Event-Driven Simulation

- The basis of VHDL simulation is event processing. VHDL simulators are event-driven simulators.
- There are four essential concepts to event-driven simulation:
  - simulation time
  - transaction generation
  - event processing
  - delta time
Simulation Time

• The simulator models time – this is called simulation time.
• Simulation time is an integral multiple of the resolution limit.
• The simulator cannot measure time delays less than the resolution limit.
• For gate-level simulations the resolution limit may be quite fine, possibly down to 1ps. For RTL simulations, there is no need to specify a fine resolution since we are only interested in clock-cycle by clock-cycle behaviour and the transfer functions are described with zero delay or unit delay. In this case, a resolution limit of 1ns is usually used.
Simulation Cycle

- **Simulation alternates between two modes:**
- **Statement Execution:**
  - each statement is recalculated *only* if its sensitivity list has an *event*.
  - this creates a new value for each target signal and a time at which the signal gets the value. This value/time pair is called a *transaction* and is added to a *transaction queue*.
  - target signals are not updated during statement execution
- **Event Processing**
  - transactions for the current time are taken off the queue
  - if a transaction represents a change in the value of the signal it becomes an *event*, causing the signal to be updated with the new value.
  - Signals are updated only at the next wait statement
    - A process with a sensitivity list has an implicit wait at the end of the process
    - Don’t mix sensitivity lists and wait statements: must have one or the other.
• When a transaction is created it is added to the queue for the scheduled time
• Each event generation phase consumes one time-step’s transactions
Transactions

• For example:

\[ a \leq '0' \text{ after 1 ns, '1' after 2 ns; } \]

• This signal assignment queues two transactions for signal a
  1. value '0' and time delay of 1ns
  2. value '1' and time delay of 2ns
Zero-Delay Transactions

- It is also possible to have a zero-delay assignment:

  ```
  a <= '0';
  ```

- the signal is still not updated immediately,
- **BUT**, a transaction still needs to be scheduled
  - If the transaction were scheduled at the current time:
    - It could be put at the start of the transaction queue
    - It could be put at the end of the transaction queue (easier to program)
  - Therefore the exact behaviour would depend on the simulator (not good
    – but see Verilog!)
Delta Time

- Zero-delay transactions are a problem – how do you handle a transaction for the current time?
- Answer: don’t! Schedule them for one delta time later
  - a delta time is infinitesimally small

- Each event generation phase consumes one delta time-step’s transactions
Event-Driven Simulation Model

- Example to illustrate the two-phase simulation cycle using delta time:
Initial State

- The start state of the circuit is all inputs set to 0

- What happens if input b changes to 1 at simulation time 20ns?
Initial Transaction Queue

- **Transaction processing phase:**
  - Initially b is 0, so this transaction represents a change from 0->1
  - This is therefore an event and so b is updated to 1
  - the transaction list for this delta is discarded

- **Statement Execution Phase**
  - all assignments with inputs that have events get recalculated:
    ```
    sum1 <= a xor b;
    c1 <= a and b;
    ```
  - this creates new transactions for sum1 and c1
After One Delta

- **Transaction processing phase:**
  - Initially sum1 is 0, so this transaction represents a change from 0->1
  - This is therefore an event and so sum1 is updated to 1
  - Initially c1 is 0, so this does not represent a change – no event is generated
  - the transaction list for this delta is discarded

- **Statement Execution Phase**
  - all assignments with inputs that have events get recalculated:
    
    ```
    sum2 <= sum1 xor c;
    c2 <= sum1 and c;
    ```
  - this creates new transactions for sum2 and c2
After Two Deltas

- **Transaction processing phase:**
  - Initially sum2 is 0, so this transaction represents a change from 0->1
  - This is therefore an event and so sum2 is updated to 1
  - Initially c2 is 0, so this does not represent a change – no event is generated
  - the transaction list for this delta is discarded

- **Statement Execution Phase**
  - all assignments with inputs that have events get recalculated:
    ```
    sum <= sum2;
    ```
  - this creates a new transaction for sum
After Three Deltas

- **Transaction processing phase:**
  - Initially sum is 0, so this transaction represents a change from 0->1
  - This is therefore an event and so sum is updated to 1
  - the transaction list for this delta is discarded

- **Statement Execution Phase**
  - there are no statements triggered by this, so delta processing has finished and simulation time can move on
Transaction Summary

- This shows all the transactions that occurred