IC Lab Formal Verification Bonus Quick Test 2023 Spring

Name: 劉峻瑋 **Student ID:** 0810917 **Account:** iclab054

1. Bonus:

   (a) What is Formal verification?

   What's the difference between Formal and Pattern based verification?

   And list the pros and cons for each.

   Formal verification tests all possible stimulus, one cycle at a time. We can set properties including assert, cover, assume, and test whether DUT meet the properties as it walks through stimulus in BFS.

   Pattern based verification generate stimulus sometimes randomly, and test whether DUT meet the pattern requirement as it walks through the random pattern in DFS.

	Formal verification	Pattern based verification
Pros	1.Less testbench effort required 2.Reveals bugs that pattern based can't find out. 3.More deterministic, none or very little randomization.	1.Test the DUT more efficiently. 2.Ignore some states that won't happen.
Cons	Take more time to stimulus	1.More testbench effort required. 2.May leave some errors since it may not test all the possible condition.

   (b) What is glue logic?

   Why will we use glue logic to simplify our SVA expression?

   Glue logic is auxiliary logic to observe and track events, and doesn't cost extra price.

   Because SVA expression sometimes will be complex, we use glue logic to simplify the SVA expression.

(c) What is the difference between Functional coverage and Code coverage?

What's the meaning of 100% code coverage, could we claim that our assertion is well enough for verification? Why?

Functional coverage tells whether you test all required functionality of your design, describes user defined SVA cover property. Convenient for creating comprehensive coverage involving variables values/transitions/crosses. May be incomplete due to human error.

Code coverage tells whether you executed every code in your design, each if-else go through their true and false states or not, which is easy to generate automatically, and guaranteed to be structurally complete. May not capture all meaningful behavior of design.

100% code coverage means we have executed all the code. But we can't claim that our assertion is well verification. The reason is code coverage only check whether our code is executed, but not the correctness.

(d) What is the difference between COI coverage and proof coverage for realizing checker's completeness? Try to explain from the meaning, relationship, and tool effort perspective.

COI coverage is called Cone-of-Influence, each assertion affected by some cover items, and the union of the all assert COI is COI coverage. COI is the maximum potential of proof coverage. COI analysis is a faster measurement.

Proof coverage is to find the region cannot truly influence assertion status, represents the portion of design verified by formal engines. Proof coverage is subsets of COI. COI doesn't require a proof to take place, while proof coverage does. Identify more unchecked code than COI measurement, with greater tool effort. Proof coverage is a slower measurement.

(e) What are the roles of ABVIP and scoreboard separately?

Try to explain the definition, objective, and the benefit.

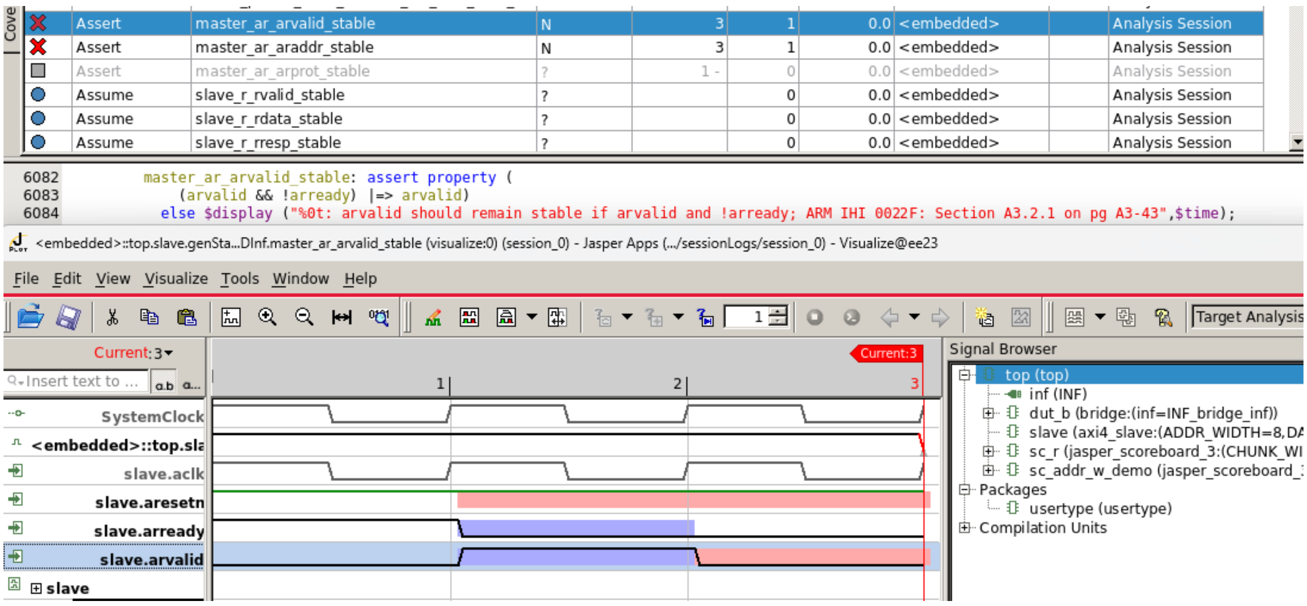
ABVIP is called Assertion Based Verification Intellectual Properties. Are a comprehensive set of checkers and RTL that check for protocol compliance. It's benefit designers to design easier and more quickly.

Scoreboard behaves like a monitor, observe input data and output data of DUV. It's often used to check data packet dropped, duplicated data packets, order of data packets, order of data packets. Benefit is to reduce state-space complexity, also reduce barrier of adoption.

(f) List four bugs in Lab Exercise

What is the answer of the Lab Exercise?

1. arvalid should remain stable if arvalid and !arready

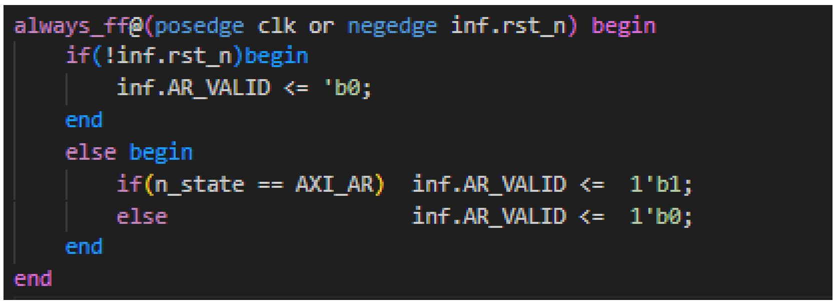


✗	Assert	master_ar_arvalid_stable	N	3	1	0.0	<embedded>	Analysis Session
✗	Assert	master_ar_araddr_stable	N	3	1	0.0	<embedded>	Analysis Session
▢	Assert	master_ar_arprot_stable	?	1 -	0	0.0	<embedded>	Analysis Session
●	Assume	slave_r_rvalid_stable	?		0	0.0	<embedded>	Analysis Session
●	Assume	slave_r_rdata_stable	?		0	0.0	<embedded>	Analysis Session
●	Assume	slave_r_rresp_stable	?		0	0.0	<embedded>	Analysis Session

```
        master_ar_arvalid_stable: assert property (
            (arvalid && !arready) |=> arvalid)
          else $display ("%0t: arvalid should remain stable if arvalid and !arready; ARM IHI 0022F: Section A3.2.1 on pg A3-43",$time);
```

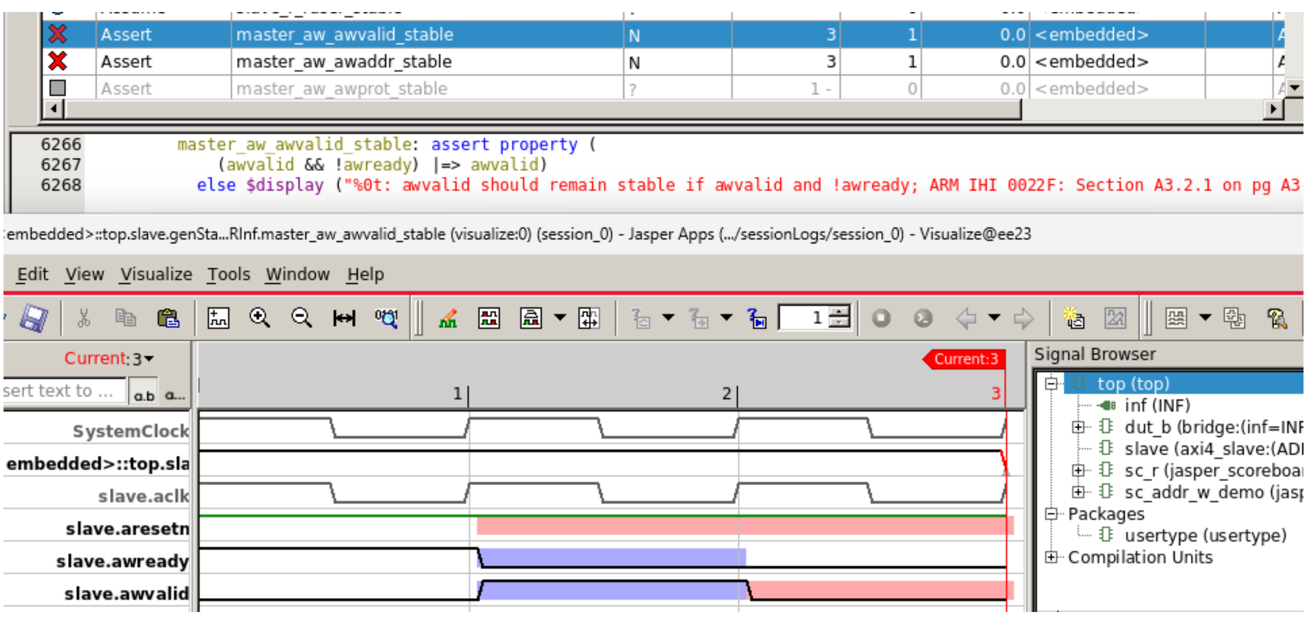
<embedded>::top.slave.genSta...DInf.master_ar_arvalid_stable (visualize:0) (session_0) - Jasper Apps (.../sessionLogs/session_0) - Visualize@ee23

Ans: else if statement modify into (n_state == AXI_AR) condition.



```
always_ff@(posedge clk or negedge inf.rst_n) begin
    if(!inf.rst_n)begin
        inf.AR_VALID <= 'b0;
    end
    else begin
        if(n_state == AXI_AR)  inf.AR_VALID <=  1'b1;
        else                   inf.AR_VALID <=  1'b0;
    end
end
```

2. awvalid should remain stable if awvalid and !awready;

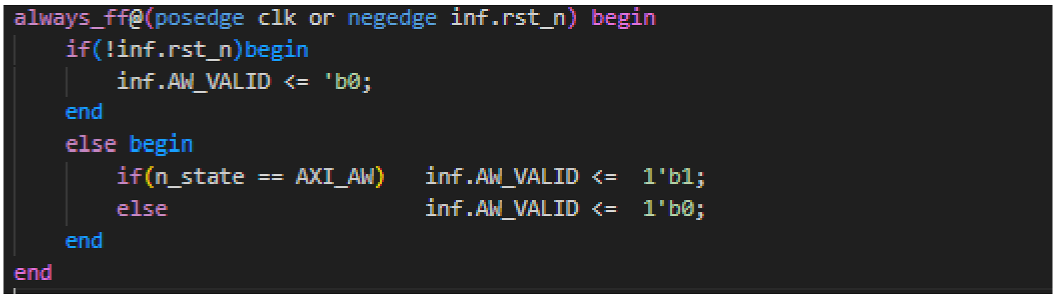


✗	Assert	master_aw_awvalid_stable	N	3	1	0.0	<embedded>
✗	Assert	master_aw_awaddr_stable	N	3	1	0.0	<embedded>
▢	Assert	master_aw_awprot_stable	?	1 -	0	0.0	<embedded>

```
        master_aw_awvalid_stable: assert property (
            (awvalid && !awready) |=> awvalid)
          else $display ("%0t: awvalid should remain stable if awvalid and !awready; ARM IHI 0022F: Section A3.2.1 on pg A3
```

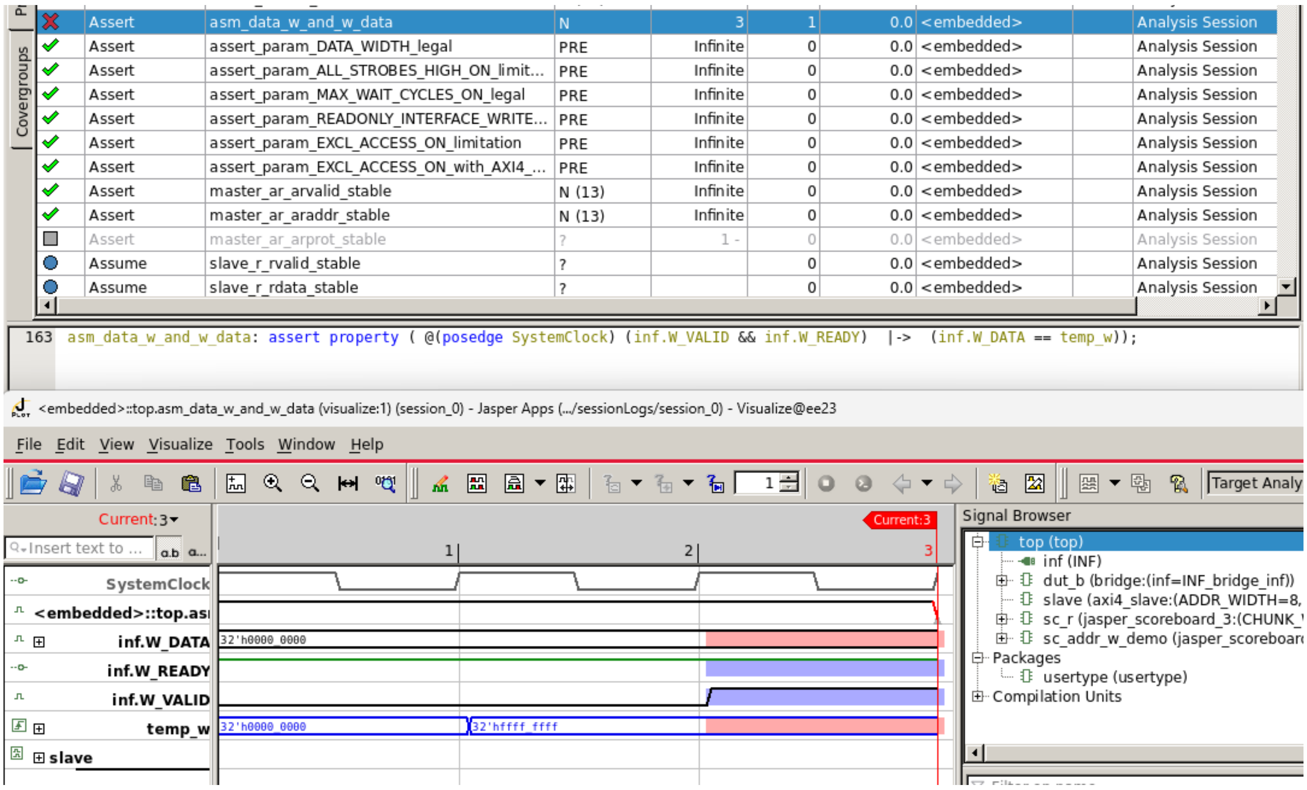
embedded>::top.slave.genSta...RInf.master_aw_awvalid_stable (visualize:0) (session_0) - Jasper Apps (.../sessionLogs/session_0) - Visualize@ee23

Ans: else if statement modify into (n_state == AXI_AW) condition.



```
always_ff@(posedge clk or negedge inf.rst_n) begin
    if(!inf.rst_n)begin
        inf.AW_VALID <= 'b0;
    end
    else begin
        if(n_state == AXI_AW)   inf.AW_VALID <=  1'b1;
        else                    inf.AW_VALID <=  1'b0;
    end
end
```

3. W_Data != temp_w when writing

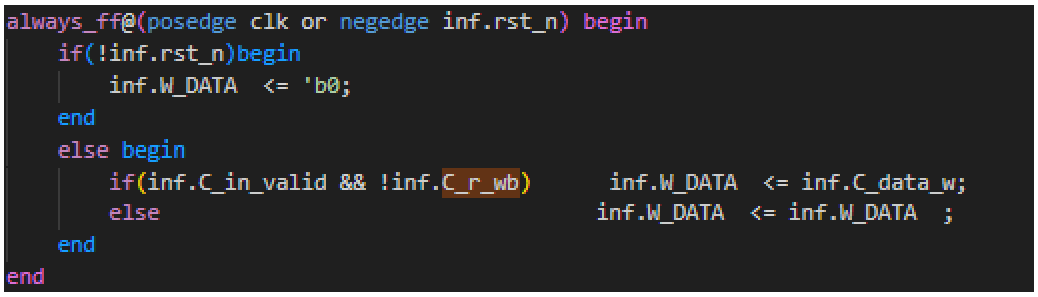


✗	Assert	asm_data_w_and_w_data	N	3	1	0.0	<embedded>		Analysis Session
✓	Assert	assert_param_DATA_WIDTH_legal	PRE	Infinite	0	0.0	<embedded>		Analysis Session
✓	Assert	assert_param_ALL_STROBES_HIGH_ON_limit...	PRE	Infinite	0	0.0	<embedded>		Analysis Session
✓	Assert	assert_param_MAX_WAIT_CYCLES_ON_legal	PRE	Infinite	0	0.0	<embedded>		Analysis Session
✓	Assert	assert_param_READONLY_INTERFACE_WRITE...	PRE	Infinite	0	0.0	<embedded>		Analysis Session
✓	Assert	assert_param_EXCL_ACCESS_ON_limitation	PRE	Infinite	0	0.0	<embedded>		Analysis Session
✓	Assert	assert_param_EXCL_ACCESS_ON_with_AXI4_...	PRE	Infinite	0	0.0	<embedded>		Analysis Session
✓	Assert	master_ar_arvalid_stable	N (13)	Infinite	0	0.0	<embedded>		Analysis Session
✓	Assert	master_ar_araddr_stable	N (13)	Infinite	0	0.0	<embedded>		Analysis Session
▢	Assert	master_ar_arprot_stable	?	1 -	0	0.0	<embedded>		Analysis Session
●	Assume	slave_r_rvalid_stable	?		0	0.0	<embedded>		Analysis Session
●	Assume	slave_r_rdata_stable	?		0	0.0	<embedded>		Analysis Session

```
asm_data_w_and_w_data: assert property ( @(posedge SystemClock) (inf.W_VALID && inf.W_READY)  |->  (inf.W_DATA == temp_w));
```

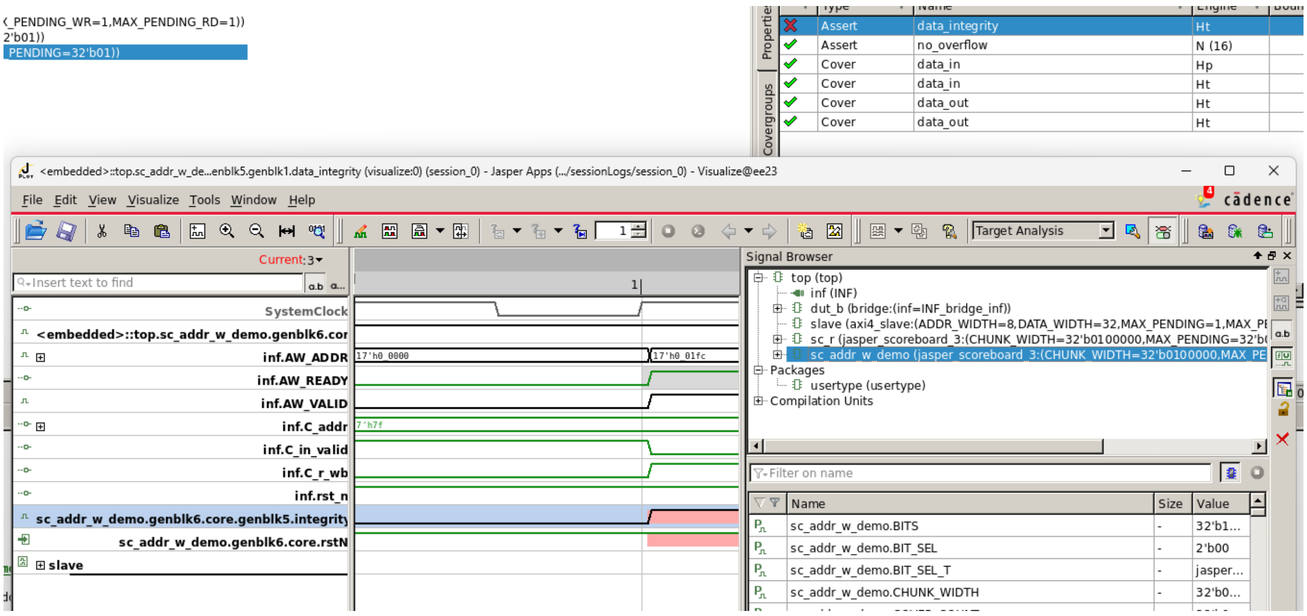
<embedded>::top.asm_data_w_and_w_data (visualize:1) (session_0) - Jasper Apps (.../sessionLogs/session_0) - Visualize@ee23

Ans: modify the else if condition into (inf.C_in_valid && !inf.C_r_wb)



```
always_ff@(posedge clk or negedge inf.rst_n) begin
    if(!inf.rst_n)begin
        inf.W_DATA  <= 'b0;
    end
    else begin
        if(inf.C_in_valid && !inf.C_r_wb)      inf.W_DATA  <= inf.C_data_w;
        else                                  inf.W_DATA  <= inf.W_DATA  ;
    end
end
```

4.data integrity went wrong


```
(_PENDING_WR=1,MAX_PENDING_RD=1))
2'b01))
PENDING=32'b01))
```

	Type	Name	Engine
✗	Assert	data_integrity	Ht
✓	Assert	no_overflow	N (16)
✓	Cover	data_in	Hp
✓	Cover	data_in	Ht
✓	Cover	data_out	Ht
✓	Cover	data_out	Ht

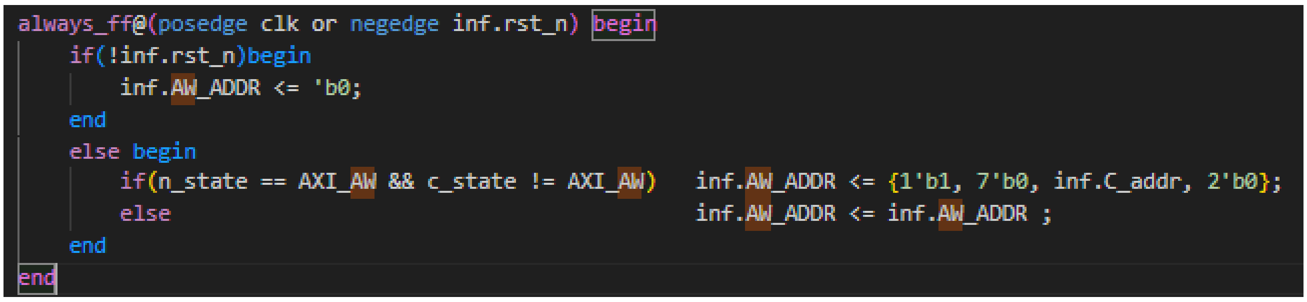
<embedded>::top.sc_addr_w_de...enblk5.genblk1.data_integrity (visualize:0) (session_0) - Jasper Apps (.../sessionLogs/session_0) - Visualize@ee23

```
SystemClock
<embedded>::top.sc_addr_w_demo.genblk6.cor
inf.AW_ADDR          17'h0_0000    17'h0_01fc
inf.AW_READY
inf.AW_VALID
inf.C_addr           7'h7f
inf.C_in_valid
inf.C_r_wb
inf.rst_n
sc_addr_w_demo.genblk6.core.genblk5.integrity
sc_addr_w_demo.genblk6.core.rstN
slave
```

Signal Browser
```
top (top)
  inf (INF)
  dut_b (bridge:(inf=INF_bridge_inf))
  slave (axi4_slave:(ADDR_WIDTH=8,DATA_WIDTH=32,MAX_PENDING=1,MAX_PE
  sc_r (jasper_scoreboard_3:(CHUNK_WIDTH=32'b0100000,MAX_PENDING=32'b(
  sc_addr_w_demo (jasper_scoreboard_3:(CHUNK_WIDTH=32'b0100000,MAX_PE
Packages
  usertype (usertype)
Compilation Units
```

Name	Size	Value
sc_addr_w_demo.BITS	-	32'b1...
sc_addr_w_demo.BIT_SEL	-	2'b00
sc_addr_w_demo.BIT_SEL_T	-	jasper...
sc_addr_w_demo.CHUNK_WIDTH	-	32'b0...

Ans: Change inf.AW_ADDR into the correct answer and correct concatenate data type.


```
always_ff@(posedge clk or negedge inf.rst_n) begin
    if(!inf.rst_n)begin
        inf.AW_ADDR <= 'b0;
    end
    else begin
        if(n_state == AXI_AW && c_state != AXI_AW)    inf.AW_ADDR <= {1'b1, 7'b0, inf.C_addr, 2'b0};
        else                                          inf.AW_ADDR <= inf.AW_ADDR ;
    end
end
```