Outros Módulos: IPs do Quartus para a Criação de Multiplicadores e Divisores

1 - Scripts em Verilog referentes aos MULTIPLICADORES

1.1 - `signed_multiplier`

- Multiplicador para VALORES SINALIZADOS.

```
// synopsys translate_off
``timescale 1 ps / 1 ps

// synopsys translate_on
module signed_multiplier #( parameter DATA_WIDTH=32, parameter END_IDX=DATA_WIDTH-1, parameter END_IDX2=
                  ( dataa, datab, result );
//----------------------------------------------------------------------------------------
    input   signed [END_IDX:0]  dataa;
    input   signed [END_IDX:0]  datab;
    output[END_IDX2:0] result;
//----------------------------------------------------------------------------------------
    wire [END_IDX2:0] sub_wire0;
    wire [END_IDX2:0] result = sub_wire0[END_IDX2:0];

    lpm_mult    lpm_mult_component (
                .dataa( dataa ),
                .datab( datab ),
                .result( sub_wire0 ),
                .aclr( 1'b0 ),
                .clken( 1'b1 ),
                .clock( 1'b0 ),
                .sclr( 1'b0 ),
                .sum( 1'b0 )  );
    defparam
        lpm_mult_component.lpm_hint = "MAXIMIZE_SPEED=5",
        lpm_mult_component.lpm_representation = "SIGNED",
        lpm_mult_component.lpm_type = "LPM_MULT",
        lpm_mult_component.lpm_widtha = DATA_WIDTH,
        lpm_mult_component.lpm_widthb = DATA_WIDTH,
        lpm_mult_component.lpm_widthp = (DATA_WIDTH*2);
endmodule
```

1.2 - `unsigned_multiplier`

- Multiplicador para VALORES NÃO-SINALIZADOS.

```
// synopsys translate_off
``timescale 1 ps / 1 ps
```

```
// synopsys translate_on
module unsigned_multiplier #( parameter DATA_WIDTH=32, parameter END_IDX=DATA_WIDTH-1, parameter END_ID
                    ( dataa, datab, result );
//-----------------------------------------------------------------------------------------
    input   [END_IDX:0]  dataa;
    input   [END_IDX:0]  datab;
    output[END_IDX2:0]  result;
//-----------------------------------------------------------------------------------------
    wire [END_IDX2:0] sub_wire0;
    wire [END_IDX2:0] result = sub_wire0[END_IDX2:0];

    lpm_mult    lpm_mult_component (
                .dataa( dataa ),
                .datab( datab ),
                .result( sub_wire0 ),
                .aclr( 1'b0 ),
                .clken( 1'b1 ),
                .clock( 1'b0 ),
                .sclr( 1'b0 ),
                .sum( 1'b0 ) );
    defparam
        lpm_mult_component.lpm_hint = "MAXIMIZE_SPEED=5",
        lpm_mult_component.lpm_representation = "UNSIGNED",
        lpm_mult_component.lpm_type = "LPM_MULT",
        lpm_mult_component.lpm_widtha = DATA_WIDTH,
        lpm_mult_component.lpm_widthb = DATA_WIDTH,
        lpm_mult_component.lpm_widthp = (DATA_WIDTH*2);
endmodule
```

---

2 - Scripts em Verilog referentes aos DIVISORES

2.1 - `divide_remainder_signed`

- Divisor para VALORES SINALIZADOS.

```
// synopsys translate_off
``timescale 1 ps / 1 ps


// synopsys translate_on
module divide_remainder_signed #( parameter DATA_WIDTH=32, parameter END_IDX=DATA_WIDTH-1 )
                                ( denom, numer, quotient, remain );
//-------------------------------------------------------------------------------------------------
    input   signed [END_IDX:0]  denom;
    input   signed  [END_IDX:0]  numer;
    output [END_IDX:0]  quotient;
    output [END_IDX:0]  remain;
//-------------------------------------------------------------------------------------------------
    wire [END_IDX:0] sub_wire0;
    wire [END_IDX:0] sub_wire1;
    wire [END_IDX:0] quotient = sub_wire0[END_IDX:0];
    wire [END_IDX:0] remain = sub_wire1[END_IDX:0];
    lpm_divide  LPM_DIVIDE_component (
                .denom( denom ),
                .numer( numer ),
                .quotient( sub_wire0 ),
                .remain( sub_wire1 ),
                .aclr( 1'b0 ),
                .clken( 1'b1 ),
                .clock( 1'b0 )     );
    defparam
        LPM_DIVIDE_component.lpm_drepresentation = "SIGNED",
        LPM_DIVIDE_component.lpm_hint = "LPM_REMAINDERPOSITIVE=TRUE",
        LPM_DIVIDE_component.lpm_nrepresentation = "SIGNED",
        LPM_DIVIDE_component.lpm_type = "LPM_DIVIDE",
        LPM_DIVIDE_component.lpm_widthd = DATA_WIDTH,
        LPM_DIVIDE_component.lpm_widthn = DATA_WIDTH;
endmodule
```

2.2 - divide_remainder_unsigned

- Divisor para VALORES NÃO-SINALIZADOS.

```
// synopsys translate_off
``timescale 1 ps / 1 ps


// synopsys translate_on
module divide_remainder_unsigned #( parameter DATA_WIDTH=32, parameter END_IDX=DATA_WIDTH-1 )
                                  ( denom, numer, quotient, remain );
//----------------------------------------------------------------------------------------------------
    input    [END_IDX:0]  denom;
    input    [END_IDX:0]  numer;
    output [END_IDX:0]  quotient;
    output [END_IDX:0]  remain;
//----------------------------------------------------------------------------------------------------
    wire [END_IDX:0] sub_wire0;
    wire [END_IDX:0] sub_wire1;
    wire [END_IDX:0] quotient = sub_wire0[END_IDX:0];
    wire [END_IDX:0] remain = sub_wire1[END_IDX:0];

    lpm_divide  LPM_DIVIDE_component (
                .denom( denom ),
                .numer( numer ),
                .quotient( sub_wire0 ),
                .remain( sub_wire1 ),
                .aclr( 1'b0 ),
                .clken( 1'b1 ),
                .clock( 1'b0 )   );
    defparam
        LPM_DIVIDE_component.lpm_drepresentation = "UNSIGNED",
        LPM_DIVIDE_component.lpm_hint = "LPM_REMAINDERPOSITIVE=TRUE",
        LPM_DIVIDE_component.lpm_nrepresentation = "UNSIGNED",
        LPM_DIVIDE_component.lpm_type = "LPM_DIVIDE",
        LPM_DIVIDE_component.lpm_widthd = DATA_WIDTH,
        LPM_DIVIDE_component.lpm_widthn = DATA_WIDTH;
endmodule
```