Post-Layout Simulation

Analog and Digital Turbo Simulator (ADiT)
Analog & Digital Turbo Simulator (ADiT)
Analog & Digital Turbo Simulator (ADiT)

Input format:
- ELDO, HSPICE, Spectre
- VCD, ECVD, I/O vector stimulus
- Verilog-A
- DSPF parasitic
- Synthesized Verilog gate-level

ADiT

SPICE Netlist

ADMS

Eldo

UDM
GUDM
UDRM

Waveforms
- WDB
- FSDB
- PSF

EZWave or Other Viewer

Standard Cell Library

.VLOG2SP

Spice netlist

.P. Verilog (Gate)
Analog and Digital Turbo Simulator (ADiT)

- Fast-SPICE simulation of analog and mixed-signal transistor-level circuits
- 10-100 times faster than SPICE
- Supports Eldo, HSPICE & Spectre netlist formats
- Integrated into ADVance MS
- Examples in $MGC_AMS_HOME/examples/adit/$
ADiT features

- A turbo engine using simplified matrix solver and the dominant-pole approximation (DPA) to enhance simulation speed
- An alternative built-in SPICE solver
- Device model libraries fully compatible with Eldo and HSPICE
- Flexible user control of simulation accuracy
- Post-layout simulation and parasitics reduction
- Behavioral modeling with Verilog-A
- Eldo compatible reliability simulation
- Hi-Z state and Hi-Z induced leakage current detection
- Integrated into Mentor Graphics IC flow, providing a complete front-to-back design and verification environment
**ADiT command line**

- `adit inpfile`
  - `-d` - turn on debug mode
  - `-eldo` – Eldo compatible format (default)
  - `-engine engine_id`
  - `-h` – display ADiT usage info on screen
  - `-m0 | -m 0` – hierarchical parser (vs flattening)
  - `-monte method_ID`
  - `-noinit` – disable adit.ini file
  - `-o outfile` – output log file
  - `-outpath path_name` – output directory
  - `-sckspice` – solve subcircuits with SPICE engine, disabling MOS table
  - `-spice` – enable SPICE engine (default is “turbo”)
  - `-v tool_name`
AdiT input & output files

Netlist From Schematic

Simulation Commands

Complete Netlist Simulation File (.cir, .sp)

ADiT Simulation

Simulation Log File (.log)
Results text files (.XPA, .ICO, .MT)

MOSFET Look-Up Tables

Simulation Results Files (.wdb)

Waveform Viewer
ADiT input/output files

- `<input_name>`.xxx – ADiT input file
- `<input_name>`.log – output with messages, statistics, etc
- `<input_name>`.IC0 – ckt state var’s at t=0
- `<input_name>_0.wdb` – graphic: transient analysis
- `<input_name>`.XPA – expansion of subcircuits
- `<input_name>`.PAR – parameter evaluation
- `<input_name>`.DCPOW0 – power dissipation from DC analysis
- `<input_name>`.DC0.wdb – graphic: DC transfer curve
- `<input_name>`.TB0.wdb – graphic: transient analysis (turbo)
- `<input_name>`.TR0.wdb – graphic: transient analysis (spice)
Simulation control

- **Input stimulus:**
  - SPICE voltage sources
  - “Bus” patterns
    - Time/value pairs
    - List of patterns with common timing
  - Digital test vectors

- **Results checks:**
  - Measurements: voltages, currents, calculations
  - Bus value checks (value at given time)
  - Test vectors: actual vs. expected values
Voltage force functions (1)

- DC value
  - $V_{\text{signame}} \text{ A 0 DC 5}$

  - $V$ indicates voltage
  - Force name
  - Between circuit nodes A and GND (node 0)
  - Value (volts)

- Alternate format: $V_{\text{signame}} \text{ A 0 5}$  (DC is default)
**Force functions (2)**

- **Pulse/square wave**

  - **V**signame B 0 **pulse** 0 5 0 0.1N 0.1N 20N 40N

<table>
<thead>
<tr>
<th>Nodes</th>
<th>Initial Voltage</th>
<th>Pulsed Voltage</th>
<th>Rise time</th>
<th>Fall time</th>
<th>Pulse width</th>
<th>Period</th>
</tr>
</thead>
<tbody>
<tr>
<td>v1</td>
<td></td>
<td>v2</td>
<td>tr</td>
<td>tf</td>
<td>tw</td>
<td>tp</td>
</tr>
</tbody>
</table>

Delay from start of period for waveform to begin - td

---

![Diagram](image)
Force functions (3)

- **Pattern wave** *(for logic 0 & 1 values)*

**Vname B 0 pattern 5 0 5n 0.1n 0.1n 10n 011010 R**

- Between circuit Nodes B & GND (node 0)
- Logic 1 & 0 voltages
- Delay to waveform begin
- Rise & Fall Time between changes
- Duration of bit value
- Bit pattern
- Repeat the pattern (optional)

```
0 1 1 0 0 1 0
```

*delay*  
*pattern*
• **Piecewise-linear wave** (digital if only two voltages)

\[ V_{\text{signame}} B \ 0 \ \text{pwl} \ (0n \ 0 \ 5n \ 0 \ 5.1n \ 5 \ 10n \ 5 \ 10.1n \ 0 \ R) \]

Nodes

- \( V_0 \) to \( V_1 \)
- \( V_2 \) to \( V_3 \)
- \( V_3 \) to \( V_4 \)

Other options: 
- \( R=\text{value} \) (time at which to begin repeat – one of \( T_n \) values)
  - default = 0 if no value specified
- \( TD=\text{value} \) (delay before waveform begins)
Example: modulo-7 counter

```
.include $ADK/technology/ic/models/tsmc035.mod
.include mod7b.pex.netlist
.topcell mod7b

.vdd VDD 0 dc 5
.vnd GND 0 dc 0
.vclk clk 0 pulse (0, 5, 0, 1n, 1n, 15n, 30n)
.vin0 i[0] 0 dc 5
.vin1 i[1] 0 dc 0
.vin2 i[2] 0 dc 5
.vload load 0 dc 0
.vreset reset 0 pulse (5, 0, 0, 1n, 1n, 5n, 0n)
.vcount count 0 dc 5

.probe v(q[2]) v(q[1]) v(q[0]) v(reset) v(clk)
.trans 0.1n 400n tmax=0.5n
```
EZwave waveform viewer
(results for previous circuit)

Double-click signal names to display.

Double-click TRAN (transient)
Making measurements

Measure delay from a “trigger” to a “target” condition

```
.MEASURE TRAN q0rise TRIG v(clk) VAL=2.5 RISE=1 TD=20n + TARG v(q[0]) VAL=2.5 RISE=2
```

- **TRAN** — measure a “transient”
- **q0rise** — variable name to which measurement assigned
- **TRIG/TARG** — defined trigger/target conditions
- **RISE=n** — condition is the nth rising edge (can also use FALL)
- **VAL=n** — trigger/target voltage level
- **TD=n** — delay before looking for trigger condition
Measurement examples

.param vddval=5.0
.param diffval1=’vddval/1.5’
.param outval2=’0.8*vddval’

.meas tran var1 always v(out1, out2) val=’diffval1’ win=100n fall
+ targ v(out2) val=’outval2’ fall=1

.meas tran var2 periodic at=52n win=100n targ v(out2)
+ val=’0.5*vddval’ rise=1

.MEAS RMSPWR RMS POWER
.MEAS APWR AVG POWER

The last two lines of examples show the POWER keyword specified.
Working with buses in ADiT

• Create a bus:
  
  \texttt{.SETBUS inbus i[2] i[1] i[0]} \quad \text{(name and components – also i[2:0])}

• Plot bus values:
  
  \texttt{.PLOTBUS inbus VTH=2.5 BASE=HEX RADIX=UNSIGNED}
  \texttt{.PLOTBUS inbus VTH1=1 VTH2=3 BASE=BIN RADIX=SIGNED}

  \text{(plots X if VTH1 < value < VTH2)}

• Stimulate a bus:
  
  \texttt{.SIGBUS inbus VHI=5 VLO=0 BASE=DEC SIGNED=NONE 10ns 5}

  \text{Specify one or more time-value pairs (apply value at specified time)}

  \texttt{.SIGBUS inbus VHI=5 VLO=0 BASE=DEC SIGNED=NONE}
  \texttt{+ TRISE=1n TFALL=1n THOLD=40n PATTERN 5 2 3 7}

  \text{Define waveform timing and pattern of values to apply}
Working with buses in ADiT (2)

- Check values on a bus at specified times & write any errors to a file

- Form 1: List time/value pairs (time at which value expected)
  
  .CHECKBUS inbus VTH=2.5 BASE=DEC
  + 50ns 5
  + 70ns 6

- Form 2: Specify timing and pattern of expected values
  
  .CHECKBUS inbus VTH=2.5 TSAMPLE=30n TDELAY=10n
  + BASE=DEC PATTERN 5 6 0 1 2
  
  Expect value 5 at t=10ns, 6 at t=40ns, 0 at t=70ns, etc.
ADiT– test vector file

- Verify design functionality/behavior
  - apply test vectors
    - ADiT Test Vector Format
    - VCD (Value Change Dump) Format – converted to ADiT format
  - capture outputs
  - compare outputs to expected result
  - vectors/outputs from behavioral simulation

- Command to execute a test vector file:
  `.VEC design.vec`

Test vector file (next slide)
ADiT test vector file format

Part 1: Vector Pattern Definition – define vector signals

- **Radix**: 1st non-comment line – define digit radix (#bits/digit)
  - 1 = binary, 3 = octal, 4 = hex

- **Node names** – in order of position within the vectors
  - Bus notation: d[2:0] or d[2-0] => d2 d1 d0

- **IO (direction) definitions**
  - i = input, o = output, b = bidirectional, x = ignored
  - Output signals are expected values – checked during simulation

Example (modulo-7 counter)

```
; Vector Pattern Definition
radix 1 1 3 3
io i i i i o
nodename load count i[[2-0]] q[[2:0]]
```
ADiT vector file format (2)

- **Part 2: Waveform Parameter Settings**
  - Tunit 0.1n (time unit, default = 1ns)
  - Slope 0.1 (rise & fall times, default = 0.1ns)
  - Trise 0.1 (rise time – overrides Slope value)
  - Tfall 0.1 (fall time – overrides Slope value)
  - Td 10 (global time delay for vector signals)
  - VOH 3 (logic threshold for sampling “1” output, default 3.3v)
  - VOL 2 (logic threshold for sampling “0” output, default 0v)
  - VTH 2.5 (logic threshold for outputs – if VOH/VOL not given)
  - VIH 5 (logic 1 voltage forced onto ckt inputs, default 3.3v)
  - VIL 0.8 (logic 0 voltage forced onto ckt inputs, default 0v)
  - CHKDELAY 10 (delay from vector applied to check of outputs)
  - CHKDELAY MAX (check outputs right before next vector applied)
Waveform definition example

; Waveform Parameter Settings

tunit ns              -- all times in units of 1ns
slope 0.1            -- rise and fall time
voh 3                -- V > 3V = logic 1
vol 1                -- V < 1V = logic 0
vih 5                -- Apply 5V for logic 1
vil 0                -- Apply 0V for logic 0
chkdelay max        -- Check outputs at max possible time
ADiT vector file format (3)

- **Tabular Data Format** – with arbitrary time steps
  - \[ T1 \quad s1 \quad s2 \quad s3 \quad \text{--time, values of signals 1 2 3 …} \]
  - \[ T2 \quad s1 \quad s2 \quad s3 \]

- **Tabular Data Format** – with uniform time step
  - \[ \text{PERIOD 20} \quad \text{-- apply vectors at 0, 20, 40, 60, …} \]
  - \[ s1 \quad s2 \quad s3 \quad \text{-- only signal values listed} \]
  - \[ s1 \quad s2 \quad s3 \]

- Tabular data states:
  - 0 = drive to ground/VIL, 1 = drive high to VIH
  - Z = high impedance, X = don’t care (set to ground)
  - Expected outputs: 0 < VOL, 1 > VOH, X (don’t care)
Test vector file example (form 1)

Generate clock and reset as separate voltage sources
 ; Vector Pattern Definition
 radix          1  1  3  3
 io             i  i  i  o
 nodename load count i[[2-0]] q[[2:0]]
 ...
 chkdelay max
 ; Tabular Data - time  in1 in2 in3 expected-output
 0  1  0  5  X
 20 1  0  5  0
 40 1  0  5  5
 60 0  1  5  5
 80 0  1  5  6
Test vector file example (form 2)

Generate clock and reset as separate voltage sources

; Vector Pattern Definition
radix 1 1 3 3
io i i i i o
nodename load count i[[2-0]] q[[2:0]]
...
chkdelay max

**period 30**

; Tabular Data (omit time) – in1 in2 in3 expected-out
1 0 5 X
1 0 5 0
1 0 5 5
0 1 5 5
0 1 5 6
Behavioral simulation listing

<table>
<thead>
<tr>
<th>is</th>
<th>clk</th>
<th>i</th>
<th>q</th>
</tr>
</thead>
<tbody>
<tr>
<td>delta</td>
<td>reset</td>
<td>load</td>
<td>count</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>+0</td>
<td>U U U U UU UU</td>
</tr>
<tr>
<td>0</td>
<td>+1</td>
<td>0</td>
<td>1 0 1 01 1 UU</td>
</tr>
<tr>
<td>2</td>
<td>+0</td>
<td>0</td>
<td>1 0 1 01 1 UU</td>
</tr>
<tr>
<td>2</td>
<td>+2</td>
<td>0</td>
<td>1 0 1 01 1 C00</td>
</tr>
<tr>
<td>7</td>
<td>+0</td>
<td>0</td>
<td>1 1 0 101 1 C00</td>
</tr>
<tr>
<td>10</td>
<td>+0</td>
<td>1</td>
<td>1 0 1 01 1 C00</td>
</tr>
<tr>
<td>10</td>
<td>+2</td>
<td>1</td>
<td>1 0 1 01 1 C00</td>
</tr>
<tr>
<td>20</td>
<td>+0</td>
<td>0</td>
<td>1 0 1 01 1 101</td>
</tr>
<tr>
<td>30</td>
<td>+0</td>
<td>1</td>
<td>1 0 1 01 1 101</td>
</tr>
<tr>
<td>30</td>
<td>+2</td>
<td>1</td>
<td>1 0 1 01 1 110</td>
</tr>
<tr>
<td>40</td>
<td>+0</td>
<td>0</td>
<td>1 0 1 01 1 110</td>
</tr>
<tr>
<td>50</td>
<td>+0</td>
<td>1</td>
<td>1 0 1 01 1 110</td>
</tr>
<tr>
<td>50</td>
<td>+2</td>
<td>1</td>
<td>1 0 1 01 1 C00</td>
</tr>
<tr>
<td>60</td>
<td>+0</td>
<td>0</td>
<td>1 0 1 01 1 C00</td>
</tr>
<tr>
<td>70</td>
<td>+0</td>
<td>1</td>
<td>1 0 1 01 1 C00</td>
</tr>
<tr>
<td>70</td>
<td>+2</td>
<td>1</td>
<td>1 0 1 01 1 C01</td>
</tr>
<tr>
<td>80</td>
<td>+0</td>
<td>0</td>
<td>1 0 1 01 1 C00</td>
</tr>
<tr>
<td>90</td>
<td>+0</td>
<td>1</td>
<td>1 0 1 01 1 C00</td>
</tr>
<tr>
<td>90</td>
<td>+2</td>
<td>1</td>
<td>1 0 1 01 1 C00</td>
</tr>
<tr>
<td>100</td>
<td>+0</td>
<td>0</td>
<td>1 0 1 01 1 C00</td>
</tr>
<tr>
<td>110</td>
<td>+0</td>
<td>1</td>
<td>1 0 1 01 1 C00</td>
</tr>
<tr>
<td>110</td>
<td>+2</td>
<td>1</td>
<td>1 0 1 01 1 C01</td>
</tr>
<tr>
<td>120</td>
<td>+0</td>
<td>1 0 1 01 1 C01</td>
<td></td>
</tr>
<tr>
<td>130</td>
<td>+0</td>
<td>1 0 1 01 1 C01</td>
<td></td>
</tr>
<tr>
<td>130</td>
<td>+2</td>
<td>1 0 1 01 1 C01</td>
<td></td>
</tr>
<tr>
<td>140</td>
<td>+0</td>
<td>1 0 1 01 1 C01</td>
<td></td>
</tr>
<tr>
<td>150</td>
<td>+0</td>
<td>1 0 1 01 1 C01</td>
<td></td>
</tr>
<tr>
<td>150</td>
<td>+2</td>
<td>1 0 1 01 1 C01</td>
<td></td>
</tr>
<tr>
<td>150</td>
<td>+2</td>
<td>1 0 1 01 1 C01</td>
<td></td>
</tr>
<tr>
<td>160</td>
<td>+0</td>
<td>1 0 1 01 1 C01</td>
<td></td>
</tr>
<tr>
<td>170</td>
<td>+0</td>
<td>1 0 1 01 1 C01</td>
<td></td>
</tr>
<tr>
<td>170</td>
<td>+2</td>
<td>1 0 1 01 1 C01</td>
<td></td>
</tr>
<tr>
<td>180</td>
<td>+0</td>
<td>0 0 1 01 1 C01</td>
<td></td>
</tr>
<tr>
<td>190</td>
<td>+0</td>
<td>0 0 1 01 1 C01</td>
<td></td>
</tr>
<tr>
<td>190</td>
<td>+0</td>
<td>0 0 1 01 1 C01</td>
<td></td>
</tr>
<tr>
<td>190</td>
<td>+2</td>
<td>0 0 1 01 1 C01</td>
<td></td>
</tr>
<tr>
<td>200</td>
<td>+0</td>
<td>0 0 1 01 1 C01</td>
<td></td>
</tr>
<tr>
<td>210</td>
<td>+0</td>
<td>0 0 1 01 1 C01</td>
<td></td>
</tr>
<tr>
<td>210</td>
<td>+2</td>
<td>0 0 1 01 1 C01</td>
<td></td>
</tr>
<tr>
<td>220</td>
<td>+0</td>
<td>0 0 1 01 1 C01</td>
<td></td>
</tr>
<tr>
<td>230</td>
<td>+0</td>
<td>0 0 1 01 1 C01</td>
<td></td>
</tr>
<tr>
<td>240</td>
<td>+0</td>
<td>0 0 1 01 1 C01</td>
<td></td>
</tr>
<tr>
<td>250</td>
<td>+0</td>
<td>0 0 1 01 1 C01</td>
<td></td>
</tr>
</tbody>
</table>

Corresponding VCD test vector file

```plaintext
# Test vector file for modulo 7 counter
CODEFILE
UNITS ps
RISE_TIME 50
FALL_TIME 50
INPUTS clk, reset, load, count, i[2], i[1], i[0];
OUTPUTS q[2] (to=max), q[1] (to=max), q[0] (to=max);
CODING(BOM)
#
RADIX <1111333;#
#
80  <01105>X;
82000 <00105>0;
87000 <01105>0;
810000 <11105>5;
820000 <01015>5;
830000 <11015>6;
840000 <0115>6;
850000 <11015>0;
860000 <0115>0;
870000 <11015>1;
860000 <0115>1;
890000 <1115>2;
810000 <0115>2;
811000 <0115>3;
812000 <0115>3;
813000 <1115>4;
814000 <0115>4;
815000 <1115>5;
816000 <0115>5;
817000 <1115>6;
818000 <0115>6;
819000 <1115>0;
820000 <01103>0;
821000 <11103>3;
822000 <01103>3;
823000 <11103>3;
824000 <01103>3;
825000 <11103>3;
END
```
VCD test vector file
(clock generated separately by voltage source)

`vpulse vclk clk 0`
`pulse(0 3.3 10n .5n .5n 10n 20n)`

Can mix other simulation commands with test vector application.
Extract waveform characteristics

- `.EXTRACT TRAN [LABEL=name] [FILE=name] [VECT] [CATVECT]`

$MACRO | FUNCTION

- **TRAN**: transient analysis (only analysis mode supported)
- **LABEL=name**: label name for extraction result
- **FILE=name**: dump extraction results to file (and *.MT)
- **VECT**: extract all vectors that match (o/w first value only)
- **FUNCTION**: predefined functions
  - **EX_XMIN, EX_XMAX**: X values of function (o/w 1st time of transient)
  - **EX_YMIN/EX_YMAX**: min/max voltage
  - See timing functions on next page
EXTRACT: timing functions

- **TFALL(wave [,VH=val] [,VL=val] [,BEFORE=val] [,AFTER=val] [,OCCUR=num])**
  - Returns the fall time of the numth falling edge on wave.
  - BEFORE/AFTER define the acquisition window
  - **TRISE** (same format)

  - Returns the propagation delay between wave1 and wave2. VTHIN and VTHOUT are the transition thresholds of wave1 and wave2, respectively. If VTHIN ≡ VTHOUT, it is possible to specify only VTH.
  - **TPDUU** – same, but wave1 and wave 2 rising
  - **TPDUD, TPDDU, TPDDD** – same formats
Example

TPD (V(IN), V(OUT), OCCUR=1)
= 75n - 10n
= 65n

TPD (V(IN), V(OUT), OCCUR=2)
= 110n - 100n
= 10n

TPD (V(IN), V(OUT))
= (65n+10n)/2
= 37.5n
EXTRACT Examples

- `.EXTRACT TRAN LABEL=tpd1 VECT TPDUD (v(CLK), v(N_NX40_X_reg_Q_0_MN11_g))`
  - Propagation delay from rising edge of clock to the falling edge of a D flip flop (for determining max clock period)

- `.EXTRACT TRAN LABEL=tpd2 VECT TPDUU (v(CLK), v(N_NX40_X_reg_Q_0_MN11_g))`
  - Propagation delay from rising edge of clock to the rising edge of the same D flip flop (for determining max clock period)
Summary

- Simulation at each stage of ASIC design
  - behavioral model
  - synthesized netlist
  - pre-layout schematic/netlist
  - post-layout netlist

- **ADMS Package** combines 3 technologies to cover the above
  - digital (VHDL, Verilog)
  - analog/mixed-signal (VHDL-AMS, Verilog-A, ADiT)
  - transistor level (Eldo, ADiT)

- **ASIC Design Kit** (ADK) supports all tools in the design flow, including simulation (VHDL, Verilog, SPICE models)