

# Gate-Level Timing Analysis

VITAL Models & SDF Files

# Timing simulation with VHDL models

---

- ▶ Synthesized structural models comprise cells from a library (ADK library, for example)
  - ▶ Each library cell has been *characterized* to determine delays, constraints, etc.
  - ▶ Most current libraries represent timing information using the **VITAL** standard
    - ▶ VHDL Initiative Toward ASIC Libraries
    - ▶ IEEE Standard 1076.4
- ▶ Synthesis tools create a Standard Delay Format (SDF) file of timing data for each cell in the design for use with VITAL models
  - ▶ SDF = IEEE Standard 1497
  - ▶ Designed for back-annotation of netlists with delay data



# Example

---

- ▶ The modulo 7 counter was synthesized using one of the technologies in the ADK
- ▶ The next slides show a VITAL model of a D flip flop from the library.
- ▶ The following slides contain a partial SDF file, showing the timing parameters extracted for a couple of gates in the synthesized design



# VITAL model of a D flip flop

---

```
library IEEE; use IEEE.STD_LOGIC_1164.all;  
use IEEE.VITAL_Primitives.all; -- Primitive logic functions  
use IEEE.VITAL_Timing.all;    -- Delay/Constraint calculation func's
```

```
entity dff is
```

```
  generic (
```

```
    tpd_CLK : VitalDelayType01Z := VitalZeroDelay01Z;  
    tpd_D : VitalDelayType01Z := VitalZeroDelay01Z;  
    tpd_CLK_Q : VitalDelayType01Z := VitalZeroDelay01Z;  
    tpd_CLK_QB : VitalDelayType01Z := VitalZeroDelay01Z;  
    tsetup_D_CLK_noedge_posedge : VitalDelayType := 0 ns;  
    thold_D_CLK_noedge_posedge : VitalDelayType := 0 ns;  
    tpw_CLK_posedge : VitalDelayType := 0 ns;  
    tpw_CLK_negedge : VitalDelayType := 0 ns;  
    TimingChecksOn : BOOLEAN := TRUE;  
    InstancePath : STRING := "*"
```

```
  );
```

```
-- VitalDelayType01Z = delays for 0-1,1-0,0-Z,1-Z,Z-0,Z-1 changes
```

---

▶ Continued

# DFF model (continued)

---

```
port (  
    D : in STD_LOGIC;  
    CLK : in STD_LOGIC;  
    Q : out STD_LOGIC;  
    QB : out STD_LOGIC  
);  
attribute VITAL_LEVEL0 of dff : entity is TRUE;  
end dff;
```

```
architecture dff_arch of dff is  
    attribute VITAL_LEVEL1 of dff_arch : architecture is TRUE;  
    signal CLK_ipd : STD_LOGIC := 'X';  
    signal D_ipd : STD_LOGIC := 'X';
```

Continued

---



# DFF model (continued)

---

begin

```
WireDelay : Block begin -- Input wire delays
    VitalWireDelay (CLK_ipd, CLK, tipd_CLK);
    VitalWireDelay (D_ipd, D, tipd_D);
end Block;
```

```
VitalBehavior : Process (CLK_ipd, D_ipd)
    VARIABLE Tviol_0 : X01 := '0';
    VARIABLE SetupHoldInfo_0 : VitalTimingDataType :=
                                                VitalTimingDataInit;

    VARIABLE Pviol_0 : X01 := '0';
    VARIABLE PeriodDataInfo_0 : VitalPeriodDataType :=
                                                VitalPeriodDataInit;

    VARIABLE Violation_0 : X01 := '0';
    VARIABLE PrevData_0 : STD_LOGIC_VECTOR(0 to 2);
    VARIABLE Results_0 : STD_LOGIC_VECTOR(0 to 2);
```

Continued

---



# DFF model (continued)

```
CONSTANT DFF_table : VitalStateTableType := (  
    ('X', '-', '-', '-', '-', 'X', 'X', 'X'), ('-', '^', '0', '0', '-', 'X', '0', '1'),  
    ('-', '^', '1', '0', '-', 'X', '1', '0'), ('-', '^', '0', '1', '0', 'X', '0', '1'),  
    ('-', '^', '1', '1', '1', 'X', '1', '0'), ('-', 'f', '0', '-', '0', '1', 'S', 'S'),  
    ('-', 'f', '0', '-', '1', '1', 'X', 'X'), ('-', 'f', '1', '-', '1', '1', 'S', 'S'),  
    ('-', 'f', '1', '-', '0', '1', 'X', 'X'), ('-', 'v', 'B', 'B', '-', 'X', 'S', 'S'),  
    ('-', 'r', '0', '-', '0', '0', 'S', 'S'), ('-', 'r', '0', '-', '-', '0', 'X', 'X'),  
    ('-', 'r', '1', '-', '1', '0', 'S', 'S'), ('-', 'r', '1', '-', '-', '0', 'X', 'X'),  
    ('-', '/', '0', '-', '-', 'X', '0', '1'), ('-', '/', '1', '-', '-', 'X', '1', '0'),  
    ('-', '\', '-', '-', '-', 'X', 'S', 'S'), ('-', '*', '-', '-', '-', 'X', 'X', 'X'),  
    ('-', 'B', '*', '-', '-', 'X', 'S', 'S'), ('-', '-', '*', '-', '-', 'X', 'S', 'S'),  
    ('-', '-', '-', '-', '-', 'S', 'S', 'S')  
);
```

```
ALIAS INT_RES_0 : STD_LOGIC is Results_0(1);
```

```
ALIAS INT_RES_1 : STD_LOGIC is Results_0(2);
```

```
VARIABLE GlitchData_Q : VitalGlitchDataType;
```

```
VARIABLE GlitchData_QB : VitalGlitchDataType;
```

Continued



# DFF model (continued)

---

```
begin -- process begin
```

```
-----  
--      TIMING SECTION      --  
-----
```

```
if (TimingChecksOn) then --check constraint violations
```

```
    VitalSetupHoldCheck (
```

```
        TestSignal => D_ipd, TestSignalName => "D",
```

```
        RefSignal => CLK_ipd, RefSignalName => "CLK",
```

```
        SetupHigh => tsetup_D_CLK_noedge_posedge,
```

```
        SetupLow => tsetup_D_CLK_noedge_posedge,
```

```
        HoldHigh => thold_D_CLK_noedge_posedge,
```

```
        HoldLow => thold_D_CLK_noedge_posedge,
```

```
        CheckEnabled => TRUE,
```

```
        RefTransition => 'R',
```

```
        TimingData => SetupHoldInfo_0,
```

```
        MsgOn => TRUE, XOn => TRUE,
```

```
        HeaderMsg => InstancePath & "/dff",
```

```
        Violation => Tviol_0,
```

```
        MsgSeverity => ERROR );
```

Continued



# DFF model (continued)

---

```
VitalPeriodPulseCheck (  
    TestSignal => CLK_ipd, TestSignalName => "CLK",  
    Period => 0 ns,  
    PulseWidthHigh => tpw_CLK_posedge,  
    PulseWidthLow => tpw_CLK_negedge,  
    CheckEnabled => TRUE,  
    PeriodData => PeriodDataInfo_0,  
    MsgOn => TRUE, XOn => TRUE,  
    HeaderMsg => InstancePath & "/dff",  
    Violation => Pviol_0,  
    MsgSeverity => ERROR  
);  
end if;
```

Continued

---



# DFF model (continued)

---

```
-----  
--    FUNCTIONALITY SECTION    --  
-----
```

```
-- Functional behavior of D flip flop
```

```
Violation_0 := Tviol_0 or Pviol_0;
```

```
VitalStateTable (
```

```
    StateTable => DFF_table,
```

```
    DataIn => STD_LOGIC_VECTOR'(
```

```
        Violation_0, CLK_ipd, D_ipd
```

```
    ),
```

```
    NumStates => 2,
```

```
    Result => Results_0,
```

```
    PreviousDataIn => PrevData_0
```

```
);
```

Continued

---



# DFF model (continued)

---

```
--      PATH DELAY SECTION      --  
-----
```

```
VitalPathDelay01Z (  -- delay to output Q  
    OutSignal => Q,  
    OutSignalName => "Q",  
    OutTemp => INT_RES_0,  
    Paths => (  
        0 => (  
            InputChangeTime => CLK_ipd'LAST_EVENT,  
            PathDelay => tpd_CLK_Q,  
            PathCondition => TRUE  
        )  
    ),  
    GlitchData => GlitchData_Q,  
    Mode => OnDetect,  
    MsgOn => TRUE, Xon => TRUE,  
    MsgSeverity => WARNING  );
```

Continued



# DFF model (continued)

---

```
VitalPathDelay01Z (          -- delay to output QB
    OutSignal => QB,
    OutSignalName => "QB",
    OutTemp => INT_RES_1,
    Paths => (
        0 => (
            InputChangeTime => CLK_ipd'LAST_EVENT,
            PathDelay => tpd_CLK_QB,
            PathCondition => TRUE
        )
    ),
    GlitchData => GlitchData_QB,
    Mode => OnDetect,
    MsgOn => TRUE, Xon => TRUE,
    MsgSeverity => WARNING
);
end Process VitalBehavior;
end dff_arch;
```

---



# Standard Delay File (SDF)

---

- ▶ Produced by synthesis tool (Leonardo)
- ▶ Contains VITAL data values for all cells in the netlist
- ▶ Example (for synthesized counter and ADK library):

SDF Header – common to all cells in this synthesized netlist

```
(DELAYFILE
(SDFVERSION "2.0")
(DESIGN "count4")
(DATE "Thu Feb 09 15:49:13 2006")
(VENDOR "Exemplar Logic, Inc., Alameda")
(PROGRAM "LeonardoSpectrum Level 3")
(VERSION "2005a.82")
(DIVIDER /)
(VOLTAGE)
(PROCESS)
(TEMPERATURE)
(TIMESCALE 1 ns)
```



# SDF data for DFFR instance Q\_0

---

- path delays and constraints
- one for each cell instance in the circuit

```
(CELL
  (CELLTYPE "dffr")
  (INSTANCE Q_0)
  (DELAY
    (ABSOLUTE
      (PORT D (::0.00) (::0.00))
      (PORT CLK (::0.00) (::0.00))
      (PORT R (::0.00) (::0.00))
      (IOPATH CLK Q (::0.51) (::0.55))
      (IOPATH R Q (::0.00) (::0.60))
      (IOPATH CLK QB (::0.39) (::0.30))
      (IOPATH R QB (::0.44) (::0.00))))
  (TIMINGCHECK
    (SETUP D (posedge CLK) (0.47))
    (HOLD D (posedge CLK) (-0.06))))
```

(Min: Typical: Max) Select at simulation setup

Input pin delays

Path delays from reset/clock to outputs

Constraints

---



# SDF data for NAND instance ix170

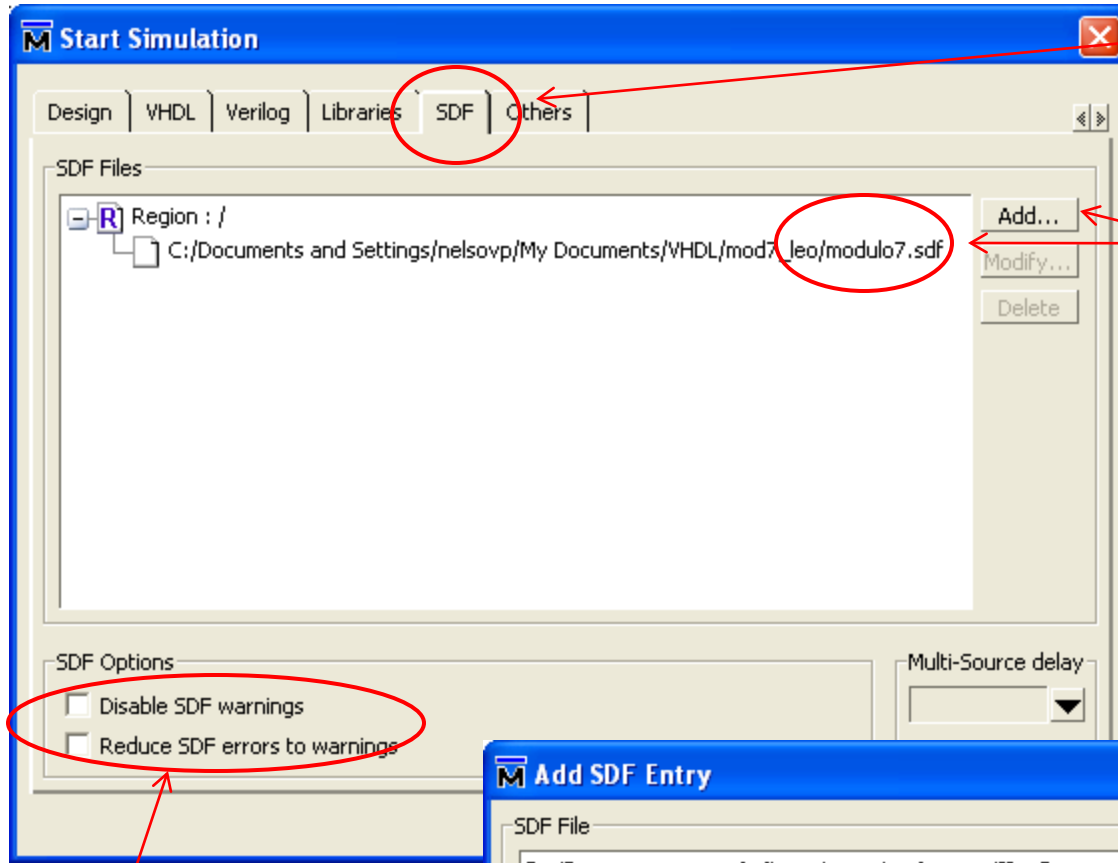
---

-- path delays only (no constraints)

```
(CELL
  (CELLTYPE "nand02")
  (INSTANCE ix170)
  (DELAY
    (ABSOLUTE
      (PORT A0 (::0.00) (::0.00))
      (PORT A1 (::0.00) (::0.00))
      (IOPATH A0 Y (::0.09) (::0.08))
      (IOPATH A1 Y (::0.12) (::0.10))))))
```



# Timing simulation in Modelsim

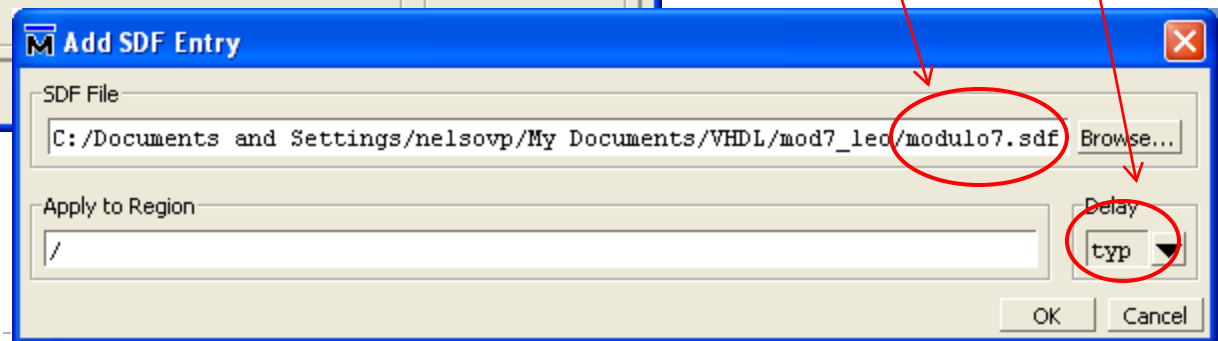


Select SDF tab when starting simulation.

Click Add to select SDF file

Select min, typ or max delay

Reduce error to warning to permit simulation to continue after error



# Simulating with SDF & do file

---

```
vsim modulo7 -sdftyp modulo7_0.sdf -sdfnoerror -L adk -t ps
```

## Compile options

-sdftyp modulo7\_0.sdf

-Apply delays from SDF file modulo7\_0.sdf

-Can also use -sdfmin or -sdfmax.

-sdfnoerror -Reduce SDF errors to warnings to enable simulation with missing hold times, etc.

-L adk -Library of gate-level VITAL models.

-t ps -Timing resolution consistent with SDF.



# Simulating with SDF & testbench

---

```
vsim -sdftyp /modulo7_bench/UUT=modulo7_0.sdf  
-sdfnoerror -L adk -t ps modulo7_bench
```

## Compile options

`-sdftyp /modulo7_bench/UUT=modulo7_1.sdf`

-Apply delays only to design instance (UUT).

-Can also use `-sdfmin` or `-sdfmax`.

`-sdfnoerror`

-Reduce SDF errors to warnings to enable simulation with missing hold times, etc.

`-L adk`

-Library of gate-level VITAL models.

`-t ps`

-Timing resolution consistent with SDF.

---



# Testbench: *modulo7\_bench.vhd*

```
LIBRARY ieee; USE ieee.std_logic_1164.all;  
USE ieee.numeric_std.all;
```

```
ENTITY modulo7_bench is end modulo7_bench;
```

```
ARCHITECTURE test of modulo7_bench is
```

```
  component modulo7
```

```
  PORT (reset,count,load,clk: in std_logic;
```

```
        I: in std_logic_vector(2 downto 0);
```

```
        Q: out std_logic_vector(2 downto 0));
```

```
  end component;
```

```
  for all: modulo7 use entity work.modulo7(Behave);
```

```
  signal clk : STD_LOGIC := '0';
```

```
  signal res, cnt, ld: STD_LOGIC;
```

```
  signal din, qout: std_logic_vector(2 downto 0);
```

```
begin
```

```
-- instantiate the component to be tested
```

```
UUT: modulo7 port map(res,cnt,ld,clk,din,qout);
```

Alternative  
to “do” file

Continue on  
next slide



# Testbench: modulo7\_bench.vhd

qint = expected outputs of UUT

```
clk <= not clk after 10 ns;
```

```
PI: process
```

```
  variable qint: UNSIGNED(2 downto 0);
```

```
  variable i: integer;
```

```
begin
```

```
  qint := "000";
```

```
  din <= "101"; res <= '1';
```

```
  cnt <= '0'; ld <= '0';
```

```
  wait for 10 ns;
```

```
  res <= '0';    --activate reset for 10ns
```

```
  wait for 10 ns;
```

```
  assert UNSIGNED(qout) = qint
```

```
    report "ERROR Q not 000"
```

```
    severity WARNING;
```

```
  res <= '1';    --deactivate reset
```

```
  wait for 5 ns; --hold after reset
```

```
  ld <= '1';    --enable load
```

```
  wait until clk'event and clk = '1';
```

```
  qint := UNSIGNED(din); --loaded value
```

```
  wait for 5 ns;    --hold after load
```

```
  ld <= '0';    --disable load
```

```
  cnt <= '1';    --enable count
```

```
  for i in 0 to 20 loop
```

```
    wait until clk'event and clk = '1';
```

```
    assert UNSIGNED(qout) = qint
```

```
      report "ERROR Q not Q+1"
```

```
      severity WARNING;
```

```
    if (qint = "110") then
```

```
      qint := "000";    --roll over
```

```
    else
```

```
      qint := qint + "001"; --increment
```

```
    end if;
```

```
  end loop;
```

```
end process;
```

Print message if incorrect result

# Simulation without SDF

The screenshot shows the ModelSim SE 6.1f interface. The 'Objects' window on the left lists variables: clk (0), count (0), i (011), load (0), q (001), q\_s (001), and reset (1). The 'list' window on the right shows a simulation log with time steps and signal values. Two instances of '+2' in the time column are circled in red, with arrows pointing to the text 'Note "delta" delays for behavioral model.'

Time (ps)	Delta	/modulo7/clk	/modulo7/reset	/modulo7/count	/modulo7/load	/modulo7/i
0	+0	U	U	U	U	X
0	+1	0	1	0	0	5
10000	+0	0	0	0	0	5
10000	+2	0	0	0	0	5
20000	+0	1	1	0	0	5
30000	+0	1	1	0	1	5
40000	+0	0	1	0	1	5
60000	+0	1	1	0	1	5
60000	+2	1	1	0	1	5
70000	+0	1	1	0	0	5
80000	+0	0	1	0	0	5
90000	+0	0	1	1	0	5
100000	+0	1	1	1	0	5
100000	+2	1	1	1	0	5
120000	+0	0	1	1	0	5
140000	+0	1	1	1	0	5
140000	+2	1	1	1	0	5
160000	+0	0	1	1	0	5

Note "delta" delays for behavioral model.

# Simulation using SDF file

Note actual delays within test circuit.

Delta delays at testbench level.

ns	delta	/modulo7_bench/clk	/modulo7_bench/qout
0.000	+0	0 U U U UUU	XXX
0.000	+1	0 1 0 0 101	XXX
1.500	+0	1 1 0 0 101	XXX
2.000	+1	1 0 0 0 101	XXX
2.140	+0	1 0 0 0 101	OXO
2.150	+0	1 0 0 0 101	000
3.000	+0	0 0 0 0 101	000
4.000	+1	0 1 0 0 101	000
4.500	+0	1 1 0 0 101	000
5.000	+1	1 1 0 1 101	000
6.000	+0	0 1 0 1 101	000
7.500	+0	1 1 0 1 101	000
7.660	+0	1 1 0 1 101	100
7.670	+0	1 1 0 1 101	101
8.500	+1	1 1 1 0 101	101
9.000	+0	0 1 1 0 101	101
10.500	+0	1 1 1 0 101	101
10.610	+0	1 1 1 0 101	100
10.680	+0	1 1 1 0 101	110
12.000	+0	0 1 1 0 101	110
13.500	+0	1 1 1 0 101	110

load  
count