7.1 Accurate Gate Level Modeling - Back Annotation of Delays

- For post-layout simulation, it is important to use realistic delays. SDF (standard delay format), along with VITAL modes, accomplishes this. VITAL models create the blanks (generics). SDF fills in the blanks. SDF can include data for gates or interconnects.

- Gate Example

```plaintext
(CELL
 (CELLTYPE 'DFF')
 (INSTANCE top/b/q)
 (DELAY
 (ABSOLUTE
 (ICPATH (posedge clk) q (2:3:4) (5:6:7))
 (PROP clk (2:3:4) (5:6:7))
 )
 )
 (TIMINGCHECK
 (SETUPHOLD d (posedge clk) (3:4:5) (-1:-1:-1))
 (WIDTH clk (4:4:7:5:11:3))
 )
)
```
7.1 Accurate Gate Level Modeling - Back Annotation of Delays

- Interconnect Example

```
ABSOLUTE
    {INTERCONNECT d1.y c.r1.a (0.01:0.02:0.03)}
    {INTERCONNECT d1.y c.r2.a (0.03:0.04:0.05)}
    {INTERCONNECT d1.y r3.a (0.05:0.06:0.07)}
    {INTERCONNECT b2.y c.r1.a (0.04:0.05:0.06)}
    {INTERCONNECT b2.y c.r2.a (0.02:0.03:0.04)}
    {INTERCONNECT b2.y r3.a (0.02:0.03:0.04)}
}
```

VITAL Library Information

```
Combinational Cell OR
	/* Library Source Code */

cell (OR)
    (area : 9.0000 ;
     pin(Y) [direction : output ;
                capacitance : 0.5000 ;
               max_fanout : 15.0000 ;
                function : "!(A + B + C) D" ;
                  timing() [intrinsic_rise : 0.1234 ;
                             rise_resistance : 0.1500 ;
                             intrinsic_fall : 0.4500 ;
                             fall_resistance : 0.1600 ;
                             related_pin : "A" ;]
                  timing() [intrinsic_rise : 0.1234 ;
                             rise_resistance : 0.1500 ;
                             intrinsic_fall : 0.4500 ;
                             fall_resistance : 0.1600 ;
                             related_pin : "B" ;]

    pin(A) [direction : input ;
             capacitance : 1.0000 ;
             fanout_load : 1.0000 ;]
    pin(B) [direction : input ;
             capacitance : 1.0000 ;
             fanout_load : 1.0000 ;]
```
7.1 Accurate Gate Level Modeling - VITAL Entity Description

use IEEE.VITAL_Timing.all;
-- entity declaration --
entity ONAND is
  generic(
    TimingChecksOn: Boolean := True;
    InstancePath: STRING := "";
    Xon: Boolean := False;
    MsgOn: Boolean := True;
    tpd_A_Y   : VitalDelayType01:= (0.123 ns, 0.450 ns);
    tpd_B_Y   : VitalDelayType01 := (0.123 ns, 0.450 ns);
    tpd_C_Y   : VitalDelayType01 := (0.123 ns, 0.450 ns);
    tpd_D_Y   : VitalDelayType01 := (0.180 ns, 0.530 ns);
    tipd_A    : VitalDelayType01 := (0.000 ns, 0.000 ns);
    tipd_B    : VitalDelayType01 := (0.000 ns, 0.000 ns);
    tipd_C    : VitalDelayType01 := (0.000 ns, 0.000 ns);
    tipd_D    : VitalDelayType01 := (0.000 ns,0.000 ns));
  port(
    Y                           : out   STD_LOGIC;
    A                           : in    STD_LOGIC;
    B                           : in    STD_LOGIC;
    C                           : in    STD_LOGIC;
    D                           : in    STD_LOGIC);
attribute VITAL_LEVEL0 of ONAND : entity is TRUE;
end ONAND;

7.1 Accurate Gate Level Modeling - VITAL Architecture Description

library IEEE;
use IEEE.VITAL_Primitives.all;
--library LIBVUOF;
--use LIBVUOF.VTABLES.all;
architecture VITAL of ONAND is
attribute VITAL_LEVEL1 of VITAL : architecture is TRUE;
SIGNAL A_ipd  : STD_ULOGIC := 'X';
SIGNAL B_ipd  : STD_ULOGIC := 'X';
SIGNAL C_ipd  : STD_ULOGIC := 'X';
SIGNAL D_ipd  : STD_ULOGIC := 'X';
begin
-- INPUT PATH DELAYS
WireDelay : block
begin
  VitalWireDelay (A_ipd, A, tipd_A);
  VitalWireDelay (B_ipd, B, tipd_B);
  VitalWireDelay (C_ipd, C, tipd_C);
  VitalWireDelay (D_ipd, D, tipd_D);
end block;
7.1 Accurate Gate Level Modeling - VITAL Architecture Description (cont.)

```
-- BEHAVIOR SECTION
VITALBehavior : process (A_ipd, B_ipd, C_ipd, D_ipd)
-- functionality results
VARIABLE Results: STD_LOGIC_VECTOR(1 to 1):=(others => 'X');
ALIAS Y_zd : STD_LOGIC is Results(1);
-- output glitch detection variables
VARIABLE Y_GlitchData : VitalGlitchDataType;
begin
-- Functionality Section
Y_zd := (NOT ((D_ipd) AND ((B_ipd) OR (A_ipd) OR C_ipd)));
-- Path Delay Section
VitalPathDelay01 (OutSignal => Y,
GlitchData => Y_GlitchData,
OutSignalName => "Y",
OutTemp => Y_zd,
Paths => (0 => (A_ipd'last_event, tpd_A_Y, TRUE),
1 => (B_ipd'last_event, tpd_B_Y, TRUE),
2 => (C_ipd'last_event, tpd_C_Y, TRUE),
3 => (D_ipd'last_event, tpd_D_Y, TRUE)),
Mode => OnDetect,
Xon => Xon,
MmsgOn => MmsgOn,
MmsgSeverity => WARNING);
end process;
end VITAL;
```

7.1 Accurate Gate Level Modeling - VITAL Timing Checks

```
use work.SYSTEM_4.all;
entity FFRS is
  generic(FFDEL,SPIKE_WIDTH: TIME);
  port(R,S: in MVL4; Q,QN: out MVL4);
end FFRS;
architecture BEHAV of FFRS is
begin
process(R,S)
variable R_LAST_EVT, S_LAST_EVT: TIME := 0 ns;
variable BOTH: BOOLEAN:= FALSE;
begin
-----------Check for X's and Z's on inputs
  assert not(R='X') report "X  on R" severity WARNING;
  assert not(R='Z') report "Z  on R" severity WARNING;
  assert not(S='X') report "X  on S" severity WARNING;
  assert not(S='Z') report "Z  on S" severity WARNING;
-------Spike Detection
  assert (NOW = 0 NS) or ((NOW - R_LAST_EVT) > SPIKE_WIDTH)
    report "Spike On R" severity WARNING;
  R_LAST_EVT := NOW;
  assert (NOW = 0 NS) or ((NOW - S_LAST_EVT) > SPIKE_WIDTH)
    report "Spike On S" severity WARNING;
  S_LAST_EVT := NOW;
end process;
end FFRS;
```