Implementation of DSP48E1

Ajinkya.S.Raghuwanshi

JULY 2021

1 Introduction

The DSP48E1 slice supports many independent functions. These functions include multiply, multiply accumulate (MACC), multiply add, three-input add, barrel shift, widebus multiplexing, magnitude comparator, bitwise logic functions, pattern detect, and wide counter. The architecture also supports cascading multiple DSP48E1 slices to form wide math functions, DSP filters, and complex arithmetic without the use of general FPGA logic

2 Procedure for Implementation:

1. Techmap will provide the DSP48E1 in form of its instances.

Code snippet:

```
module Multiplier (output reg P,reg PATTERNDETECT,..... input A,B, INMODE,ALUMODE,
CARRYIN,RSTA,RSTB,...);
  parameter [4:0] IS_C_INVERTED = 5'b0;
  generate case (|IS_C_INVERTED)
    5'b0: always @(posedge C) if (R) P<= 1'b0; else if (ALUMODE == 4’b0000) P <= A*B;
    5'b1: always @(posedge C) if (R) P <= 1'b0; else if (ALUMODE == 4’b0001) P <=
ACIN*B;
...
...
...
...
...
  endcase endgenerate
Endmodule
…
…
..
```

2. The architectural description of the DSP is to be added in the ”arch.timing.xml” file, it will included the inputs, outputs, interconnects, sub-types, etc of the DSP.

Code snippet:

```
 <! -- Defining the DSP -->
    <pb\_dsp name="DSP_block" >
    <!-- Input ports -->
       <input_ports>
          <port name = "A" num_pins="30">
          <port name = "B" num_pins="18">
          <port name = "C" num_pins="48">
```

```
    <port name = "D" num_pins="25">
    <port name = "OPMODES" num_pins="7">
    <port name = "ALUMODES" num_pins="4">
    <port name = "CARRYIN" num_pins="1">
    <port name = "CARRYSEL" num_pins="3">
    <port name = "INMODE" num_pins="5">
    <port name = "CEA1" num_pins="1">
    <port name = "CEA2" num_pins="1">
    <port name = "CEB1" num_pins="1">
    <port name = "CEB2" num_pins="1">
    <port name = "CEC" num_pins="1">
    <port name = "CED" num_pins="1">
    <port name = "CEM" num_pins="1">
    <port name = "CEP" num_pins="1">
    <port name = "CEAD" num_pins="1">
    <port name = "CEALUMODE" num_pins="1">
    <port name = "CECTRL" num_pins="1">
    <port name = "CECARRYIN" num_pins="1">
    <port name = "CEINMODE" num_pins="1">
    <port name = "RSTA" num_pins="1">
    <port name = "RSTB" num_pins="1">
    <port name = "RSTC" num_pins="1">
    <port name = "RSTD" num_pins="1">
    <port name = "RSTM" num_pins="1">
    <port name = "RSTP" num_pins="1">
    <port name = "RSTCTRL" num_pins="1">
    <port name = "RSTALLCARRYIN" num_pins="1">
    <port name = "RSTALUMODE" num_pins="1">
    <port name = "RSTINMODE" num_pins="1">
    <port name = "CLK" num_pins="1">
    <port name = "ACIN" num_pins="30">
    <port name = "BCIN" num_pins="18">
    <port name = "PCIN" num_pins="48">
    <port name = "CARRYCASCIN" num_pins="1">
    <port name = "MULTSIGNIN" num_pins="1">
  </input_ports>
  <!-- Output port -->
  <output_ports>
    <port name = "ACOUT" num_pins="30">
    <port name = "BCOUT" num_pins="18">
    <port name = "PCOUT" num_pins="48">
    <port name = "P" num_pins="48">
    <port name = "CARRYOUT" num_pins="4">
    <port name = "CARRYCASCOUT" num_pins="1">
    <port name = "MULTSIGNOUT" num_pins="1">
    <port name = "PATTERNDETECT" num_pins="1">
    <port name = "PATTERNBDETECT" num_pins="1">
    <port name = "UNDERFLOW" num_pins="1">
```

```
    <port name = "OVERFLOW" num_pins="1">
  </output_port>
  <!-- Mapping A, D, and Pre-adder Logic>
     <models>
  <mode name="ALU_OPMODE_mode">
  <pb\_type name="A_D_PREADDER" num\_pb="1" blif_model=".subcircuit A_D_PRE">
       <!-- Sub type Inputs -->
       <input name="A_REG" num_pins="30">
       <input name="ACIN_REG" num_pins="30">
       <input name="D_REG" num_pins="25">
       <!-- Sub type Output>
       <output name="ACOUT_REG" num_pins="30">
       <output name="X_MUX_REG" num_pins="30">
       <output name="A_MULT_REG" num_pins="25">
       <equivalent_sites>
          <site pb_type="ADPRE_0" pin_mapping="custom">
            <direct from="A" to="A_REG"/>
            <direct from="ACIN" to="ACIN_REG"/>
            <direct from="D" to="D_REG">
            <direct from="ACOUT_REG" to="ACOUT"/>
            <direct from="X_MUX_REG" to="X_MUX"/>
            <direct from="A_MULT_REG" to="A_MULT"/>
          </site>
        </equivalent_sites>
     </pb\_type>
  </mode>
  ..
  ...
  …
     </models>
</pb\_dsp>
```

3. BLIF (model) file contains the technology on which the FPGA is implemented.

Code snippet:

```
.model top
.inputs clk,
.outputs led[0] led[1] led[2] led[3]
.subckt GND GND=$false
.subckt VCC VCC=$true
.subckt VCC VCC=$undef
.names counter[0] $auto$alumacc.cc:485:replace_alu$1415.Y[0]
0 1
.subckt CARRY4_VPR CO0=$abc$2028$aiger2027$38 CO1=$abc$2028$aiger2027$39
CO2=$abc$2028$aiger2027$40 CO3=$abc$2028$aiger2027$41 DI0=$true DI1=$false
DI2=$false DI3=$false O0=$abc$2028$aiger2027$34
O1=$auto$alumacc.cc:485:replace_alu$1415.Y[1]
```

```
O2=$auto$alumacc.cc:485:replace_alu$1415.Y[2]
O3=$auto$alumacc.cc:485:replace_alu$1415.Y[3]
S0=$auto$alumacc.cc:485:replace_alu$1415.Y[0] S1=counter[1] S2=counter[2]
S3=counter[3]
.cname $auto$alumacc.cc:485:replace_alu$1415.genblk1.slice[0].genblk1.carry4
.attr module_not_derived 00000000000000000000000000000001
.attr src
```

4. Yosys will be used for the synthesis of the model.

Code snippet:

```
  DSP48E1  sum1 (
  .CLK(clk),          // input wire CLK
  .CARRYIN(1'b0),
  .A({{12{1'b0}},a}),             // input wire [17 : 0] A
  .C({48{1'b0}}),
  .P(p),            // output wire [47 : 0] P
  .B(b),              // input wire [17:0] B
  .D({25{1'b1}}),
  .OPMODE(7'b1001000),
  .ALUMODE(4'b0000),
  .CARRYINSEL(3'b000),
  .INMODE(4'b0000),
  .CEA1(1'b1),
  .CEA2(1'b1),
  .CEB1(1'b1),
  .CEB2(1'b1),
  .CEC(1'b1),
  .CED(1'b1),
  .CEM(1'b1),
  .CEP(1'b1),
  .CEAD(1'b1),
  .CEALUMODE(1'b1),
  .CECTRL(1'b1),
  .CECARRYIN(1'b1),
  .CEINMODE(1'b1),
  .RSTA(1'b0),
  .RSTB(1'b0),
  .RSTC(1'b0),
  .RSTD(1'b0),
  .RSTM(1'b0),
  .RSTP(1'b0),
  .RSTCTRL(1'b0),
  .RSTALLCARRYIN(1'b0),
  .RSTALUMODE(1'b0),
  .RSTINMODE(1'b0),
  .ACIN({30{1'b0}}),
  .BCIN({18{1'b0}}),
```

```
  .PCIN({48{1'b0}}),
  .CARRYCASCIN(1'b0),
  .MULTSIGNIN(1'b0),
  .ACOUT(),
  .BCOUT(),
  .PCOUT(),
  .CARRYOUT(),
  .CARRYCASCOUT(),
  .MULTSIGNOUT(),
  .PATTERNDETECT(),
  .PATTERNBDETECT(),
  .OVERFLOW(),
  .UNDERFLOW()
);
endmodule
```

5. Then the VPR will be used to place and route as well as generate the Bitstream.

6. ABC helps in the logic optimization.

3 Modeling using VPR

The Place and Route process in VPR consists of several steps:

- Packing (combinines primitives into complex blocks)
- Placement (places complex blocks within the FPGA grid)
- Routing (determines interconnections between blocks)
- Analysis (analyzes the implementation)

Each of these steps provides additional configuration options that can be used to customize the whole process.

1

3.1 Packing

The packing tries to combine primitive netlist blocks (e.g. LUTs, FFs) into groups, called Complex Blocks (as specified in the FPGA architecture file). The results from the packing process are written into a .net file.

A detailed description of the .net file format can be found in the Packed Netlist Format (.net) section.

3.2 Placement

This step assigns a location to the Complex Blocks (produced by packing) with the FPGA grid, while optimizing for wirelength and timing. The output from this step is written to the .place file, which contains the physical location of the blocks from the .net file.

3.3 Routing

This step determines how to connect the placed Complex Blocks together, according to the netlist connectivity and the routing resources of the FPGA chip.

The router uses a Routing Resource (RR) Graph to represent the FPGA's available routing resources. The RR graph can be created in two ways:
1. Automatically generated by VPR from the FPGA architecture description
, or
2. Loaded from an external RR graph file.
The output of routing is written into a .route file. The file describes each connection from input to its output through different routing resources of the FPGA. Each net starts with a SOURCE node and ends in a SINK node, potentially passing through complex block input/output pins (IPIN/OPIN nodes) and horizontal/vertical routing wires (CHANX/CHANY nodes). The pair of numbers in round brackets provides information on the (x, y) resource location on the VPR grid. The additional field provides information about the specific node.

3.4 Analysis

This step analyzes the resulting implementation, producing information about:
1. Resource usage (e.g. block types, wiring)
2. Timing (e.g. critical path delays and timing paths)
3. Power (e.g. total power used, power broken down by blocks

4. Multi-Mode Logics of DSP48E1

Mode:

Interconnect_mode:

This is the mode required if the operation is only to make an interconnect between two nodes, where there is no extra operation required. For ex: In a case where the only job is to make an pass the output such as ACIN to ACOUT, or A to ACOUT, etc. Here we just have to change the ports the connection is applied to.

Small code snippet:

```
<mode name="Interconnect_mode">
<pb_type name="Connection" num_pb="1" blif_model=".subckt sync_or_async">
  <input  name="ACIN"  num_pins="1"/>
  <output name="ACOUT"  num_pins="1"/>

</pb_type>
<interconnect>

  <direct name="Q"  input="ACIN" output="ACOUT" />
</interconnect>
</mode>
```

Mode:

Multiply_INMODE_mode:

In this mode we present the operation of simple multiplication with direct input. For ex: Operation where we have to multiply inputs (A[24:0] , B[17:0]), here we give the output as the (CARRYCASOUT, CARRYOUT[3:0], MULTSIGNOUT, P[47:0], PATTERNDETECT, PATTERNBDETECT, PCOUT[47:0]). This can be used for various inputs of ACIN, A, BCIN,B, D,etc.

Small code snippet:

```
       <mode name="Interconnect_mode">
       <pb_type name="Connection" num_pb="1" blif_model=".subckt sync_or_async">
         <input  name="ACIN"  num_pins="1"/>
         <input name="A"  num_pins="1"/>
         <input name="BCIN"  num_pins="1"/>
         <input name="B"  num_pins="1"/>
         <input name="INMODE"  num_pins="1"/>
         <input name="C"  num_pins="1"/>
         <input name="D"  num_pins="1"/>
         <output name="CARRYCASCOUT"  num_pins="1"/>
         <output name="CARRYOUT"  num_pins="1"/>
         <output name="MULTSIGNOUT"  num_pins="1"/>
         <output name="P"  num_pins="1"/>
         <output name="PATTERNBDETECT"  num_pins="1"/>
         <output name="PATTERNDETECT"  num_pins="1"/>
         <output name="PCOUT"  num_pins="1"/>
       </pb_type>
       <interconnect>
         <!-- Defining the interconnections during this MODE -->
       </interconnect>
       <!-- Defininig the inner sub types for the model>
             <pb_type name=”A*B_operation” blif_model=”.names1” num_pb=”1”
class=”multiply”>
                   <!-- Inputs -->
                   <!-- Outputs -->
                   <!-- Delay Matrix -->
                   <!-- Defining the inner logic -- >
             </pb_type>
<pb_type name=”A+B_operation” blif_model=”.names2” num_pb=”1” class=”addition”>
                   <!-- Inputs -->
                   <!-- Outputs -->
                   <!-- Delay Matrix -->
                   <!-- Defining the inner logic -- >
             </pb_type>
```

```
<pb_type name=”pattern_operation” blif_model=”.names3” num_pb=”1” class=”pattern”>
                    <!-- Inputs -->
                    <!-- Outputs -->
                    <!-- Delay Matrix -->
                    <!-- Defining the inner logic -- >
            </pb_type>
...
...
...
     </mode>
```

Mode:

ALUMODE_OPMODE_mode:

In this mode we decide the operation based on the inputs to the ALU. This will decide which inputs to consider and what outputs should be sent at the output.

Small code snippet:

```
      <mode name="ALUMODE_OPMODE_mode">
       <pb_type name="Connection" num_pb="1" blif_model=".subckt operation">
         <input  name="ACIN"  num_pins="1"/>
         <input name="A"  num_pins="1"/>
         <input name="BCIN"  num_pins="1"/>
         <input name="B"  num_pins="1"/>
         <input name="INMODE"  num_pins="1"/>
         <input name="C"  num_pins="1"/>
         <input name="D"  num_pins="1"/>
         <input name="OPMODE"  num_pins="1"/>
         <input name=”ALUMODE"  num_pins="1"/>
         <output name="CARRYCASCOUT"  num_pins="1"/>
         <output name="CARRYOUT"  num_pins="1"/>
         <output name="MULTSIGNOUT"  num_pins="1"/>
         <output name="P"  num_pins="1"/>
         <output name="PATTERNBDETECT"  num_pins="1"/>
         <output name="PATTERNDETECT"  num_pins="1"/>
         <output name="PCOUT"  num_pins="1"/>
       </pb_type>
       <interconnect>
         <!-- Defining the interconnections during this MODE -->
       </interconnect>
      <!-- Defining the inner sub types for the model>
```

```
      <!-- Defining for the adder operation of the ALU -->
            <pb_type name=”Adder” blif_model=”.names1” num_pb=”1”
class=”simple_adder”>
                  <!-- Inputs -->
                  <!-- Outputs -->
                  <!-- Delay Matrix -->
                  <!-- Defining the inner logic -- >
            </pb_type>
<pb_type name=”Pre adder for A” blif_model=”.names2” num_pb=”1” class=”addition”>
                  <!-- Inputs -->
                  <!-- Outputs -->
                  <!-- Delay Matrix -->
                  <!-- Defining the inner logic -- >
            </pb_type>
<pb_type name=”Logic_operation” blif_model=”.names3” num_pb=”1” class=”logic”>
                  <!-- Inputs -->
                  <!-- Outputs -->
                  <!-- Delay Matrix -->
                  <!-- Defining the inner logic -- >
            </pb_type>
...
...
...
     </mode>
```

Mode:

Output_mode:

In this mode the DSP is required to forward what kind of output depending on the carry and other inputs to the DSP from the present as well as past outputs. For ex: the MACC operation, Patterndetect,etc.

Small code snippet:

```
      <mode name="OUTPUT_mode">
       <pb_type name="Connection" num_pb="1" blif_model=".subckt operation">
         <input name="CARRYINSEL"  num_pins="1"/>
         <input name=”CREG"  num_pins="1"/>
         <output name="CARRYCASCOUT"  num_pins="1"/>
         <output name="CARRYOUT"  num_pins="1"/>
         <output name="MULTSIGNOUT"  num_pins="1"/>
         <output name="P"  num_pins="1"/>
         <output name="PATTERNBDETECT"  num_pins="1"/>
```

```
          <output name="PATTERNDETECT"  num_pins="1"/>
          <output name="PCOUT"  num_pins="1"/>
        </pb_type>
        <interconnect>
          <!-- Defining the interconnections during this MODE -->
        </interconnect>
        <!-- Defining the inner sub types for the model>
        <!-- Defining for the operation of the Output -->
              <pb_type name=”Signout” blif_model=”.names1” num_pb=”1” class=”flags”>
                    <!-- Inputs -->
                    <!-- Outputs -->
                    <!-- Delay Matrix -->
                    <!-- Defining the inner logic -- >
              </pb_type>
<pb_type name=”Patternderected” blif_model=”.names2” num_pb=”1”
class=”Bypass/Mask”>
                    <!-- Inputs -->
                    <!-- Outputs -->
                    <!-- Delay Matrix -->
                    <!-- Defining the inner logic -- >
              </pb_type>

...
...
...
     </mode>
```

Mode:

Flag_mode:

This mode helps in determining the flags for the underflow and overflow of the DSP, based on the inputs (PATTERNDETECTPAST, PATTERBDETECT, PATTERNBDETECT) and outputs (OVERFLOW, UNDERFLOW).

Small code snippet:

```
        <mode name="FLAG_mode">
        <pb_type name="Connection" num_pb="1" blif_model=".subckt operation">
          <input  name="ACIN"  num_pins="1"/>
          <input name="A"  num_pins="1"/>
          <input name="BCIN"  num_pins="1"/>
          <input name="B"  num_pins="1"/>
          <input name="C"  num_pins="1"/>
```

```
          <input name="D"  num_pins="1"/>
          <output name="CARRYCASCOUT"  num_pins="1"/>
          <output name="CARRYOUT"  num_pins="1"/>
          <output name="MULTSIGNOUT"  num_pins="1"/>
          <output name="P"  num_pins="1"/>
          <output name="PATTERNBDETECT"  num_pins="1"/>
          <output name="PATTERNDETECT"  num_pins="1"/>
          <output name="PCOUT"  num_pins="1"/>
        </pb_type>
        <interconnect>
          <!-- Defining the interconnections during this MODE -->
        </interconnect>
       <!-- Defining the inner sub types for the model>
       <!-- Defining for the ALU flag use of the ALU -->
              <pb_type name=”Underflow” blif_model=”.names1” num_pb=”1”
class=”under”>
                    <!-- Inputs -->
                    <!-- Outputs -->
                    <!-- Delay Matrix -->
                    <!-- Defining the inner logic -- >
             </pb_type>
<pb_type name=”Overflow” blif_model=”.names2” num_pb=”1” class=”over”>
                    <!-- Inputs -->
                    <!-- Outputs -->
                    <!-- Delay Matrix -->
                    <!-- Defining the inner logic -- >
             </pb_type>
...
...
...
     </mode>
```