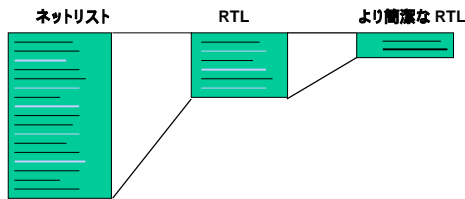
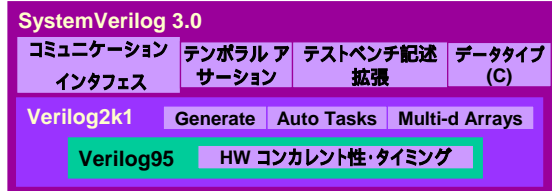


こんな話も聞けたりします！
 これからのデザインの必須アイテム = 新しいRTL設計検証言語ってどうよ？

SystemVerilog 3.0

より良いデザイン記述は検証しやすい記述をも意味する



より簡潔に書かれた合成可能なRTLは
 一般に 1/3 - 1/5 の記述量の削減となる



第11回 FPGA/PLD Design Conference

SystemVerilog 記述例(1)・構造体

```
typedef struct {
    byte R,G,B;
} RGB;

const RGB BLUE = {0,0,255};
RGB Frame[639:0][479:0];
Frame[x][y] = BLUE;
```

定数リテラル
 二次元アレイ
 コピー

SystemVerilog 記述例(3) : always_*

- 設計意図を明確に区別
 - always_comb
 - always_latch
 - always_ff

```
always_comb begin
    tmp1 = a & b;
    tmp2 = c & d;
    y = tmp1 ! tmp2;
end

always_comb
    if (en) q <= d;

always_latch
    if (en) q <= d;
    else q <= q;
```

正しい使用法
 順序回路が検測されるのでエラー
 フィードバックが検測されるのでエラー
 正しい使用法
 センシティブリストが不明確なのでエラー

SystemVerilog 記述例(2)・interface

```
module sqart_m (SX, sx_soc, SX_en, SX_w_data, SX_w_clk,
    SX_w_clk,
    SX_cpu_BusMode, SX_cpu_Addr, SX_cpu_Sel, SX_cpu_Data,
    SX_cpu_Rd_Dst, SX_cpu_Wr_Rst, SX_cpu_Rdy_Dstck, rd, ck);
    input SX_soc;
    input [7:0] SX_cpu_Addr;
    input SX_cpu_Sel;
    input [7:0] SX_cpu_Data;
    input SX_cpu_Rd_Dst;
    input SX_cpu_Wr_Rst;
    input SX_cpu_Rdy_Dstck;
    input rd;
    input ck;
    wire SX_w_soc;
    wire SX_w_sel;
    wire [7:0] SX_w_data;
    wire SX_w_clk;
    wire SX_w_rst;
    wire [7:0] SX_cpu_Addr;
    wire SX_cpu_Sel;
    wire [7:0] SX_cpu_Data;
    wire SX_cpu_Rd_Dst;
    wire SX_cpu_Wr_Rst;
    wire SX_cpu_Rdy_Dstck;
    wire rd;
    wire ck;
endmodule
```

```
interface utopia_1
    wire soc; // start of cell
    wire en; // enable
    wire [7:0] data; // data
    wire clk; // cell enable
    // ATM layer clock
endinterface

interface cpu_1(input bit rst;
    wire BusMode;
    logic [11:0] Addr;
    logic Sel;
    wire [7:0] Data;
    logic Rd_Dst;
    logic Wr_Rst;
    wire Rdy_Dstck;
endinterface

module sqart_m(utopia_1 ut, cpu_1 cpu,
    input bit rst);
endmodule
```

Verilog95
 SystemVerilog
 1/3 以下へ記述量を削減
 記述ミスによるインタフェ
 イス誤りの低減



第11回 FPGA/PLD Design Conference