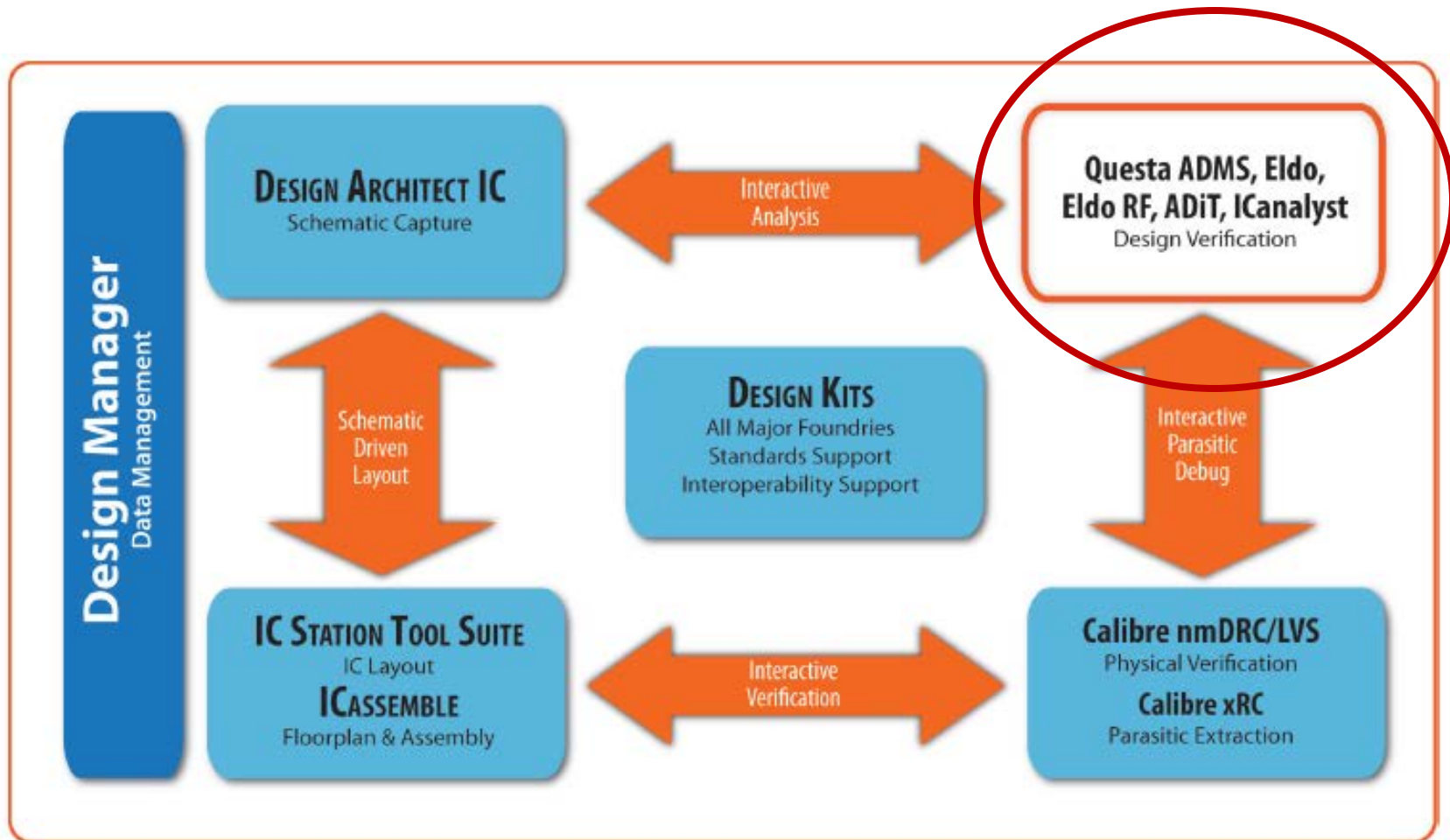
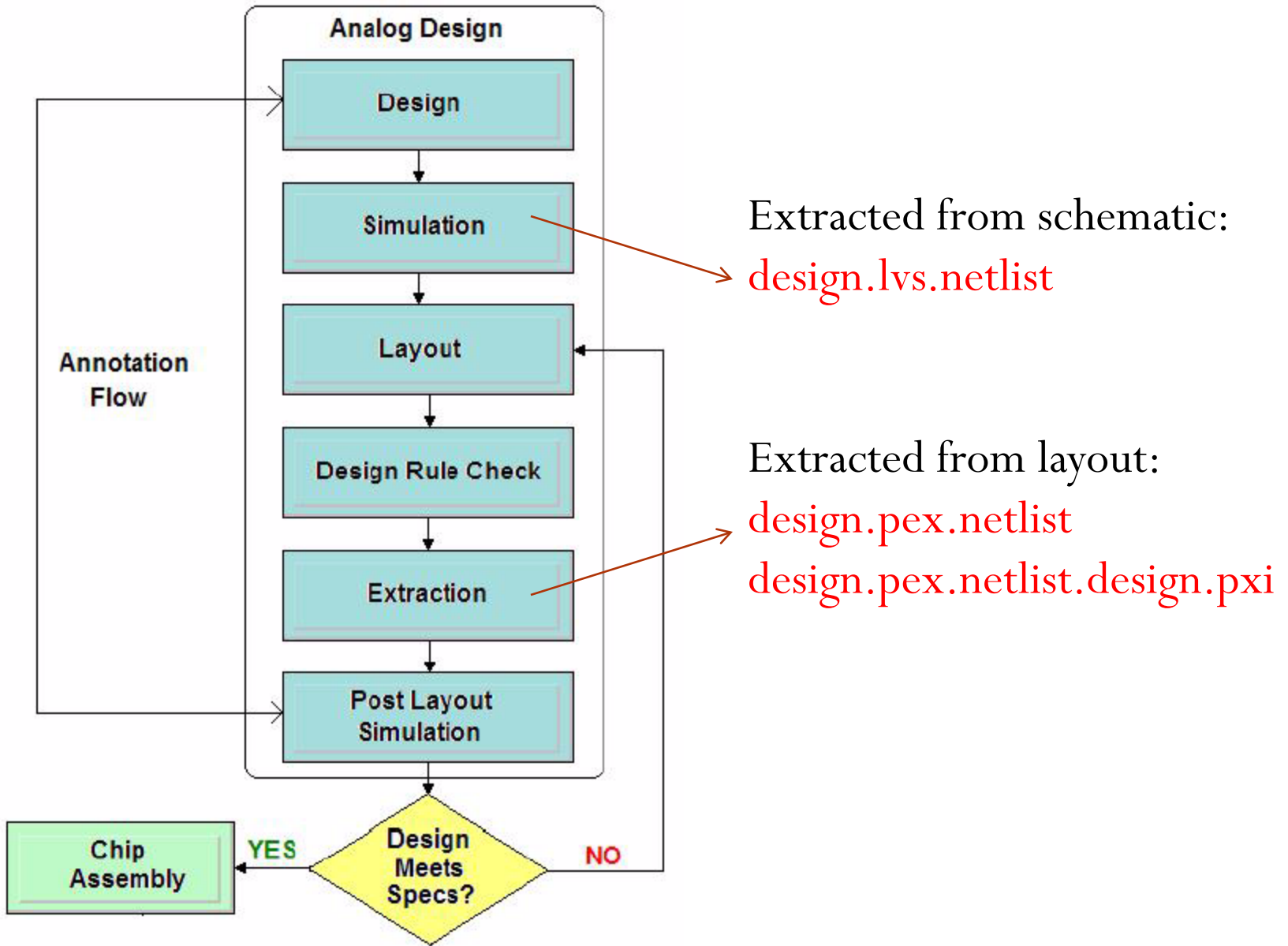


Post-Layout Simulation

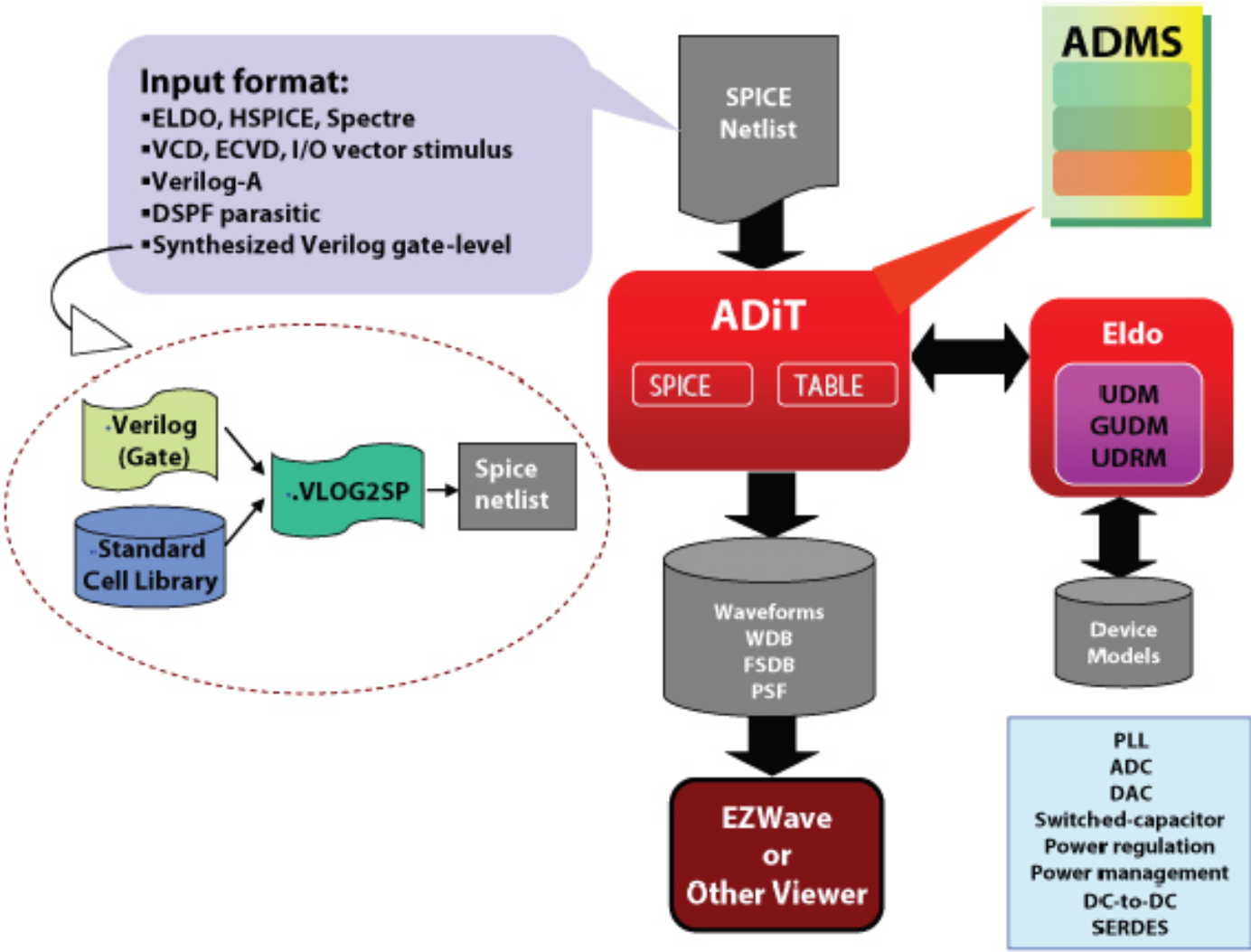
Analog and Digital Turbo Simulator (ADiT)

Analog & Digital Turbo Simulator (ADiT)





Analog & Digital Turbo Simulator (ADiT)



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 infile**

[-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

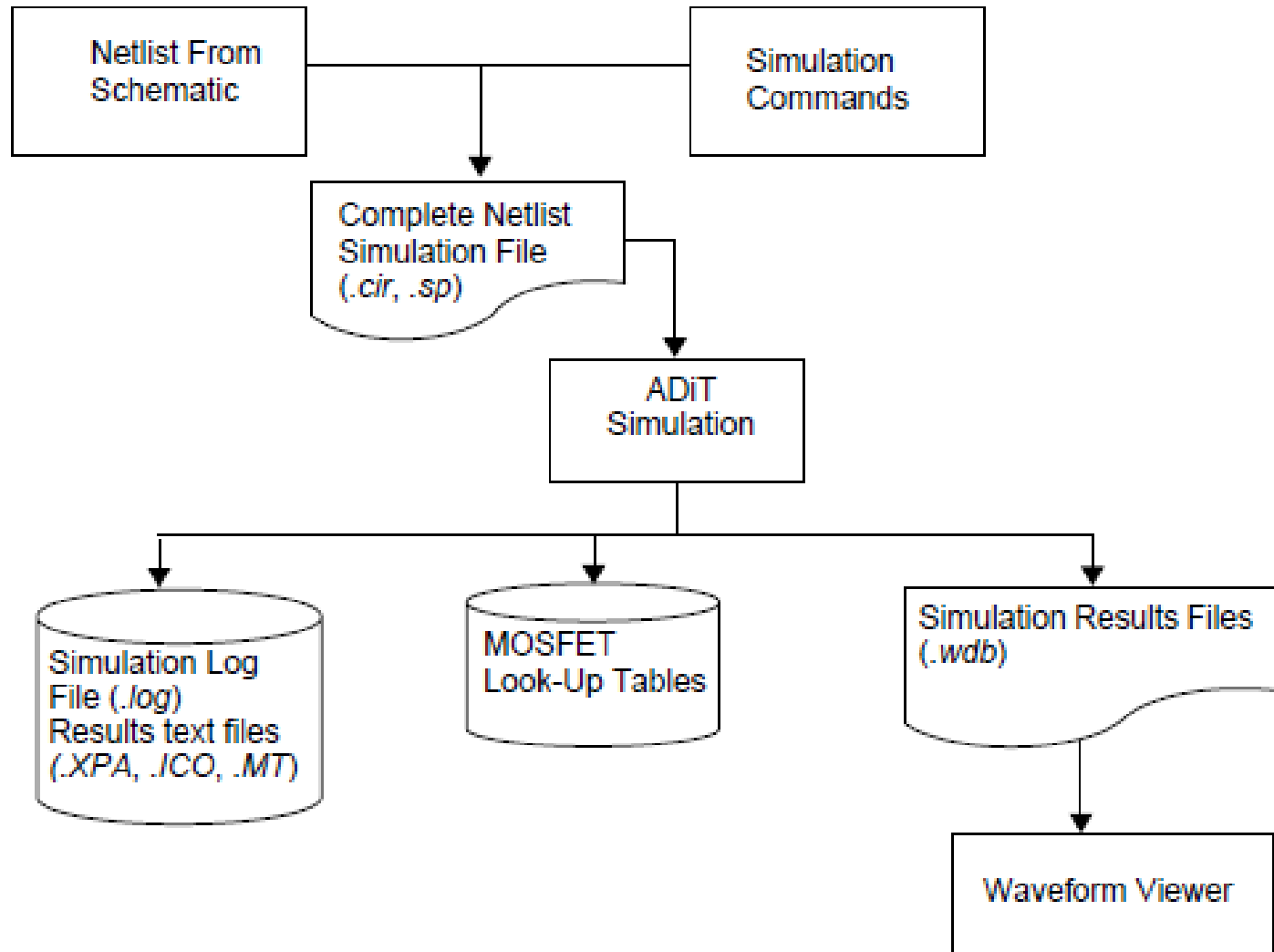
[-outpath path_name] – output directory

[-sckspice] – solve subcircuits with SPICE engine, disabling MOS table

[-spice] – enable SPICE engine (default is “turbo”)

[-v tool_name]

AdiT input & output files



ADiT input/output files

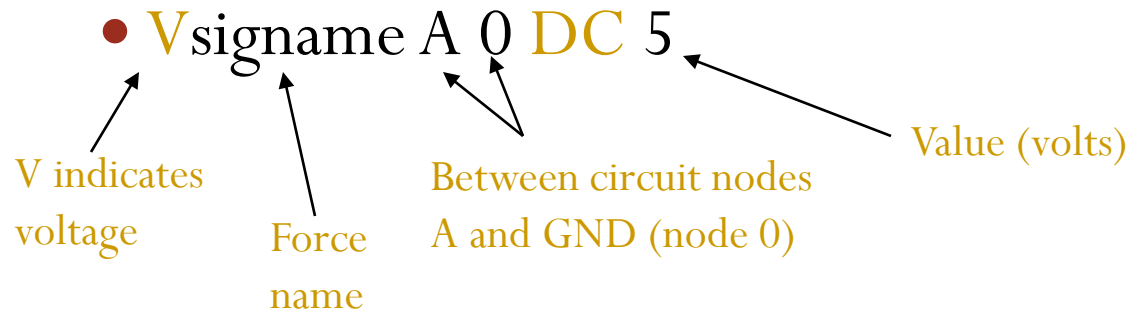
- `<input_name>.xxx` – ADiT input file
- `<input_name>.log` – output with messages, statistics, etc
- `<input_name>.IC0` – ckt state var's at $t=0$
- `<input_name>_0.wdb` – graphic: transient analysis
- `<input_name>.XPA` – expansion of subcircuits
- `<input_name>.PAR` – parameter evaluation
- `<input_name>.DCPOW0` – 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



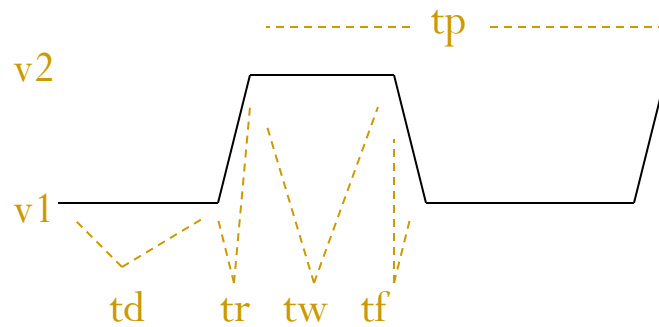
- Alternate format: `Vsignature A 0 5` (DC is default)

Force functions (2)

- Pulse/square wave

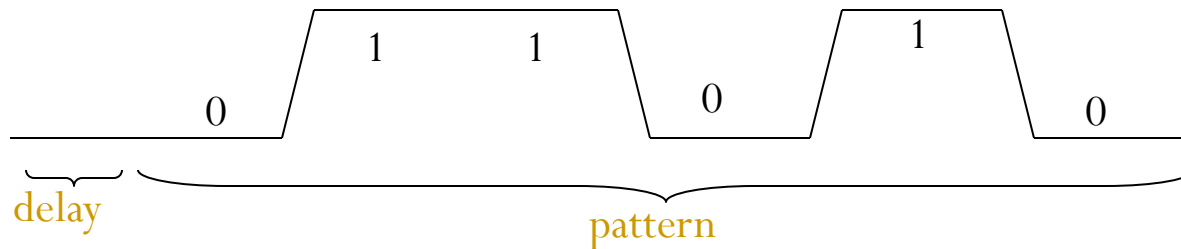
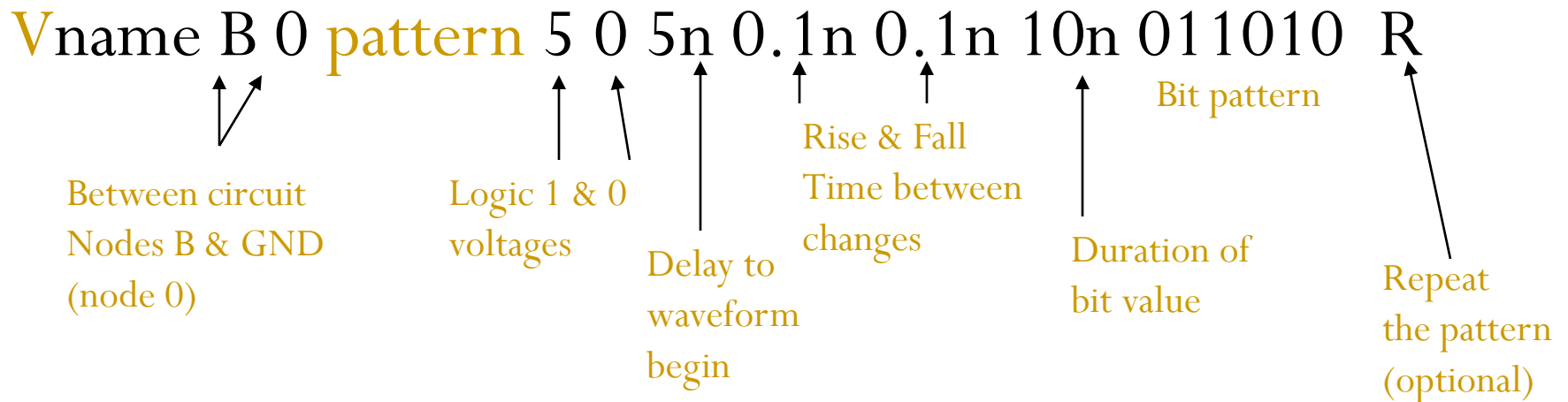
- `Vsigname B 0 pulse 0 5 0 0.1N 0.1N 20N 40N`

Nodes →
 Initial Voltage $v1$ →
 Pulsed Voltage $v2$ →
 Rise time t_r →
 Fall time t_f →
 Pulse width t_w →
 Period t_p →
 Delay from start of period for waveform to begin - t_d



Force functions (3)

- Pattern wave (for logic 0 & 1 values)



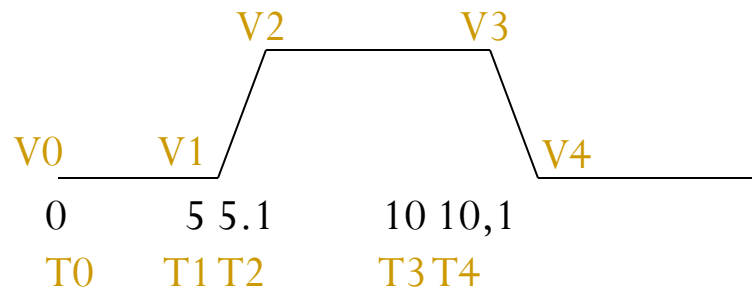
Force functions (4)

- Piecewise-linear wave (digital if only two voltages)

Vsignature B 0 pwl (0n 0 5n 0 5.1n 5 10n 5 10.1n 0 R)

Nodes

Repeat
(optional)



Other options: R=value (time at which to begin repeat – one of Tn values)

default = 0 if no value specified

TD=value (delay before waveform begins)

Example: modulo-7 counter

```
.INCLUDE $ADK/technology/ic/models/tsmc035.mod
```

```
.INCLUDE mod7b.pex.netlist
```

```
.TOPCELL MOD7B
```

```
vvdd VDD 0 dc 5
```

```
vgnv 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
```

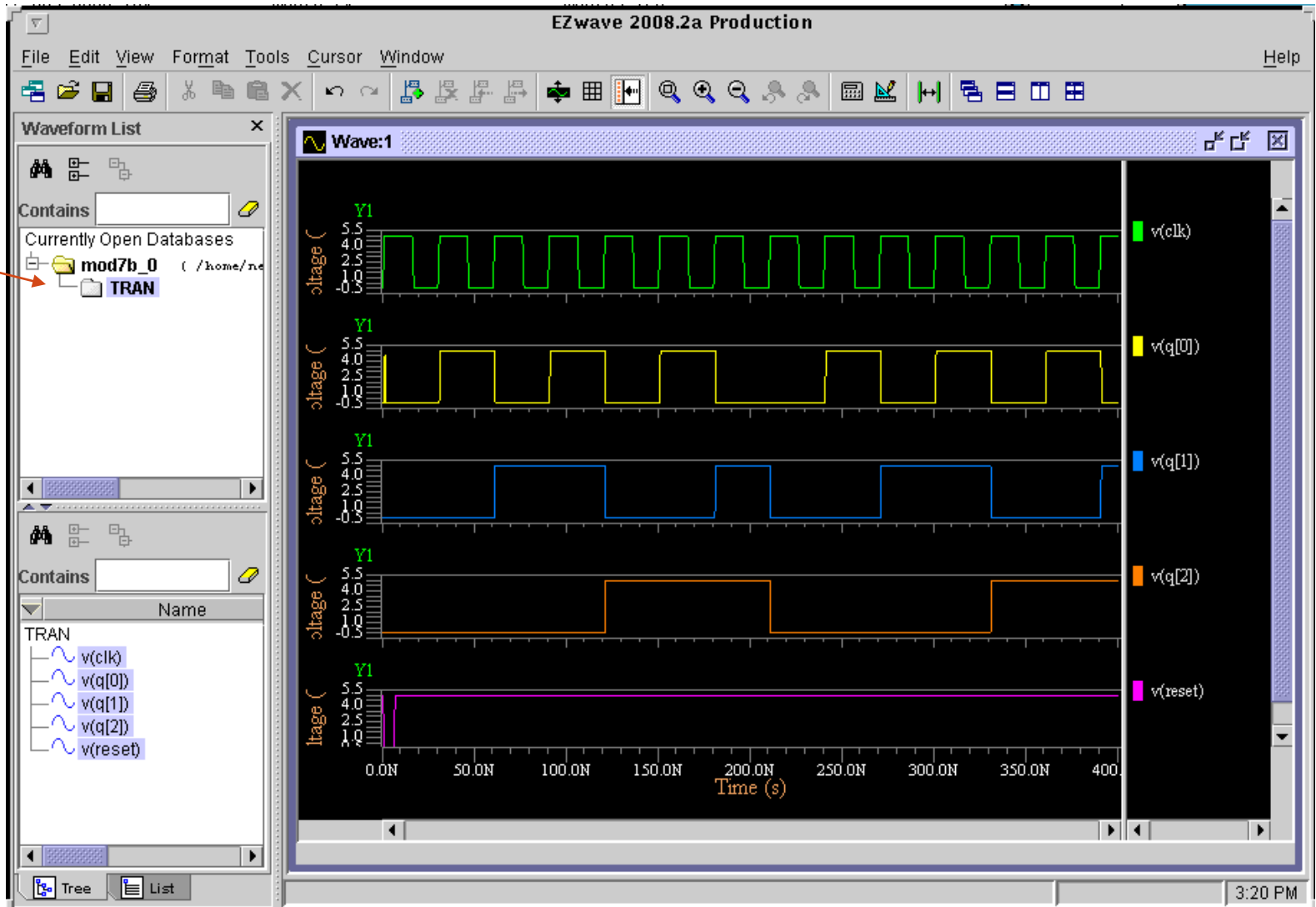
```
vreset reset 0 pulse (5 0 0 1n 1n 5n 0n )
```

```
vcount count 0 dc 5
```

```
.PROBE v(q[2]) v(q[1]) v(q[0]) v(reset) v(clk)
```

```
.TRAN 0.1n 400n tmax=0.5n
```

EZwave waveform viewer (results for previous circuit)



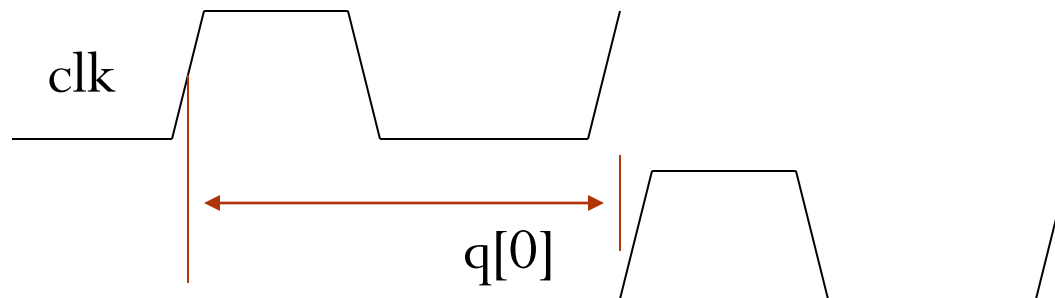
Making measurements

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:

```
.SETBUS inbus i[2] i[1] i[0]    (name and components)
```

- Plot bus values:

```
.PLOTBUS inbus VTH=2.5 BASE=HEX RADIX=UNSIGNED
```

```
.PLOTBUS inbus VTH1=1 VTH2=3 BASE=BIN RADIX=SIGNED
```

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

- Stimulate a bus:

```
.SIGBUS inbus VHI=5 VLO=0 BASE=DEC SIGNED=NONE 10ns 5
```

Specify one or more time-value pairs (apply value at specified time)

```
.SIGBUS inbus VHI=5 VLO=0 BASE=DEC SIGNED=NONE
```

```
+ TRISE=1n TFALL=1n THOLD=40n PATTERN 5 2 3 7
```

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
 - Alternate: d[[2:0]] => d[2] d[1] d[0]
- **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 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 – ifVOH/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	-- $V > 3v = \text{logic } 1$
vol 1	-- $V < 1v = \text{logic } 0$
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 s1 s2 s3 --time, values of signals 1 2 3 ...

T2 s1 s2 s3

- Tabular Data Format – with uniform time step

PERIOD 20 -- apply vectors at 0, 20, 40, 60, ...

s1 s2 s3 -- only signal values listed

s1 s2 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: $0 < V_{OL}$, $1 > V_{OH}$, 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    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

```
list.lst - WordPad
File Edit View Insert Format Help
|
fs          clk          i  q
delta      reset
          load
          count
0 +0      U U U U UUU UUU
0 +1      0 1 1 0 101 UUU
2 +0      0 0 1 0 101 UUU
2 +2      0 0 1 0 101 000
7 +0      0 1 1 0 101 000
10 +0     1 1 1 0 101 000
10 +2     1 1 1 0 101 101
20 +0     0 1 0 1 101 101
30 +0     1 1 0 1 101 101
30 +2     1 1 0 1 101 110
40 +0     0 1 0 1 101 110
50 +0     1 1 0 1 101 110
50 +2     1 1 0 1 101 000
60 +0     0 1 0 1 101 000
70 +0     1 1 0 1 101 000
70 +2     1 1 0 1 101 001
80 +0     0 1 0 1 101 001
90 +0     1 1 0 1 101 001
90 +2     1 1 0 1 101 010
100 +0    0 1 0 1 101 010
110 +0    1 1 0 1 101 010
110 +2    1 1 0 1 101 011
120 +0    0 1 0 1 101 011
130 +0    1 1 0 1 101 011
130 +2    1 1 0 1 101 100
140 +0    0 1 0 1 101 100
150 +0    1 1 0 1 101 100
150 +2    1 1 0 1 101 101
160 +0    0 1 0 1 101 101
170 +0    1 1 0 1 101 101
170 +2    1 1 0 1 101 110
180 +0    0 1 0 1 101 110
190 +0    1 1 0 1 011 110
190 +2    1 1 0 1 011 000
200 +0    0 1 1 0 011 000
210 +0    1 1 1 0 011 000
210 +2    1 1 1 0 011 011
220 +0    0 1 1 0 011 011
230 +0    1 1 1 0 011 011
240 +0    0 1 1 0 011 011
250 +0    1 1 1 0 011 011
```

Corresponding VCD test vector file

```
mod7b.tv - WordPad
File Edit View Insert Format Help
# Test vector file for modulo7 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(ROM)
#
RADIX <11113>3;
#
@0        <01105>X;
@2000    <00105>0;
@7000    <01105>0;
@10000   <11105>5;
@20000   <01015>5;
@30000   <11015>6;
@40000   <01015>6;
@50000   <11015>0;
@60000   <01015>0;
@70000   <11015>1;
@80000   <01015>1;
@90000   <11015>2;
@100000  <01015>2;
@110000  <11015>3;
@120000  <01015>3;
@130000  <11015>4;
@140000  <01015>4;
@150000  <11015>5;
@160000  <01015>5;
@170000  <11015>6;
@180000  <01015>6;
@190000  <11013>0;
@200000  <01103>0;
@210000  <11103>3;
@220000  <01103>3;
@230000  <11103>3;
@240000  <01103>3;
@250000  <11103>3;
END
```

VCD test vector file

(clock generated separately by voltage source)

```
m7.tv - WordPad
File Edit View Insert Format Help
# Test vector file for modulo7 counter
CODEFILE
UNITS ps
RISE_TIME 50
FALL_TIME 50
INPUTS reset, load, count, i[2], i[1], i[0];
OUTPUTS q[2] (to=max), q[1] (to=max), q[0]
(to=max);
CODING(ROM)
#
RADIX <1113>3;
#
@0 <1105>X;
@2000 <0105>0;
@7000 <1105>0;
@10000 <1105>5;
@20000 <1015>5;
@30000 <1015>6;
@50000 <1015>0;
@70000 <1015>1;
@90000 <1015>2;
@110000 <1015>3;
@130000 <1015>4;
@150000 <1015>5;
@170000 <1015>6;
@190000 <1013>0;
@200000 <1103>0;
@210000 <1103>3;
@230000 <1103>3;
@250000 <1103>3;
END
```



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.
 - ***TRISE*** (same format)
- ***TPD***(*wave1*, *wave2* [, *VTH=val*] [, *VTHIN=val*] [, *VTHOUT=val*] [, *BEFORE=val*] [, *AFTER=val*] [, *OCCUR=num*])
 - Returns the propagation delay between wave1 and wave2. *VTHIN* and *VTHOUT* are the transition thresholds of wave1 and wave2, respectively. If $VTHIN \equiv VTHOUT$, it is possible to specify only *VTH*.
 - ***TPDUU*** – same, but wave1 and wave 2 rising
 - ***TPDUD***, ***TPDDU***, ***TPDDD*** – same formats

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)

Linux .bashrc for Mentor Graphics tools

- # Set up Modelsim

```
MODELTECH=/linux_apps/mentor/modelsim/6.5b/modeltech
```

```
MGLS_LICENSE_FILE=27011@theon.eng.auburn.edu
```

```
export MODELTECH MGLS_LICENSE_FILE
```

```
PATH=$MODELTECH/linux:$PATH
```

```
export PATH
```

```
# Set up ASIC Design Kit
```

```
export ADK=/linux_apps/ADK3.1
```

```
PATH=$ADK/bin:$PATH
```

```
# Set up Mentor Graphics IC Flow and AMS Simulation
```

```
export PATH=/linux_apps/mentor/icflow/2008.1_rhelx86linux/icflow_home/bin:$PATH
```

```
export MGC_HOME=/linux_apps/mentor/icflow/2008.1_rhelx86linux/icflow_home
```

```
export MGC_LOCATION_MAP=/linux_apps/mentor/mgc_location_map
```

```
export MGC_AMS_HOME=/linux_apps/mentor/Eldo/2009.1
```

```
export PATH=/linux_apps/mentor/Eldo/2009.1/bin:$PATH
```

```
export MGC_WD=/home/nelson/nelsovp/ELEC5250_6250
```

```
export ADK_TECH=tsmc025
```