# •RTL RAM VERIFICATION USING SYSTEMVERILOG AND UVM.

• Under supervision of: Eng. Sherif Hosny

# Table of Contents

3
3
6
6
6
·····
8
g
12
13
1
1
1
17
19

# A class-based UVM Verification for synchronous RAM.

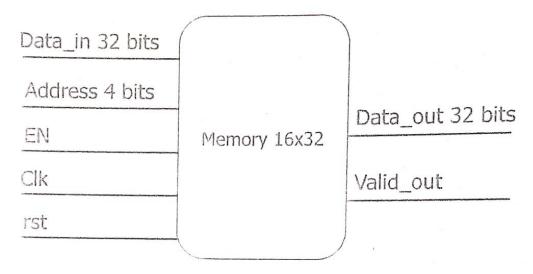
# RAM design:

The design is memory-based circuit that stores data in writing case and out the data in the reading case. It is synchronized with a clock edge and has asynchronous reset signal. The design help management between different parts of a system.

# RAM design specifications:

specification	Details
Inputs	<ul> <li>- Data_in :32-bits input data signal.</li> <li>- Address :4-bits input address signal.</li> <li>- EN : 1-bit input enable signal.</li> <li>- CLK :1-bit clk signal.</li> <li>- RST :1-bit synchronize active high reset signal.</li> <li>- W_R :1-bit read /write signal (0 for write, 1 for read).</li> </ul>
Outputs	-Data_out :32-bits output data signalvalid_out :1-bit output refers to new data out.
Behavior	<ul> <li>-Reset: it is synchronize reset so on high level reset all locations on the memory should be equal zero.</li> <li>-Read operation: in this case memory should out data on output.in this case rst signal should be =0, EN signal should be =1 and W_R = 1 for work on read mode.</li> <li>-write operation: in this case memory should monitor the input data and save it on the input address.in this case rst signal should be =0, EN signal should be =1 and W_R = 0 for work on write mode.</li> </ul>

# RTL design Architecture.



Note: the following RTL code was made by AI copilot tool.

## RTL design Modules:

## • Memory.v

```
module Memory (
    input [31:0] Data_in,
    input [3:0] Address,
    input EN,
    input CLK,
    input RST,
    input W_R, // 0 for write, 1 for read
    output reg [31:0] Data_out,
    output reg valid_out
);
    //declare index
    reg [4:0] i;
    // Declare memory array of size 16x32
    reg [31:0] memory [0:15];
    always @(posedge CLK or posedge RST) begin
        if (RST) begin
            // Reset memory and outputs
            Data_out <= 32'b0;</pre>
```

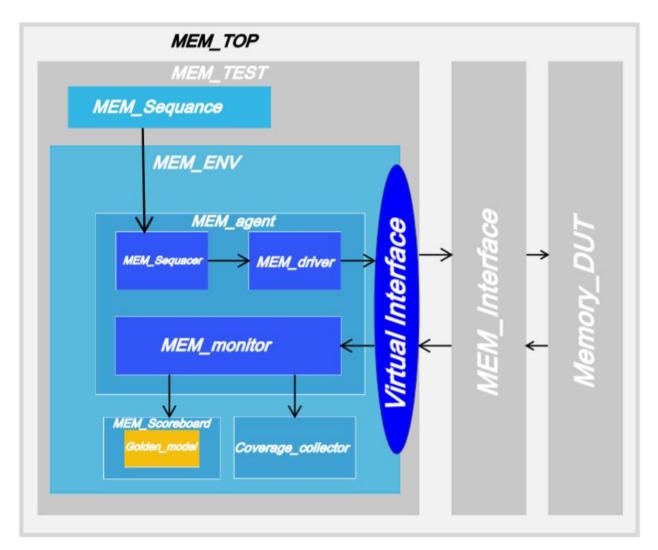
```
4
```

# test plan:

Feature	Check list	Stimulus
Reset behavior	-All memory locations should be equal zeroThe memory does not have response about all inputs in case rst signal highin case rst signal is low we can test other features.	Rst :0->1->0
Write all memory location	-in this check we should write all memory locations for all memory locations.	Rst =0 EN=1 W_R=0 -other inputs randomize them.
Read all memory location	-in this check we should read all memory locations for all memory locations that we write it from the previous scenariocheck the values are right as we write in the previous scenario.	Rst =0 EN=1 W_R=1 -other inputs randomize them.
Randomize number of stimulus inputs between Read and write cases	-in this case we will check No error has been found From check the output data	Rst =0 EN=1 -other inputs randomize them.
Randomize all inputs and check the correction of the outputs (in this case we	in this case we will check No error has been found	All inputs randomize

check all feature with random order)

From check the output data



## environment:

# Parts of environment code:

• Top Module.

```
`timescale 1ns/1ps
`include"MEM_INTERFACE.sv"
import uvm_pkg::*;
import MEM_Package::*;
module MEM_TOP;
```

```
parameter clk_period =10;
bit clk;
//interface instantiation
MEM INTERFACE intf(clk);
Memory mem_dut(.CLK(clk),
                .Data_in(intf.Data_in),
                .Data out(intf.Data out),
                .Address(intf.Address),
                .EN(intf.EN),
                .RST(intf.RST),
                .W_R(intf.W_R),
                .valid_out(intf.valid_out)
                );
//clk triggering
initial begin
    clk=0;
    forever
    begin
        #(clk_period/2) clk=~clk;
    end
end
initial begin
    uvm_config_db#(virtual
MEM_INTERFACE)::set(null,"uvm_test_top","top2tast",intf);
    run test("MEM TEST");
end
endmodule
```

#### • Pack file

```
package MEM_Package;
import uvm_pkg::*;
parameter size_of_memory_location=16;
  `include"uvm_macros.svh"
  `include "MEM_seq_item.sv"
  `include "MEM_Sequancer.sv"
  `include "MEM_Driver.sv"
  `include "MEM_Monitor.sv"
  `include "MEM_Agent.sv"
  `include "MEM_Agent.sv"
  `include "MEM_Scoreboard.sv"
  `include "MEM_coverage_collector.sv"
```

```
`include "MEM_Sequance.sv"
  include "MEM_ENV.sv"
  include "MEM_TEST.sv"
endpackage
```

#### AES TEST

```
class MEM TEST extends uvm test;
`uvm_component_utils(MEM_TEST)
// Constructor
function new(string name = "MEM_TEST", uvm_component parent = null);
super.new(name, parent);
endfunction: new
//declare the environment and sequence
MEM_env my_env;
MEM_Sequence my_seq;
//declare the interface
virtual MEM_INTERFACE vif;
//declare the build function
function void build_phase(uvm_phase phase);
super.build_phase(phase);
my env = MEM env::type id::create("my env", this);
my_seq = MEM_Sequence::type_id::create("my_seq", this);
if (
    !uvm_config_db #(virtual MEM_INTERFACE)::get(this,"", "top2tast", vif)
    ) begin
        `uvm fatal(get full name(), "[MEM_TEST] vif not get");
end
    uvm config db #(virtual MEM INTERFACE)::set(this, "my env", "test2env",
vif);
endfunction: build phase
//declare the connect function
function void connect phase(uvm phase phase);
super.connect_phase(phase);
endfunction: connect_phase
// Task: run phase
task run_phase(uvm_phase phase);
super.run_phase(phase);
```

```
phase.raise_objection(this);
$display("Starting the test");
my_seq.start(my_env.my_agent.m_sequencer);
phase.drop_objection(this);
$display("Ending the test");
endtask: run_phase
endclass: MEM_TEST
```

#### Sequence class

```
class MEM_Sequence extends uvm_sequence;
uvm_object_utils(MEM_Sequence)
///**declare my_trans object**///
MEM_Transaction my_trans;
///**Constructor**///
extern function new(string name = "MEM_Sequence");
extern task body();
//***********************
///**Task for Test RST at first time**///
extern task RST();
//**********************************
////****read_n_number****////
extern task read_n_number(input int num_to_read);
////****write_n_number****///
extern task write_n_number(input int num_to_write);
//*******************************
////****rand_task****////
extern task rand_task(input int num_to_rand);
endclass: MEM_Sequence
```

```
/////**** Constructor****////
function MEM_Sequence::new(string name = "MEM_Sequence");
    super.new(name);
   endfunction: new
task MEM_Sequence::body();
 int num 2 rand;
///**test rst at first time **///
   #10;
   RST();
///**test read all memory sequencially**///
read_n_number(size_of_memory_location);
//test write all memory sequencially
write_n_number(size_of_memory_location);
//test read all memory sequencially
read n number(size of memory location);
//rand task to test the randomization of the transaction
num 2 rand =$urandom range(100000,50000);
rand_task(num_2_rand);
endtask: body
///**Task for Test RST at first time**///
task MEM Sequence::RST();
my_trans=MEM_Transaction::type_id::create("my_trans");
start item(my trans);
my_trans.RST.rand_mode(0); //I put it as without it rst will be randomize on the next
step
                            //set RST to 1 and will not affect by the randomize
my trans.RST = 1;
assert(my_trans.randomize()); //randomize the my_trans
$display("at time(%0t): [SQEUANCER] the randomized my_trans RST :%p",$realtime,my_trans);
finish_item(my_trans);
///***this iteration to fall the RST with the same task****///
my_trans=MEM_Transaction::type_id::create("my_trans");
start_item(my_trans);
//my trans.RST.rand mode(0); //I put it as without it rst will be randomize on the
my_trans.RST = 0;
my trans.EN = 0;
```

```
assert(my_trans.randomize(Address,Data_in,W_R)); //randomize the my_trans
$display("at time(%0t): [SQEUANCER] the randomized my_trans RST :%p",$realtime,my_trans);
finish_item(my_trans);
my trans.RST.rand mode(1);
 $display("at time(%0t):[SQEUANCER] Driver Done on RST check",$realtime);
endtask:RST
//*Task to write with certain number to task*//
task MEM Sequence::write n number(input int num to write);
    int i = 0;
    my_trans=MEM_Transaction::type_id::create("my_trans");
    repeat(num_to_write) begin
    start_item(my_trans);
    /*initial values for do the functionality of the task without the randomization*/
    my trans.EN = 1;
    my_trans.W_R = 0;
    my_trans.RST = 0;
    assert(my_trans.randomize(Address,Data_in));
    $display("[SQEUANCER] the randomized transaction on write_n_number task
:%p",my_trans);
    finish item(my trans);
    $display("[SQEUANCER] send to driver number : %0d on write_n_number task ",i);
    i++;
    end
    /*change the inputs after finish the required function from task*/
    my trans=MEM Transaction::type id::create("my trans");
    start_item(my_trans);
    my trans.EN = 0;
    my_trans.W_R = 0;
    my_trans.RST=0;
    assert(my trans.randomize(Address,Data in));
    finish_item(my_trans);
endtask:write_n_number
//***************
//**Task to read with certain number to task**//
task MEM_Sequence::read_n_number(input int num_to_read);
int i = 0;
        repeat(num_to_read) begin
        my_trans=MEM_Transaction::type_id::create("my_trans");
        start_item(my_trans);
        my trans.EN = 1;
```

```
my_trans.W_R = 1;
        my_trans.RST=0;
        assert(my_trans.randomize(Address,Data_in));
        $display("at time(%0t): [SQEUANCER] send to driver no to read on read_n_number
task: %0d",$time,i);
      // if(my_trans.W_R)
        //$stop;
        finish_item(my_trans);
        $display("at time(%0t): [SQEUANCER] Driver Done****",$time);
        i++;
   end
    /****this iteration to fall the valid out from the inputs****/
   my_trans=MEM_Transaction::type_id::create("my_trans");
    start item(my trans);
   my_trans.EN = 0;
   my trans.W R = 0;
   my_trans.RST=0;
   assert(my_trans.randomize(Address,Data_in));
   finish_item(my_trans);
endtask:read_n_number
 task MEM_Sequence::rand_task(input int num_to_rand);
 for(int i=0;i<num to rand;i++) begin</pre>
   my_trans=MEM_Transaction::type_id::create("my_trans");
   start_item(my_trans);
   assert(my_trans.randomize());
   finish_item(my_trans);
 end
 /****this iteration to fall the valid_out from the inputs****/
   my_trans=MEM_Transaction::type_id::create("my_trans");
   start_item(my_trans);
   my trans.EN = 0;
   my_trans.W_R = 0;
   my_trans.RST=0;
    assert(my_trans.randomize(Address,Data_in));
    finish_item(my_trans);
 endtask:rand_task
```

#### • Env class

```
class MEM_env extends uvm_env;
`uvm_component_utils(MEM_env)
```

```
// Constructor
function new(string name = "MEM_env", uvm_component parent = null);
super.new(name, parent);
endfunction: new
//declare the conponents
MEM_Agent my_agent;
MEM_Scoreboard my_scoreboard;
MEM_coverage_collector my_collector;
//declare the virtual interface
virtual MEM INTERFACE my vif;
//declare the build function
function void build_phase(uvm_phase phase);
super.build phase(phase);
my_agent = MEM_Agent::type_id::create("my_agent", this);
my_scoreboard = MEM_Scoreboard::type_id::create("my_scoreboard", this);
my_collector = MEM_coverage_collector::type_id::create("my_collector", this);
if(
    !uvm_config_db #(virtual MEM_INTERFACE)::get(this, "", "test2env", my_vif)
`uvm fatal(get full name(), "[MEM env] vif not get")
uvm_config_db #(virtual MEM_INTERFACE)::set(this, "my_agent", "env2agent", my_vif);
endfunction: build_phase
//declare the connect function
function void connect phase(uvm_phase phase);
super.connect_phase(phase);
my_agent.m_monitor.port.connect(my_collector.analysis_export);
my_agent.m_monitor.port.connect(my_scoreboard.imp);
endfunction: connect phase
// Task: run phase
task run_phase(uvm_phase phase);
super.run_phase(phase);
endtask: run phase
endclass: MEM_env
```

• Agent class

```
class MEM_Agent extends uvm_agent;
  `uvm_component_utils(MEM_Agent)
 // Components
 MEM Driver m driver;
 MEM_Sequencer m_sequencer;
 MEM_Monitor m_monitor;
 // Constructor
 function new(string name = "MEM_Agent", uvm_component parent = null);
 super.new(name, parent);
 endfunction: new
 //interface
 virtual MEM_INTERFACE m_vif;
 // Build Phase
   function void build_phase(uvm_phase phase);
   super.build_phase(phase);
    m_driver = MEM_Driver::type_id::create("m_driver", this);
   m_sequencer = MEM_Sequencer::type_id::create("m_sequencer", this);
    m monitor = MEM Monitor::type id::create("m monitor", this);
    if(
      !uvm_config_db #(virtual MEM_INTERFACE)::get(this, "", "env2agent",
m_vif)
    `uvm_fatal(get_full_name(), "[MEM_Agent] vif not get")
    uvm_config_db #(virtual MEM_INTERFACE)::set(this, "m_driver",
"driver2agent", m vif);
    uvm_config_db #(virtual MEM_INTERFACE)::set(this, "m_monitor",
"monitor2agent", m_vif);
 endfunction: build phase
 // Connect Phase
    function void connect phase(uvm_phase phase);
   super.connect_phase(phase);
   m driver.seg item port.connect(m sequencer.seg item export);
    endfunction: connect phase
 // Task: run phase
    task run_phase(uvm_phase phase);
    super.run_phase(phase);
    endtask: run_phase
endclass
```

#### Coverage collector

```
class MEM coverage collector extends uvm subscriber #(MEM Transaction);
`uvm_component_utils(MEM_coverage_collector)
//transaction object
MEM_Transaction tran_mon2sub;
//coverage group
covergroup cov;
ENABLE: coverpoint tran_mon2sub.EN { bins H_EN={1'd1};
                                     bins L_EN={1'd0};
RESET: coverpoint tran_mon2sub.RST{ bins H_RST={1'd1};
                                    bins L RST={1'd0};
TRANS_RST:coverpoint tran_mon2sub.RST{bins trans_H2L=(1'd1 => 1'd0);
W R: coverpoint tran_mon2sub.W_R{ bins H_W_R={1'd1};
                                  bins L_W_R={1'd0};
TRANS_W_R:coverpoint tran_mon2sub.W_R{
                                    bins trans_L2H=(1'd0 => 1'd1);
ADDRESS: coverpoint tran mon2sub.Address{bins low add ={4'd0};
                                         bins med_add[]={[4'd1:4'd14]};
                                         bins high_add={4'd15};
DATA_IN: coverpoint tran_mon2sub.Data_in{bins low_data ={32'h0};
                                         bins med_data ={[32'h1:32'hfffffffe]};
                                         bins high_data ={32'hffffffff};
check_check_W_R_all_addresses :cross_ENABLE, W_R_,DATA_IN , ADDRESS ; //check_read and
write on all addresses
check_rst :cross RESET, ENABLE, W_R, ADDRESS ,DATA_IN ; //check rst is domenant on
all cases on the other values for variables
endgroup:cov
 // Constructor
 function new(string name = "MEM_coverage_collector", uvm_component parent = null);
 super.new(name, parent);
```

```
cov = new();
  endfunction: new
  //declare the build function
  function void build_phase(uvm_phase phase);
  super.build_phase(phase);
  endfunction: build_phase
  //declare the connect function
  function void connect_phase(uvm_phase phase);
  super.connect_phase(phase);
  endfunction: connect_phase
 task run_phase(uvm_phase phase);
  super.run_phase(phase);
 endtask: run_phase
 //write function
 function void write(MEM_Transaction t);
 tran_mon2sub =t;
 if(tran_mon2sub.inp_trans ==1)
  cov.sample();
 endfunction: write
endclass
```

## Results:

#### • Assertion coverage:

```
ASSERTION RESULTS:
Name
                   File(Line)
                                                  Failure
/mem_package/mem_sequancer/RST/immed__56
                     C:/Users/Wello/Desktop/material/GP/MY GITHUB MEM/SV/Sequencer Class.sv(56)
/mem_package/mem_sequancer/RST/immed__44
                     C:/Users/Wello/Desktop/material/GP/MY GITHUB_MEM/SV/Sequencer Class.sv(44)
/mem_package/mem_sequancer/read_n_number/immed__170
                     C:/Users/Wello/Desktop/material/GP/MY GITHUB_MEM/SV/Sequencer Class.sv(170)
                                                        0
/mem_package/mem_sequancer/read_n_number/#ublk#101438645#152/immed 158
                     C:/Users/Wello/Desktop/material/GP/MY GITHUB_MEM/SV/Sequencer Class.sv(158)
/mem_package/mem_sequancer/write_n_number/immed__196
                     C:/Users/Wello/Desktop/material/GP/MY GITHUB_MEM/SV/Sequencer Class.sv(196)
/mem_package/mem_sequancer/write_n_number/#ublk#101438645#183/immed__184
                     C:/Users/Wello/Desktop/material/GP/MY GITHUB MEM/SV/Sequencer Class.sv(184)
/mem_package/mem_scoreboard/run/#ublk#101438645#70/immed__72
                     C:/Users/Wello/Desktop/material/GP/\overline{\text{MY}} GITHUB MEM/SV/Scoreboard Class.sv(72)
```

### • Code coverage:

```
-----Statement Details-----
Statement Coverage for file Tbench.sv --
                                                                                     `timescale 1ns/1ns
                                                                                    timescale Ins/Ins
/////**NDTE: import and include not defind inside a module and interface cann't defind inside package**////
import mem_package::*; // Uncommented this line to import the package
'include "Interface.sv"
module MEM_top;
parameter clk_period =10;
bit clk;
                                                                                        // Instantiate the MemoryInterface
MemoryInterface I_F(clk);
                                                                                        10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
24
24
25
26
27
28
29
30
31
                                                                                               .EN(I F.EN),
                                                                                              .EN(I_F.EN),
.CLK(I_F.clk),
.RST(I_F.RST),
.W_R(I_F.W_R),
.Data_out(I_F.Data_out),
.valid_out(I_F.valid_out)
                                                                                        initial begin
                                                                                                clk = 0;
forever #(clk_period/2) clk = ~clk;
                                                                  156714
                                                                                        end
                                                                                        initial
                                                                                         begin
                                                                                          mem_env mem_env_inst;
mem_env_inst = new(I_F);
                                                                                         mem_env_inst.run();
                                                                                    endmodule
```

```
Toggle Coverage:
  Enabled Coverage
                       Active
                               Hits
                                     Misses % Covered
                                      -----
  Toggle Bins
                                         0
                                           100.00
Toggle Coverage for File Tbench.sv --
                                 Node
                                        1H->0L
                                                0L->1H "Coverage"
     Line
Total Node Count
                       1
Toggled Node Count =
                       1
Untoggled Node Count =
                      0
Toggle Coverage
                  100.00% (2 of 2 bins)
```

# • Function coverage:

OVERGROUP COVERAGE:				
vergroup	Metric	Goal	Stat	tus
 TYPE /mem_package/mem_coverage_collector/cov	100.00%	100	Cove	 ered
covered/total bins:	603	603		
missing/total bins:	0	603		
% Hit:	100.00%	100		
Coverpoint cov::ENABLE	100.00%	100.00% 100		ered
covered/total bins:	2	2		
missing/total bins:	0	2		
% Hit:	100.00%	100		
Coverpoint cov::RESET	100.00%	100	Cove	ered
covered/total bins:	2	2		
missing/total bins:	0	2		
% Hit:	100.00%	100		
Coverpoint cov::TRANS_RST	100.00%	100	Cove	ered
covered/total bins:	1	1		
missing/total bins:	0	1		
% Hit:	100.00%	100		
Coverpoint cov::W_R	100.00%	100	Cove	ered
covered/total bins:	2	2		
missing/total bins:	0	2		
% Hit:	100.00%	100		
Coverpoint cov::TRANS_W_R	100.00%	100	Cove	ered
covered/total bins:	1	1		
missing/total bins:	0	1		
% Hit:	100.00%	100		
Coverpoint cov::ADDRESS	100.00%	100	Cove	ered
covered/total bins:	16	16		
missing/total bins:	0	16		
% Hit:	100.00%	100	_	
Coverpoint cov::DATA_IN	100.00%	100	Cove	ered
covered/total bins:	3	3		
missing/total bins:	0	3		
% Hit:	100.00%	100		
Coverpoint W_R	100.00		100	Covered
<pre>covered/total bins: missing/total bins:</pre>		2 Ø	2 2	
% Hit:	100.00	%	100	
bin H_W_R	39440 38906		1	Covered
bin L_W_R Coverpoint TRANS_W_R	100.00		1 100	Covered Covered
covered/total bins:		1	1	
missing/total bins: % Hit:	100.00	0 %	1 100	
bin trans_L2H	1961	.9	1	Covered
Coverpoint ADDRESS covered/total bins:	100.00	9% .6	100 16	Covered
missing/total bins:		.6 Ø	16	
% Hit:	100.00		100	6
bin low_add bin med_add[1]	481 484		1 1	Covered Covered
bin med_add[2]	494	3	1	Covered
bin med_add[3] bin med add[4]	502 476		1 1	Covered Covered
bin med_add[4] bin med_add[5]	476 474		1	Covered
bin med_add[6]	493		1	Covered
bin med_add[7] bin med_add[8]	49 <i>6</i> 489		1 1	Covered Covered
bin med_add[9]	482	.7	1	Covered
bin med_add[10] bin med_add[11]	488 491		1 1	Covered Covered
oin med_add[ii]			1	Covered
bin med_add[12]	491	.0		
bin med_add[12] bin med_add[13] bin med_add[14]	491 500 499	96	1 1	Covered Covered

5				
Cross check_check_W_R_all_addresses	100.00%	100	Covered	
covered/total bins:	192	192		- 1
missing/total bins:	0	192		- 1
% Hit:	100.00%	100		- 1
bin <h_en,h_w_r,low_data,low_add></h_en,h_w_r,low_data,low_add>	391	1	Covered	- 1
bin <l_en,h_w_r,low_data,low_add></l_en,h_w_r,low_data,low_add>	425	1	Covered	- 1
bin <h_en,l_w_r,low_data,low_add></h_en,l_w_r,low_data,low_add>	389	1	Covered	- 1
bin <l_en,l_w_r,low_data,low_add></l_en,l_w_r,low_data,low_add>	400	1	Covered	- 1
bin <h_en,h_w_r,med_data,low_add></h_en,h_w_r,med_data,low_add>	399	1	Covered	- 1
bin <l_en,h_w_r,med_data,low_add></l_en,h_w_r,med_data,low_add>	407	1	Covered	- 1
bin <h_en,l_w_r,med_data,low_add></h_en,l_w_r,med_data,low_add>	376	1	Covered	- 1
bin <l_en,l_w_r,med_data,low_add></l_en,l_w_r,med_data,low_add>	409	1	Covered	- 1
bin <h_en,h_w_r,high_data,low_add></h_en,h_w_r,high_data,low_add>	372	1	Covered	
bin <l_en,h_w_r,high_data,low_add></l_en,h_w_r,high_data,low_add>	409	1	Covered	- 1
bin <h_en,l_w_r,high_data,low_add></h_en,l_w_r,high_data,low_add>	421	1	Covered	- 1
bin <l_en,l_w_r,high_data,low_add></l_en,l_w_r,high_data,low_add>	419	1	Covered	- 1
bin <h_en,h_w_r,low_data,med_add[1]></h_en,h_w_r,low_data,med_add[1]>	361	1	Covered	- 1
bin <h_en,h_w_r,low_data,med_add[2]></h_en,h_w_r,low_data,med_add[2]>	408	1	Covered	- 1
bin <h_en,h_w_r,low_data,med_add[3]></h_en,h_w_r,low_data,med_add[3]>	419	1	Covered	- 1
bin <h_en,h_w_r,low_data,med_add[4]></h_en,h_w_r,low_data,med_add[4]>	405	1	Covered	- 1
bin <h_en,h_w_r,low_data,med_add[5]></h_en,h_w_r,low_data,med_add[5]>	409	1	Covered	- 1
bin <h_en,h_w_r,low_data,med_add[6]></h_en,h_w_r,low_data,med_add[6]>	429	1	Covered	- 1
bin <h_en,h_w_r,low_data,med_add[7]></h_en,h_w_r,low_data,med_add[7]>	427	1	Covered	- 1
bin <h_en,h_w_r,low_data,med_add[8]></h_en,h_w_r,low_data,med_add[8]>	417	1	Covered	- 1
bin <h_en,h_w_r,low_data,med_add[9]></h_en,h_w_r,low_data,med_add[9]>	355	1	Covered	
bin <h_en,h_w_r,low_data,med_add[10]></h_en,h_w_r,low_data,med_add[10]>	417	1	Covered	

