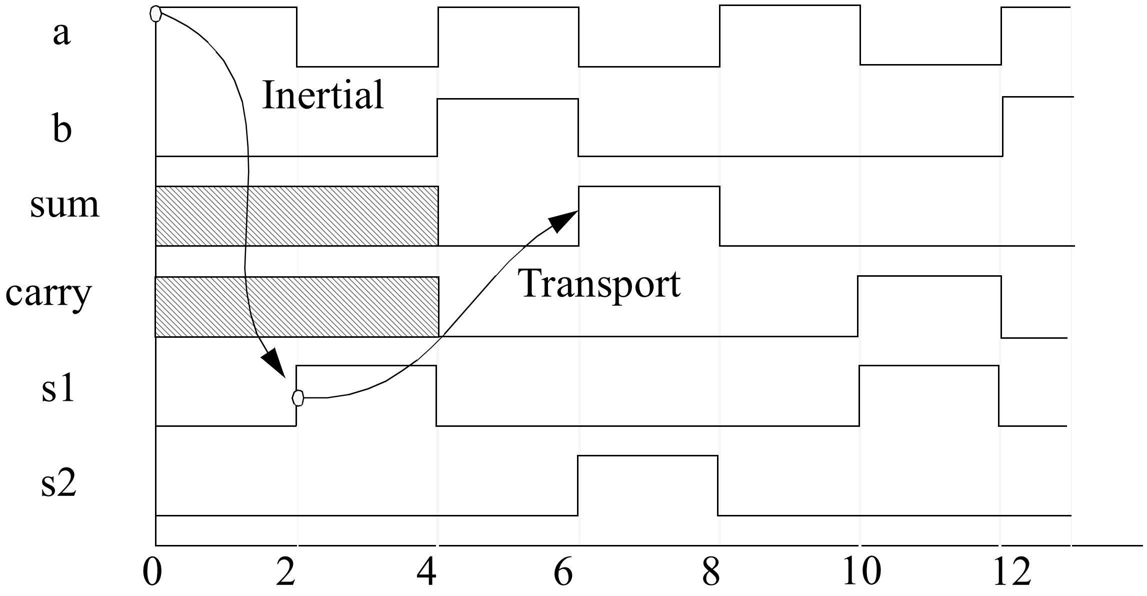
Understanding Delays

- Inertial delay
  - default delay model
  - suitable for modeling delays through devices such as gates
  - only propagates input pulses longer than the inertial delay
- Transport Delay
  - models delays through devices with very small inertia, e.g, wires
  - all input events are propagated to output signals
- Delta delay
  - used when propagation delays are not specified
  - infinitesimally small delays are automatically inserted by the simulator to preserve the correct ordering of events

Inertial & Transport Delays: Example

```
architecture transport_delay of
half_adder is
signal s1, s2: std_logic:= ‘0’;
begin
s1 <= (a xor b) after 2 ns; -- inertial delay
s2 <= (a and b) after 2 ns;
sum <= transport s1 after 4 ns;
carry <= transport s2 after 4 ns; -- transport delay
end transport_delay;
```

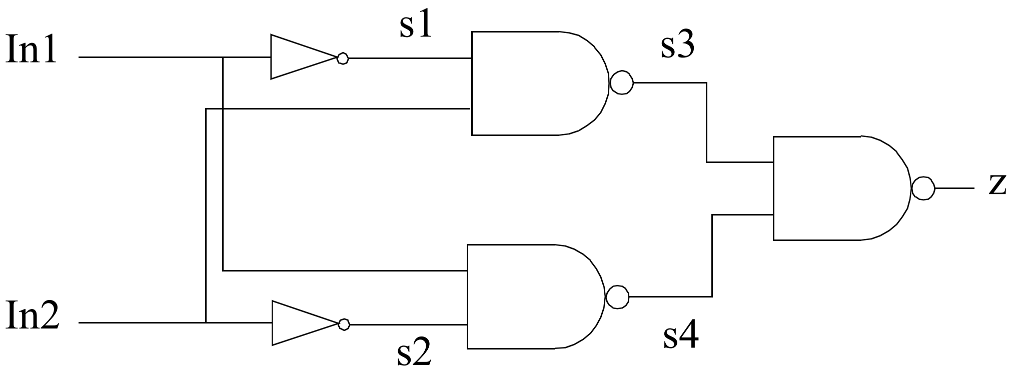


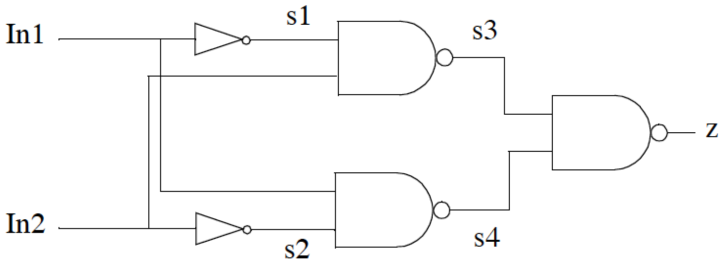


Delta Delays: Example

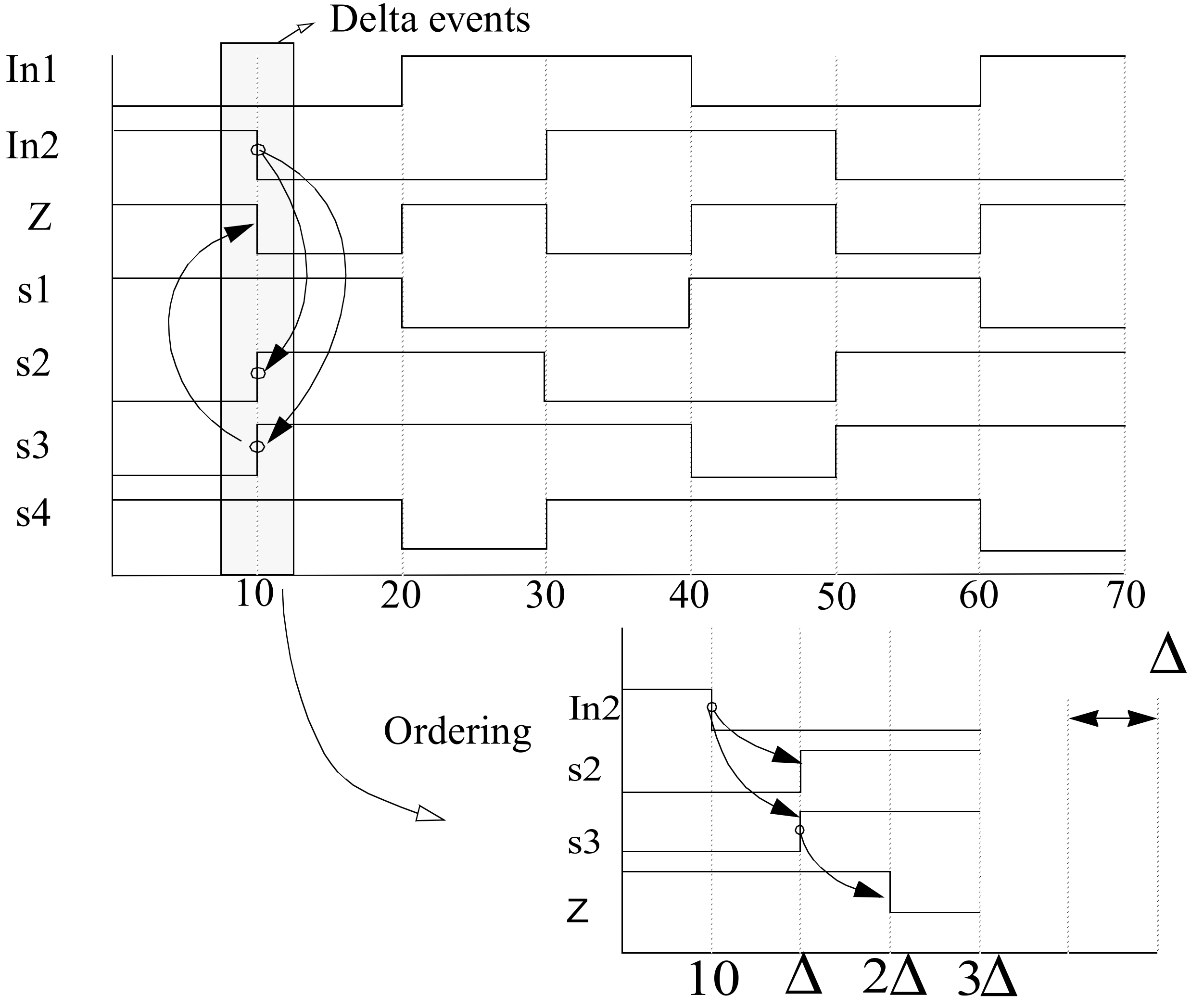
```
library IEEE;
use IEEE.std_logic_1164.all;
entity combinational is
port (In1, In2: in std_logic;
     z  : out std_logic);
end combinational;
```

```
architecture behavior of
combinational is
signal s1, s2, s3, s4: std_logic:= '0';
begin
s1 <= not  In1;
s2 <= not In2;
s3 <= not (s1 and In2);
s4 <= not (s2 and In1);
z <= not (s3 and s4);
end behavior;
```

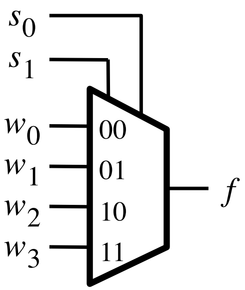




Delta Delays: Example



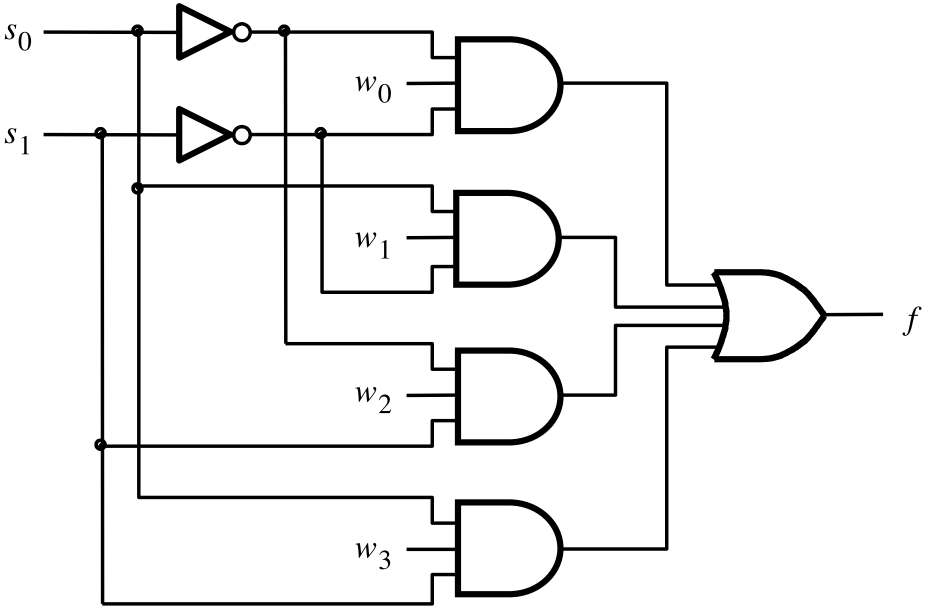
Simulating a 4-to-1 multiplexer



(a) Graphic symbol

s_1	s_0	f
0	0	w_0
0	1	w_1
1	0	w_2
1	1	w_3

(b) Truth table



(c) Circuit

Q1: Assuming s1 = s0 = 0; w0 = w2 = 0; and w1 = w3 = 1, sketch the delta transitions of all affected signals if s0 transitions from 0→1 @ 10ns; depict the state of the event queue prior to each transition.

Q2: Repeat Q1 assuming gates have a 2ns inertial delay and wires have a 1ns transport delay