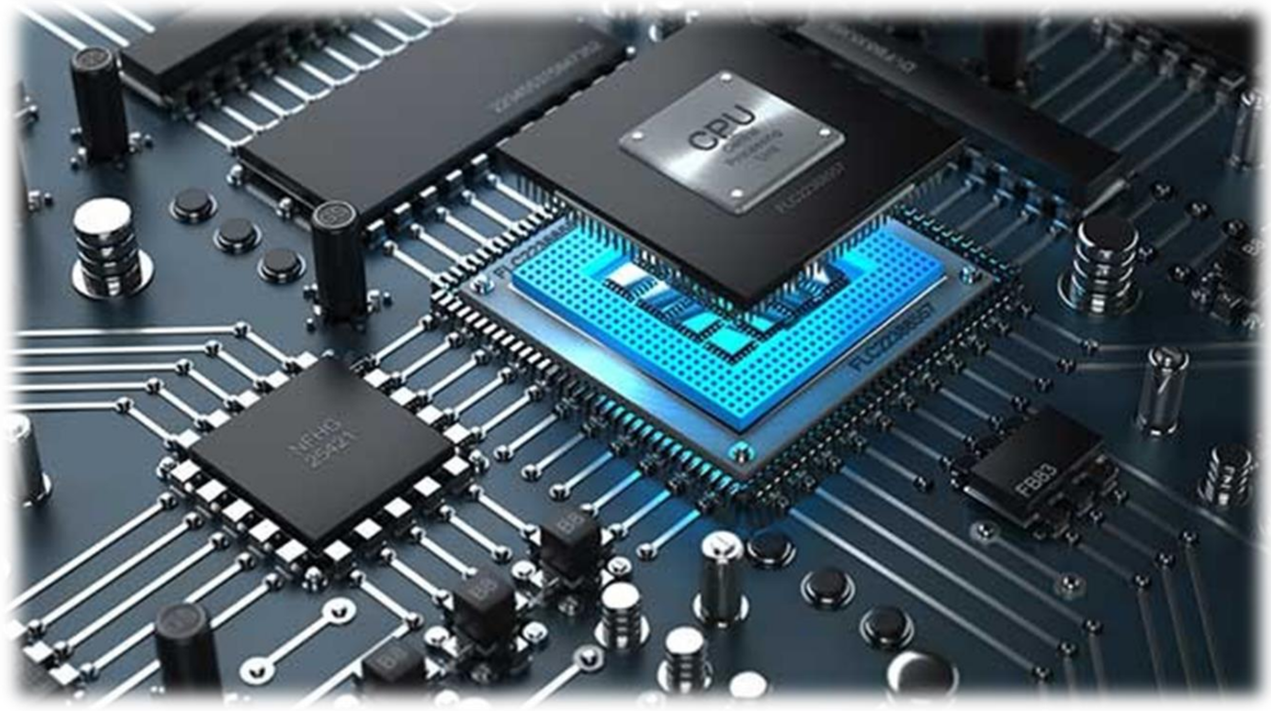


NAME : George Jan George Shaffik

Project : Brent and Kung parameterized adder

GitHub link :

https://github.com/georgejan9/Brent-Kung-parameterized-Adder

Table of Contents

1. Specs:

You are asked to design a 64-bit parallel prefix adder using Brent-kung Carry network through the following the steps below and attach all the required captures and codes

Fig.3 below show an example how to implement a size-8 prefix sum network using brent-kung Architecture each layer is encoded with different color to help you extract the pattern

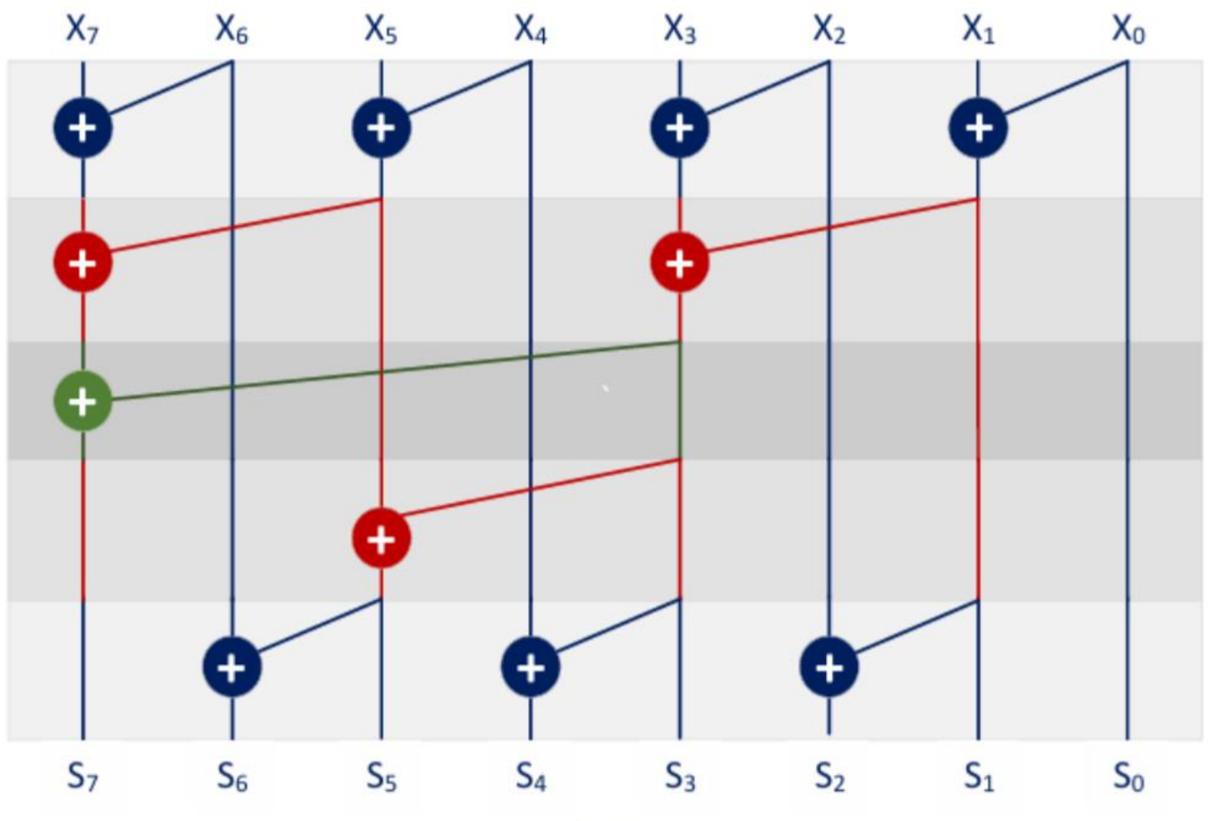


Figure 3

Fig.4 show the steps how the example in fig.3 is build step by step first layer one which contains (8-1 adders) and change the problem from 8-input problem to a smaller one with only 4-inputs then we apply the same idea by using (4-1 adders) and split the problem to half please spend some time to convince yourself and make sure you understand the pattern very well then start to map this into a carry network of 64-bit adder by removing each + sign with a carry operator you will find the implementation of carry operator below in fig.4

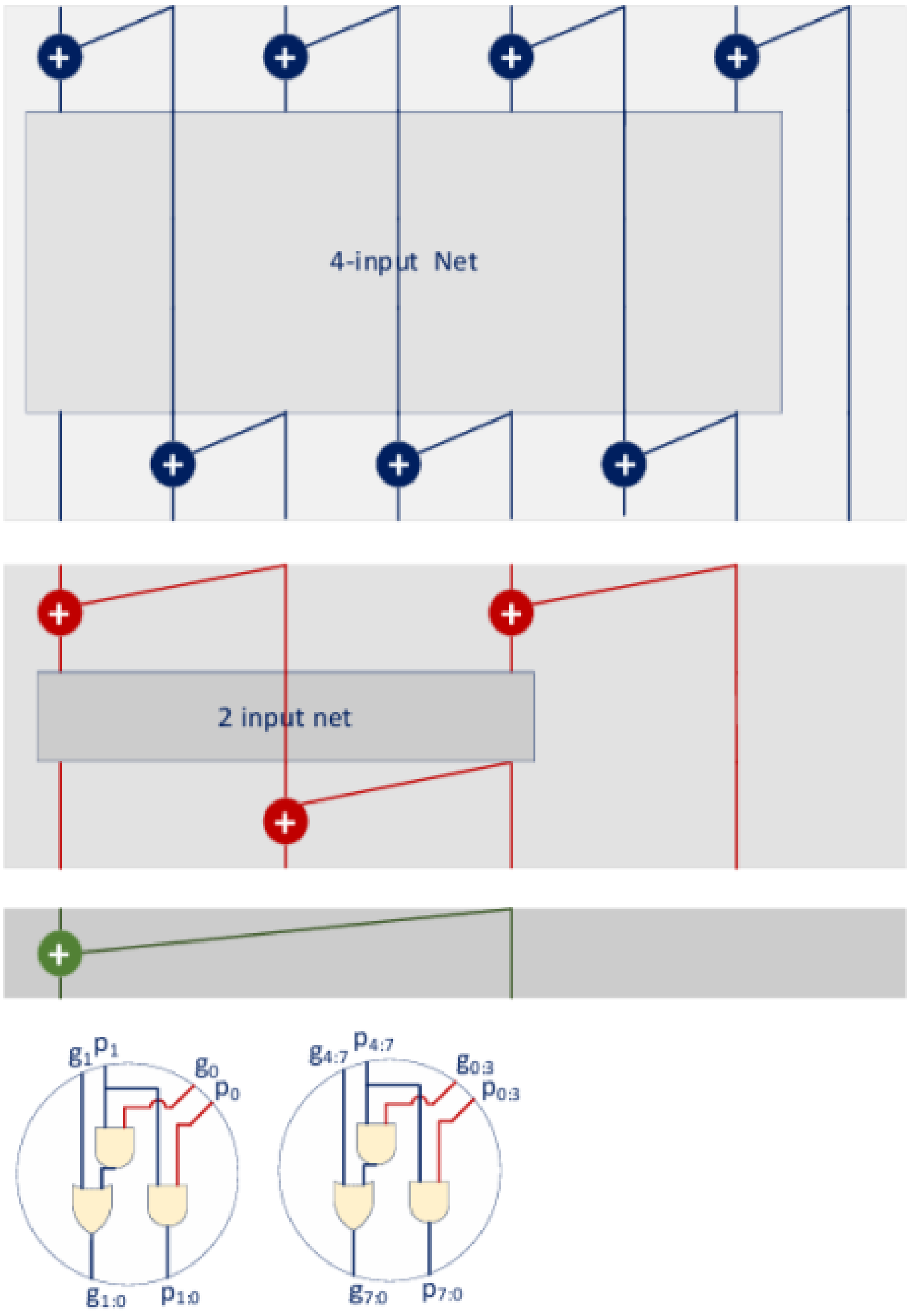
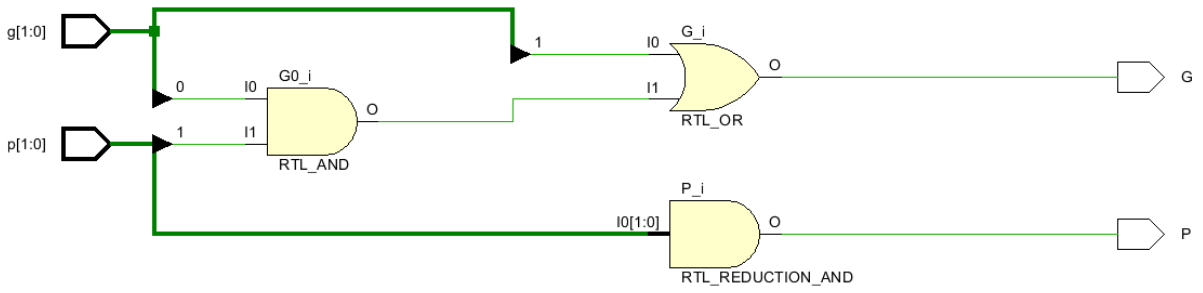


Figure 4

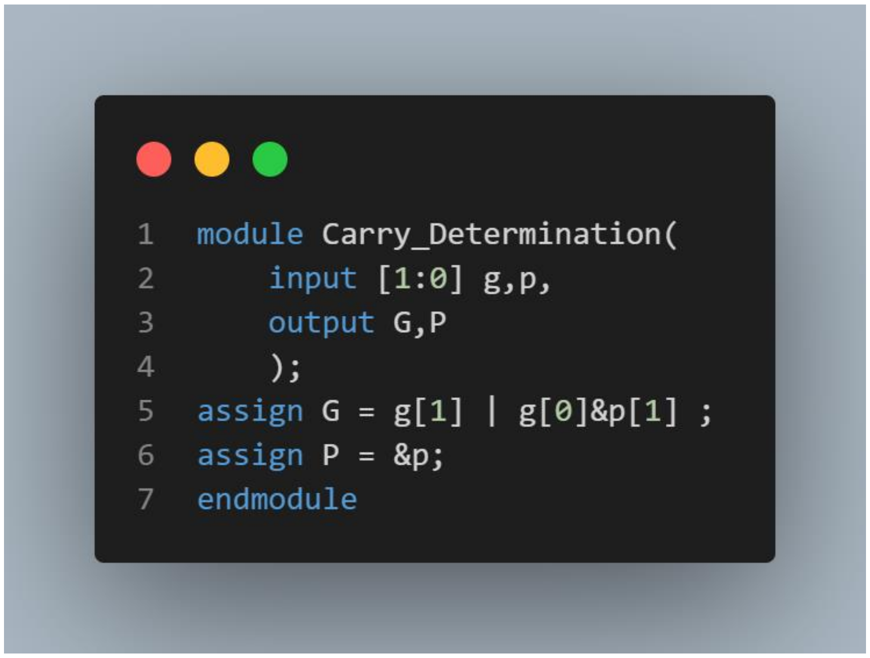
- Inputs {A, B, Cin} (A,B are 64-bits inputs and Cin is 1-bit)
- Outputs {Sum, Cout} (sum is 64-bits output and Cout is 1-bit)

2. Carry Determination:

a) RTL schematic :



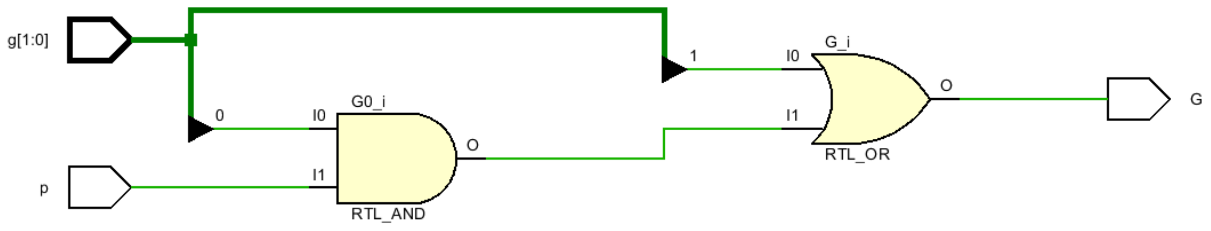
b) RTL code :



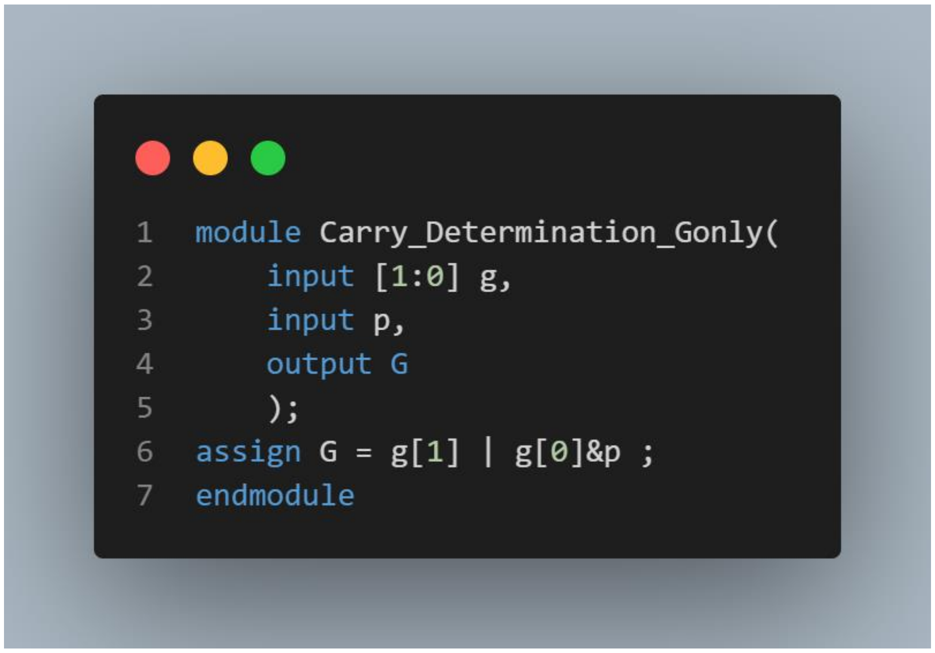
```
module Carry_Determination(
    input [1:0] g,p,
    output G,P
    );
assign G = g[1] | g[0]&p[1] ;
assign P = &p;
endmodule
```

3. Carry_Determination_Gonly

a) RTL schematic :



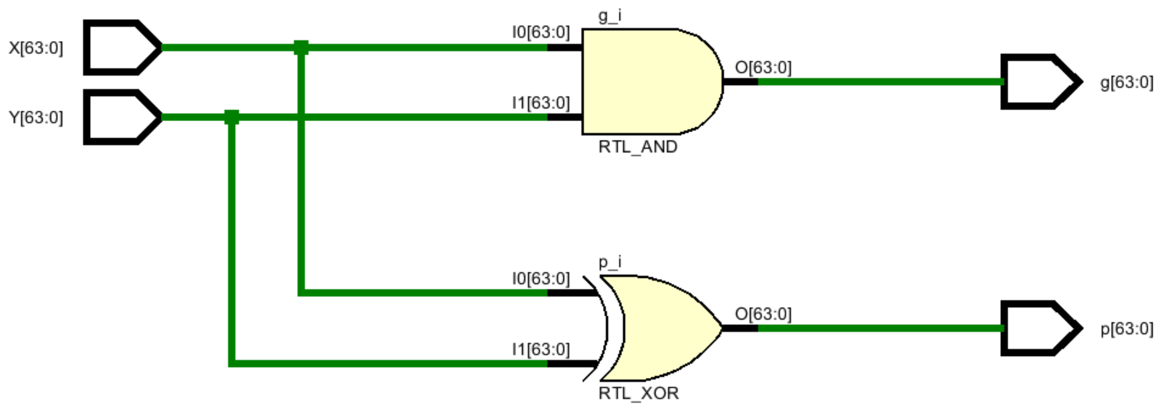
b) RTL code :



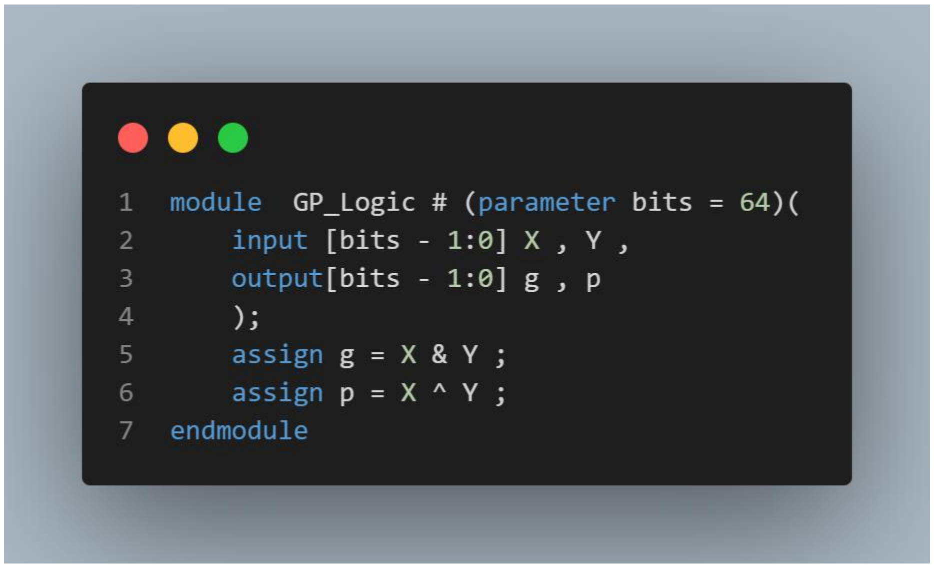
```
module Carry_Determination_Gonly(
    input [1:0] g,
    input p,
    output G
    );
assign G = g[1] | g[0]&p ;
endmodule
```

4. GP Logic

a) RTL schematic :



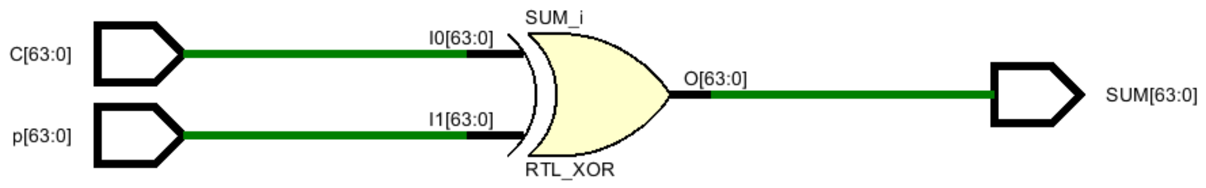
b) RTL code :



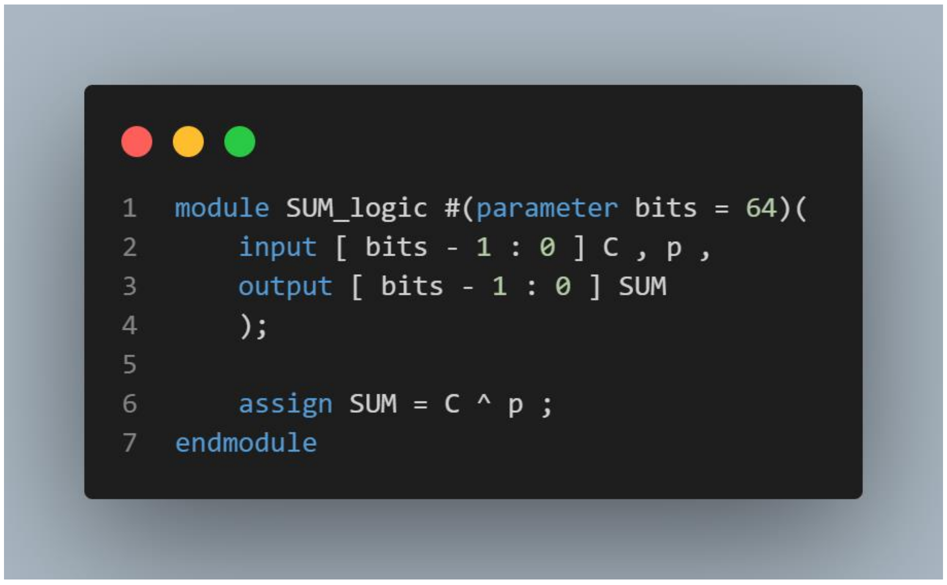
```verilog
module  GP_Logic # (parameter bits = 64)(
    input [bits - 1:0] X , Y ,
    output[bits - 1:0] g , p
    );
    assign g = X & Y ;
    assign p = X ^ Y ;
endmodule
```

5. SUM logic

a) RTL schematic :



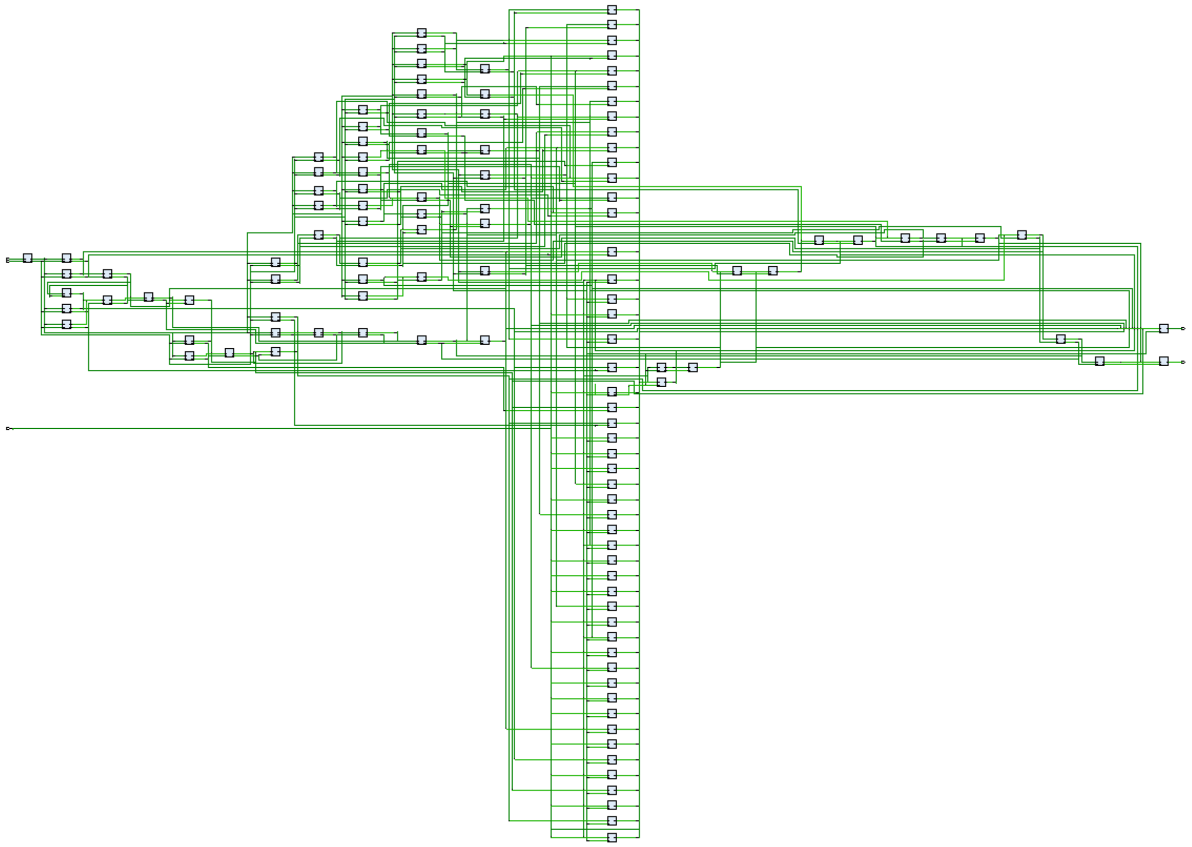
b) RTL code :



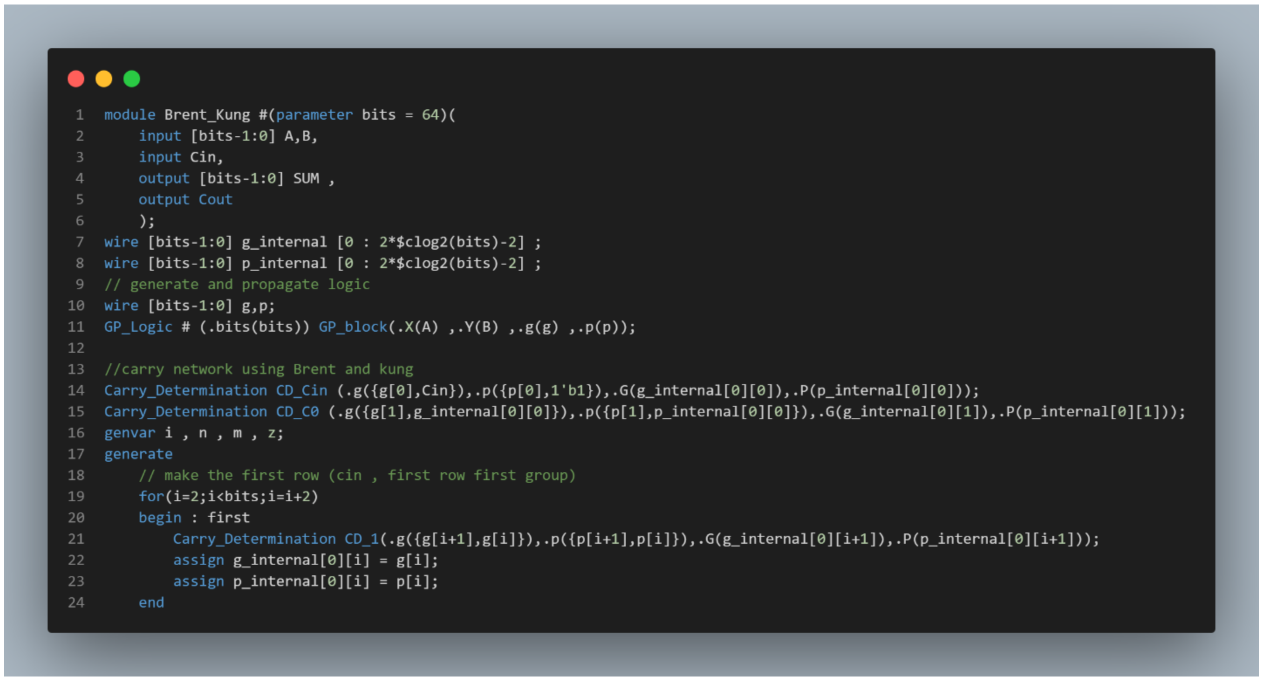
```
module SUM_logic #(parameter bits = 64)(
    input [ bits - 1 : 0 ] C , p ,
    output [ bits - 1 : 0 ] SUM
    );

    assign SUM = C ^ p ;
endmodule
```

6. Brent Kung Adder

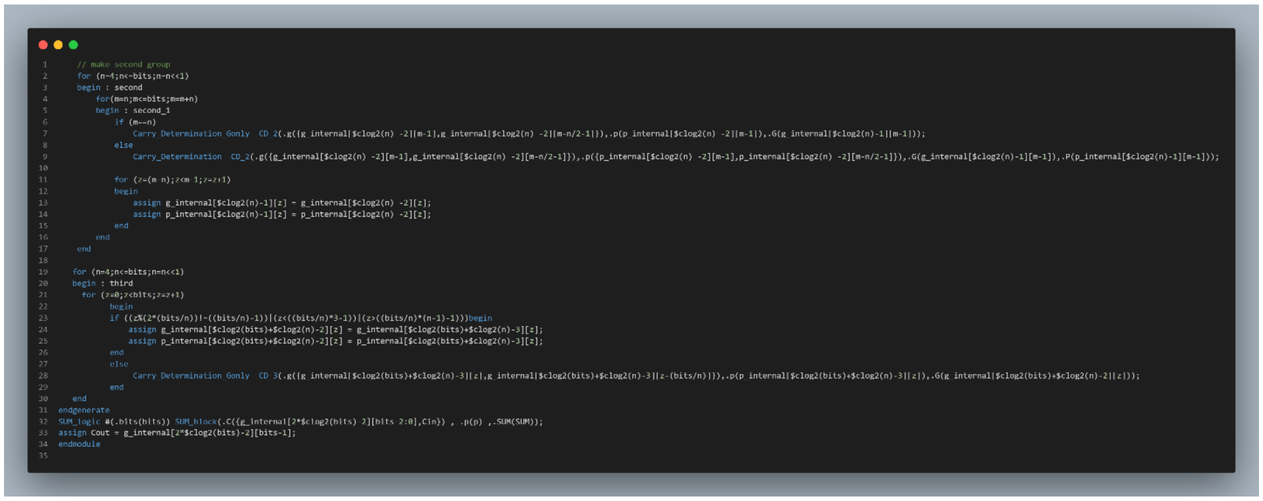
a) RTL schematic :



b) RTL code :



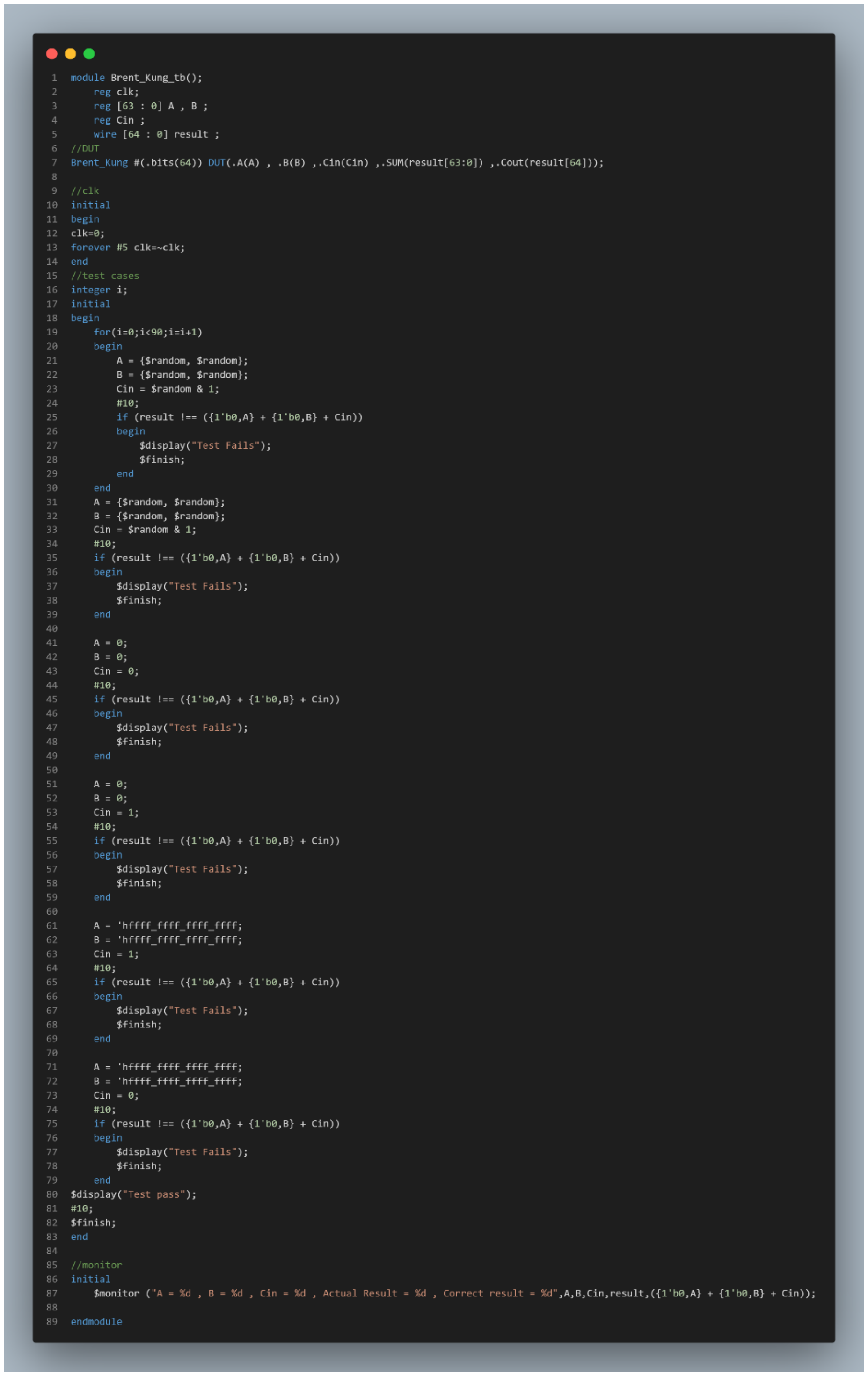
```verilog
module Brent_Kung #(parameter bits = 64)(
    input [bits-1:0] A,B,
    input Cin,
    output [bits-1:0] SUM ,
    output Cout
    );
wire [bits-1:0] g_internal [0 : 2*$clog2(bits)-2] ;
wire [bits-1:0] p_internal [0 : 2*$clog2(bits)-2] ;
// generate and propagate logic
wire [bits-1:0] g,p;
GP_Logic # (.bits(bits)) GP_block(.X(A) ,.Y(B) ,.g(g) ,.p(p));

//carry network using Brent and kung
Carry_Determination CD_Cin (.g({g[0],Cin}),.p({p[0],1'b1}),.G(g_internal[0][0]),.P(p_internal[0][0]));
Carry_Determination CD_C0 (.g({g[1],g_internal[0][0]}),.p({p[1],p_internal[0][0]}),.G(g_internal[0][1]),.P(p_internal[0][1]));
genvar i , n , m , z;
generate
    // make the first row (cin , first row first group)
    for(i=2;i<bits;i=i+2)
    begin : first
        Carry_Determination CD_1(.g({g[i+1],g[i]}),.p({p[i+1],p[i]}),.G(g_internal[0][i+1]),.P(p_internal[0][i+1]));
        assign g_internal[0][i] = g[i];
        assign p_internal[0][i] = p[i];
    end
```



```verilog
     // make second group
     for (n=4;n<=bits;n=n<<1)
     begin : second
         for(m=n;m<=bits;m=m+n)
         begin : second_1
             if (m==n)
                 Carry_Determination_Gonly  CD_2(.g({g_internal[$clog2(n) -2][m-1],g_internal[$clog2(n) -2][m-n/2-1]}),.p(p_internal[$clog2(n) -2][m-1]),.G(g_internal[$clog2(n)-1][m-1]));
             else
                 Carry_Determination  CD_2(.g({g_internal[$clog2(n) -2][m-1],g_internal[$clog2(n) -2][m-n/2-1]}),.p({p_internal[$clog2(n) -2][m-1],p_internal[$clog2(n) -2][m-n/2-1]}),.G(g_internal[$clog2(n)-1][m-1]),.P(p_internal[$clog2(n)-1][m-1]));

             for (z=(m-n);z<m-1;z=z+1)
             begin
                 assign g_internal[$clog2(n)-1][z] = g_internal[$clog2(n) -2][z];
                 assign p_internal[$clog2(n)-1][z] = p_internal[$clog2(n) -2][z];
             end
         end
     end

    for (n=4;n<=bits;n=n<<1)
    begin : third
      for (z=0;z<bits;z=z+1)
            begin
            if ((z%(2*(bits/n))!=((bits/n)-1))|(z<((bits/n)*3-1))|(z>((bits/n)*(n-1)-1)))begin
                assign g_internal[$clog2(bits)+$clog2(n)-2][z] = g_internal[$clog2(bits)+$clog2(n)-3][z];
                assign p_internal[$clog2(bits)+$clog2(n)-2][z] = p_internal[$clog2(bits)+$clog2(n)-3][z];
            end
            else
                Carry_Determination_Gonly  CD_3(.g({g_internal[$clog2(bits)+$clog2(n)-3][z],g_internal[$clog2(bits)+$clog2(n)-3][z-(bits/n)]}),.p(p_internal[$clog2(bits)+$clog2(n)-3][z]),.G(g_internal[$clog2(bits)+$clog2(n)-2][z]));
            end
    end
endgenerate
SUM_logic #(.bits(bits)) SUM_block(.C({g_internal[2*$clog2(bits)-2][bits-2:0],Cin}) , .p(p) ,.SUM(SUM));
assign Cout = g_internal[2*$clog2(bits)-2][bits-1];
endmodule
```

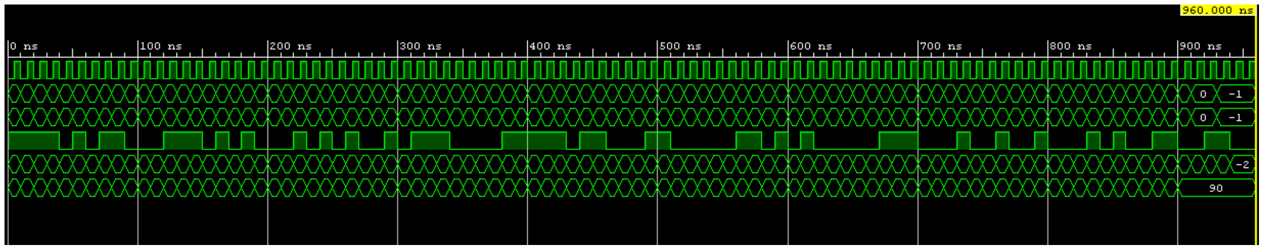
7. testbench code for 64 adder

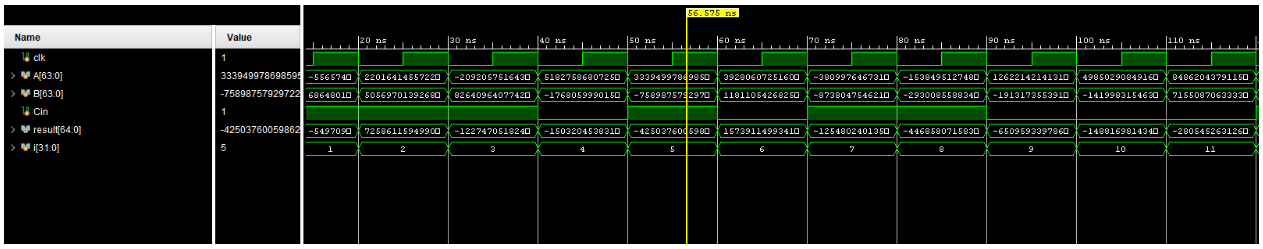
a) code



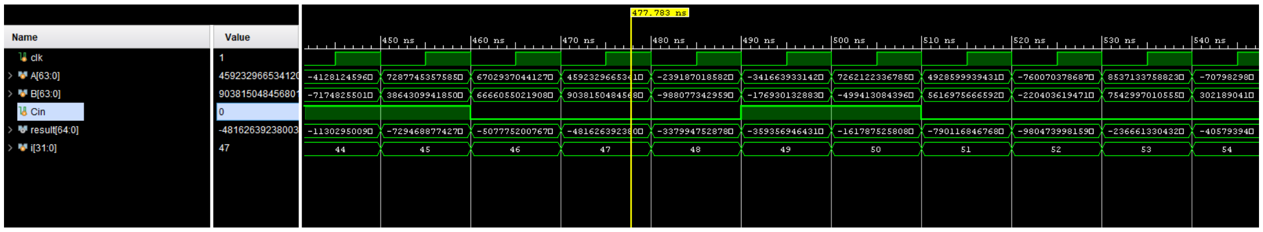
```verilog
module Brent_Kung_tb();
    reg clk;
    reg [63 : 0] A , B ;
    reg Cin ;
    wire [64 : 0] result ;
//DUT
Brent_Kung #(.bits(64)) DUT(.A(A) , .B(B) ,.Cin(Cin) ,.SUM(result[63:0]) ,.Cout(result[64]));

//clk
initial
begin
clk=0;
forever #5 clk=~clk;
end
//test cases
integer i;
initial
begin
    for(i=0;i<90;i=i+1)
    begin
        A = {$random, $random};
        B = {$random, $random};
        Cin = $random & 1;
        #10;
        if (result !== ({1'b0,A} + {1'b0,B} + Cin))
        begin
            $display("Test Fails");
            $finish;
        end
    end
    A = {$random, $random};
    B = {$random, $random};
    Cin = $random & 1;
    #10;
    if (result !== ({1'b0,A} + {1'b0,B} + Cin))
    begin
        $display("Test Fails");
        $finish;
    end

    A = 0;
    B = 0;
    Cin = 0;
    #10;
    if (result !== ({1'b0,A} + {1'b0,B} + Cin))
    begin
        $display("Test Fails");
        $finish;
    end

    A = 0;
    B = 0;
    Cin = 1;
    #10;
    if (result !== ({1'b0,A} + {1'b0,B} + Cin))
    begin
        $display("Test Fails");
        $finish;
    end

    A = 'hffff_ffff_ffff_ffff;
    B = 'hffff_ffff_ffff_ffff;
    Cin = 1;
    #10;
    if (result !== ({1'b0,A} + {1'b0,B} + Cin))
    begin
        $display("Test Fails");
        $finish;
    end

    A = 'hffff_ffff_ffff_ffff;
    B = 'hffff_ffff_ffff_ffff;
    Cin = 0;
    #10;
    if (result !== ({1'b0,A} + {1'b0,B} + Cin))
    begin
        $display("Test Fails");
        $finish;
    end
$display("Test pass");
#10;
$finish;
end

//monitor
initial
    $monitor ("A = %d , B = %d , Cin = %d , Actual Result = %d , Correct result = %d",A,B,Cin,result,({1'b0,A} + {1'b0,B} + Cin));

endmodule
```

b) Waveform





Name	Value
clk	1
A[63:0]	333949978698595
B[63:0]	-75898757929722
Cin	1
result[64:0]	-42503760059862
i[31:0]	5



Name	Value
clk	1
A[63:0]	459232966534120
B[63:0]	903815048456801
Cin	0
result[64:0]	-48162639238003
i[31:0]	47

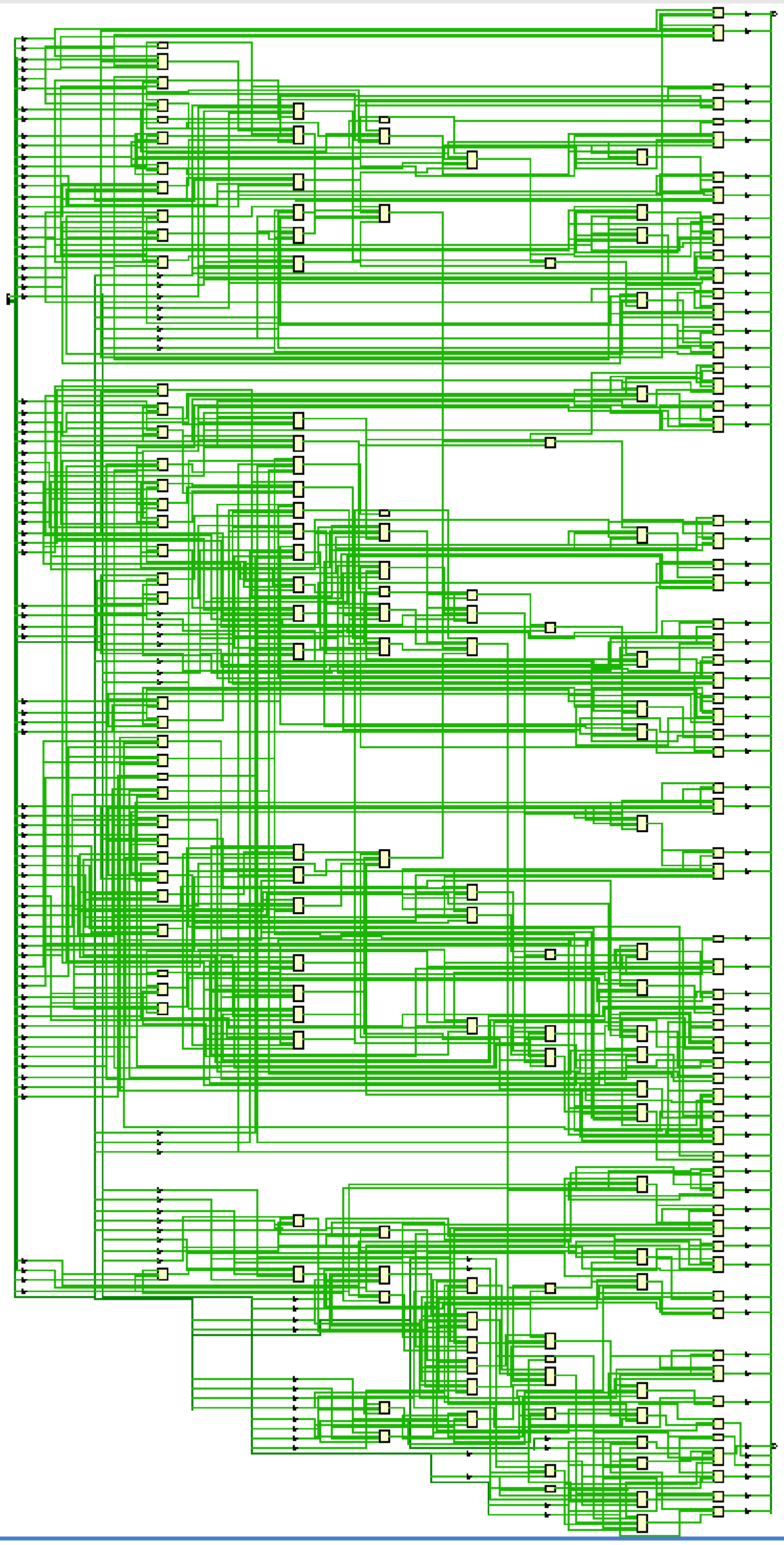
c) Test-Bench log

```
# run 1000ns
A = 13873724035538695460 , B = 12821843124548851209 , Cin = 1 , Actual Result = 26695567160087546670 , Correct result = 26695567160087546670
A = 12881003454748662157 , B =   68648015077265938 , Cin = 1 , Actual Result = 12949651469825928096 , Correct result = 12949651469825928096
A =  2201641455722951030 , B =  5056970139268044781 , Cin = 1 , Actual Result =  7258611594990995812 , Correct result =  7258611594990995812
A = 16354686557274711238 , B =  8264096407742567082 , Cin = 1 , Actual Result = 24618782965017278321 , Correct result = 24618782965017278321
A =  5182758680725214738 , B = 16678684083553855986 , Cin = 0 , Actual Result = 21861442764279070724 , Correct result = 21861442764279070724
A =  3339499786985950917 , B = 10856868280737343677 , Cin = 1 , Actual Result = 14196368067723294595 , Correct result = 14196368067723294595
A =   392806072516043363 , B =  1181105426825683584 , Cin = 0 , Actual Result =  1573911499341726947 , Correct result =  1573911499341726947
A = 14636767606389656733 , B =  9708696527496787987 , Cin = 1 , Actual Result = 24345464133886444721 , Correct result = 24345464133886444721
A = 16908248946225175915 , B = 15516658485362641410 , Cin = 1 , Actual Result = 32424907431587817326 , Correct result = 32424907431587817326
A =  1262214214131741391 , B = 16533570519791396106 , Cin = 0 , Actual Result = 17795784733923137497 , Correct result = 17795784733923137497
A =  4985029084916006386 , B = 17026760919071634241 , Cin = 0 , Actual Result = 22011790003987640627 , Correct result = 22011790003987640627
A =  8486204379115426441 , B =  7155087063333889462 , Cin = 0 , Actual Result = 15641291442449315903 , Correct result = 15641291442449315903
A =  1517674775637721788 , B = 13301309419248130571 , Cin = 1 , Actual Result = 14818984194885852360 , Correct result = 14818984194885852360
A = 11370568966393189711 , B = 13772907602663322426 , Cin = 1 , Actual Result = 25143476569056512138 , Correct result = 25143476569056512138
A =  7826213649870593009 , B =  2753383051867391842 , Cin = 1 , Actual Result = 10579596701737984852 , Correct result = 10579596701737984852
A =  8966009316984856975 , B = 14971186305873830071 , Cin = 0 , Actual Result = 23937195622858687046 , Correct result = 23937195622858687046
A =  4962464903874723931 , B = 16727282506821153353 , Cin = 1 , Actual Result = 21689747410695877285 , Correct result = 21689747410695877285
A =  5413660551199325265 , B = 16225570890230628108 , Cin = 0 , Actual Result = 21639231441429953373 , Correct result = 21639231441429953373
A =  2230105614896487031 , B = 13764408377445505810 , Cin = 1 , Actual Result = 15994513992341992842 , Correct result = 15994513992341992842
A =  1140131019468957497 , B =  4816933528651167443 , Cin = 0 , Actual Result =  5957064548120124940 , Correct result =  5957064548120124940
A =  2633281184802167131 , B =  1516779772123983423 , Cin = 0 , Actual Result =  4150060956926150554 , Correct result =  4150060956926150554
A = 14346793370242726022 , B =  9022286730986779292 , Cin = 0 , Actual Result = 23369080101229505314 , Correct result = 23369080101229505314
A = 15099360788828526451 , B = 15695768951379830831 , Cin = 1 , Actual Result = 30795129740208357283 , Correct result = 30795129740208357283
A =  8933169098761243972 , B = 17512558205003839179 , Cin = 0 , Actual Result = 26445727303765083151 , Correct result = 26445727303765083151
A = 17763009388186943785 , B = 12871206808937786586 , Cin = 1 , Actual Result = 30634216197124730372 , Correct result = 30634216197124730372
A =  4391319123166926047 , B = 16750027002164150084 , Cin = 0 , Actual Result = 21141346125331076131 , Correct result = 21141346125331076131
A =   535874541159099819 , B = 14798472797048647132 , Cin = 1 , Actual Result = 15334347338207746952 , Correct result = 15334347338207746952
A =  3102678150718489283 , B = 12959519588151322958 , Cin = 0 , Actual Result = 16062197738869812241 , Correct result = 16062197738869812241
A = 11245077294247557558 , B = 15864990407916095353 , Cin = 0 , Actual Result = 27110067702163652911 , Correct result = 27110067702163652911
A =  9385268545392631187 , B =  7911533562622888537 , Cin = 1 , Actual Result = 17296802108015519725 , Correct result = 17296802108015519725
A = 13160686262500243417 , B =  7294505156530791030 , Cin = 0 , Actual Result = 20455191419031034447 , Correct result = 20455191419031034447
A = 11747387042220437909 , B =  8922301525300386564 , Cin = 1 , Actual Result = 20669688567520824474 , Correct result = 20669688567520824474
A =  4900353488475568820 , B = 10849176093999290664 , Cin = 1 , Actual Result = 15749529582474859485 , Correct result = 15749529582474859485
A =  9545348660785355822 , B = 18349818812540470556 , Cin = 1 , Actual Result = 27895167473325826379 , Correct result = 27895167473325826379
A =  4842890801916121116 , B = 11436827748005578970 , Cin = 0 , Actual Result = 16279718549921700086 , Correct result = 16279718549921700086
A = 13399621161371157616 , B =  3403229686307854778 , Cin = 0 , Actual Result = 16802850847679012394 , Correct result = 16802850847679012394
A = 10170524451826697685 , B =  1983661053437804729 , Cin = 0 , Actual Result = 12154185505264502414 , Correct result = 12154185505264502414
A =  1379683848030855616 , B =  4510781078361217718 , Cin = 0 , Actual Result =  5890464926392073334 , Correct result =  5890464926392073334
A =  4324580959197419398 , B = 17117479437854219134 , Cin = 1 , Actual Result = 21442060397051638533 , Correct result = 21442060397051638533
A = 18242082054079327097 , B =   834301828324453985 , Cin = 1 , Actual Result = 19076383882403781083 , Correct result = 19076383882403781083
A = 12133707092539707782 , B = 11147684740919482869 , Cin = 1 , Actual Result = 23281391833459190652 , Correct result = 23281391833459190652
A = 16349602413318075841 , B =  2716369090733257880 , Cin = 1 , Actual Result = 19065971504051333722 , Correct result = 19065971504051333722
A = 14217888809699436268 , B = 15297462018997014862 , Cin = 1 , Actual Result = 29515350828696451131 , Correct result = 29515350828696451131
A =   507702856292928673 , B = 14976369404079793894 , Cin = 0 , Actual Result = 15484072260372722567 , Correct result = 15484072260372722567
A = 14318619477080153386 , B = 11271918571903909790 , Cin = 1 , Actual Result = 25590538048984063177 , Correct result = 25590538048984063177
A =  7287745357585638344 , B =  3864309941850292499 , Cin = 1 , Actual Result = 11152055299435930844 , Correct result = 11152055299435930844
A =  6702937044127833526 , B =  6666055021908896708 , Cin = 0 , Actual Result = 13368992066036730234 , Correct result = 13368992066036730234
A =  4592329665341201076 , B =  9038150484568019078 , Cin = 0 , Actual Result = 13630480149909220154 , Correct result = 13630480149909220154
A = 16054873887888187698 , B = 17458666730750463364 , Cin = 0 , Actual Result = 33513540618638651062 , Correct result = 33513540618638651062
A = 15030104742282426793 , B = 18269813940825729422 , Cin = 1 , Actual Result = 33299918683108156216 , Correct result = 33299918683108156216
A =  7262122336785174767 , B = 13452613229740090678 , Cin = 1 , Actual Result = 20714735566525265446 , Correct result = 20714735566525265446
A =  4928599939431120747 , B =  5616975666592330926 , Cin = 0 , Actual Result = 10545575606023451673 , Correct result = 10545575606023451673
A = 10846040286833205032 , B = 16242707878991322955 , Cin = 0 , Actual Result = 27088748165824527987 , Correct result = 27088748165824527987
A =  8537133758823990541 , B =  7542997010555837720 , Cin = 0 , Actual Result = 16080130769379828261 , Correct result = 16080130769379828261
A = 11366914191769216065 , B =  3021890419496909272 , Cin = 0 , Actual Result = 14388804611266125337 , Correct result = 14388804611266125337
A = 17394866462248411739 , B =  4155700350800880132 , Cin = 0 , Actual Result = 21550566813049291871 , Correct result = 21550566813049291871
A = 15853590670494027026 , B = 17476137976323969593 , Cin = 1 , Actual Result = 33329728646817996620 , Correct result = 33329728646817996620
A =  4652321224067190571 , B =  1420134885176870344 , Cin = 1 , Actual Result =  6072456109244060916 , Correct result =  6072456109244060916
A =  9377121165401419295 , B = 14653728670031356760 , Cin = 0 , Actual Result = 24030849835432776055 , Correct result = 24030849835432776055
A = 14637385530755101714 , B =  3081259659758985905 , Cin = 1 , Actual Result = 17718645190514087620 , Correct result = 17718645190514087620
A =  8840878282947373867 , B = 10646828546434270125 , Cin = 0 , Actual Result = 19487706829381643992 , Correct result = 19487706829381643992
A = 17570695985827996839 , B = 11878901842382393529 , Cin = 1 , Actual Result = 29449597828210390369 , Correct result = 29449597828210390369
A =  2919371946606596041 , B =  1533155899126603425 , Cin = 0 , Actual Result =  4452527845733199466 , Correct result =  4452527845733199466
```

```
A =  4729067146647576645 , B = 13350380628440597116 , Cin = 0 , Actual Result = 18079447775088173761 , Correct result = 18079447775088173761
A = 13249496025490911848 , B =  8656800224540757382 , Cin = 0 , Actual Result = 21906296250031669230 , Correct result = 21906296250031669230
A = 10667187107119141184 , B = 16338807408598821366 , Cin = 0 , Actual Result = 27005994515717962550 , Correct result = 27005994515717962550
A =  2080224568498347892 , B =  2169757815884376752 , Cin = 0 , Actual Result =  4249982384382724644 , Correct result =  4249982384382724644
A =   787062400018076514 , B =   856419605301776097 , Cin = 1 , Actual Result =  1643482005319852612 , Correct result =  1643482005319852612
A = 14066435461400093641 , B =  6986945324478090021 , Cin = 1 , Actual Result = 21053380785878183663 , Correct result = 21053380785878183663
A =   276451341057385050 , B =   435262620490758956 , Cin = 1 , Actual Result =   711713961548144007 , Correct result =   711713961548144007
A =  6584515227711274043 , B = 14921778647611196151 , Cin = 0 , Actual Result = 21506293875322470194 , Correct result = 21506293875322470194
A = 10388179400313599829 , B =  4134314846647423648 , Cin = 0 , Actual Result = 14522494246961023477 , Correct result = 14522494246961023477
A =  9717906036438529501 , B = 13604321662470232395 , Cin = 0 , Actual Result = 23322227698908761896 , Correct result = 23322227698908761896
A = 13682167097951544573 , B =  9350486590232884111 , Cin = 1 , Actual Result = 23032653688184428685 , Correct result = 23032653688184428685
A =  6411223375201583901 , B = 14599751440736034628 , Cin = 0 , Actual Result = 21010974815937618529 , Correct result = 21010974815937618529
A =  2988724748783024365 , B =  5106157895831935992 , Cin = 0 , Actual Result =  8094882644614960357 , Correct result =  8094882644614960357
A =  2552130617057339268 , B =  5208551029808468620 , Cin = 1 , Actual Result =  7760681646865807889 , Correct result =  7760681646865807889
A =  4790518853978120554 , B =  4498831581596335274 , Cin = 0 , Actual Result =  9289350435574455828 , Correct result =  9289350435574455828
A = 14240769648404090298 , B = 10557315099291156134 , Cin = 0 , Actual Result = 24798084747695246432 , Correct result = 24798084747695246432
A = 11857833000371029154 , B = 11441887698858848276 , Cin = 1 , Actual Result = 23299720699229877431 , Correct result = 23299720699229877431
A =  2562677138703003740 , B = 11290377177201022854 , Cin = 0 , Actual Result = 13853054315904026594 , Correct result = 13853054315904026594
A = 10818297406556333520 , B =   281957693562570124 , Cin = 0 , Actual Result = 11100255100118903644 , Correct result = 11100255100118903644
A =  8388798627972512017 , B =  4171601488556564043 , Cin = 0 , Actual Result = 12560400116529076060 , Correct result = 12560400116529076060
A =  5912926848739575075 , B = 11287995635579577801 , Cin = 1 , Actual Result = 17200922484319152877 , Correct result = 17200922484319152877
A =  3711922014871098580 , B =  6354901988360077725 , Cin = 0 , Actual Result = 10066824003231176305 , Correct result = 10066824003231176305
A = 17057694583570480969 , B = 10233669411680597792 , Cin = 1 , Actual Result = 27291363995251078762 , Correct result = 27291363995251078762
A = 10757835253712445923 , B = 12598589463391762632 , Cin = 0 , Actual Result = 23356424717104208555 , Correct result = 23356424717104208555
A = 10784335686462253444 , B =  9907668569352404793 , Cin = 0 , Actual Result = 20692004255814658237 , Correct result = 20692004255814658237
A = 16729631061642086745 , B =  5473326653731629678 , Cin = 1 , Actual Result = 22202957715373716424 , Correct result = 22202957715373716424
A =  6739025218940073894 , B =  9792035119273966592 , Cin = 1 , Actual Result = 16531060338214040487 , Correct result = 16531060338214040487
A =  3217849806052614668 , B =  7482508470734637509 , Cin = 0 , Actual Result = 10700358276787252177 , Correct result = 10700358276787252177
A =                    0 , B =                    0 , Cin = 0 , Actual Result =                    0 , Correct result =                    0
A =                    0 , B =                    0 , Cin = 1 , Actual Result =                    1 , Correct result =                    1
A = 18446744073709551615 , B = 18446744073709551615 , Cin = 1 , Actual Result = 36893488147419103231 , Correct result = 36893488147419103231
A = 18446744073709551615 , B = 18446744073709551615 , Cin = 0 , Actual Result = 36893488147419103230 , Correct result = 36893488147419103230
Test pass
```

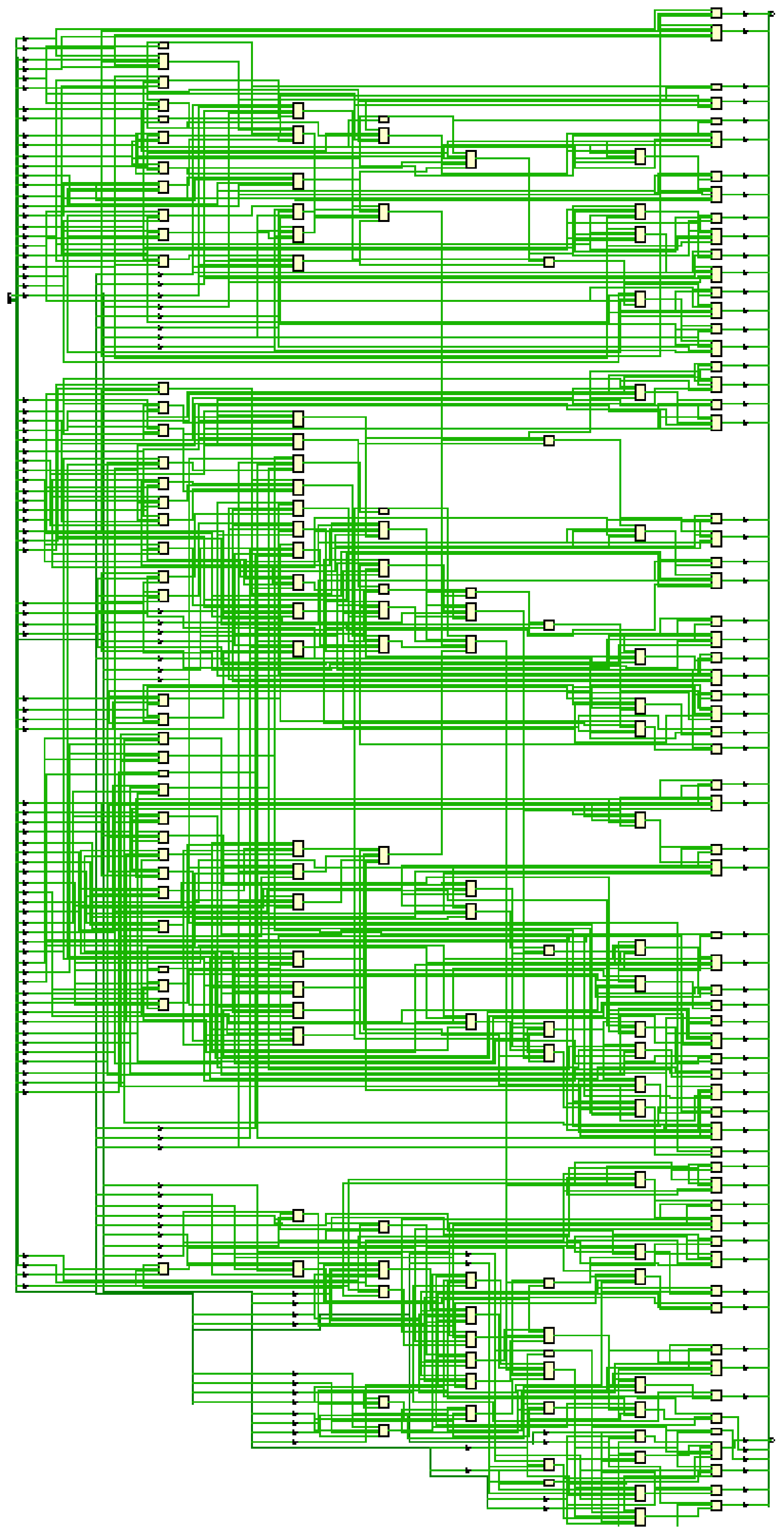
8. SYNTHESIS

a) Synthesis schematic

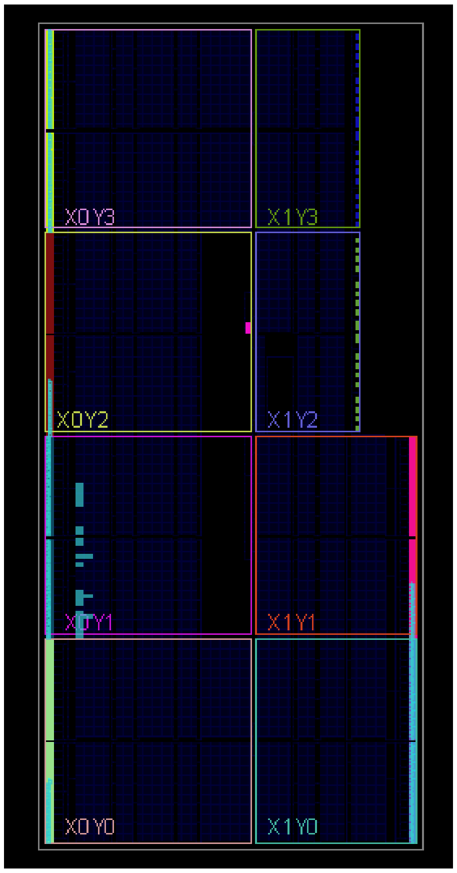


9. IMPLEMENTATION

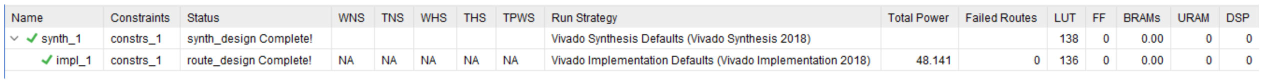
a) Schematic



b) Floorplan

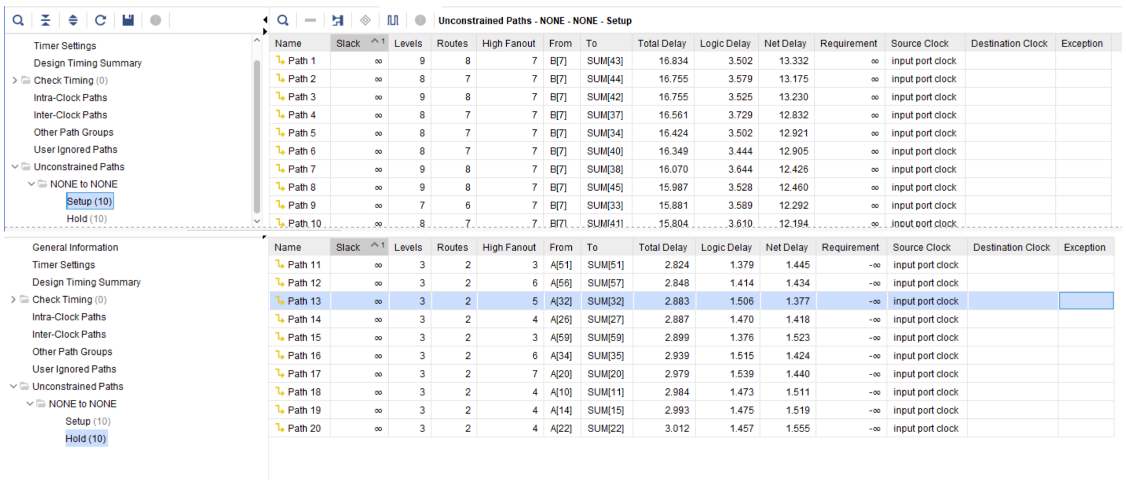


10. Design Runs



Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Run Strategy	Total Power	Failed Routes	LUT	FF	BRAMs	URAM	DSP
✓ synth_1	constrs_1	synth_design Complete!						Vivado Synthesis Defaults (Vivado Synthesis 2018)			138	0	0.00	0	0
✓ impl_1	constrs_1	route_design Complete!	NA	NA	NA	NA	NA	Vivado Implementation Defaults (Vivado Implementation 2018)	48.141	0	136	0	0.00	0	0

11. Timing



Timer Settings
Design Timing Summary
Check Timing (0)
Intra-Clock Paths
Inter-Clock Paths
Other Path Groups
User Ignored Paths
Unconstrained Paths
NONE to NONE
Setup (10)
Hold (10)

Unconstrained Paths - NONE - NONE - Setup

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock	Exception
Path 1	∞	9	8	7	B[7]	SUM[43]	16.834	3.502	13.332	∞	input port clock		
Path 2	∞	8	7	7	B[7]	SUM[44]	16.755	3.579	13.175	∞	input port clock		
Path 3	∞	9	8	7	B[7]	SUM[42]	16.755	3.525	13.230	∞	input port clock		
Path 4	∞	8	7	7	B[7]	SUM[37]	16.561	3.729	12.832	∞	input port clock		
Path 5	∞	8	7	7	B[7]	SUM[34]	16.424	3.502	12.921	∞	input port clock		
Path 6	∞	8	7	7	B[7]	SUM[40]	16.349	3.444	12.905	∞	input port clock		
Path 7	∞	9	8	7	B[7]	SUM[38]	16.070	3.644	12.426	∞	input port clock		
Path 8	∞	9	8	7	B[7]	SUM[45]	15.987	3.528	12.460	∞	input port clock		
Path 9	∞	7	6	7	B[7]	SUM[33]	15.881	3.589	12.292	∞	input port clock		
Path 10	∞	8	7	7	B[7]	SUM[41]	15.804	3.610	12.194	∞	input port clock		

General Information
Timer Settings
Design Timing Summary
Check Timing (0)
Intra-Clock Paths
Inter-Clock Paths
Other Path Groups
User Ignored Paths
Unconstrained Paths
NONE to NONE
Setup (10)
Hold (10)

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock	Exception
Path 11	∞	3	2	3	A[51]	SUM[51]	2.824	1.379	1.445	-∞	input port clock		
Path 12	∞	3	2	6	A[56]	SUM[57]	2.848	1.414	1.434	-∞	input port clock		
Path 13	∞	3	2	5	A[32]	SUM[32]	2.883	1.506	1.377	-∞	input port clock		
Path 14	∞	3	2	4	A[26]	SUM[27]	2.887	1.470	1.418	-∞	input port clock		
Path 15	∞	3	2	3	A[59]	SUM[59]	2.899	1.376	1.523	-∞	input port clock		
Path 16	∞	3	2	6	A[34]	SUM[35]	2.939	1.515	1.424	-∞	input port clock		
Path 17	∞	3	2	7	A[20]	SUM[20]	2.979	1.539	1.440	-∞	input port clock		
Path 18	∞	3	2	4	A[10]	SUM[11]	2.984	1.473	1.511	-∞	input port clock		
Path 19	∞	3	2	4	A[14]	SUM[15]	2.993	1.475	1.519	-∞	input port clock		
Path 20	∞	3	2	4	A[22]	SUM[22]	3.012	1.457	1.555	-∞	input port clock		