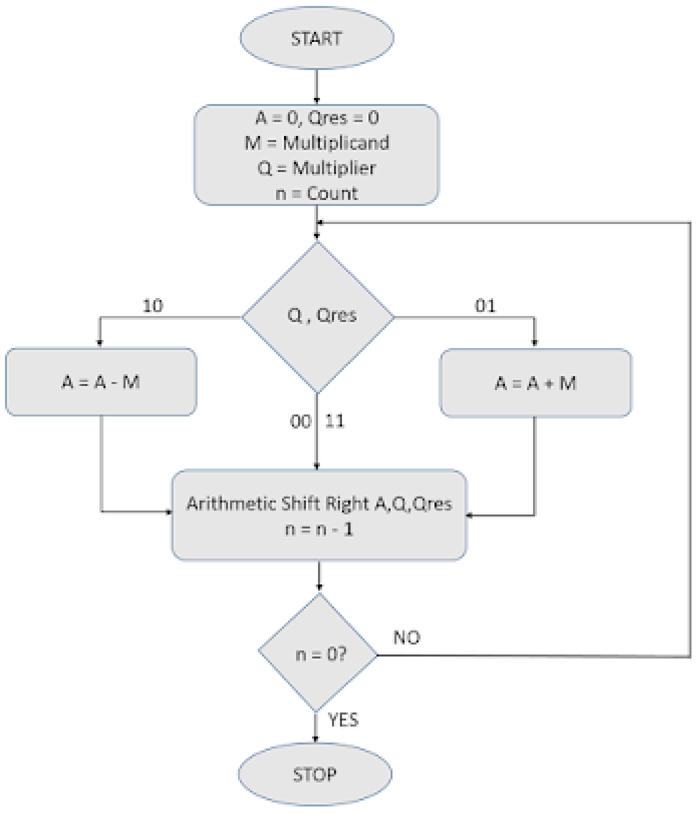
CA ASSIGNMENT
Yosys implementation of Booth Multiplication algorithm

Name- Asmit Kumar Panika
Id- 2020UCP1165

Booth's multiplication

Booth's Multiplication Algorithm is a commonly used algorithm for multiplication of two signed numbers.

Flowchart:



Verilog Code:

```
module booth_multiplication(clk,rst,start,X,Y,valid,Z);

input clk;
input rst;
input start;
input signed [3:0]X,Y;
output signed [7:0]Z;
output valid;

reg signed [7:0] Z,next_Z,Z_temp;
reg next_state, pres_state;
reg [1:0] temp,next_temp;
reg [1:0] count,next_count;
```

```verilog
reg valid, next_valid;

parameter IDLE = 1'b0;
parameter START = 1'b1;

always @ (posedge clk or negedge rst)
begin
if(!rst)
begin
  Z          <= 8'd0;
  valid      <= 1'b0;
  pres_state <= 1'b0;
  temp       <= 2'd0;
  count      <= 2'd0;
end
else
begin
  Z          <= next_Z;
  valid      <= next_valid;
  pres_state <= next_state;
  temp       <= next_temp;
  count      <= next_count;
end
end

always @ (*)
begin
case(pres_state)

IDLE:
begin
next_count = 2'b0;
next_valid = 1'b0;
if(start)
begin
    next_state = START;
    next_temp  = {X[0],1'b0};
    next_Z     = {4'd0,X};
end
else
begin
    next_state = pres_state;
    next_temp  = 2'd0;
    next_Z     = 8'd0;
end
end

START:
begin
    case(temp)
    2'b10:   Z_temp = {Z[7:4]-Y,Z[3:0]};
    2'b01:   Z_temp = {Z[7:4]+Y,Z[3:0]};
    default: Z_temp = {Z[7:4],Z[3:0]};
    endcase
next_temp  = {X[count+1],X[count]};
```

```
next_count = count + 1'b1;
next_Z     = Z_temp >>> 1;
next_valid = (&count) ? 1'b1 : 1'b0;
next_state = (&count) ? IDLE : pres_state;
end
endcase
end
Endmodule
```

Testbench:

```
module booth_tb;

reg clk,rst,start;
reg signed [3:0]X,Y;
wire signed [7:0]Z;
wire valid;

always #5 clk = ~clk;

booth_multiplication inst (clk,rst,start,X,Y,valid,Z);

initial
$monitor($time,"X=%d, Y=%d, valid=%d, Z=%d ",X,Y,valid,Z);
initial
begin
X=5;Y=7;clk=1'b1;rst=1'b0;start=1'b0;
#10 rst = 1'b1;
#10 start = 1'b1;
#10 start = 1'b0;
@valid
#10 X=-4;Y=6;start = 1'b1;
#10 start = 1'b0;
end
endmodule
```

Synthesis Report:

```
Running ABC command: "<yosys-exe-dir>/yosys-abc" -s -f
<abc-temp-dir>/abc.script 2>&1
```

```
ABC: ABC command line: "source <abc-temp-dir>/abc.script"

ABC:

ABC: + read_blif <abc-temp-dir>/input.blif

ABC: + read_library <abc-temp-dir>/stdcells.genlib

ABC: Entered genlib library with 13 gates from file
"<abc-temp-dir>/stdcells.genlib"

ABC: + strash

ABC: + dretime

ABC: + map

ABC: + write_blif <abc-temp-dir>/output.blif



Re-integrating ABC results.

ABC RESULTS:            OR cells:         2

ABC RESULTS:            NOR cells:        2

ABC RESULTS:            AND cells:        1

ABC RESULTS:            NOT cells:        18

ABC RESULTS:            XNOR cells:       6

ABC RESULTS:            ORNOT cells:      3
ABC RESULTS:            NAND cells:         3

ABC RESULTS:            XOR cells:          10

ABC RESULTS:            MUX cells:          38

ABC RESULTS:            ANDNOT cells:       26

ABC RESULTS:            internal signals:   95

ABC RESULTS:            input signals:      22

ABC RESULTS:            output signals:     12

Removing temp directory.



Printing statistics.

=== booth_multiplication ===

Number of wires:                109

Number of wire bits:            145

Number of public wires:         11

Number of public wire bits:     35

Number of memories:             0

Number of memory bits:          0
Number of processes:          0

Number of cells:              130

$_ANDNOT_              26

$_AND_                 1

$_DFF_P_               16

$_MUX_                 38

$_NAND_                3

$_NOR_                 2

$_NOT_                 17
```

Checking the synthesis

```
$_ORNOT_               3

$_OR_                  2

$_SDFFE_PP0N_          2

$_SDFFE_PP0P_          5

$_SDFFE_PP1N_          1

$_XNOR_                4
```

$_XOR_	10