BTF List Compiled by Cliff Cummings, Chairperson of BTF - October 10, 2001

I guess the only perk I get as chair of the BTF is prioritizing which mistakes are most critical. The following is the list of changes with assigned priority. In some cases, the changes (designated as **BTF-PR##**) are in response to Problem Reports being tracked on the BTF Gnats Reflector, being maintained by Stefen Boyd.

Regards - Cliff Cummings

CRITICAL CHANGES

I consider these changes to be critical. As chair of the task force that proposed these enhancements to the Verilog Standard, I will be publicly very critical of the published IEEE document if these corrections are not made before publication.

Somehow, between the second-to-last DRAFT document and last draft document, All occurrences of nonblocking were changed to non blocking. The correct spelling per the 1364-1995 Standard is nonblocking without any space or hyphen. I have examined the entire 1364-2001 PDF file and fortunately, every single occurrence of "on blocking" (with either space or hypen) can be replaced with "onblocking" to globally fix the problem in the entire document.

(Global change)
REPLACE: on blocking
WITH: onblocking

(Global change)
REPLACE: on-blocking
WITH: onblocking
(NOTE: all spellings of nonblocking should not have spaces nor hyphens. This is consistent with the 1995 version of the IEEE Standard))

Page 25 - Section 3.5 - add paragraph:
(Due to a number of unfortunate circumstances, vendors are already implementing implicit net declarations incorrectly - we must add the following indented paragraph to the end of section 3.5 to clarify the implicit wire declaration enhancement. Indenting to match the indenting of the two last paragraphs already in section 3.5)
ADD:
If an identifier appears on the left-hand side of a continuous assignment statement, and that identifier has not been declared previously, an implicit scalar net declaration of the default net type is assumed.

Page 140 - Section 9.7.5 - 4th paragraph:
(This was a requested correction that was sent to Yatin on 04/25/2001 to be changed in Draft 6 before it was sent to the IEEE - The following replacement provides critical clarification on how the @* implicit event_expression list works)

REPLACE:
Net and variables which appear on the right hand side of assignments, in function and task calls, or case and conditional expressions shall all be included by these rules.
WITH:
Nets and variables which appear on the right hand side of assignments, in function and task calls, in case and conditional expressions, as an index variable on the left hand side of assignments or as variables in case item expressions shall all be included by these rules.

Page 141 - Section 9.7.5 - Add Examples 5 & 6 just before section 9.7.6:

(This was a requested correction that was sent to Yatin on 04/25/2001 to be added to Draft 6 before it was sent to the IEEE - The following two examples provide critical clarification on how the @* implicit event_expression list works)

Example 5

```
always @*  begin // same as @(a or en)
  y = 8'hff;
  y[a] = !en;
end
```

Example 6

```
always @* begin // same as @(state or go or ws)
  next = 4'b0;
  case (1'b1)
    state[IDLE] : if (go)  next[READ] = 1'b1;
                  else     next[IDLE] = 1'b1;
    state[READ] :          next[DLY]  = 1'b1;
    state[DLY]  : if (!ws) next[DONE] = 1'b1;
                  else     next[READ] = 1'b1;
    state[DONE] :          next[IDLE] = 1'b1;
  endcase
end
```

Page 161 - Section 10.3.5 - Example: ram_model
(This was a requested correction that was sent to Yatin on 04/25/2001 to be added to Draft 6 before it was sent to the IEEE - The clogb2 function is very broken in this example in Draft 6. The corrected clogb2 function has been tested and works, even with Verilog-1995)
REPLACE:

```
module ram_model (address, write, chip_select, data);
 parameter data_width = 8;
 parameter ram_depth = 256;
 localparam adder_width = clogb2(ram_depth);
 input [adder_width - 1:0] address;
 input write, chip_select;
 inout [data_width - 1:0] data;

  //define the clogb2 function
 function integer clogb2;
  input depth;
  integer i,result;
  begin
   for (i = 0; 2 ** i < depth; i = i + 1)
       result = i + 1;
       clogb2 = result;
  end
 endfunction

 reg [data_width - 1:0] data_store[0:ram_depth - 1];
  //the rest to the ram model
```

WITH:

```
module ram_model (address, write, chip_select, data);
 parameter data_width = 8;
 parameter ram_depth = 256;
 localparam adder_width = clogb2(ram_depth);
 input [adder_width - 1:0] address;
 input write, chip_select;
 inout [data_width - 1:0] data;
```

```
  //define the clogb2 function
 function integer clogb2;
  input [31:0] value;
  for (clogb2=0; value>0; clogb2=clogb2+1)
       value = value>>1;
 endfunction

 reg [data_width - 1:0] data_store[0:ram_depth - 1];
  //the rest to the ram model
```

Pages 178-179 - Section 12.1.3.4 - Example 8 (Replace the entire example with the following):
(Apologies - Example 8 in Draft 6 was full of errors and problems - we suggest replacing it with the following model that has been reasonably tested to the extent that we could replicate Verilog-2001 functionality with near-equivalent constructs in Verilog-1995 - The existing Draft 6 example has no less that 17 typos and syntax errors. Readers of the Draft 6 example will become very confused unless the typos are fixed as shown below)
REPLACE EXAMPLE 8 WITH:

```
module dimm (adr, ba, rasx, casx, csx, wex, dqm, cke, data, clk, dev_id);
  parameter [31:0] MEM_SIZE  = 8,  // in mbytes
                   MEM_WIDTH = 16;
  input [10:0] adr;
  input        ba;
  input        rasx, casx, csx, wex;
  input [ 7:0] dqm;
  input        cke;
  inout [63:0] data;
  input        clk;
  input [ 4:0] dev_id;

  genvar       i;

  generate
    case ({MEM_SIZE, MEM_WIDTH})
      {32'd8, 32'd16}: // 8Meg x 16 bits wide.
      begin
        // The generated instance names are word[3].p, word[2].p,
        // word[1].p, word[0].p, and the task read_mem
        for (i=0; i<4; i=i+1) begin:word
          sms_16b216t0 p (.clk(clk), .csb(csx), .cke(cke),
              .ba(ba[0]), .addr(adr[10:0]), .rasb(rasx),
              .casb(casx), .web(wex), .udqm(dqm[2*i+1]),
              .ldqm(dqm[2*i]), .dqi(data[15+16*i:16*i]),
              .dev_id(dev_id[4:0]));
        end
        task read_mem;
          input  [31:0] address;
          output [63:0] data;
          begin
            word[3].p.read_mem(address, data[63:48]);
            word[2].p.read_mem(address, data[47:32]);
            word[1].p.read_mem(address, data[31:16]);
            word[0].p.read_mem(address, data[15:0]);
          end
        endtask
      end

      {32'd16, 32'd8}: // 16Meg x 8 bits wide.
```

```
        begin
          // The generated instance names are byte[7].p, byte[6].p,
          // byte[5].p, byte[4].p, byte[3].p, byte[2].p, byte[1].p,
          // byte[0].p and the task read_mem
          for (i=0; i<8; i=i+1) begin:byte
            sms_16b208t0 p (.clk(clk), .csb(csx), .cke(cke),
                .ba(ba[0]), .addr(adr[10:0]), .rasb(rasx),
                .casb(casx), .web(wex), .dqm(dqm[i]),
                .dqi(data[7+8*i:8*i]),.dev_id(dev_id[4:0]));
          end
          task read_mem;
            input  [31:0] address;
            output [63:0] data;
            begin
              byte[7].p.read_mem(address, data[63:56]);
              byte[6].p.read_mem(address, data[55:48]);
              byte[5].p.read_mem(address, data[47:40]);
              byte[4].p.read_mem(address, data[39:32]);
              byte[3].p.read_mem(address, data[31:24]);
              byte[2].p.read_mem(address, data[23:16]);
              byte[1].p.read_mem(address, data[15:8]);
              byte[0].p.read_mem(address, data[7:0]);
            end
          endtask
        end
      // Other memory cases ...
    endcase
  endgenerate
endmodule
```

Bullet Lists?
What happened to the bulleted lists between Draft 6 and this IEEE copy?? Are they going to be restored (Examples: Page 179 - section 12.2 - list after 2nd paragraph / Page 354 - Section 19.4 - list at bottom of page 354 / I'm sure there are others - do I need to find all of them or can this be done easily?)

The Index is missing
I assume that it will be generated for the final document?

IMPORTANT ERRORS IN THE STANDARD

These are syntax, wording and reference errors that should be corrected before publication. This goes to the professionalism of the IEEE and its effort to produce an error-free document.

BTF-PR8: Page 46 - Section 4.1.7 - 3rd & 2nd to last paragraphs
REPLACE: When two operands of unequal bit lengths are used, the smaller operand shall be zero filled on the most significant bit side to extend to the size of the larger operand.
WITH: When two operands of unequal bit lengths are used and one or both of the operands is unsigned, the smaller operand shall be zero filled on the most significant bit side to extend to the size of the larger operand.

REPLACE: When both operands of a relational expression are signed integral operands (an integer, or a unsized, unbased integer) then the expression shall be interpreted as a comparison between signed values. When either operand of a relational expression is a real operand then the other operand shall be converted to an equivalent real value, and the expression shall be interpreted as a comparison between two real values.

WITH: When both operands of a relational expression are signed ~~integral~~ operands (an integer, a signed reg data type, or an unsized, unbased integer) then the expression shall be interpreted as a comparison between signed values. When either operand of a relational expression is a real operand then the other operand shall be converted to an equivalent real value, and the expression shall be interpreted as a comparison between two real values.

BTF-PR18: Page 60 - Section 4.4.1 - 8th row of Table 29, 1st column - missing >>> and <<< operators
REPLACE:
i op j, where op is:
>> << **
WITH:
i op j, where op is:
>> << ** >>> <<<

BTF-PR10: Page 80 - Section 7.1.6 - Port declarations for Example 3
(This is an obvious coding mistake - previously identified and should have been fixed in Draft 6)
REPLACE:
```
module busdriver (busin, bushigh, buslow, enh, enl);
input [15:0] in;
output [7:0] bushigh, buslow;
input enh, enl;
```
WITH:
```
module busdriver (busin, bushigh, buslow, enh, enl);
input [15:0] busin;
output [7:0] bushigh, buslow;
input enh, enl;
```

BTF-PR36: Page 139 - 4th line under syntax box 9-9
REPLACE: Its occurrence can be recognized by using the event control syntax described in
WITH: Its occurrence can be recognized by using the event control syntax described in 9.7.2
(NOTE: in the PDF document, there appears to be an empty hyperlink where 9.7.2 should be displayed)

BTF-PR7: Page 169 - Section 12.1.3 - 7th paragraph - Typo
REPLACE: Parameter redefinition using ~~by the~~ ordered or named parameter = value assignment or defparam statements can also be declared within the generate scope.
WITH: Parameter redefinition using ordered or named parameter = value assignment or defparam statements can also be declared within the generate scope.

BTF-PR39: Page 186 - Section 12.3.3 - last line on bottom of page 186 - Mistake
REPLACE: The inputs are ored together
WITH: The inputs are tied together

BTF-PR26: Page 222 - Syntax box 14-6 - 5th line from the bottom - end of line - missing ,
Same problem: Page 726 - Section A.7.4 - 8th line from the top - end of line - missing ,
REPLACE:
```
| t01_path_delay_expression , t10_path_delay_expression , t0z_path_delay_expression ,
  tz1_path_delay_expression , t1z_path_delay_expression , tz0_path_delay_expression
  t0x_path_delay_expression , tx1_path_delay_expression , t1x_path_delay_expression ,
  tx0_path_delay_expression , txz_path_delay_expression , tzx_path_delay_expression
```
WITH: (bold font for the missing ,)
```
| t01_path_delay_expression , t10_path_delay_expression , t0z_path_delay_expression ,
  tz1_path_delay_expression , t1z_path_delay_expression , tz0_path_delay_expression ,
  t0x_path_delay_expression , tx1_path_delay_expression , t1x_path_delay_expression ,
  tx0_path_delay_expression , txz_path_delay_expression , tzx_path_delay_expression
```

Page 322 - Section 17.10.2 - *Examples* - reg declaration syntax error
REPLACE: **reg** `8*32:1 testname;`
WITH: **reg** `[8*32:1] testname;`

BTF-PR15: Page 354 - Section 19.4 - 2nd list item at the bottom of the page - Typo
REPLACE: the the
WITH: the

FORMATTING ERRORS - LESS IMPORTANT CHANGES

These corrections have mostly to do with fonts and are requested only if the IEEE is willing to make changes to be consistent with the descriptions set forth in section 1. There are probably hundreds of these. These are the ones that have been reported to the Behavioral Task Force. These changes are not critical to publication of the IEEE Verilog-2001 Standard at this time.

Page iii - 2nd paragraph after "INTRODUCTION"
"The Verilog <funny character> ..."
Is this funny character really supposed to be here?

BTF-PR29 - Page 7 - Syntax 2-1 and
BTF-PR29 - Page 716 - Section A.8.7
(The second [ ]'s enclosing size should NOT be bold)
REPLACE:
decimal_number ::=
unsigned_number
| **[** size **]** decimal_base unsigned_number
| **[** size **]** decimal_base x_digit { _ }
| **[** size **]** decimal_base z_digit { _ }
binary_number ::= **[** size **]** binary_base binary_value
octal_number ::= **[** size **]** octal_base octal_value
hex_number ::= **[** size **]** hex_base hex_value
WITH:
decimal_number ::=
unsigned_number
| [size] decimal_base unsigned_number
| [size] decimal_base x_digit { _ }
| [size] decimal_base z_digit { _ }
binary_number ::= [size] binary_base binary_value
octal_number ::= [size] octal_base octal_value
hex_number ::= [size] hex_base hex_value

Page 36 - *Examples* formatting
The comment for the last parameter declaration in the example wraps to the next line

BTF-PR24 - Page 37 - Syntax 3-5 and
BTF-PR24 - Page 229 - Syntax 14-7 and
BTF-PR24 - Page 716 - Section A.2.4
(The second $ should be bold)
REPLACE:
| **PATHPULSE$**specify_input_terminal_descriptor$specify_output_terminal_descriptor
= (reject_limit_value [**,** error_limit_value] **) ;**
WITH:
| **PATHPULSE$**specify_input_terminal_descriptor**$**specify_output_terminal_descriptor
= (reject_limit_value [**,** error_limit_value] **) ;**

BTF-PR31 - Page 193 - Syntax 12-7 and
BTF-PR31 - Page 734 - Section A.9.4
(The inner [ ]'s should be bold)
REPLACE:
simple_hierarchical_branch3 ::=
    simple_identifier [[unsigned_number]]
        [{ **.**simple_identifier [[unsigned_number]] }]
escaped_hierarchical_branch4 ::=
    escaped_identifier [[unsigned_number]]
        [{ **.**escaped_identifier [[unsigned_number]] }]
WITH:

simple_hierarchical_branch3 ::=
    simple_identifier [[unsigned_number]]
        [{ .simple_identifier [[unsigned_number]] }]
escaped_hierarchical_branch4 ::=
    escaped_identifier [[unsigned_number]]
        [{ .escaped_identifier [[unsigned_number]] }]

Page 357 - Section 19.5 - paragraph before "*Examples*"
The word "**the**" should not be bold.

OTHER ISSUES - NO CHANGE TO BE MADE AT THIS TIME

Other BTF Problem reports need to be investigated. Some request new wording to replace existing wording. Some request more examples. Some require more investigation, proposals and voting by the BTF.

NOTE: Section 4.1.8 does not describe equality comparisons between two operands of different sizes (signed or unsigned). I believe wording about operands of different sizes was removed because establishing equality between two unsigned variables is not bit-width dependent. Establishing equality between two signed variables is similarly not bit-width dependent, and testing equality between two operands where one is signed and the other is not, requires that the signed operand be treated as an unsigned operand, defaulting to the case of testing equality between two unsigned operands. Is this correct?

NOTE: Section 4.1.10 discusses bit-wise operations, and also states that a shorter operand will be zero-filled. Again, I think the assumption was that bit-wise operations are performed on unsigned operands, even if both operands were declared to be signed. Whether or not an engineer should perform bit-wise operations on signed variables is a whole other question.

BTF-PR9 - Truncation Description Missing
I think Paul is right. This is so simple that we may have overlooked it in both the 1995 and 2001 Standards. I cannot find a description of truncation anywhere, not even Section 2.5.1 on Integer constants. In 2.5.1 the Standard describes "Automatic left padding" but does not describe intentional or unintentional truncation. I also can't find truncation discussed in the section on assignments. Paul is right that MSBs truncate, and many of us teach this in our classes, but I can't find the official description in the IEEE Standard. I would not withhold publication because of this omission (the rules are generally understood) but Paul is right that a description should be included in the Standard.

Can anyone find a description in the Standard for truncation?

BNF Fixes Might Be Required - But we don't have time to investigate and do them right now
BTF-PR12
BTF-PR20
BTF-PR21
BTF-PR22
BTF-PR23
BTF-PR25
BTF-PR28
BTF-PR29
BTF-PR30
BTF-PR32
BTF-PR33
BTF-PR34
BTF-PR37
BTF-PR43