Training Verilog basic (7 day Fulltime)

Day 1:

* Hierarchical Modeling Concepts:

Top-down vs Bottom-up;

Module, Instance;

Design vs Stimulus Block: initial statement, always statement.

* Logic gate: and, or, xor, not, flip-flop, mux, demux.
* Data types:

Nets vs Registers; Vectors; Arrays; memories

Integer, real and time register data types

Parameter vs localparam; String.

System Tasks and Complies Directives

Day 2:

* Dataflow modelling:

Assigment ; Delay ; Expression , Operator and Operand

Operator types: Logical operator vs Bitwise operator?

* Behavioral modelling:

Proceduiral Assigments: Blocking vs NonBlocking?

Day 3:

* Condition statements vs multiway statements? Parallel case, full case?
* Loops: While; for, repeat, forever.
* Block types.

Day 4:

* Tasks vs Functions?
* Procedural continuous assignment: assign/deassign; force/release
* Conditional Compilation and Execution
* Time scales.

Day 5:

* FSM: Moore vs Mealy? Onehot vs onecold?
* Reset synchronous vs reset asynchronous?

Day 6:

* Multidirve
* Race Condition

Day 7:

* Metastability - Clock and reset

Bài tập:

1: Thiết kế bộ Mux 4 -1

2: Thiết kế bộ đếm

* Mạch reset không đồng bộ, tích cực mức thấp.
* Tín hiệu bật cho phép bộ đếm hoạt động, tín hiệu tắt bộ đếm về 0.
* Tín hiệu đếm tăng cho phép bộ đếm đến giá trị A(parameter), khi bằng A xuất tín hiệu báo done.
* Tín hiệu đếm giảm cho phép bộ đếm giảm về 0, khi giảm về 0 xuất tín hiệu báo done.

3: Phân tích timing, thiết kế mạch logic theo chuẩn I2C.

Yêu cầu:

* IP gửi lệnh đọc, ghi dữ liệu xuống I2C Slave.
* IP cần có FSM.
* IP theo chuẩn THE I2C-BUS SPECIFICATION VERSION 2.1 JANUARY 2000 của Philips Semiconductors.

Note:

* Tool sử dụng mô phỏng dùng Questasim.
* Bài 1,2 làm quen với ngôn ngữ, tool mô phỏng.
* Bài 3: Sau khi hoàn thành các lệnh đọc/ghi dữ liệu;
* Thiết kế cần giao tiếp với CPU thông qua chuẩn APB.
* Dữ liệu được đẩy vào FIFO trước khi thực hiện lệnh ghi; được lưu vào FIFO sau khi đọc dữ liệu từ slave.
* Thời gian hoàn thành 3 bài tập sau khi kết thúc ngày 7.

Day 1:

* Phép so sánh a === b chỉ trả về kết quả x hoặc z còn a == b có thì kết quả có thể ko xác định
* Phép dịch >>> thêm bit 1 vào đầu , <<< thì them bit 0 vào sau
* Fifo có xung clk và điều khiển bằng cạnh
* Latch ko có xung clk và điều khiển bằng mức

Day 4:

* task vs function khá giống nhau nhưng function không thể điều khiểm nhiều hơn 1 đầu ra , không chứa các độ trễ thời gian còn task thì ngược lại
* deassign : xóa bỏ lệnh gán của biến trc đó

assign a=b;

#40 deassign a;

Sau #40 thì giá trị a sẽ giữ nguyên kể cả giá trị b có thay đổi

* force/ release : được dùng trong testbench dùng để ghi đè và xóa bỏ lệnh 1 cách ép buộc

assign d = a&b&c; (\*)

a=1,b=0,c=1;

#10 force d=a|b|c; // ghi đè cái lệnh (\*)

Force e= a|b|c;

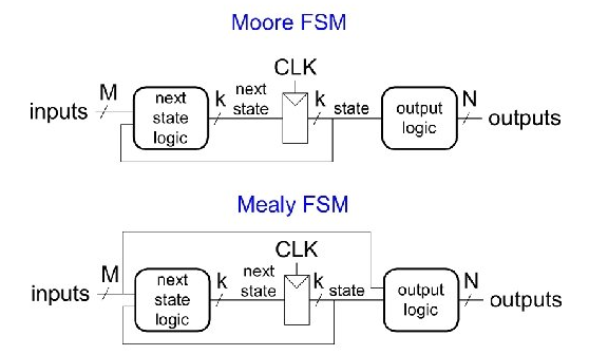
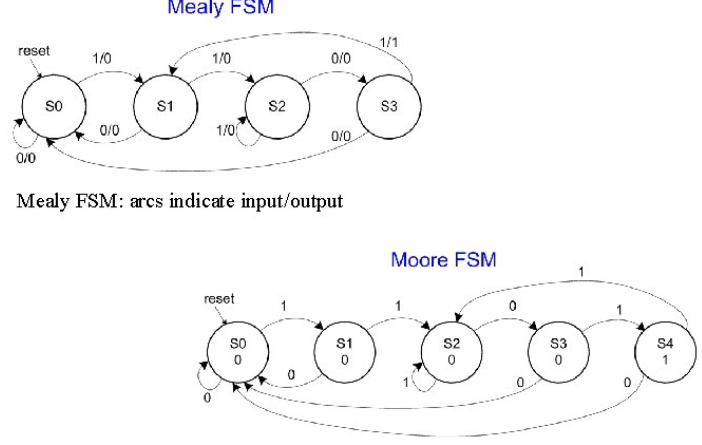
#10 release d;

Release e;// trả về giá trị ban đầu chức năng của lệnh (\*) đc khôi phục

* timescale khung đơn vị time / độ chính xác time

timescale 10ns / 1ns 

Day 5:

* Mô hình moore : đầu ra chỉ phụ thuộc vào trạn thái hiện tại
* Mô hình mealy : đầu ra phụ thuộc vào trạng thái hiện tại và đầu vào
*  
* Mã one hot ví dụ : 0 0 0 0 0 1 còn mã one cold thì ngược lại các bit

0 0 0 0 1 0

0 0 0 1 0 0

0 0 1 0 0 0

* Reset có đồng bộ là khi reset đc kích hoạt đồng bộ theo xung clock còn bất đồng bộ thì ko theo xung clock – thêm tham số reset vào trong danh sách độ nhạy của khối always thì là bất đồng bộ và ngược lại

Các lệnh dùng trong verify

* Rand : cách để khai báo 1 biến để dung lệnh randum giá trị. Rand bit [1:0] cnt ;
* Mailbox : giao tiếp các thư giữa các tiến trình.
  + Có 2 loại mailbox là
    - Chung : có thể gửi và nhận bất kì loại nào . mailbox mailbox\_name
    - không giới hạn : chỉ gửi và nhận loại dữ liệu cụ thể. Mailbox#(type) mail\_name
  + 1 số lệnh thao tác với mailbox
    - New() tạo 1 mailbox , trans = new() ;
    - Put() nơi lưu trữ mailbox , gen2driv.put(trans);
    - Num() , trả về giá trị tin nhắn có trong mailbox
* Event : là 1 kiểu đối tượng tĩnh để đồng bộ hóa giữa các tiến trình hoạt động đồng thời. Để kích hoạt các sự kiện đã đc khai báo ta dùng -> hoặc ->> , các tiến trình có thể đợi 1 sự kiện bằng cách dung @ hoặc .triggered

Một số kiến trúc trong verifi

* SVA- SystemVerilog Assertion :

là một cấu trúc ngôn ngữ cung cấp 1 cách thay thế để viết các dàng buộc, bộ kiểm tra và các điểm che khuất cho thiết kế

phân loại assertion:

* Immediate assertion : giống như 1 assert if else đơn giản, không phụ thuộc vào clock và reset.

Immediate\_assertion\_name :

Assert (Q !=0)

Else

$error(“%m checker failed”);

Mã trên giống với :

Always\_comb begin

If(assertion statement)

// do nothing

Else

%error(“fail”);

End

* Concurrent assertion : cho phép mô tả các biểu thức phức tạp hơn và phụ thuộc vào clock và reset. Từ khóa property để phân biệt 2 loại assertion

// If inputs vld=1 and dat=8'h55, then ack is high 3 cycles later.

// ack sẽ bằng 1 sau 3 chu kì kể từ khi điều kiện trong ngoặc đúng

valid\_gnt\_chk: assert property (

@posdege (clk) disable iff (rst)

(vld && dat == 8'h55) |-> ##3 ack);

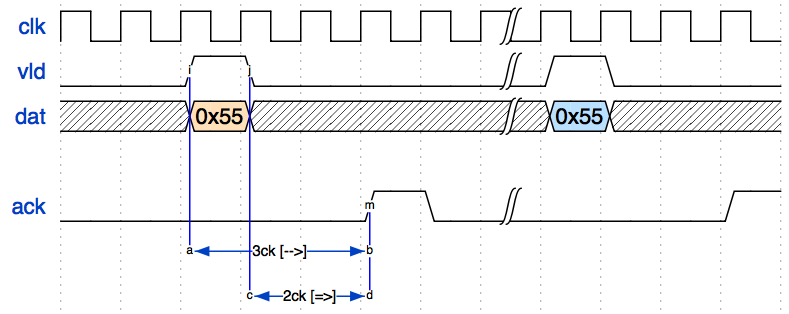
// ack sẽ bằng 1 sau n-1 chu kì kể từ khi điều kiện trong ngoặc đúng

valid\_gnt\_chk: assert property (

@posdege (clk) disable iff (rst)

(vld && dat == 8'h55) |=> ##2 ack);

// “ |-> ##1 “ = “ |=>” , dissble iff thiết lập tín hiệu không đồng bộ - thường là reset



Một số hàm sử dụng cho SVA:

* $rose : trả về true nếu bit LSB của biểu thức thay đổi thành 1 và ngược lại
* $fell : trả về true nếu bit LSB của biểu thức thay đổi thành 0 và ngược lại
* $stable : trả về true nếu giá trị của biểu thức không thay đổi và ngược lại
* $past(expression , num\_cycles) : trả về giá trị của biểu thức từ trước num\_cycles
* $countones: trả vè số 1 trong 1 biểu thức
* $onehot: trả vè true nếu chính xác 1 bit , neesuko có bit nào hoặc nhiều hơn 1 bit 1 thì là false
* $onehot0: trả về true là không có bit nào hoặc chỉ 1 bit 1 trong biểu thức

Một số toán tử sử dụng trong SVA :

* ##n ; ##[m:n] : toán tử độ trẽ cố định; khoảng thời gian tương tác
* |->: hàm chồng chéo
* |=> : hàm không trùng lặp
* [\*n];[\*m:n] : toán tử lặp liên tục; lặp liên tục trong phạm vi chu kì đc chỉ định
* [->n];[->m:n]: cho biết có 1 hay nhiều chu kì trẽ giữa một lần lặp lại của biểu thức

a[->3] = (!a[\*0:$] ##1 a ) [\*3];

* [=n];[=m:n]: a[->3] = (!a[\*0:$] ##1 a ##1 !a[\*0:$] ) [\*3];