VHDL/Verilog Simulation

Using Mentor Graphics Modelsim SE

VHDL/Verilog Simulation Tools

- Mentor Graphics "Modelsim PE" Student Edition: free download for academic course work:

http://model.com/content/modelsim-pe-student-edition-hdl-simulation

  - Full version in ECE PC labs: Broun 308 and 310
  - Full version on College of Engineering Linux Servers.
    - Refer to ELEC 5250/6250 course web site:

    http://www.eng.auburn.edu/~nelsovp/courses/elec5250_6250/
    - Use sample .bashrc file provided under "useful CAD links" to set system environment/paths
  - Additional information on the ELEC 4200 web page:

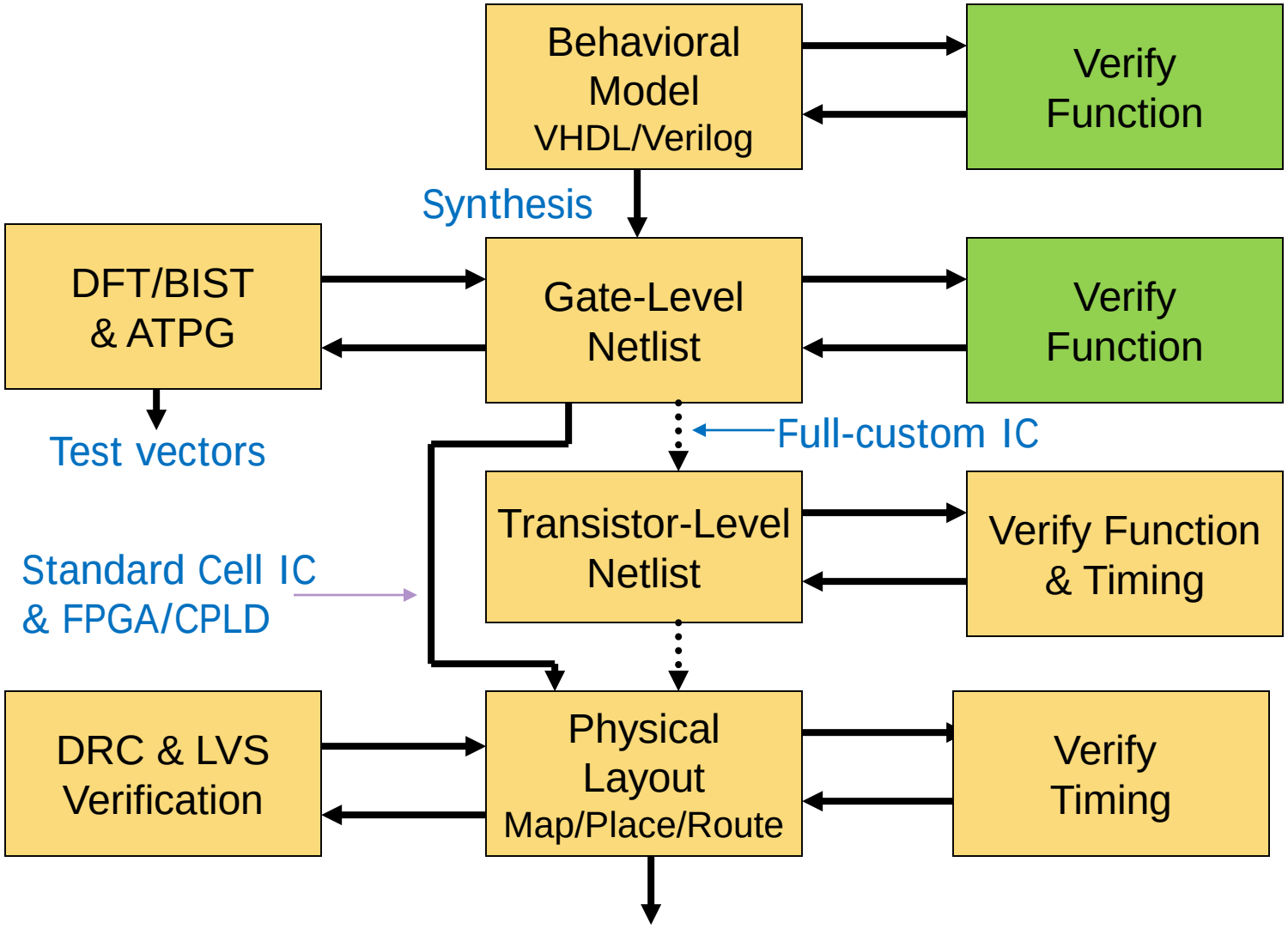
    http://www.eng.auburn.edu/~nelsovp/courses/elec4200/elec4200.html

- Aldec "Active-HDL" Student Edition: free download at:

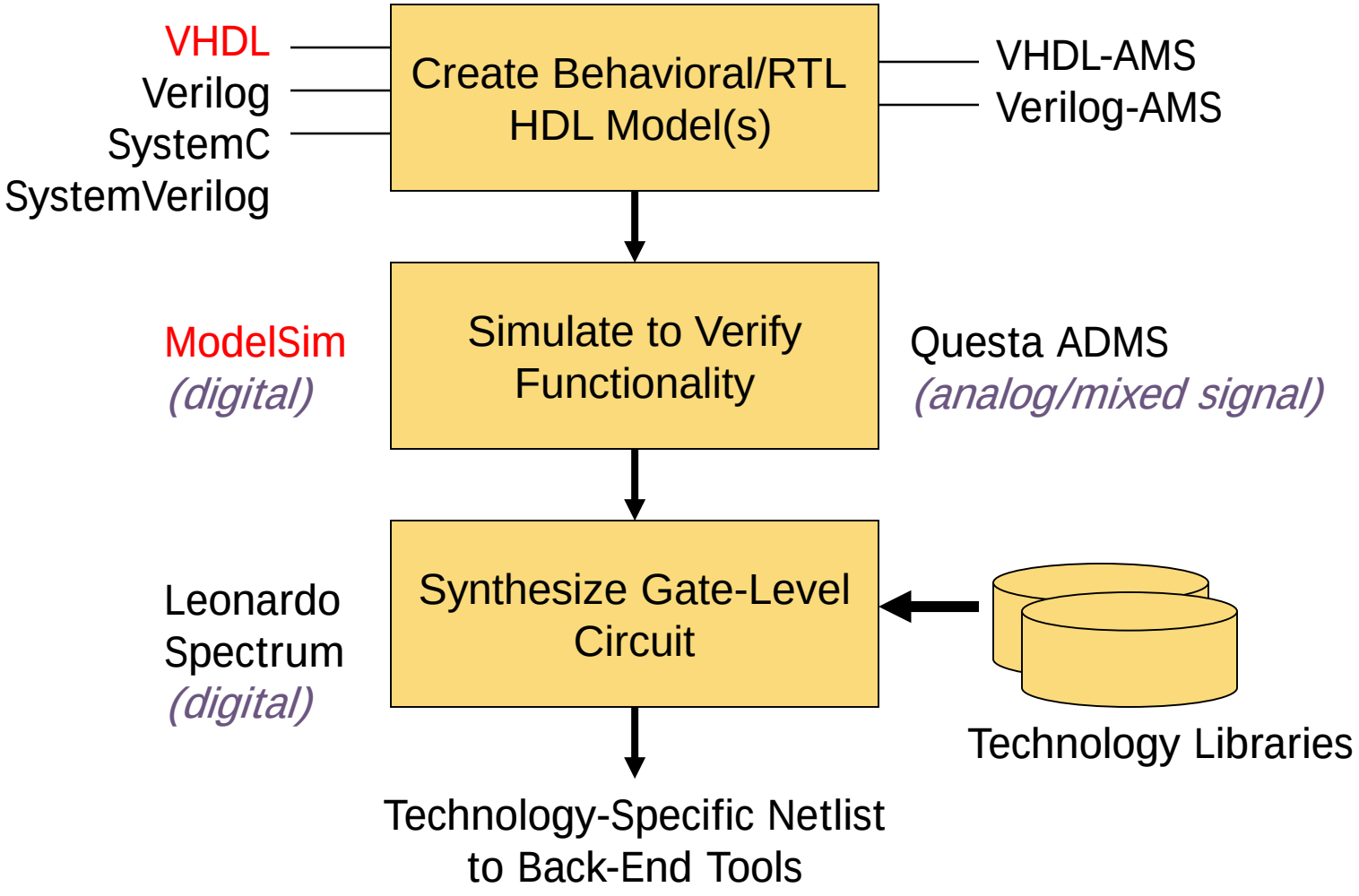
http://www.aldec.com/en/products/fpga_simulation/active_hdl_student

  - Full version installed in ELEC 4200 lab (Broun 320)

ASIC Design Flow



Behavioral Design & Verification
(mostly technology-independent)



Project simulations

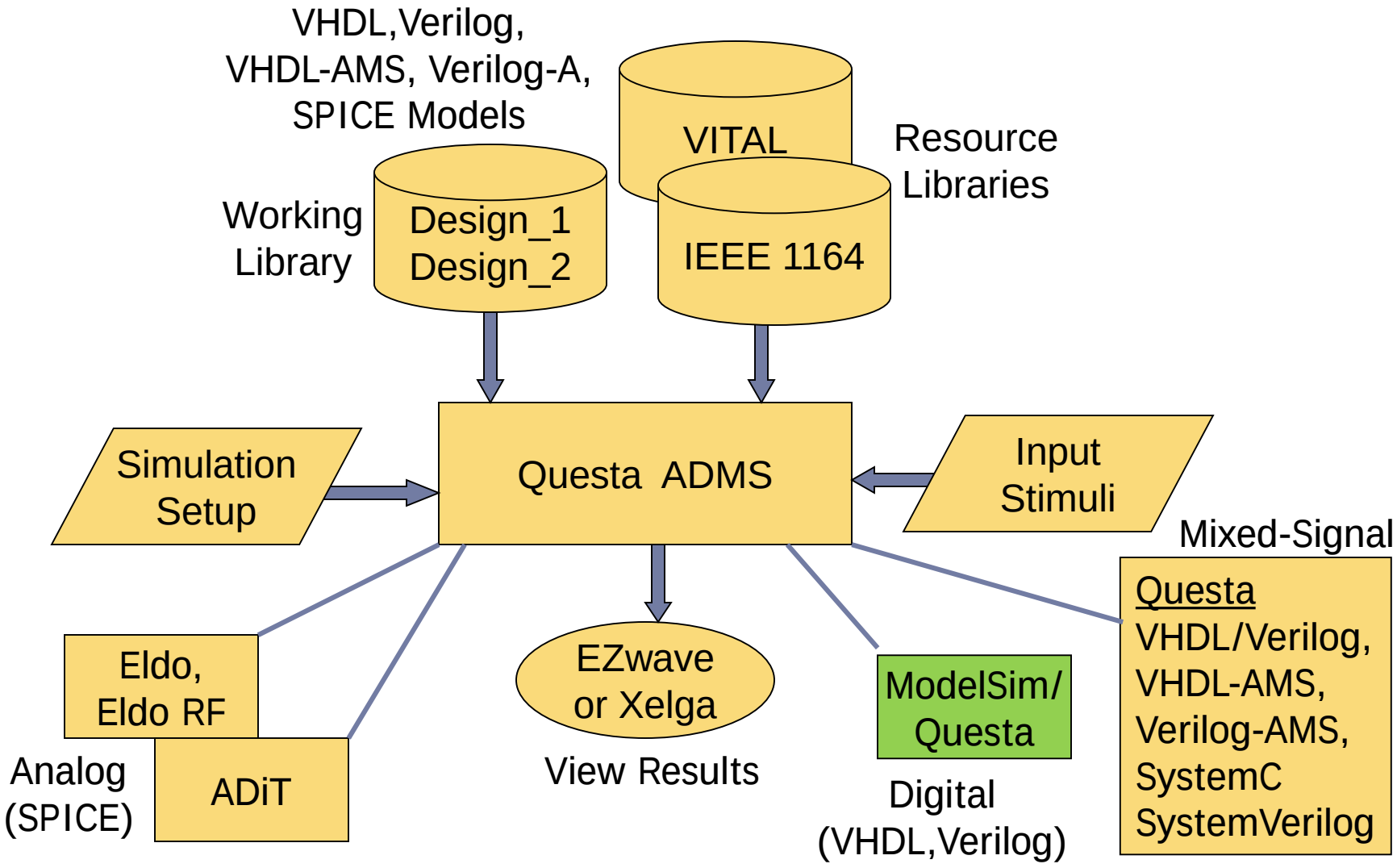
1. Behavioral/RTL – verify functionality
   - Model in VHDL/Verilog
   - Drive with "force file" or testbench
2. Post-Synthesis
   - Synthesized gate-level VHDL/Verilog netlist
   - Technology-specific VHDL/Verilog gate-level models
   - Optional SDF file (from synthesis) for timing
   - Drive with same force file/testbench as in (1)
3. Post-Layout
   - Netlist back-annotated with extracted capacitances for accurate delays

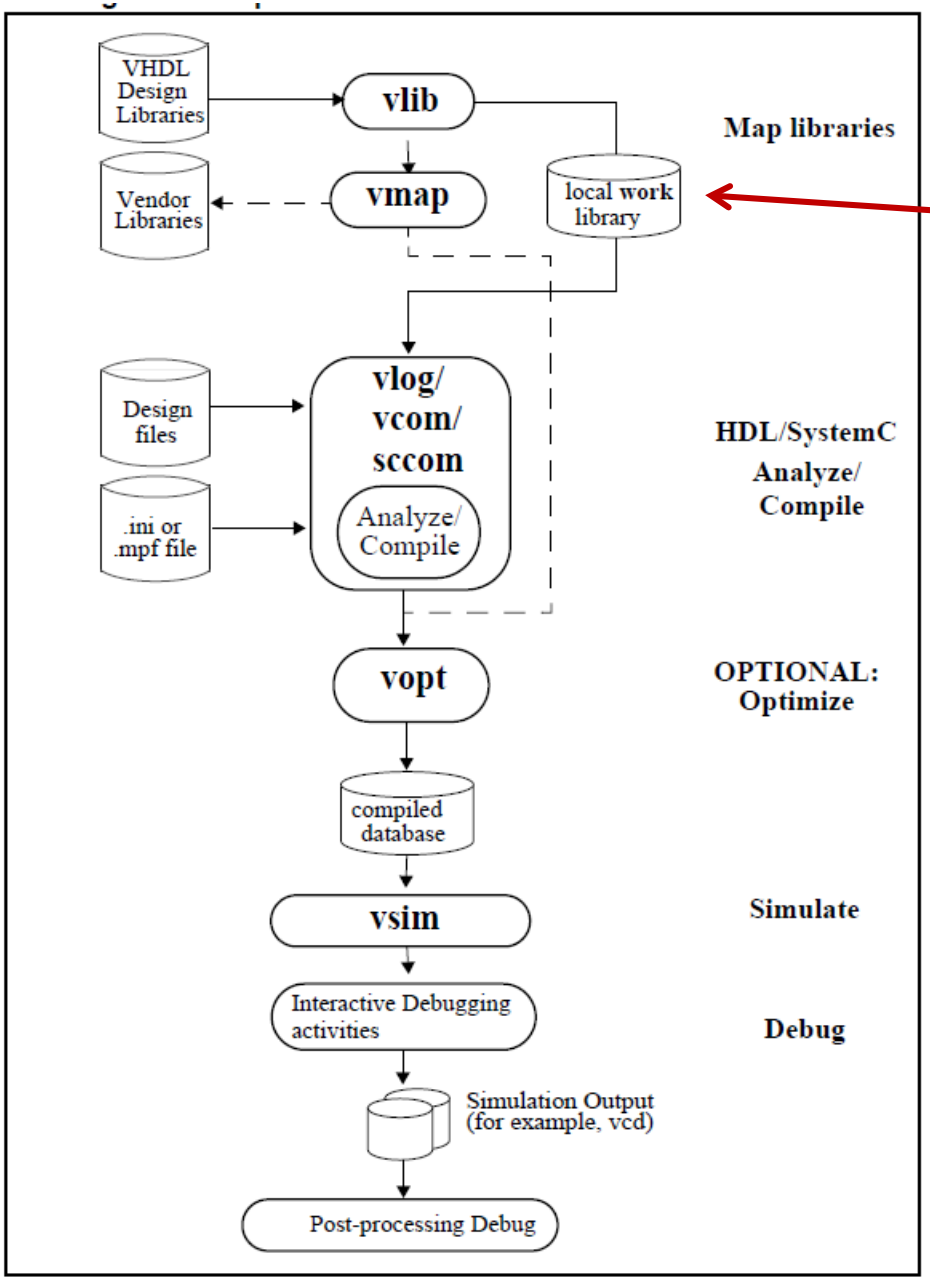
Questa ADMS

- Analog-Digital Mixed-Signal Simulator
- Language neutral – mix any supported languages within a model
- Four simulation engines:
  - Questa
    - Digital: VHDL/Verilog (ModelSim/Questa Sim)
    - Analog/mixed signal: VHDL-AMS/Verilog-AMS (Questa)
  - Eldo – General purpose analog (SPICE)
  - ADiT - Fast transistor level (SPICE)
  - Eldo RF
- Invoke stand-alone or from Design Architect-IC

- PC and Student versions are "Modelsim PE"

Digital/Mixed-Signal Simulation

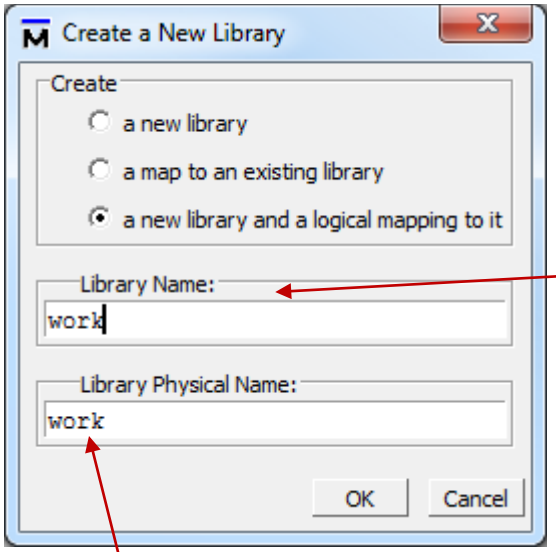




Modelsim
Operational Structure and Flow

Create working library *work* and map logical name *work* to that physical directory:

- *Command line:*
  - *> vlib work*
  - *> vmap work work*
- *From Modelsim menu:*
  - *File > New > Library*



Logical name

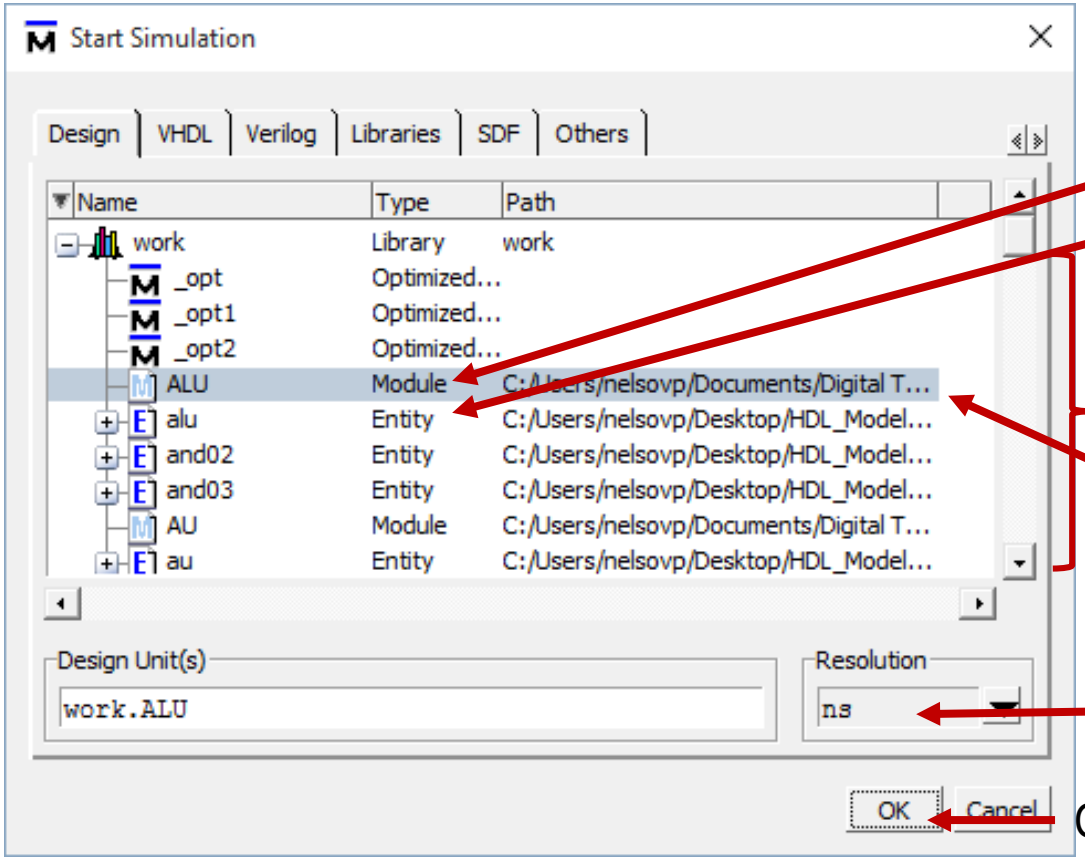
Physical name can be anything.
Enter full path name if not current directory.

Compile Verilog/VHDL model(s)

- Compiled models are placed in the working library
  - From linux or Modelsim command line:
    - *vlog count4.v* *(Verilog model)*
    - *vcom count4.vhd* *(VHDL model)*
  - From Modelsim menu:
    - *Select:* *Compile > Compile*
    - *Select file in Navigator window and click Compile*
  - To place the compiled model in a different library:
    - Use compile option *-work lib-name*

Simulate a VHDL model

- From Modelsim menu select: *Simulate > Start Simulation*



Verilog module
VHDL entity

Expand *work* library

Select top-level model (design unit)

Select time resolution for simulation

Click OK to start simulation

- OR – in Modelsim Transcript window enter: *vsim work.count4*

Example: 4-bit binary counter

- VHDL model *(count4.vhd)*
  - Create working library: *vlib work*
  - Map name "work": *vmap work work*
  - Compile: *vcom count4.vhd*
  - Simulate: *vsim count4(rtl)*
- ModelSim simulation-control inputs
  - ModelSim "Macro" file *(count4.do)*
  - Testbench (VHDL or Verilog)
- ModelSim results
  - *List (table)* and/or *Waveform (logic analyzer)*

-- count4.vhd 4-bit parallel-load synchronous counter

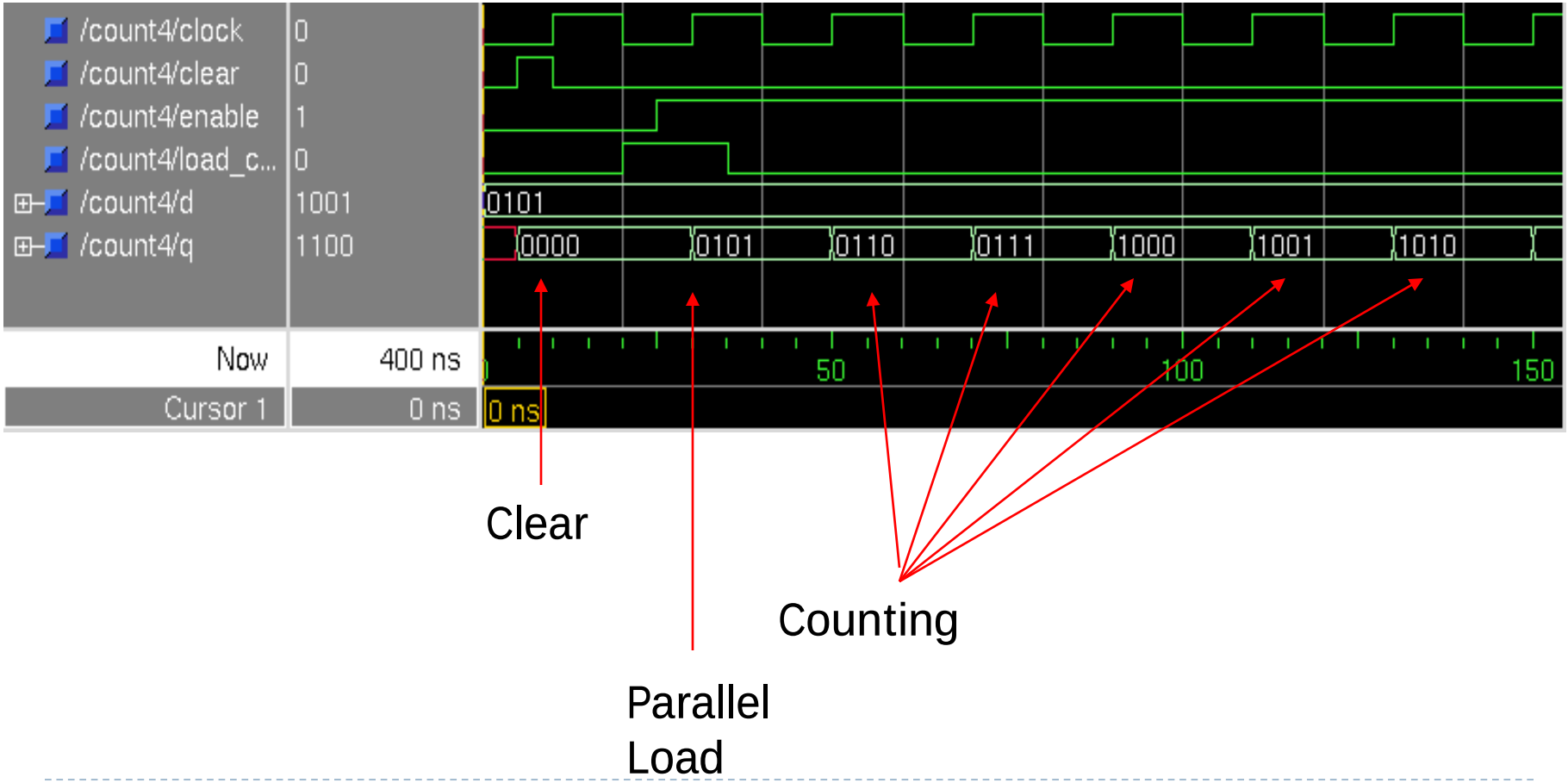
```
LIBRARY ieee;
USE ieee.std_logic_1164.all;  USE ieee.numeric_std.all;
ENTITY count4 IS
  PORT (clock,clear,enable,load_count : IN STD_LOGIC;
        D: IN  STD_LOGIC _VECTOR(3 downto 0);
        Q: OUT STD_LOGIC _VECTOR (3 downto 0);
END count4;
ARCHITECTURE rtl OF count4 IS
    SIGNAL CNTint : UNSIGNED(3 downto 0);  -- counter internal state
BEGIN
  PROCESS(clear, clock)    -- synchronous load/count, asynchronous clear
  BEGIN
    IF (clear = '1') THEN
        CNTint <= "0000";
    ELSIF (clock'EVENT AND clock='1') THEN
        IF (enable = '1') THEN
          IF (load_count = '1') THEN
              CNTint <= UNSIGNED(D);
          ELSE
              CNTint <= CNTint + 1;
          END IF;
        END IF;
    END IF;
  END PROCESS;
  Q <= STD_LOGIC _VECTOR (CNTint);
END rtl;
```

Modelsim TCL "macro" file: *count4.do*

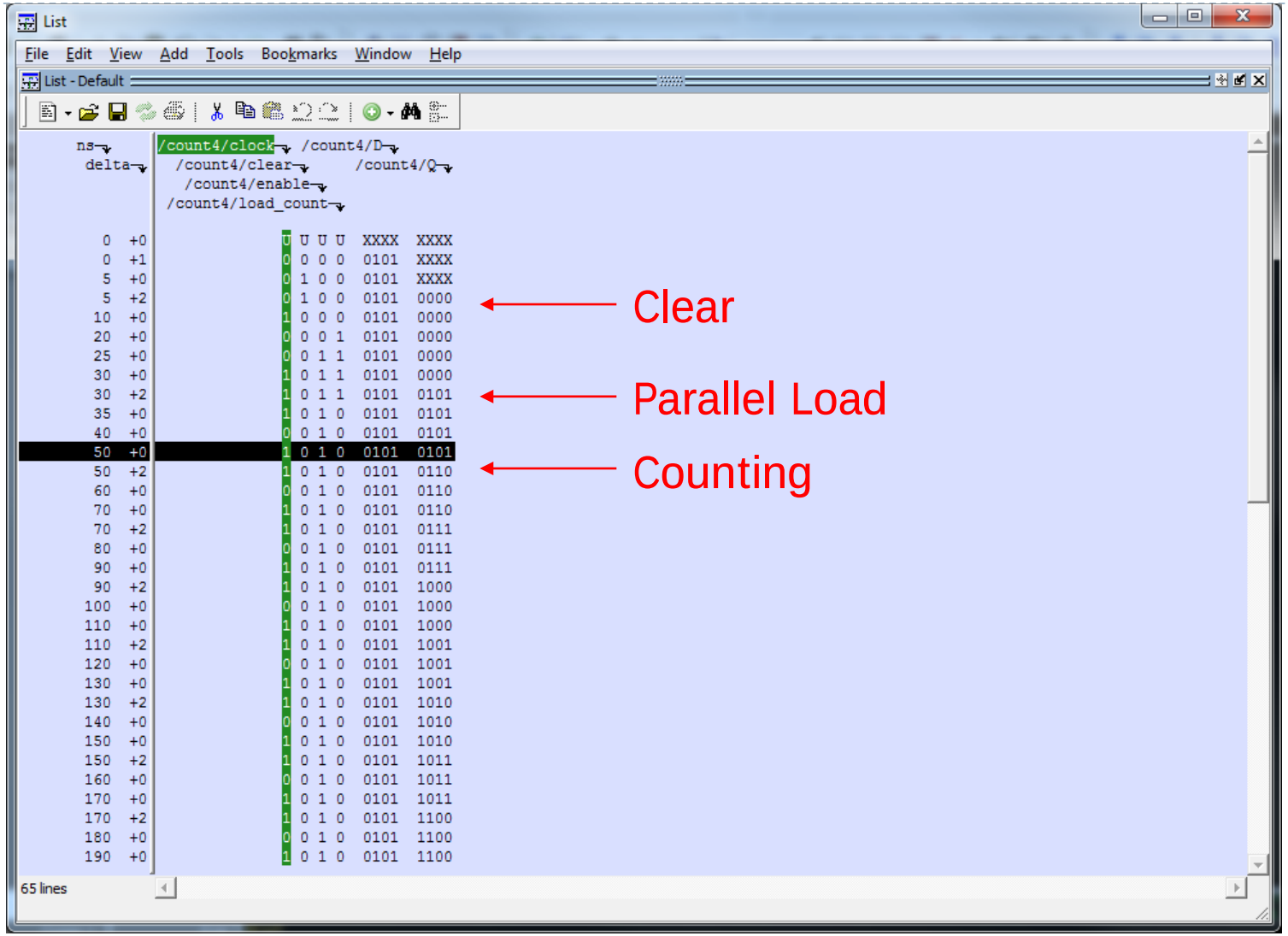
```
add wave /clock /clear /enable /load_count
add wave –hex /D /Q
add list /clock /clear /enable /load_count
add list –hex /D /Q
force /clock 0 0, 1 10 -repeat 20
force /clear 0 0, 1 5, 0 10
force /enable 0 0, 1 25
force /load_count 0 0, 1 20, 0 35, 1 330, 0 350
force /D 10#5 0, 10#9 300
run 400
```

* To execute the macro from the Modelsim Transcript window, enter:
  do count4.do
* To execute from the Modelsim menu, select:
  Tools > Tcl > Execute Macro (select file *count4.do* in the navigator window)
* Individual commands can also be entered in the Modelsim Transcript window.

Count4 – Simulation waveform



Count4 – Simulation list window



Post-synthesis simulation (1)

- Compile the ADK standard cell library
  1. Create an adk library: *vlib adk*
  2. Compile the cell models:
     vcom $ADK/technology/adk.vhd –work adk
  3. Compile the ADK components declaration package:
     vcom $ADK/technology/adk_comp.vhd –work adk
  4. Map logical name "adk" to the physical directory:
     vmap adk adk

Post-synthesis simulation (2)

- Edit the synthesized VHDL netlist file produced by Leonardo (count4_0.vhd) to include the ADK component declarations package:

```
-- Definition of  count4
--      Thu Sep 21 10:48:09 2006
--      LeonardoSpectrum Level 3, 2005a.82
--
library IEEE;
use IEEE.STD_LOGIC_1164.all;
library adk;                    -- add these two lines
use adk.adk_components.all;

entity count4 is
   port (…..
```

Post-synthesis simulation (3)

- Compile the VHDL netlist:

  vcom count4_0.vhd

- Simulate the model:

  vsim count4 –do count4.do

  (count4.do is the "macro file" used earlier for the behavioral model)

- Verify that the synthesized circuit produces the same results as the behavioral circuit

Leonardo-synthesized netlist count4_0.vhd

```
library IEEE; use IEEE.STD_LOGIC_1164.all;
library adk;  use adk.adk_components.all;     -- ADDED BY VPN
entity count4 is
   port (
      clock : IN std_logic ;  clear : IN std_logic ;  enable : IN std_logic ;  load_count : IN std_logic ;
      D : IN std_logic_vector (3 DOWNTO 0) ;  Q : OUT std_logic_vector (3 DOWNTO 0)) ;
end count4 ;

architecture netlist of count4 is       -- rtl changed to netlist by VPN
   signal Q_3_EXMPLR, Q_2_EXMPLR, Q_1_EXMPLR, Q_0_EXMPLR, nx8, nx14, nx22,
          nx28, nx48, nx54, nx62, nx126, nx136, nx146, nx156, nx169, nx181,
          nx183, nx185, nx187, nx189: std_logic ;
```

(continue next slide)

Leonardo-synthesized netlist count4_0.vhd

```
begin
  Q(3) <= Q_3_EXMPLR ; Q(2) <= Q_2_EXMPLR ; Q(1) <= Q_1_EXMPLR ;  Q(0) <= Q_0_EXMPLR ;
  Q_0_EXMPLR_EXMPLR : dffr port map ( Q=>Q_0_EXMPLR, QB=>OPEN, D=>nx126,  CLK=>clock, R=>clear);
  ix127 : mux21_ni port map ( Y=>nx126,A0=>Q_0_EXMPLR,A1=>nx8, S0=>enable   );
  ix9 :    oai21 port map ( Y=>nx8,A0=>load_count,A1=>Q_0_EXMPLR, B0=>nx169   );
  ix170 : nand02 port map ( Y=>nx169,A0=>D(0),A1=>load_count);
  Q_1_EXMPLR_EXMPLR : dffr port map ( Q=>Q_1_EXMPLR, QB=>OPEN, D=>nx136,      CLK=>clock, R=>clear);
  ix137 : mux21_ni port map ( Y=>nx136,A0=>Q_1_EXMPLR,A1=>nx28, S0=>      enable);
  ix29 :  ao22 port map ( Y=>nx28,A0=>D(1),A1=>load_count, B0=>nx14, B1=>      nx22);
  ix15 :  or02 port map ( Y=>nx14,A0=>Q_0_EXMPLR,A1=>Q_1_EXMPLR);
  ix23 :  aoi21 port map ( Y=>nx22,A0=>Q_1_EXMPLR,A1=>Q_0_EXMPLR, B0=> load_count);
  Q_2_EXMPLR_EXMPLR : dffr port map ( Q=>Q_2_EXMPLR, QB=>OPEN, D=>nx146,  CLK=>clock, R=>clear);
  ix147 : mux21_ni port map ( Y=>nx146,A0=>Q_2_EXMPLR,A1=>nx48, S0=> enable);
  ix49 :  oai21 port map ( Y=>nx48,A0=>nx181,A1=>nx183, B0=>nx189);
  ix182 : aoi21 port map ( Y=>nx181,A0=>Q_1_EXMPLR,A1=>Q_0_EXMPLR, B0=> Q_2_EXMPLR);
  ix184 : nand02 port map ( Y=>nx183,A0=>nx185,A1=>nx187);
  ix186 : inv01 port map ( Y=>nx185,A=>load_count);
  ix188 : nand03 port map ( Y=>nx187,A0=>Q_2_EXMPLR,A1=>Q_1_EXMPLR,A2=> Q_0_EXMPLR);
  ix190 : nand02 port map ( Y=>nx189,A0=>D(2),A1=>load_count);
  Q_3_EXMPLR_EXMPLR : dffr port map ( Q=>Q_3_EXMPLR, QB=>OPEN, D=>nx156, CLK=>clock, R=>clear);
  ix157 : mux21_ni port map ( Y=>nx156,A0=>Q_3_EXMPLR,A1=>nx62, S0=> enable);
  ix63 :  mux21_ni port map ( Y=>nx62,A0=>nx54,A1=>D(3), S0=>load_count);
  ix55 :  xnor2 port map ( Y=>nx54,A0=>Q_3_EXMPLR,A1=>nx187);
end netlist ;
```

```
// Verilog description for cell count4,  LeonardoSpectrum Level 3, 2005a.82

module count4 ( clock, clear, enable, load_count, D, Q ) ;
    input clock ;
    input clear ;
    input enable ;
    input load_count ;
    input [3:0]D ;
    output [3:0]Q ;

    wire nx8, nx14, nx22, nx28, nx48, nx54, nx62, nx126, nx136, nx146;
    wire nx156, nx169, nx181, nx183, nx185, nx187, nx189;
    wire [3:0] \$dummy ;
```

(continue next slide)

// Verilog description for cell count4, LeonardoSpectrum Level 3, 2005a.82

```
    dffr Q_0__rename_rename (.Q (Q[0]), .QB (\$dummy [0]), .D (nx126), .CLK (clock), .R (clear)) ;
    mux21_ni ix127 (.Y (nx126), .A0 (Q[0]), .A1 (nx8), .S0 (enable)) ;
    oai21 ix9 (.Y (nx8), .A0 (load_count), .A1 (Q[0]), .B0 (nx169)) ;
    nand02 ix170 (.Y (nx169), .A0 (D[0]), .A1 (load_count)) ;
    dffr Q_1__rename_rename (.Q (Q[1]), .QB (\$dummy [1]), .D (nx136), .CLK (clock), .R (clear)) ;
    mux21_ni ix137 (.Y (nx136), .A0 (Q[1]), .A1 (nx28), .S0 (enable)) ;
    ao22 ix29 (.Y (nx28), .A0 (D[1]), .A1 (load_count), .B0 (nx14), .B1 (nx22) ) ;
    or02 ix15 (.Y (nx14), .A0 (Q[0]), .A1 (Q[1])) ;
    aoi21 ix23 (.Y (nx22), .A0 (Q[1]), .A1 (Q[0]), .B0 (load_count)) ;
    dffr Q_2__rename_rename (.Q (Q[2]), .QB (\$dummy [2]), .D (nx146), .CLK (clock), .R (clear)) ;
    mux21_ni ix147 (.Y (nx146), .A0 (Q[2]), .A1 (nx48), .S0 (enable)) ;
    oai21 ix49 (.Y (nx48), .A0 (nx181), .A1 (nx183), .B0 (nx189)) ;
    aoi21 ix182 (.Y (nx181), .A0 (Q[1]), .A1 (Q[0]), .B0 (Q[2])) ;
    nand02 ix184 (.Y (nx183), .A0 (nx185), .A1 (nx187)) ;
    inv01 ix186 (.Y (nx185), .A (load_count)) ;
    nand03 ix188 (.Y (nx187), .A0 (Q[2]), .A1 (Q[1]), .A2 (Q[0])) ;
    nand02 ix190 (.Y (nx189), .A0 (D[2]), .A1 (load_count)) ;
    dffr Q_3__rename_rename (.Q (Q[3]), .QB (\$dummy [3]), .D (nx156), .CLK (clock), .R (clear)) ;
    mux21_ni ix157 (.Y (nx156), .A0 (Q[3]), .A1 (nx62), .S0 (enable)) ;
    mux21_ni ix63 (.Y (nx62), .A0 (nx54), .A1 (D[3]), .S0 (load_count)) ;
    xnor2 ix55 (.Y (nx54), .A0 (Q[3]), .A1 (nx187)) ;
endmodule
```