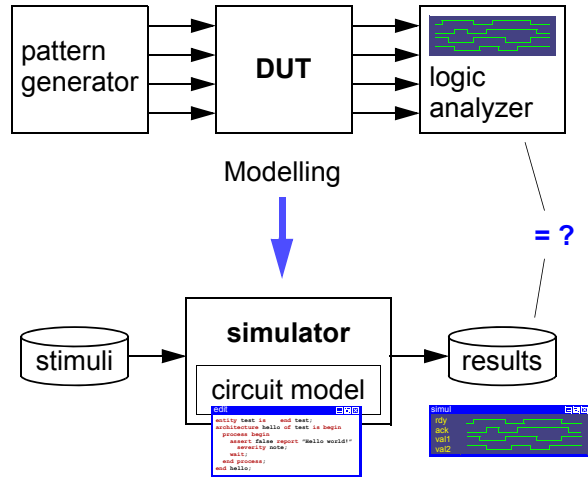




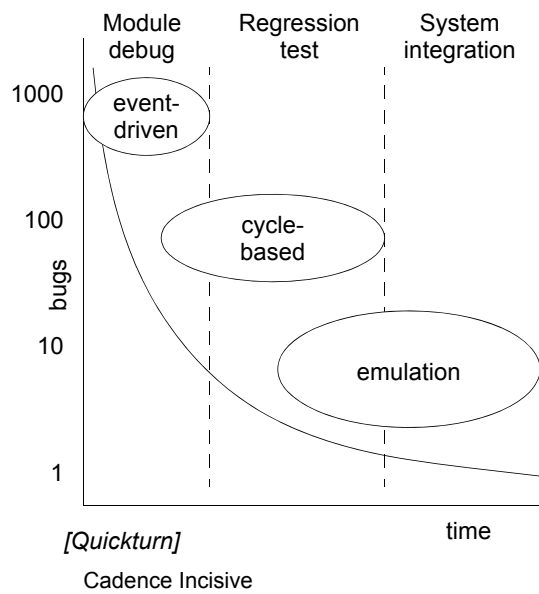
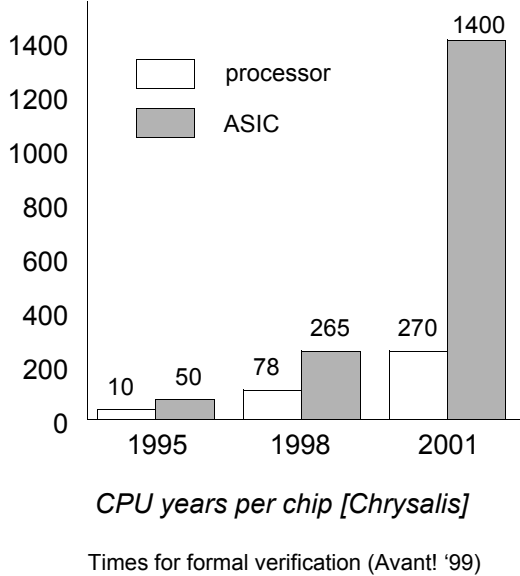
Simulation

Simulation = modelling + analysis

- Logic level simulation
- RT-level simulation
- Functional level simulation
- Behavioral level simulation
- System level simulation



Simulation problems





Simulators

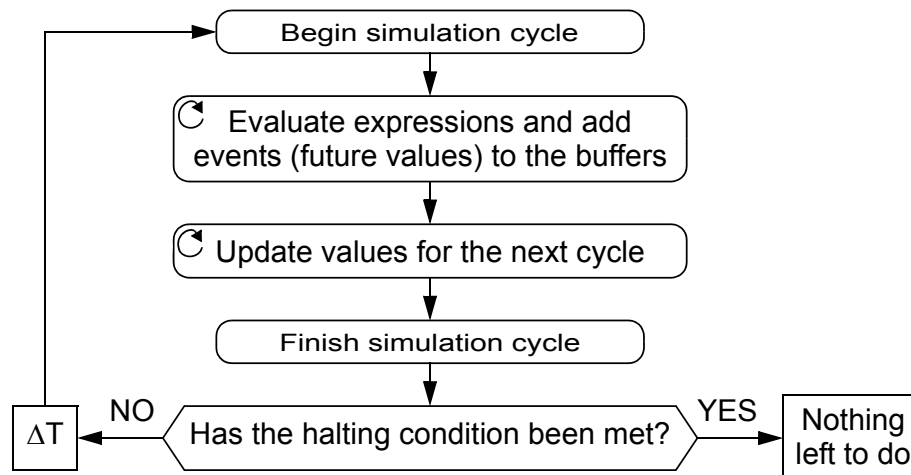
- **Time-driven:**
all components of the digital logic system are evaluated at every time step
- **Event-driven:**
system input events are kept in a time-ordered event queue

Simulation delay models

- **Unit-delay (RTL simulator)**
- **Zero-delay**
- **delta-delay (VHDL) -- δ -delay, Δ -delay**



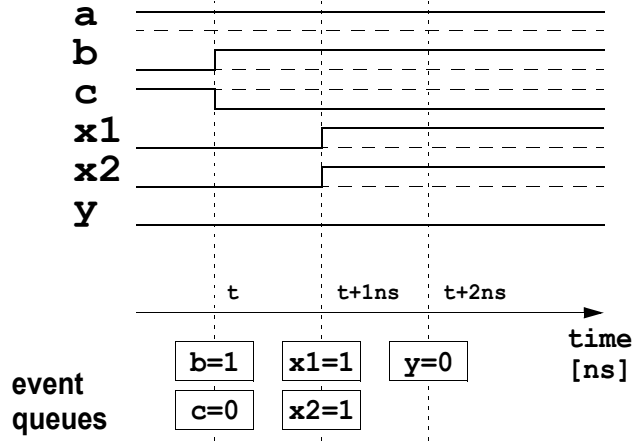
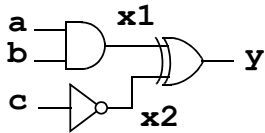
Unit-delay simulation model



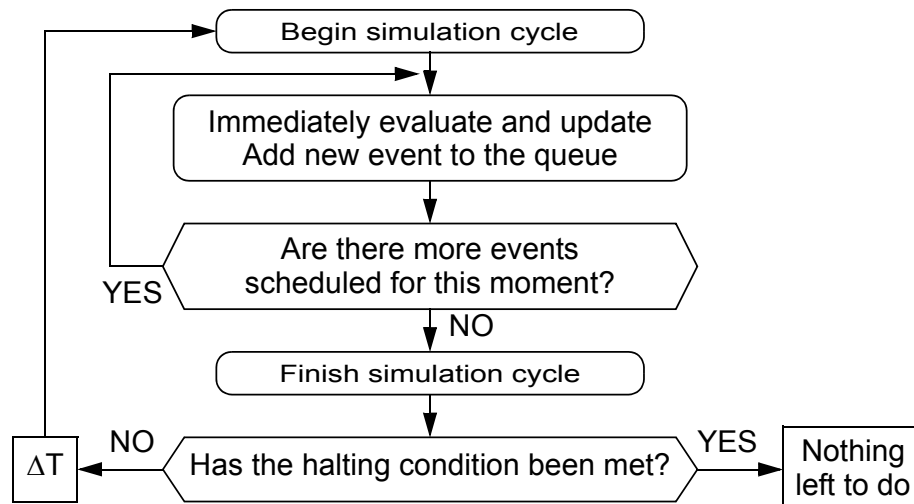


Unit-delay simulation model (example)

```
x1 <= a and b;
x2 <= not c;
y <= x1 xor x2;
```



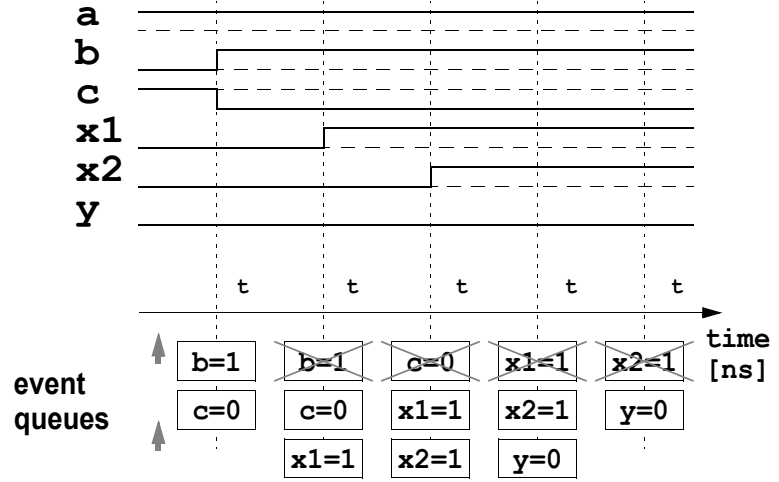
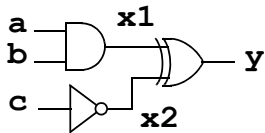
Zero-delay simulation model





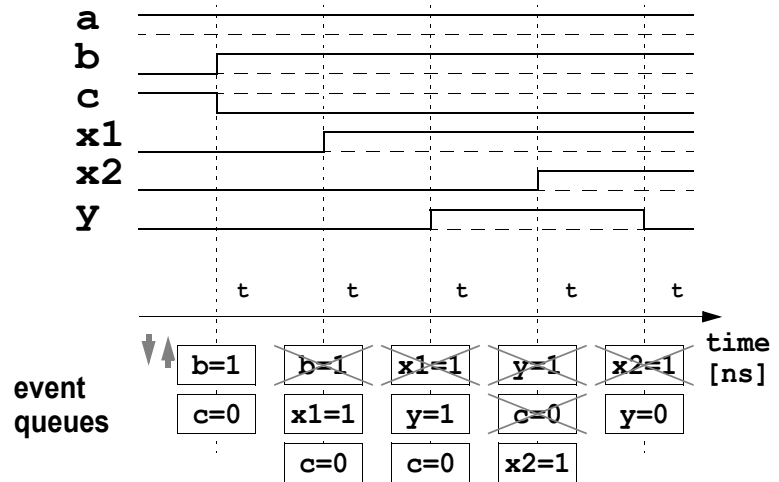
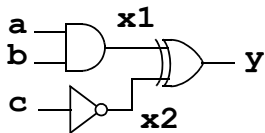
Zero-delay simulation model (example #1)

```
x1 <= a and b;
x2 <= not c;
y <= x1 xor x2;
```



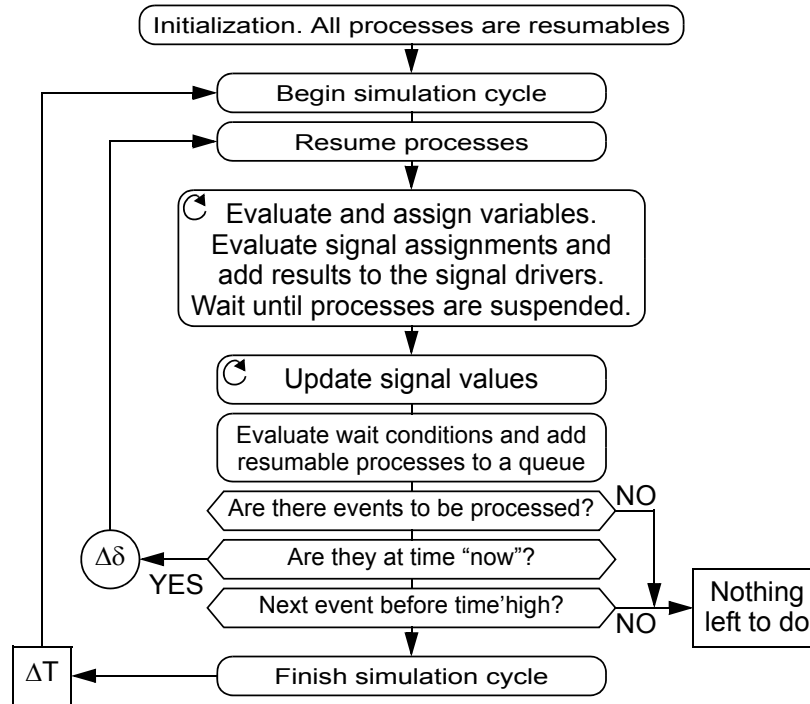
Zero-delay simulation model (example #2)

```
x1 <= a and b;
x2 <= not c;
y <= x1 xor x2;
```



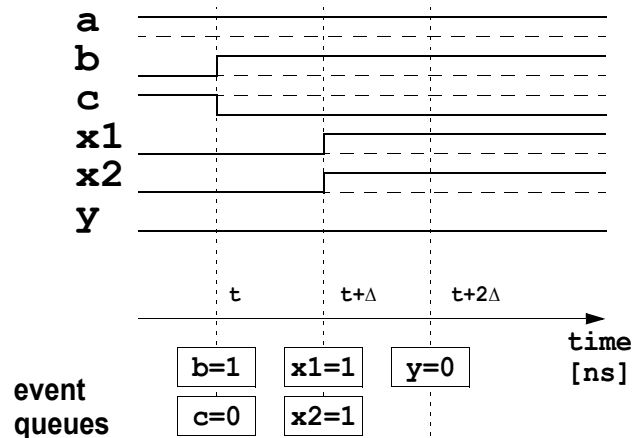
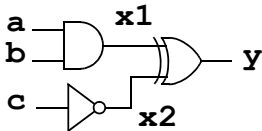


Delta-delay (VHDL) simulation model

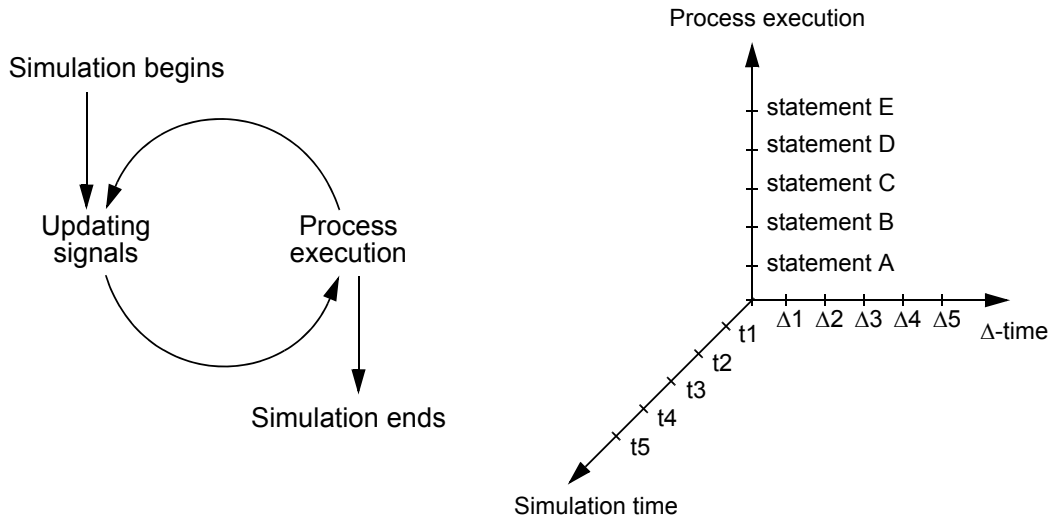


Delta-delay (VHDL) simulation model (example)

```
x1 <= a and b;
x2 <= not c;
y <= x1 xor x2;
```



VHDL simulation scheme



Drivers and delay modelling

- **Waveform generation**
`y <= '0' after 0 ns, '1' after 20 ns, '0' after 50 ns;`
 - 30 ns pulse at 20 ns after the simulation starts
 - events should be listed in the increasing order of time
- **Transport and inertial delays**
`y <= transport inp_a after 20 ns;`
 - the new value is added to the list of events at the scheduled point in time; if there are other entries in the list, scheduled at a later point in the time, these entries will be cancelled`y <= inp_a after 20 ns;`
 - a shorter pulse than 20 ns will be suppressed`y <= reject 10 ns inertial inp_a after 100 ns;`
 - definition of minimum pulse width 10 ns`y <= reject 0 ns inertial inp_a after 20 ns;`
 - equivalent to *transport* delay



Delta delay example #1

```
-- SR flip-flop
x <= not (y and lset); -- (1)
y <= not (x and reset); -- (2)
```

time	lset	x	y	reset	stm.
20 ns	_	0	1	1	(1)
20 ns + 1 Δ	0	_/	1	1	(2)
20 ns + 2 Δ	0	1	_	1	(1)
20 ns + 3 Δ	0	1	0	1	-
30 ns	_/	1	0	1	(1)
30 ns + 1 Δ	1	1	0	1	-
40 ns	1	1	0	_	(2)
40 ns + 1 Δ	1	1	_/	0	(1)
40 ns + 2 Δ	1	_	1	0	(2)
40 ns + 3 Δ	1	0	1	0	-



Delta delay example #2

```
-- SR flip-flop + delays
x <= not (y and lset)
  after 2 ns; -- (1)
y <= not (x and reset)
  after 2 ns; -- (2)
```

time	lset	x	y	reset	stm.
20 ns	_	0	1	1	(1)
22 ns	0	_/	1	1	(2)
24 ns	0	1	_	1	(1)
24 ns + 1 Δ	0	1	0	1	-
30 ns	_/	1	0	1	(1)
30 ns + 1 Δ	1	1	0	1	-
40 ns	1	1	0	_	(2)
42 ns	1	1	_/	0	(1)
44 ns	1	_	1	0	(2)
44 ns + 1 Δ	1	0	1	0	-



Delta delay example #3

- Dangers of default initialization

```
-- SR flip-flop & oscillation
x <= not (y and lset); -- (1)
y <= not (x and reset); -- (2)
```

time	lset	x	y	reset	stm.
0 ns	_/	_	_	_/	(1),(2)
0 ns + 1 Δ	1	_/	_/	1	(1),(2)
0 ns + 2 Δ	1	_	_	1	(1),(2)
0 ns + 3 Δ	1	_/	_/	1	(1),(2)
etc.	1	1	(1),(2)
0 ns	_/	_/	_/	_/	(1),(2)
0 ns + 1 Δ	1	_	_	1	(1),(2)
0 ns + 2 Δ	1	_/	_/	1	(1),(2)
0 ns + 3 Δ	1	_	_	1	(1),(2)
etc.	1	1	(1),(2)

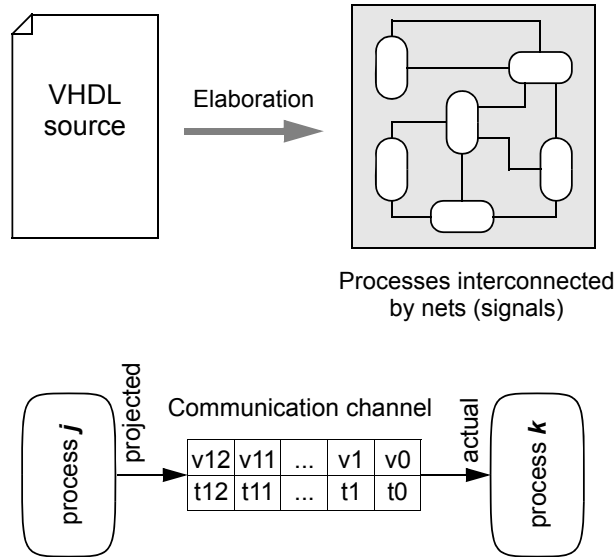


Preparing for VHDL simulation

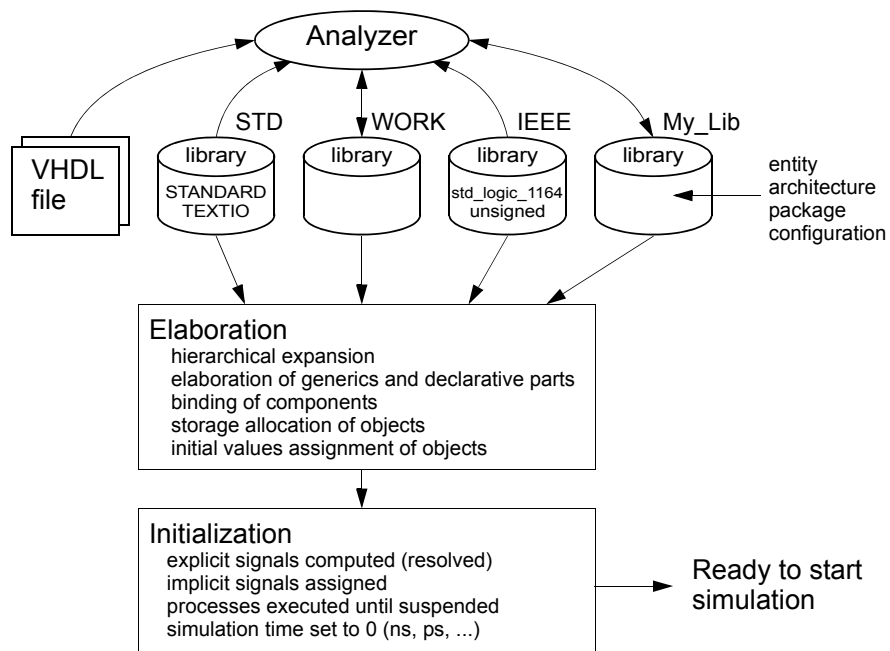
- VHDL source -->
- VHDL Analyzer (*vhdlan* in Synopsys, *vcom* in ModelSim) -->
 - VHDL intermediate format (analyzed code in library WORK) -->
- VHDL Simulator (*scsim* and GUI *scirocco* in Synopsys, *vsim* in ModelSim)
 - VHDL Elaborator (*scs* in Synopsys, *vsim* in ModelSim) -->
 - Processes interconnected by nets (internal format) -->



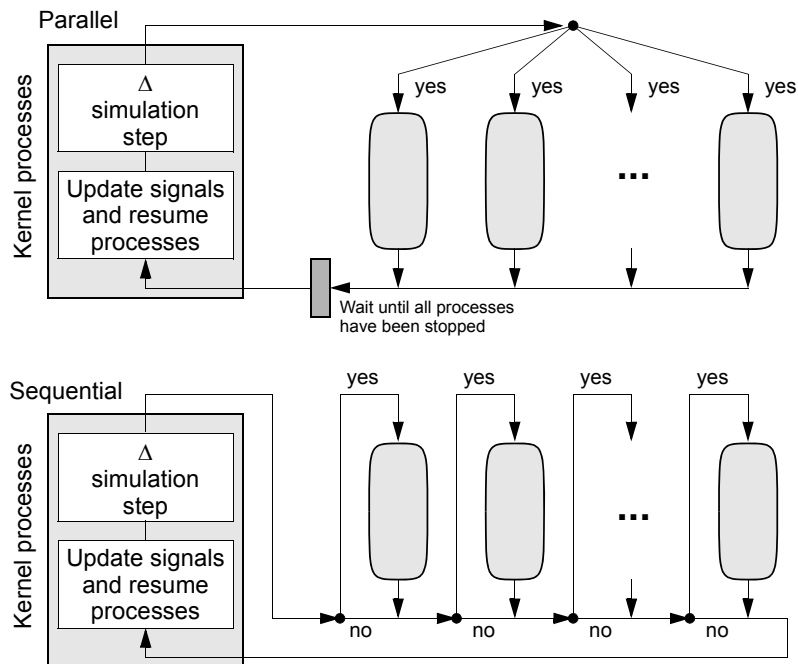
Elaboration



Compilation and elaboration



Sequential and parallel simulation

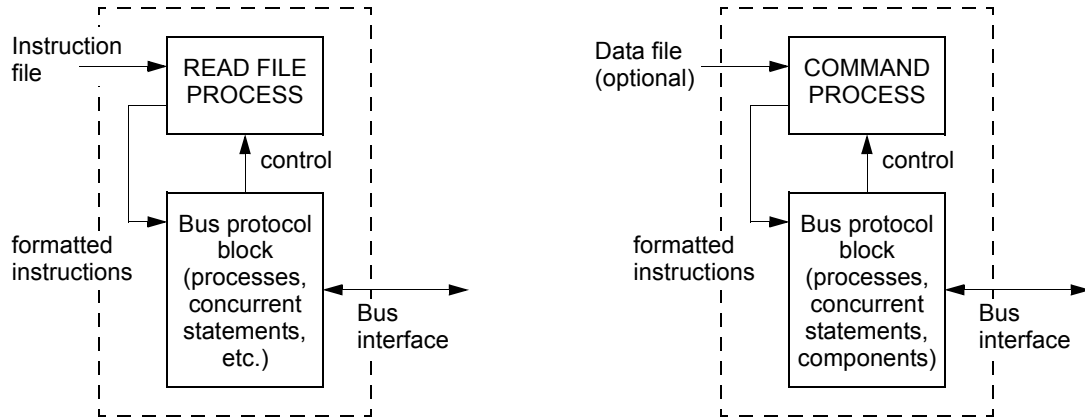


Functional Models

- **Functional Model (FM)** is a model of a component which represents both the interfaces and the internal operation or structure of the component
- **Bus Functional Model (BFM)** is a subset of the FM in that it only models the bus interfaces and bus transactions of the component
- **Unit Under Test (UUT)** represents a design itself
- **Testbench (TB)** is a VHDL component which instantiates the UUT
- A testbench may make use of FM and BFM models

BFM Modeling

- **Instruction file command format**
- **Architectural command format**



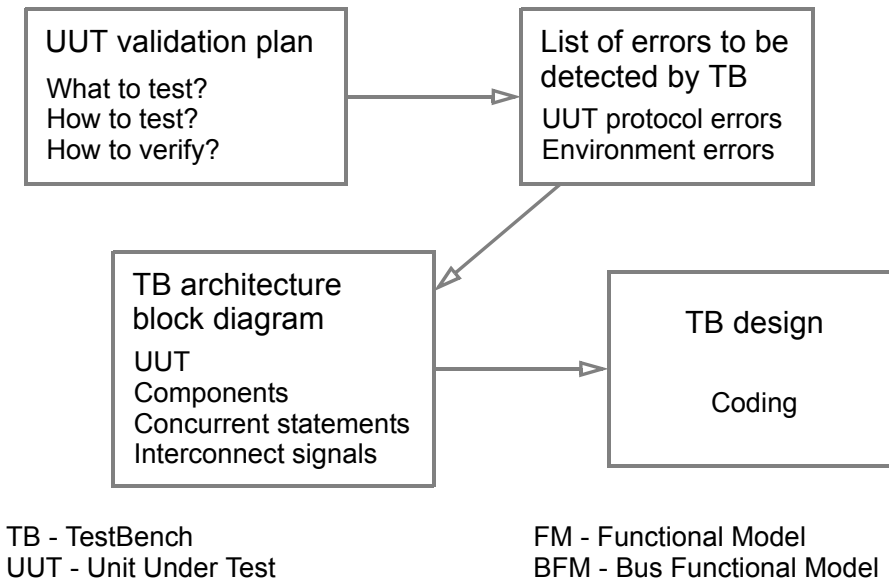
Testbench

Purposes:

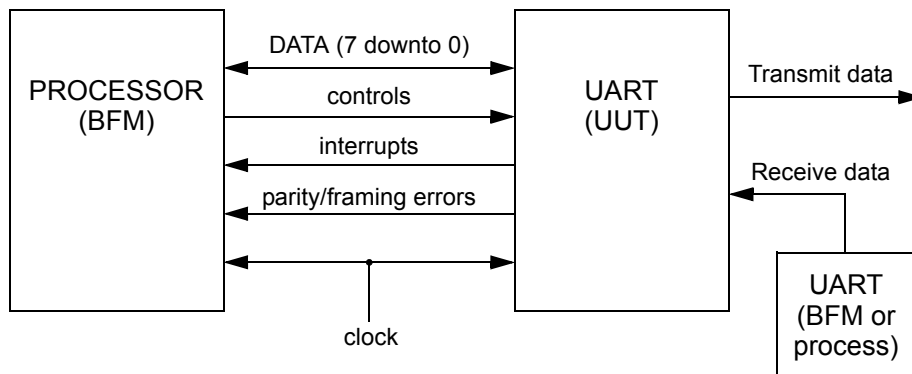
- **Stimuli generator(s)**
- **Verifier against UUT specification**
- **Report generation (human interface)**



Testbench design methodology



UART Testbench



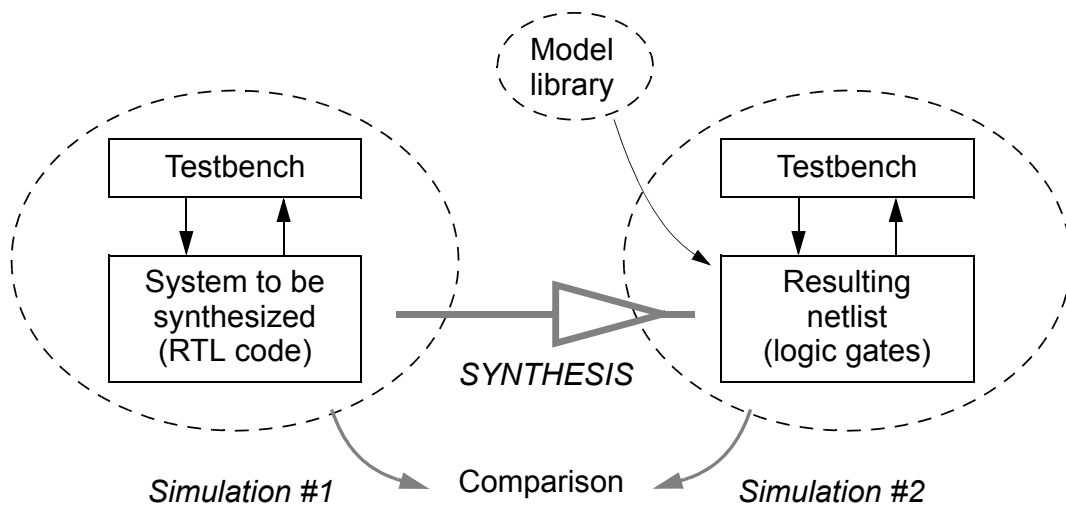


Testbench architecture elements

- **UUT** (any level)
- **Set of models which emulate the bus interfaces and bus transactions to the UUT**
- **A clock generator for the system**
- **A bus verifier to perform timing and protocol checks (+ reports)**
- **Rationale:**
 - The UUT is the whole purpose of the testbench
 - BFM's represent the environment to the UUT
 - Clocks represent another environment to the UUT
 - Bus verifier performs automatic verification function



Result validation methodology





Design refinement validation

- **Large projects & multiple teams**
 - one team -- one module
 - behavioral --> RTL --> gate level
- **Validating intermediate steps?**
 - the same test bench for all teams
 - refining & replacing the module under design
 - interactive / intelligent test benches
 - test sequence extraction



Design refinement validation

