Verilog & UVM Editor

post_design@live.com

Agent

- Overview
- Before Start
- Vlog Utilities
- UVM Utilities

Overview

- Generate following blocks automatically
  - Verilog port list for module
  - Implement for Verilog instances
  - definition for wire and reg
  - loop codes
  - user-defined template
  - Interface
  - UVM field

Before Start

- Setup environment $VLOG_AIDE_HOME

  Put scripts into $VLOG_AIDE_HOME/src
- Setup environment $VLOG_LIBRARY_PATH

  Location of template files
- Setup environment $VLOG_AIDE_RTL_PATH

  Path for design

Vlog-Utilities

- AutoPort
- AutoInst
- AutoDef
- AutoGen
- AutoInf
- Template

AutoPort

- Gvim Menu: Vlog-Utilites->AutoPort
- Gvim Command: AutoPort
- Mark: /*vlog_aide:auto

```
module auto_port(/*vlog_aide:auto_port*/);

input          test1;
input          test21, test22, test23;
input [3:0]     test3;
input [`DATA_WIDTH-1:0] test4;
output          test5, test51;
output  [15:0]  test6;
inout          test7;

endmodule
```

```
module auto_port(/*vlog_aide:auto_port*/
/*vlog_aide:auto_port begin*/
/*vlog_aide:auto_port input ports*/
test4, test3, test23, test22, test21, test1,
/*vlog_aide:auto_port output ports*/
test6, test51, test5,
/*vlog_aide:auto_port inout ports*/
test7
/*vlog_aide:auto_port end*/
);

input          test1;
input          test21, test22, test23;
input [3:0]     test3;
input [`DATA_WIDTH-1:0] test4;
output          test5, test51;
output  [15:0]  test6;
inout          test7;

endmodule
```

AutoInst

- Gvim Menu: Vlog-Utilities->Auto-Instance
- Gvim Command: AutoInst
- [illegible]uto[illegible]

```
module inst1(/*vlog_aide:auto_port*/);

parameter DIN_WIDTH = 16;

input                  inst1_in0;
input                  inst1_in1, inst1_in2;
input [5:0]            inst1_in3;
input [`DATA_WIDTH-1:0] inst1_in4;
input [DIN_WIDTH-1:0]   inst1_in5;
output                 inst1_out0, inst1_out2;
output [15:0]          inst1_out3;
output [DIN_WIDTH*2-1:0]       inst1_out4;

endmodule
```

```
module inst2(
 input                 inst2_in0,
 input                 inst2_in1, inst2_in2,
 input [DIN_WIDTH-1:0]  inst2_in3,
 input [7:0]           inst2_in4, inst2_in5,
 output reg signed [15:0]      inst2_out0,
 output [DOUT_WIDTH-1:0]       inst2_out1, inst2_out2,
 output reg [DIN_WIDTH*2-1:0]   inst2_out3
);
parameter DIN_WIDTH = 16,
          DOUT_WIDTH = 32;

endmodule
```

AutoInst

```
module top(/*vlog_aide:auto_port*/);

inst1 dut1(/*vlog_aide:auto_inst inst1.v*/);
inst2 #(8,16) dut2(/*vlog_aide:auto_inst inst2.v*/);

endmodule
```

$\rightarrow$

```
module top(/*vlog_aide:auto_port*/);

inst1 dut1(/*vlog_aide:auto_inst inst1.v*/
    /*vlog_aide:auto_inst begin*/
    /*vlog_aide:auto_inst input ports*/
    .inst1_in3     (inst1_in3[5:0]),
    .inst1_in1     (inst1_in1),
    .inst1_in2     (inst1_in2),
    .inst1_in0     (inst1_in0),
    .inst1_in5     (inst1_in5[15:0]),
    .inst1_in4     (inst1_in4[`DATA_WIDTH-1:0]),
    /*vlog_aide:auto_inst output ports*/
    .inst1_out4    (inst1_out4[31:0]),
    .inst1_out0    (inst1_out0),
    .inst1_out2    (inst1_out2),
    .inst1_out3    (inst1_out3[15:0])
    /*vlog_aide:auto_inst end*/
);
inst2 #(8,16) dut2(/*vlog_aide:auto_inst inst2.v*/
    /*vlog_aide:auto_inst begin*/
    /*vlog_aide:auto_inst input ports*/
    .inst2_in2     (inst2_in2),
    .inst2_in5     (inst2_in5[7:0]),
    .inst2_in4     (inst2_in4[7:0]),
    .inst2_in0     (inst2_in0),
    .inst2_in3     (inst2_in3[7:0]),
    .inst2_in1     (inst2_in1),
    /*vlog_aide:auto_inst output ports*/
    .inst2_out0    (inst2_out0[15:0]),
    .inst2_out3    (inst2_out3[15:0]),
    .inst2_out1    (inst2_out1[15:0]),
    .inst2_out2    (inst2_out2[15:0])
    /*vlog_aide:auto_inst end*/
);

endmodule
```

AutoInst

```
module inst1(/*vlog_aide:auto_port*/);

parameter DIN_WIDTH = 16;

input                inst1_in0;
input                inst1_in1, inst1_in2;
input [5:0]          inst1_in3;
//input [`DATA_WIDTH-1:0] inst1_in4;
input [DIN_WIDTH-1:0]    inst1_in5;
output               inst1_out0, inst1_out2;
output [15:0]        inst1_out3;
output [DIN_WIDTH*2-1:0]     inst1_out4;

endmodule
```

```
module top(/*vlog_aide:auto_port*/);

inst1 dut1(/*vlog_aide:auto_inst inst1.v*/
    /*vlog_aide:auto_inst begin*/
    /*vlog_aide:auto_inst input ports*/
    .inst1_in3     (top_in3[5:0]),
    .inst1_in1     (top_in1),
    .inst1_in2     (top_in2),
    .inst1_in0     (top_in0),
    .inst1_in5     (top_in5[15:0]),
    .inst1_in4     (top_in4[`DATA_WIDTH-1:0]),
    /*vlog_aide:auto_inst output ports*/
    .inst1_out4    (top_out4[31:0]),
    .inst1_out0    (top_out0),
    .inst1_out2    (top_out2),
    .inst1_out3    (top_out3[15:0])
    /*vlog_aide:auto_inst end*/
);
inst2 #(8,16) dut2(/*vlog_aide:auto_inst inst2.v*/
    /*vlog_aide:auto_inst begin*/
    /*vlog_aide:auto_inst input ports*/
    .inst2_in2     (inst2_in2),
    .inst2_in5     (inst2_in5[7:0]),
    .inst2_in4     (inst2_in4[7:0]),
    .inst2_in0     (inst2_in0),
    .inst2_in3     (inst2_in3[7:0]),
    .inst2_in1     (inst2_in1),
    /*vlog_aide:auto_inst output ports*/
    .inst2_out0    (inst2_out0[15:0]),
    .inst2_out3    (inst2_out3[15:0]),
    .inst2_out1    (inst2_out1[15:0]),
    .inst2_out2    (inst2_out2[15:0])
    /*vlog_aide:auto_inst end*/
);

endmodule
```

```
module top(/*vlog_aide:auto_port*/);

inst1 dut1(/*vlog_aide:auto_inst inst1.v*/
    /*vlog_aide:auto_inst begin*/
    /*vlog_aide:auto_inst input ports*/
    .inst1_in3     (top_in3[5:0]),
    .inst1_in1     (top_in1),
    .inst1_in2     (top_in2),
    .inst1_in0     (top_in0),
    .inst1_in5     (top_in5[15:0]),
    /*vlog_aide:auto_inst output ports*/
    .inst1_out4    (top_out4[31:0]),
    .inst1_out0    (top_out0),
    .inst1_out2    (top_out2),
    .inst1_out3    (top_out3[15:0])
    /*vlog_aide:auto_inst end*/
);
inst2 #(8,16) dut2(/*vlog_aide:auto_inst inst2.v*/
    /*vlog_aide:auto_inst begin*/
    /*vlog_aide:auto_inst input ports*/
    .inst2_in2     (inst2_in2),
    .inst2_in5     (inst2_in5[7:0]),
    .inst2_in4     (inst2_in4[7:0]),
    .inst2_in0     (inst2_in0),
    .inst2_in3     (inst2_in3[7:0]),
    .inst2_in1     (inst2_in1),
    /*vlog_aide:auto_inst output ports*/
    .inst2_out0    (inst2_out0[15:0]),
    .inst2_out3    (inst2_out3[15:0]),
    .inst2_out1    (inst2_out1[15:0]),
    .inst2_out2    (inst2_out2[15:0])
    /*vlog_aide:auto_inst end*/
);

endmodule
```

AutoDefine

- Generate definitions of
  - Signals of sequential blocks
  - Signals of combinational blocks
  - Signals of assign statement
  - Signals of instances’ outputs
  - Signals of module’s outputs
- Support Verilog-2001
- Gvim Menu: Vlog-Utilities->Auto-Define
- Gvim Command: AutoDef
- Mark: /*vlog_aide:auto_define*/

AutoDefine

- Coding Style

```
always @(posedge clk)
   if (!reset_n)
      a <= #`FFD 0;
   else
      a <= #`FFD b + c;
```

✗

```
always @(posedge clk)
   if (!reset_n)
      a <= #`FFD 0;
   else
      a[7:0] <= #`FFD b + c;
```

✓

```
always @(posedge clk)
   if (!reset_n)
      a <= #`FFD 8'd0;
   else
      a <= #`FFD b + c;
```

✓

```
always @(posedge clk)
   if (!reset_n)
      a <= #`FFD 0;
   else
      a <= #`FFD b + 8'h1;
```

✓

```
always @(posedge clk)
   if (!reset_n)
      a[7:0] <= #`FFD 0;
   else
      a <= #`FFD b + c;
```

✓

```
reg signed [7:0] a;
always @(posedge clk)
   if (!reset_n)
      a <= #`FFD 0;
   else
      a <= #`FFD b + c;
```

✓

AutoInterface

- Gvim Menu: Vlog-Utilities->AutoInterface
- Gvim Command: AutoInf <inf_name> <begin> <end>
- Mark (optional):

```
/*vlog_aide:auto_inf <inf_name> begin*/
 ports;
 /*vlog_aide:auto_inf <inf_name> end*/
```

AutoGenerate

- Gvim Menu: Vlog-Utilities->Auto-Generate
- Gvim Command: AutoGen
- Mark:

```
/*vlog-aide:auto_gen <var1>=<begin>, <end>,
   <step>  begin
your code;
vlog-aide:auto_gen end*/
```

AutoGenerate – Delay Chain

```
always @(posedge clk) begin
    din_d0 <= #`FFD din;
    /*vlog_aide:auto_gen i=1,10,1 begin
    din_d$(i) <= din_d$(i-1);
    vlog_aide:auto_gen i, end*/
end
```

$\rightarrow$

```
always @(posedge clk) begin
    din_d0 <= #`FFD din;
    /*vlog_aide:auto_gen i=1,10,1 begin*/
    din_d1 <= din_d0;
    din_d2 <= din_d1;
    din_d3 <= din_d2;
    din_d4 <= din_d3;
    din_d5 <= din_d4;
    din_d6 <= din_d5;
    din_d7 <= din_d6;
    din_d8 <= din_d7;
    din_d9 <= din_d8;
    din_d10 <= din_d9;
/*vlog_aide:auto_gen end*/
end
```

AutoGenerate – PE Array

```
/*vlog_aide:auto_gen i=0,9,1 begin
pe_cell cell$(i)(.clk(clk), .din(din[$(i*8+7):0]), .dout(dout_$(i)[7:0]));
vlog_aide:auto_gen i, end*/
```

$\rightarrow$

```
/*vlog_aide:auto_gen i=0,9,1 begin*/
pe_cell cell0(.clk(clk), .din(din[7:0]), .dout(dout_0[7:0]));
pe_cell cell1(.clk(clk), .din(din[15:0]), .dout(dout_1[7:0]));
pe_cell cell2(.clk(clk), .din(din[23:0]), .dout(dout_2[7:0]));
pe_cell cell3(.clk(clk), .din(din[31:0]), .dout(dout_3[7:0]));
pe_cell cell4(.clk(clk), .din(din[39:0]), .dout(dout_4[7:0]));
pe_cell cell5(.clk(clk), .din(din[47:0]), .dout(dout_5[7:0]));
pe_cell cell6(.clk(clk), .din(din[55:0]), .dout(dout_6[7:0]));
pe_cell cell7(.clk(clk), .din(din[63:0]), .dout(dout_7[7:0]));
pe_cell cell8(.clk(clk), .din(din[71:0]), .dout(dout_8[7:0]));
pe_cell cell9(.clk(clk), .din(din[79:0]), .dout(dout_9[7:0]));
/*vlog_aide:auto_gen end*/
```

AutoGenerate – 2D Memory

```
always @(posedge clk) begin
   if (rst_n) begin
      /*vlog_aide:auto_gen i=0,3,1 j=0,2,1 begin
      mem_$(i)_$(j) = 8'h0;
      vlog_aide:auto_gen i,j, end*/
   end
   else begin
      case (addr_i)
      /*vlog_aide:auto_gen i=0,3,1 begin
      $(i) : begin
         case (addr_j)
         /*vlog_aide:auto_gen j=0,2,1 begin
         $(j) : mem_$(i)_$(j) = din;
         vlog_aide:auto_gen j, end*/
         endcase
      end
      vlog_aide:auto_gen i, end*/
      endcase
   end
end
```

→

```
always @(posedge clk) begin
   if (rst_n) begin
      /*vlog_aide:auto_gen i=0,3,1 j=0,2,1 begin*/
      mem_0_0 = 8'h0;
      mem_0_1 = 8'h0;
      mem_0_2 = 8'h0;
      mem_1_0 = 8'h0;
      mem_1_1 = 8'h0;
      mem_1_2 = 8'h0;
      mem_2_0 = 8'h0;
      mem_2_1 = 8'h0;
      mem_2_2 = 8'h0;
      mem_3_0 = 8'h0;
      mem_3_1 = 8'h0;
      mem_3_2 = 8'h0;
/*vlog_aide:auto_gen end*/
   end
   else begin
      case (addr_i)
      /*vlog_aide:auto_gen i=0,3,1 begin*/
      0 : begin
         case (addr_j)
         /*vlog_aide:auto_gen j=0,2,1 begin
         $(j) : mem_0_$(j) = din;
         vlog_aide:auto_gen j, end*/
         endcase
      end
      1 : begin
         case (addr_j)
         /*vlog_aide:auto_gen j=0,2,1 begin
         $(j) : mem_1_$(j) = din;
         vlog_aide:auto_gen j, end*/
         endcase
      end
      2 : begin
         case (addr_j)
         /*vlog_aide:auto_gen j=0,2,1 begin
         $(j) : mem_2_$(j) = din;
         vlog_aide:auto_gen j, end*/
         endcase
      end
      3 : begin
         case (addr_j)
         /*vlog_aide:auto_gen j=0,2,1 begin
         $(j) : mem_3_$(j) = din;
         vlog_aide:auto_gen j, end*/
         endcase
      end
/*vlog_aide:auto_gen end*/
      endcase
   end
end
```

→

```
always @(posedge clk) begin
   if (rst_n) begin
      /*vlog_aide:auto_gen i=0,3,1 j=0,2,1 begin*/
      mem_0_0 = 8'h0;
      mem_0_1 = 8'h0;
      mem_0_2 = 8'h0;
      mem_1_0 = 8'h0;
      mem_1_1 = 8'h0;
      mem_1_2 = 8'h0;
      mem_2_0 = 8'h0;
      mem_2_1 = 8'h0;
      mem_2_2 = 8'h0;
      mem_3_0 = 8'h0;
      mem_3_1 = 8'h0;
      mem_3_2 = 8'h0;
/*vlog_aide:auto_gen end*/
   end
   else begin
      case (addr_i)
      /*vlog_aide:auto_gen i=0,3,1 begin*/
      0 : begin
         case (addr_j)
         /*vlog_aide:auto_gen j=0,2,1 begin*/
         0 : mem_0_0 = din;
         1 : mem_0_1 = din;
         2 : mem_0_2 = din;
/*vlog_aide:auto_gen end*/
         endcase
      end
      1 : begin
         case (addr_j)
         /*vlog_aide:auto_gen j=0,2,1 begin*/
         0 : mem_1_0 = din;
         1 : mem_1_1 = din;
         2 : mem_1_2 = din;
/*vlog_aide:auto_gen end*/
         endcase
      end
      2 : begin
         case (addr_j)
         /*vlog_aide:auto_gen j=0,2,1 begin*/
         0 : mem_2_0 = din;
         1 : mem_2_1 = din;
         2 : mem_2_2 = din;
/*vlog_aide:auto_gen end*/
         endcase
      end
      3 : begin
         case (addr_j)
         /*vlog_aide:auto_gen j=0,2,1 begin*/
         0 : mem_3_0 = din;
         1 : mem_3_1 = din;
         2 : mem_3_2 = din;
/*vlog_aide:auto_gen end*/
         endcase
      end
/*vlog_aide:auto_gen end*/
      endcase
   end
end
```

Template

- Gvim Command: Tmp <template_file> <arg1> <arg2> …

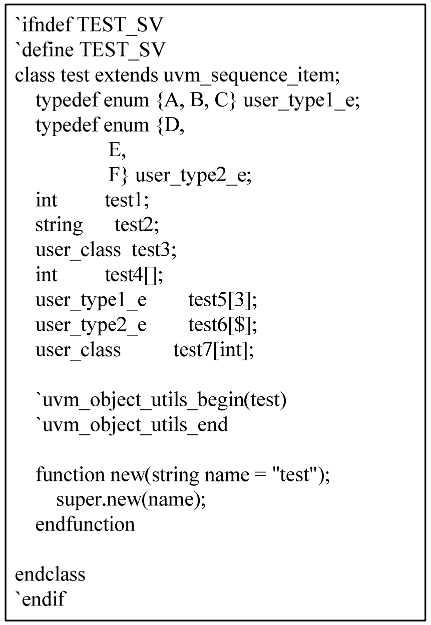
```
# flop(clk, reset)
always @(posedge $(clk)) begin
    if ($(reset)) begin

    end
    else begin

    end
end
```

UVM-Utilities

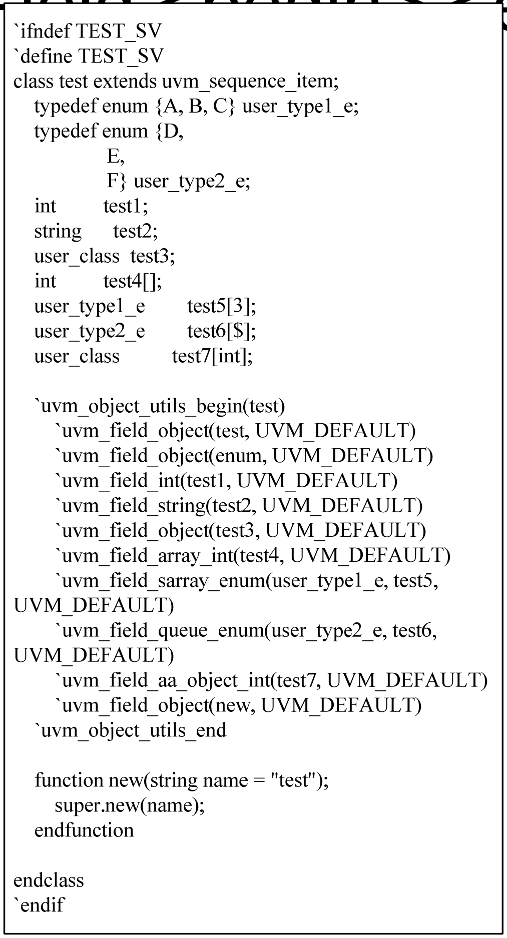
- UVM Templates
  - uvm_component
  - uvm_driver
  - uvm_agent
  - uvm_env
  - uvm_test
  - uvm_object
  - uvm_sequence_item
  - uvm_transaction
  - uvm_sequence
- Auto Field

AutoField

- Gvim Menu: UVM-Utilities->AutoField
- Gvim Command: AutoField <begin> <end> <flag>



```
`ifndef TEST_SV
`define TEST_SV
class test extends uvm_sequence_item;
   typedef enum {A, B, C} user_type1_e;
   typedef enum {D,
              E,
              F} user_type2_e;
   int        test1;
   string      test2;
   user_class  test3;
   int        test4[];
   user_type1_e        test5[3];
   user_type2_e        test6[$];
   user_class        test7[int];

   `uvm_object_utils_begin(test)
   `uvm_object_utils_end

   function new(string name = "test");
      super.new(name);
   endfunction

endclass
`endif
```



```
`ifndef TEST_SV
`define TEST_SV
class test extends uvm_sequence_item;
   typedef enum {A, B, C} user_type1_e;
   typedef enum {D,
              E,
              F} user_type2_e;
   int        test1;
   string      test2;
   user_class  test3;
   int        test4[];
   user_type1_e        test5[3];
   user_type2_e        test6[$];
   user_class        test7[int];

   `uvm_object_utils_begin(test)
      `uvm_field_object(test, UVM_DEFAULT)
      `uvm_field_object(enum, UVM_DEFAULT)
      `uvm_field_int(test1, UVM_DEFAULT)
      `uvm_field_string(test2, UVM_DEFAULT)
      `uvm_field_object(test3, UVM_DEFAULT)
      `uvm_field_array_int(test4, UVM_DEFAULT)
      `uvm_field_sarray_enum(user_type1_e, test5,
UVM_DEFAULT)
      `uvm_field_queue_enum(user_type2_e, test6,
UVM_DEFAULT)
      `uvm_field_aa_object_int(test7, UVM_DEFAULT)
      `uvm_field_object(new, UVM_DEFAULT)
   `uvm_object_utils_end

   function new(string name = "test");
      super.new(name);
   endfunction

endclass
`endif
```