**Description Phase**: Specification, Planning, Design Entry, Functional Test. **Implementation** **Phase**: Synthesis, Post-Synthesis Test, APR Parasitic Extraction & timing checks, Manufacture & Hardware Validation

HDL: 9x testing/validating

Enable Large Design, Portable Design, Explore Large solution space, Better Validated Designs, portable, self-documenting, describe multiple levels of abstraction, represent parallelism, provides many descriptive styles, Serve as input for synthesis tools.

can be compiled to semi-custom and programmable hardware implementations

Full 
Custom 
Manual 
VLSI 
Semi- 
Custom 
Standard 
Cell 
Gate 
Array 
Programmable 
PLD 
less work faster time to market 
implementation efficiencv 

Hardware Building Blocks

Transistors are switches

Use multiple transistors to make a gate

Use multiple gates to make a circuit

Standard Cells

Library of common gates and structures (cells)

Decompose hardware in terms of these cells

Arrange the cells on the chip

Connect them using metal wiring

FPGAs

“Programmable” hardware

Use small memories as truth tables of functions

Decompose circuit into these blocks

Connect using programmable routing

SRAM bits control functionality

Netlist

an ASCII text representation of the interconnect of a schematic

Port: can be bidirectional

Module Styles

Structural – connect primitives and modules

Dataflow– use continuous assignments

Behavioral – use initial and always blocks

A single module can use more than one method.

Structural

A schematic in text form (i.e. A netlist)

Build up a circuit from gates/flip-flops

Gates are primitives (part of the language)

Flip-flops themselves described behaviorally

оит 
оит 
оит 
оит 
оит 
оит 
оит 
А 
О 
о 
О 
О 
в 
О 
1 
х 
z 
А 
О 
о 
О 
О 
в 
О 
1 
х 
z 
о 
о 
О 
О 
1 
1 
о 
1 
х 
z 
о 
о 
Т 
z 
z 
о 
О 
х 
х 
в 
z 
х 
1 
1 
z 
о 
оит 
10 х х 



D 
invl 
inv2 
not 
invl (Q,nl) 
not (weak0,weak1) inv2 (n1,Q) 

reg [7:0] mem[0:99][0:3]; // 2D array of 8 bits

typedef enum reg [1:0] { IDLE, CONV, ACCM } state\_t;

state\_t state, nxt\_state;  
$display(“At time %t count = %h”,$time,cnt);

$stop //Stops simulation and allows you to still probe signals and debug

$finish //completely stops simulation, simulator relinquishes control of thread.

Empty input ports => high impedance state (z)

Empty output ports => output not used

Inertial Delay (Gates)

Suppresses pulses shorter than delay amount In reality, gates need to have inputs held a certain time before output is accurate

Transport Delay (Nets)

“Time of flight” from source to sink

计算机生成了可选文字:
out
rIse
whattheoutput
Ofthegateis
doing
not#（2，1）invl(out,in);
out
//notgatewith2unitriseand1unitfall
out
bufifl#（1，1，2）tril(out,in,en),
TurnOffdelay
Timefromcntrlto
0什
//tri-stategatewithunitrise/falland2unitOff

and #(1:2:3) g1(out,a,b); // 1 ns min, 2ns typical, 3ns max delay

and #(2:3:4, 1:2:3) g1(out,a,b); // gate has different rise,fall for min:typ:max

$monitor

Output the given values whenever one changes; Can use when simulating Structural, RTL, and/or Behavioral

$display, $strobe

Output specific information like a printf in a C program; Used in Behavioral Verilog

Dataflow Verilog

The continuous assign statement

It is the main construct of Dataflow Verilog

It is deceptively powerful & useful

assign [drive\_strength] [delay] list\_of\_net\_assignments;

list\_of\_net\_assignment ::= net\_assignment [{,net\_assignment}]

Net\_assignment ::= net\_lvalue = expression

===, !== test x’s, z’s! ONLY USE FOR testbenches

Bitwise Operators (&, |, ^, ~)



Reduction XOR

assign parity = ^data\_out; // even parity bit

When wire is used but not declared, it is implied

Latches with Continuous Assign

assign q\_out = enable ? data\_in : q\_out;

Behavioral Verilog

**initial** and **always** form basis of all behavioral Verilog

All <LHS> assignments **must be to type reg** (or logic if using SV)

Behavioral: Combinational vs Sequential

**Combinational**

Not edge-triggered

All “inputs” (RHS nets/variables) are triggers

Does not depend on clock

**Sequential**

Edge-triggered by clock signal

Only clock (and possibly reset) appear in trigger list

Can include combinational logic that feeds a FF or registe

**Blocking** “Evaluated” sequentially =

Used for combinational logic

The evaluation of subsequent statements <RHS> are blocked, until the <LHS> assignment of the current statement is completed.

**Non-Blocking Assignments** <=

“Updated” simultaneously if no delays given

Used for sequential logic

**casez**

Uses z and ? as “don’t care” bits in case items and expression

**casex**

Uses x, z, and ? as “don’t care” bits in case items and expression

**Timing**

**Tsetup**: Time D must be stable BEFORE clock edge

■ Adds to critical path delay

Clk->Q: Time from clock edge to Q changing

■ Adds to critical path delay

**Thold**: Time D must be stable AFTER clock edge

■ Sets minimum path from Q of one DFF to D of another

**Cost(mapping)** = -Slack(mapping) + Area(mapping) + [power(mapping)]

**input delay** is specified as time after the clock edge (of prior clock cycle) that the input to the DUT is valid.

**Optimization Priorities**

Design rules have priority over timing goals

Timing goals have priority over area goals

Max delay has priority over min-delay

To prioritize area constraints:

use the ignore\_tns (total negative slack) option when you specify the area constraint:

set\_max\_area -ignore\_tns 10000

**Top-Down Compilation**

Use top-down compile strategy for medium to small designs

Read in the entire design

Resolve multiple instances of any design references with uniquify

Apply attributes and constraints to the top level

Compile the design using compile or compile\_ultra

**Bottom-Up Compile Strategy**

Compile the subdesigns separately and then incorporate them

Top-level constraints are applied and the design is checked for violations.

Advantages:

Compiles large designs more quickly (divide-and-conquer)

Requires less memory than top-down compile

Disadvantages

Need to develop local constraints as well as global constraints

May need to repeat process several times to meet design goals

**Compile-once-don’t-touch Method**

uses the set\_dont\_touch command to preserve the compiled subdesign

**Ungroup** Method

The ungroup command makes unique copies of the design and removes levels of the hierarchy

**report\_design**

display characteristics of the current design

operating conditions, wire load model, output delays, etc.

parameters used by the design

**report\_area**

displays area information for the current design

number of nets, ports, cells, references

area of combinational logic, non-combinational, interconnect, total

**report\_hierarchy**

displays the reference hierarchy of the current design

tells modules/cells used and the libraries they come from

**report\_timing**

reports timing information about the design

default shows one worst case delay path

**report\_resources**

Lists the resources and datapath blocks used by the current design

**repeat Loop**

only evaluated when the loops starts

If it changes during loop execution it won’t change the number of iterations

**forever loops**

Only a **$stop**, **$finish** or a specific **disable** can end a forever loop

Sequential vs Parallel è (begin/end) vs (fork/join)

**begin/end** are used to form compound sequential statements.

**fork/join** are used to form compound parallel statements.

Statements in a parallel block are executed simultaneously.

All delay or event based control is relative to when the block was entered

**Named Blocks**

Blocks (**begin/end**) or (**fork/join**) can be named

Local variables can be declared for the named block

Variables in a named block can be accessed using hierarchical naming reference

Named blocks can be disabled (i.e. execution stopped)

**disable** Statement

Disables execution of the current block (not permanently)

由于Verilog不支持循环的continue和break，因此通过将最外层block命名为break，将循环里面的block命名为continue，通过disable continue和 disable break达成和C中continue与break同样的效果。

@ (negedge 
clk) ; 
rst n = 1 
@ (negedge 
clk) ; 
send cmd 
@ (negedge 
clk) • 
send cmd 
fork 
beg in 
. timeout 1 
// De—assert reset to DUT 
// Master sends command via UART 
// This block will error out after 70k clocks 
repeat ( 
70000 
) @ (posedge Clk) • 
$display ("ERROR: timed out waiting for transmission to complete" 
$stop() ; 
end 
beg in 
@ (posedge cmd rdy) ; 
disable timeout 1 ; 
end 
orn 
// This block waits for cmd rdy 
// Cancels timeout as soon as cmd rdy occurs 

**Assertions** (only in System Verilog)

**assert** (*true\_condition*) *pass\_statement*

**else** *fail\_statement*

**assert** (result == expected) $**display**(“self check passed”)

**else** $**fatal**(“ERR: at time %t, result not same as expected”,$**time**);

$**fatal** è Throws a fatal message to output, exits the simulator (like a $finish).

$**error** è Throws a error message to output, continues simulation.

Assertions…immediate vs concurrent

The assertion condition is **evaluated as the statement is encountered** in the test bench flow.

Concurrent assertion

allows you to define conditions that should always be true, and are checked at all times during simulation

/// check that rd & wrt are never both asserted ///

/// This will be checked at every simulation tick ///

**assert property** (!(rd && wrt))

**else** $**error**(“ERROR: both rd and wrt asserted to SRAM”);

key word property distinguishes a concurrent assertion from an immediate assertion.

Most concurrent assertions would be checked on clock ticks.

**assert property** (@(**posedge** clk) req |-> ##[1:2] ack);

A ## followed by a number or range specifies the delay from the current clock tick to the beginning of the sequence that follows.

## is often used with a range. For example: req|-> ##[0:3] gnt would mean gnt should be asserted 0 to 3 clock cycles after req.

// / [Sequence Layer/ /// 
req gnt seq; 
sequence 
& gnt) ##1 & 
////Property Layer/ /// 
property req_gnt_prop; 
@ (posedge cXk) 
disable (!rst n) 
////Directive Layer//// 
assert property 
// clock right after 
// should be low and gar should 
// be high. Next clock both low 
// is used for clock ticks 
// will not check when resetting 
/ 1" upon the sequence reg_gnd_seq 
// specified above should occur 
else $display ("ERR: req_gnt assertion failure at time 8t", $time) ; 

$fopen opens a file and returns an integer descriptor

integer fd = $fopen(“filename”);

integer fd = $fopen(“filename”, “r”);

If file cannot be open, returns a 0

Can output to more than one file simultaneously by writing to the OR ( | ) of the relevant file descriptors

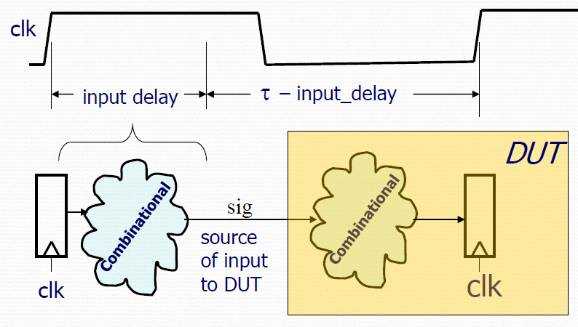
$fclose closes the file

$fclose(fd);

Writing To Files

Output statements have file equivalents

$fmonitor(), $fdisplay(), $fstrobe()

$fwrite()        // write is like a display without the \n

These system calls take the file descriptor as the first argument

$fdisplay(fd, “out=%b in=%b”, out, in);

Reading From Files

Read a binary file: $fread(destination, fd);

Can specify start address & number of locations too

Very rich file manipulation (see IEEE Standard)

$fseek(), $fflush(), $ftell(), $rewind(), …

Using $fgetc to read characters

reg signed [15:0] wide\_char;

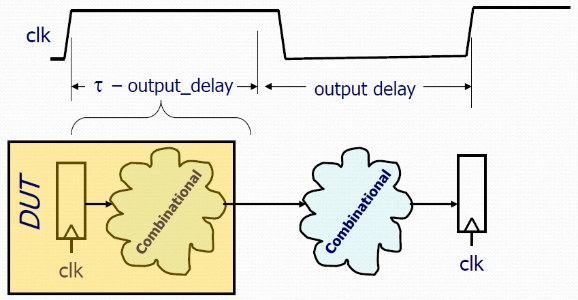
wide\_char = $fgetc(file\_handle);

Error

integer file\_handle,error;

reg [639:0] err\_str;

error = $ferror(file\_handle,err\_str);

Using $fgets to read lines

$fgets() returns the number of bytes in the line. When this is a zero you know you hit EOF.

integer file\_handle,error,indx,num\_bytes\_in\_line;

reg [256\*8:1] mem[0:255],line\_buffer;

reg [639:0] err\_str;

num\_bytes\_in\_line = $fgets(line\_buffer,file\_handle);

Using $fscanf to read files

integer file\_handle,error,indx,num\_matches;

reg [15:0] mem[0:255][0:1];

reg [639:0] err\_str;

num\_matches = $fscanf(file\_handle,”%h %h”,mem[indx][0],mem[indx][1]);

Loading Memory Data From Files

$readmemb(“<file\_name>”,<memory>);

$readmemb(“<file\_name>”,<memory>,<start\_addr>,<finish\_addr>);

$readmemh(“<file\_name>”,<memory>);

$readmemh(“<file\_name>”,<memory>,<start\_addr>,<finish\_addr>);

$readmemh Hex data…$readmemb binary data

**Functions**

Declared and referenced within a module

Used to implement combinational behavior

Contain no timing controls or tasks

Inputs/outputs

Must have at least one input argument

Has only one output (no inouts)

Function name is implicitly declared return variable

function [7: 0] aligned\_word;         // function declaration



**Tasks** can modify global signals too

Task <taskname>; input…; output …; endtask

Tasks provide the ability to

Execute common procedures from **multiple** places

**Divide** large procedures into smaller ones

Include Compiler Directives

`include “adder8.v” // include the tasks for adder8

**A generate-loop** permits making one or more instantiations (pre-synthesis) using a for-loop.

Only allowable uses of **x** is as “don’t care”

Only allowable use of z:

Constructs implying a 3-state output

**Min\_delay\_slack = clk2q – THold – Skew\_Between\_clks**

1.) Use clock tree synthesis (CTS) within the APR tool to balance clock network

异步信号使用分开always block获得更小的面积

Translation (Elaboration)

Unrolls loops, substitutes macros & parameters, computes constant functions, evaluates generate conditionals

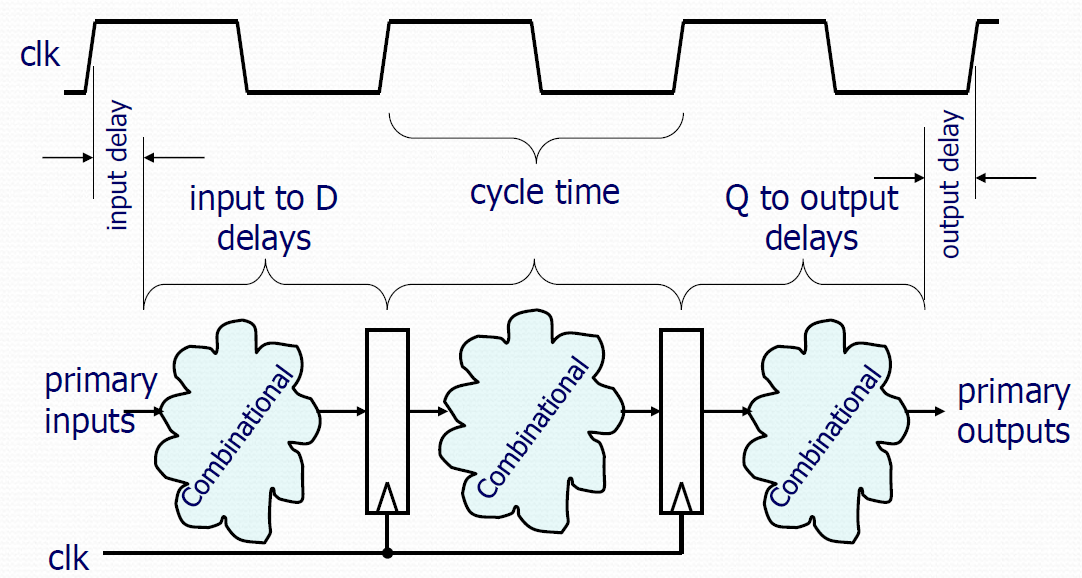
Builds a structural representation of the design

set compile\_seqmap\_enable\_output\_inversion true

Allows the mapping of sequential elements in the design to library cells whose outputs are inverted.

The register **retiming** tool moves registers through the synthesized combinational logic network to improve timing and/or area

APR: Automatic Places & Route 布线用的



**max path**  The worst case (maximum) amount of time it takes to propagate from an input of a combinational block of logic to the output.

**Clock skew** There is uncertainty (a +/- margin) in a clock’s arrival time.

**input delay** is specified as time after the clock edge (of prior clock cycle) that the input to the DUT is valid.

output delay is specified as time prior to next rising edge that the output must be valid.

**MaxDelaySlack** = ClockPeriod – clk2q – TMAX – TSU – clkSkew

**MinDelaySlack** = clk2q + TMIN – THLD – clkSkew

read\_file -format verilog {./source/ss\_a2d\_dp.v}

# Define clock and set don't mess with it #

create\_clock -name "clk" -period 20 -waveform { 0 10 } { clk }

set\_dont\_touch\_network [find port clk]

# setup pointer that contains all inputs except clock #

set prim\_inputs [remove\_from\_collection [all\_inputs] [find port clk]]

# Set input delay & drive on all inputs #

set\_input\_delay -clock clk 5 [copy\_collection $prim\_inputs]

set\_driving\_cell -lib\_cell AO33D0BWP -pin Z -from\_pin A1 -library \

tcbn40lpbwptc [copy\_collection $prim\_inputs]

# tell it por\_n strongly driven so it won't buffer

set\_drive 0.1 por\_n

# Set output delay & load on all outputs #

set\_output\_delay -clock clk 5 [all\_outputs]

set\_load 0.10 [all\_outputs]

# Wire load model allows it to estimate internal parasitics #

set\_wire\_load\_model -name TSMC32K\_Lowk\_Conservative \

-library tcbn40lpbwptc

# Max transition time is important for Hot-E reasons #

set\_max\_transition 0.10 [current\_design]

# Now actually synthesize for 1st time #

compile -map\_effort low

check\_design

## design ware component caused extra pins

report\_area

# Take a look at max & min timings #

report\_timing -path full -delay max -nworst 3

report\_timing -path full -delay min -nworst 3

## smash the hierarchy (design ware component)

ungroup –all

compile -map\_effort medium

check\_design

report\_area

#### write out final netlist ######

write –format verilog ss\_a2d\_dp –output ss\_a2d\_dp.vg

# Sets maximum area to 20,000 area units (square microns)

set\_max\_area 20000

#Sets maximum dynamic power to 10 mW (default unit for our library is mW)

set\_max\_dynamic\_power 10

**ignore\_tns** (total negative slack) option specify the area constraint:

**set\_max\_area -ignore\_tns 10000**

change priorities use **set\_cost\_priority**

set\_cost\_priority -delay

constraints: max\_transition, max\_fanout, max\_capacitance, cell\_degradation, max\_delay, min\_delay, power, area

compile options include:

-map\_effort low | medium | high (default is medium)

-area\_effort low | medium | high (default same as map\_effort)

-incremental\_mapping (may improve already-mapped)

-verify (compares initial and synthesized designs)

-ungroup\_all (collapses all levels of design hierarchy)

The **ungroup** command makes unique copies of the design and removes levels of the hierarchy

**check\_design** command to verify design consistency.

**check\_timing** to locate potential timing problems

**report\_contraints –all\_violators** (check everything)

**Uniquify** Treats every instance of a common sub module individually, so it can be optimized to its own constraints/context**.**

**Ungroup** Smashes (flattens) levels of hierarchy so optimizations can be made across boundaries.

**Minimum clk period** = Clk2q + Tmax + Tsetup

**Min delay slack** = Clk2q + Tmin – Thold

**Input delay constraint** set\_input\_delay -clock clk 1.2 Sig\_out

**Delay to input** = Clk2q + Tmax

**Average clock insertion delay** = (mindelay + maxdelay)/2