



Comparison of Wallace, Vedic and Dadda Multipliers

D.SONY

Geethanjali College of Engineering & Technology

ABSTRACT: *There are various multiplication methods that are known. of the many, three structural multipliers, Wallace, Vedic and Dadda are compared in this paper. For comparison, the three multipliers are simulated in cadence. Simulated results of the multipliers are attached. Later comparison is done among the multipliers based on the delay, area and power requirements.*

KEYWORDS : *Multipliers, Wallace, Dadda, Vedic , Vertical Crosswise, Xilinx, LUTs, delay.*

I INTRODUCTION: Multiplication is the most critical operation in every computational system. Multiplication can be well thought-out as a series of repeated additions. Since, in today's world, speed is a major factor hence multiplication must be performed faster as multiplication is the base of any signal processing device. The three structural multipliers that are compared are: 1. Wallace Multiplier. 2. Vedic Multiplier 3. Dadda Multiplier.

II WALLACE MULTIPLIER Wallace multiplier is a multiplier which multiplies two numbers in a structural format. It was devised by Australian Computer Scientist Chris Wallace in 1964. Wallace introduced parallel multiplier architecture. The Wallace tree has three steps:

1. Partial Product Generation
2. Partial Product Reduction
3. Partial Product Addition

In the first stage, as per normal multiplication process, partial products are generated. More the number of partial products more will be the height of the set of partial products generated. In the second stage, this height is reduced by placing half adders or full adders to a height of two. Here adders are placed wherever possible. Once the height of the partial products is reduced to two, then addition is performed between them to yield the final result.

In Wallace multiplier, adders are used height wise or in other words column wise. In order to proceed to next step, first the unused bits are written column wise and then the sum of the adder of that stage followed by the carry of the previous stage.

Step 1: Generation of Partial Products.

Step 2: Reduction of Partial Products generated. In this step, the height of the partial products is reduced step by step till it reaches to the height of two. The height of the intermediate steps is such that uniformity is maintained with respect to the previous stage or intermediate matrix. For reducing heights adders are used. So the reduced intermediate matrix is given below which has the height of six. Notice that vacant or unused bits are placed first followed by the sum of the adder followed by the carry of the previous step. Here calculation starts from the LSB side.

Step 3: Addition of partial products. Then the matrix of height two is added by any known adder like carry select adder or parallel adder etc. The final sum which will be the result of the multiplication

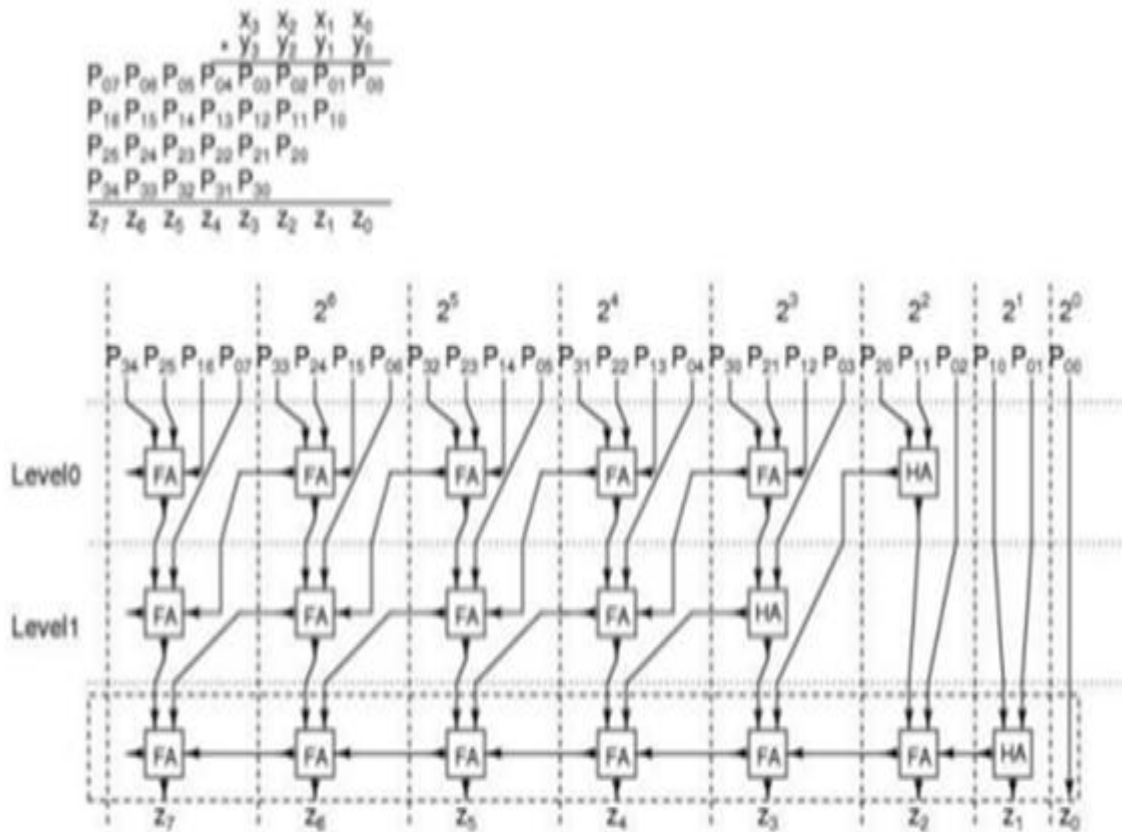


Figure 1. Wallace tree multiplier

III VEDIC MULTIPLIER

The structure of Vedic Multiplier is a Vertical Crosswise. It generates all partial products and their sum in one step. Since the partial products and their sum are calculated in Parallel. The Vedic Multiplier is one of the fastest multiplier. The Vedic Multiplier works under Algorithmic basis.

Algorithm

- ❖ Divided the Multiplicand A and Multiplier B into two equal parts, each consisting of $[N \text{ to } (N/2) - 1]$ bits and $[N/2 \text{ to } 1]$ bits respectively, where first bit parts indicates the MSB and other represents LSB.
- ❖ Represents the parts of A as A_m and A_l and parts of B as B_m and B_l . Now represents A and B as A_m, A_l and B_m, B_l respectively.
- ❖ These inputs are multiplied with Vertical and Crosswise basis,
- ❖ $A_m \times B_m, A_m \times B_l, A_l \times B_l, A_l \times B_m$
- ❖ The individual multiplication products can be obtained by the partitioning method and applying the basic building blocks.

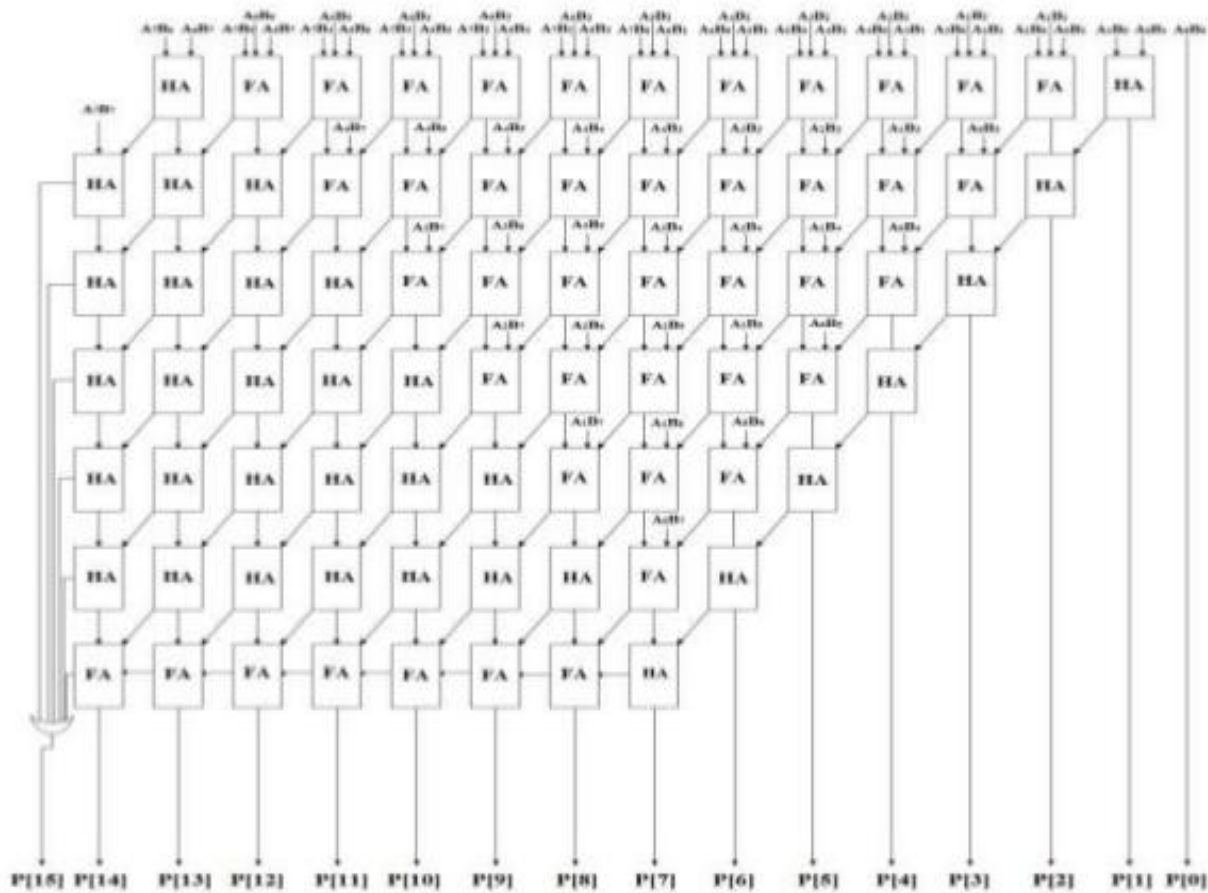


Figure 2. Vedic multiplier

IV DADDA MULTIPLIER: Dadda multiplier is another type of structural multiplier. This multiplier was invented by computer scientist Luigi Dadda in 1965. Dadda multiplier undergoes four stages i.e. generation of partial products, formation of delta structure and reduction of partial products and addition of reduced matrix. In order to realize the minimum number of reduction stages, the height of each intermediate matrix is not more than 1.5 times the height of its successor. Here also, the adders are used to add the bits height wise or column wise. Here adders are not extensively used as in case of Wallace multiplier. Here adders are used just to make the height apt of each intermediate matrix. A. Example Taking the same example that was taken in case of Wallace multiplier, that is we have two decimal numbers 99 and 78 which are to be multiplied. Step 1: Generation of Partial Products.

Step 2: Formation of delta structure. In this step, all columns of the matrix are shifted upwards to form an inverted delta structure. The structure of the inverted delta so formed

Step 3: Reduction of partial products. In Dadda multiplier, height of the partial products are reduced but after forming an inverted delta structure, as described in step 2. Here also adders are used to reduce the height of the matrix. In Dadda multiplier, first the sum from the adder is placed followed by the vacant or unused bits and later the carry from the previous stage is placed. This is the format of arrangement of heights in case of Dadda multiplier.

Step 3: Addition of partial products. Then the matrix of height two is added by any known adder like carry select adder or parallel adder etc. The final sum which will be the result of the multiplication

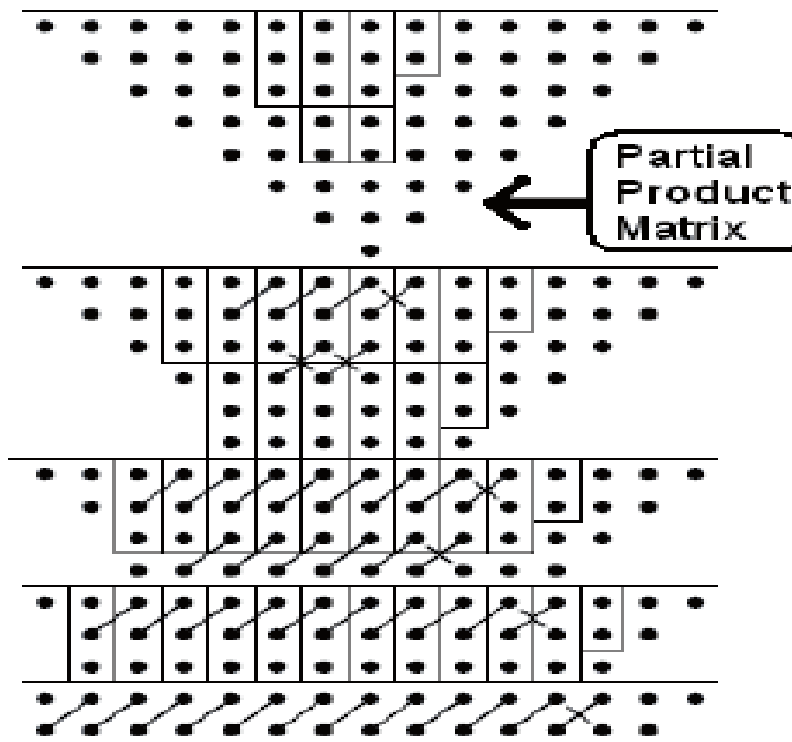


Figure 3. Dadda 8 bit multiplier

V SIMULATION RESULT: Wallace multiplier, Vedic multiplier and Dadda multiplier are simulated in cadence. The Simulation results include schematic and area, power and delay of all the three multipliers.

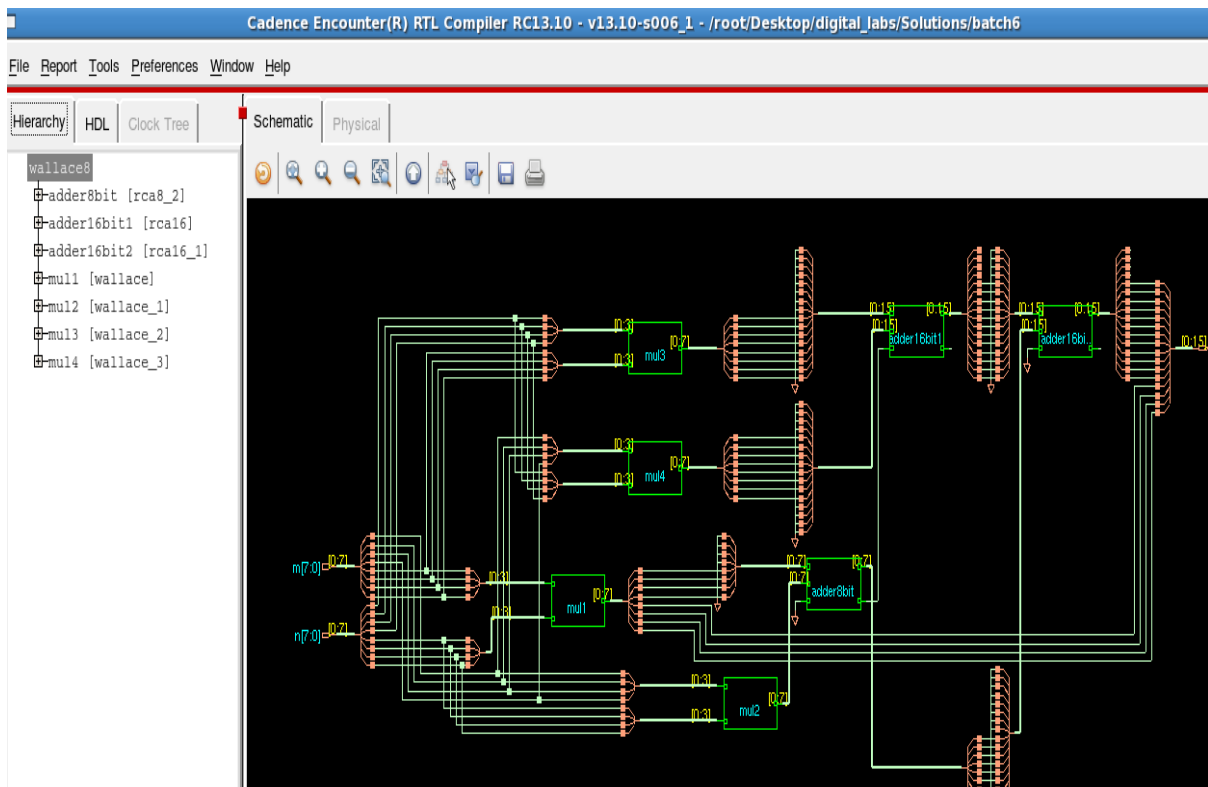


Figure 4. Wallace 8 bit multiplier schematic

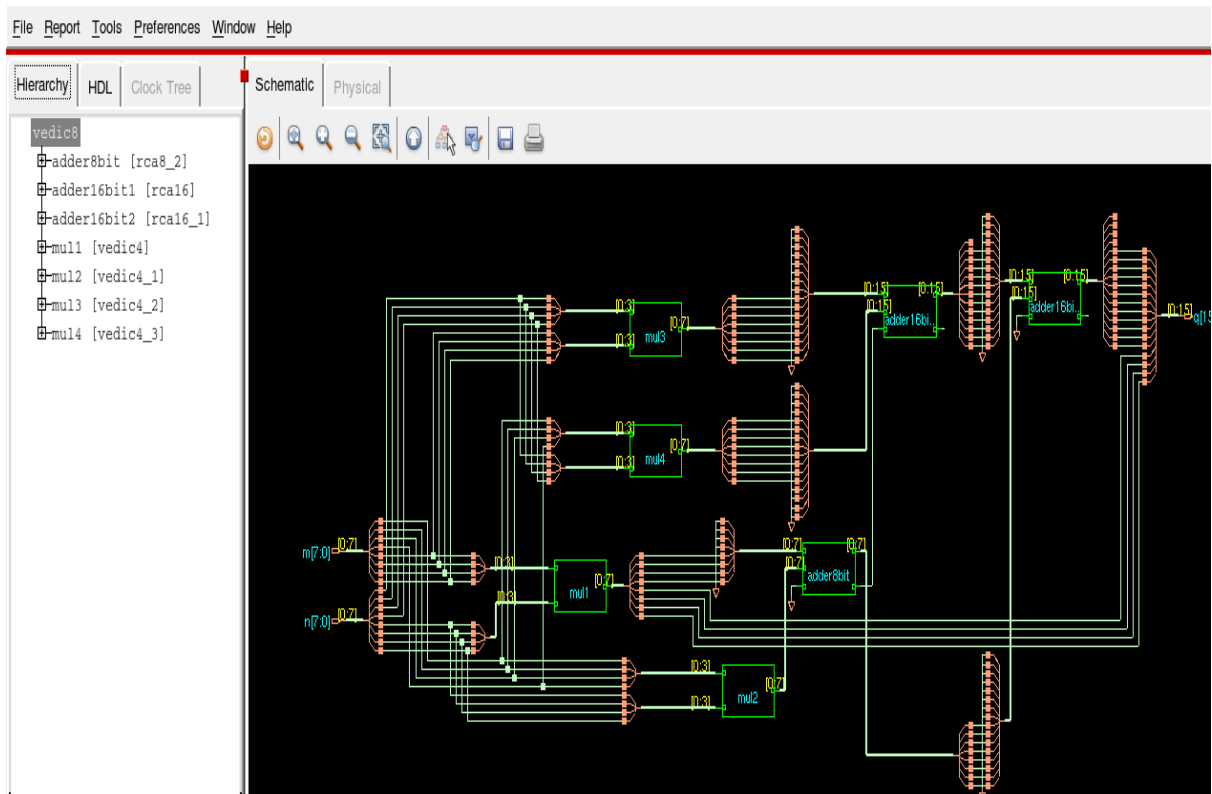


Figure 4.Vedic8 bit multiplier Schematic

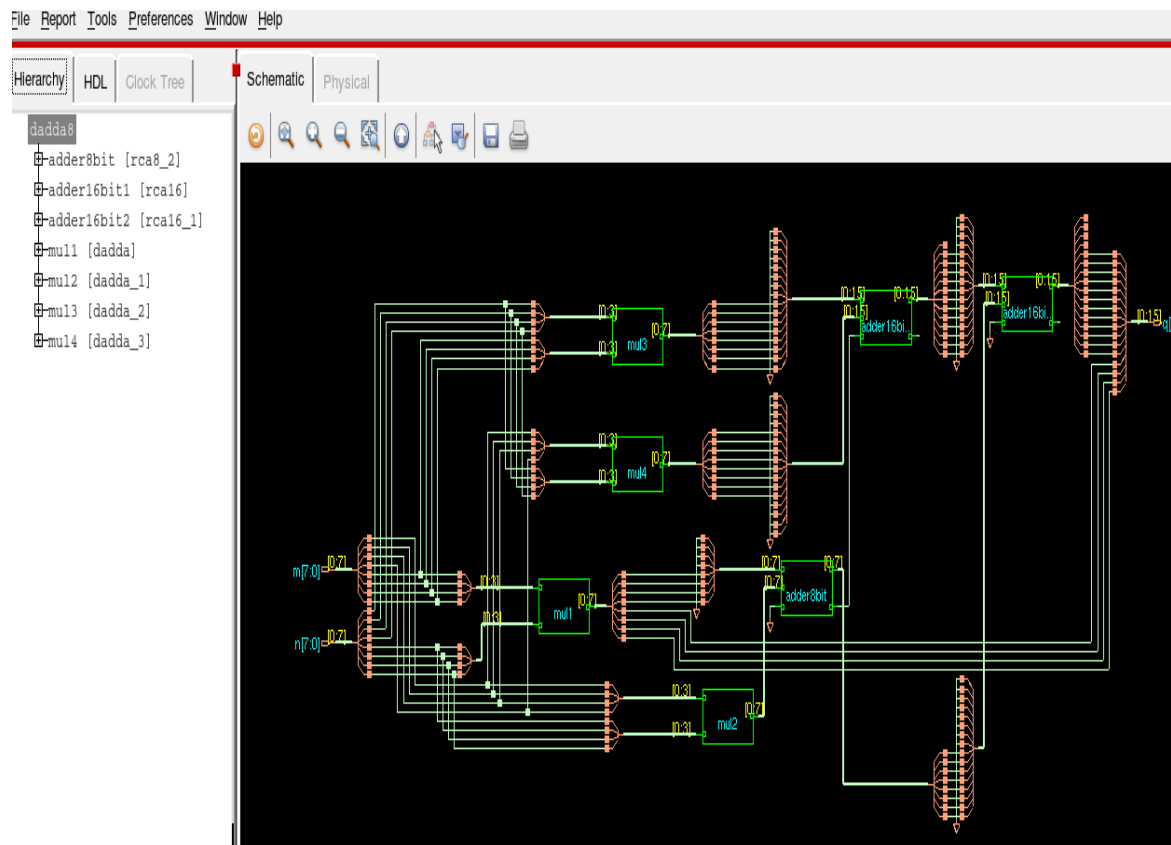
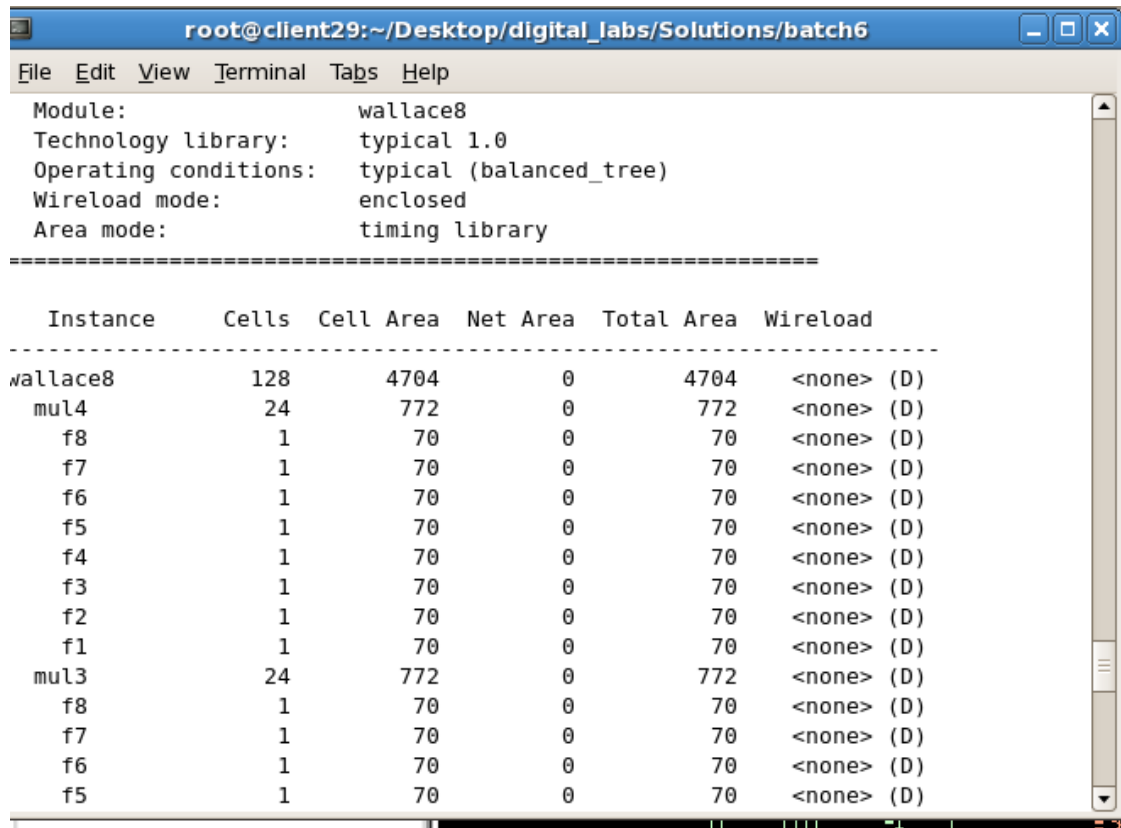


Figure 4.Dadda8 bit multiplierSchematic



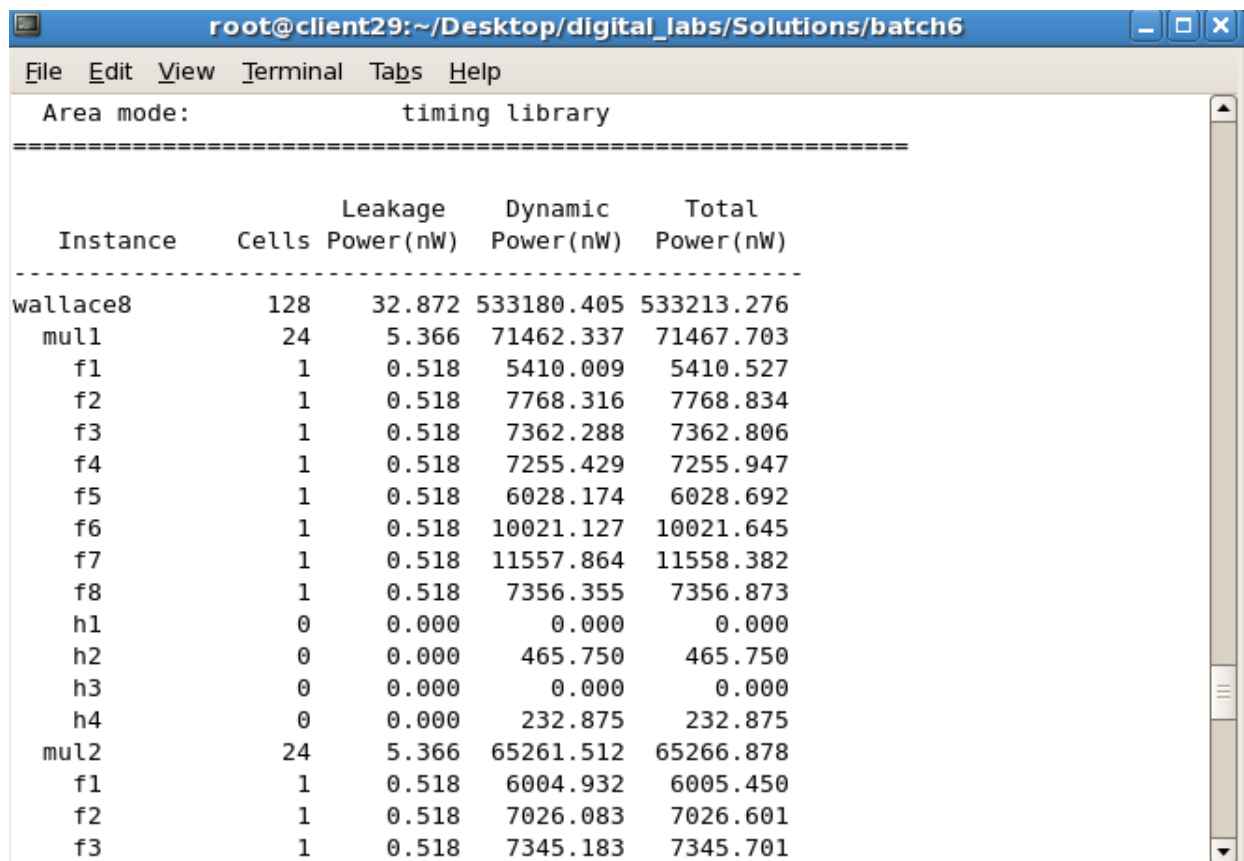
```

root@client29:~/Desktop/digital_labs/Solutions/batch6
File Edit View Terminal Tabs Help
Module: wallace8
Technology library: typical 1.0
Operating conditions: typical (balanced_tree)
Wireload mode: enclosed
Area mode: timing library
=====

```

Instance	Cells	Cell Area	Net Area	Total Area	Wireload
wallace8	128	4704	0	4704	<none> (D)
mul4	24	772	0	772	<none> (D)
f8	1	70	0	70	<none> (D)
f7	1	70	0	70	<none> (D)
f6	1	70	0	70	<none> (D)
f5	1	70	0	70	<none> (D)
f4	1	70	0	70	<none> (D)
f3	1	70	0	70	<none> (D)
f2	1	70	0	70	<none> (D)
f1	1	70	0	70	<none> (D)
mul3	24	772	0	772	<none> (D)
f8	1	70	0	70	<none> (D)
f7	1	70	0	70	<none> (D)
f6	1	70	0	70	<none> (D)
f5	1	70	0	70	<none> (D)

Figure 4. Wallace 8 bit multiplier area



```

root@client29:~/Desktop/digital_labs/Solutions/batch6
File Edit View Terminal Tabs Help
Area mode: timing library
=====

```

Instance	Cells	Leakage Power(nW)	Dynamic Power(nW)	Total Power(nW)
wallace8	128	32.872	533180.405	533213.276
mul1	24	5.366	71462.337	71467.703
f1	1	0.518	5410.009	5410.527
f2	1	0.518	7768.316	7768.834
f3	1	0.518	7362.288	7362.806
f4	1	0.518	7255.429	7255.947
f5	1	0.518	6028.174	6028.692
f6	1	0.518	10021.127	10021.645
f7	1	0.518	11557.864	11558.382
f8	1	0.518	7356.355	7356.873
h1	0	0.000	0.000	0.000
h2	0	0.000	465.750	465.750
h3	0	0.000	0.000	0.000
h4	0	0.000	232.875	232.875
mul2	24	5.366	65261.512	65266.878
f1	1	0.518	6004.932	6005.450
f2	1	0.518	7026.083	7026.601
f3	1	0.518	7345.183	7345.701

Figure5. Wallace 8 bit multiplier power

```

root@client29:~/Desktop/digital_labs/Solutions/batch6
File Edit View Terminal Tabs Help
g22/C0 ADDHXL 1 6.0 63 +123 4367 F
f1/cout
f2/cin
g22/B +0 4367
g22/C0 ADDHXL 1 6.0 63 +116 4483 F
f2/cout
f3/cin
g22/B +0 4483
g22/C0 ADDHXL 1 2.2 44 +99 4582 F
f3/cout
f4/cin
g16/B +0 4582
g16/Y XOR2XL 1 0.0 41 +195 4777 R
f4/s
fourbit1/s[3]
eig2/s[3]
adder16bit2/s[11]
q[15] out port +0 4777 R
-----
Timing slack : UNCONSTRAINED
Start-point : n[0]
End-point : q[15]

rc:/>

```

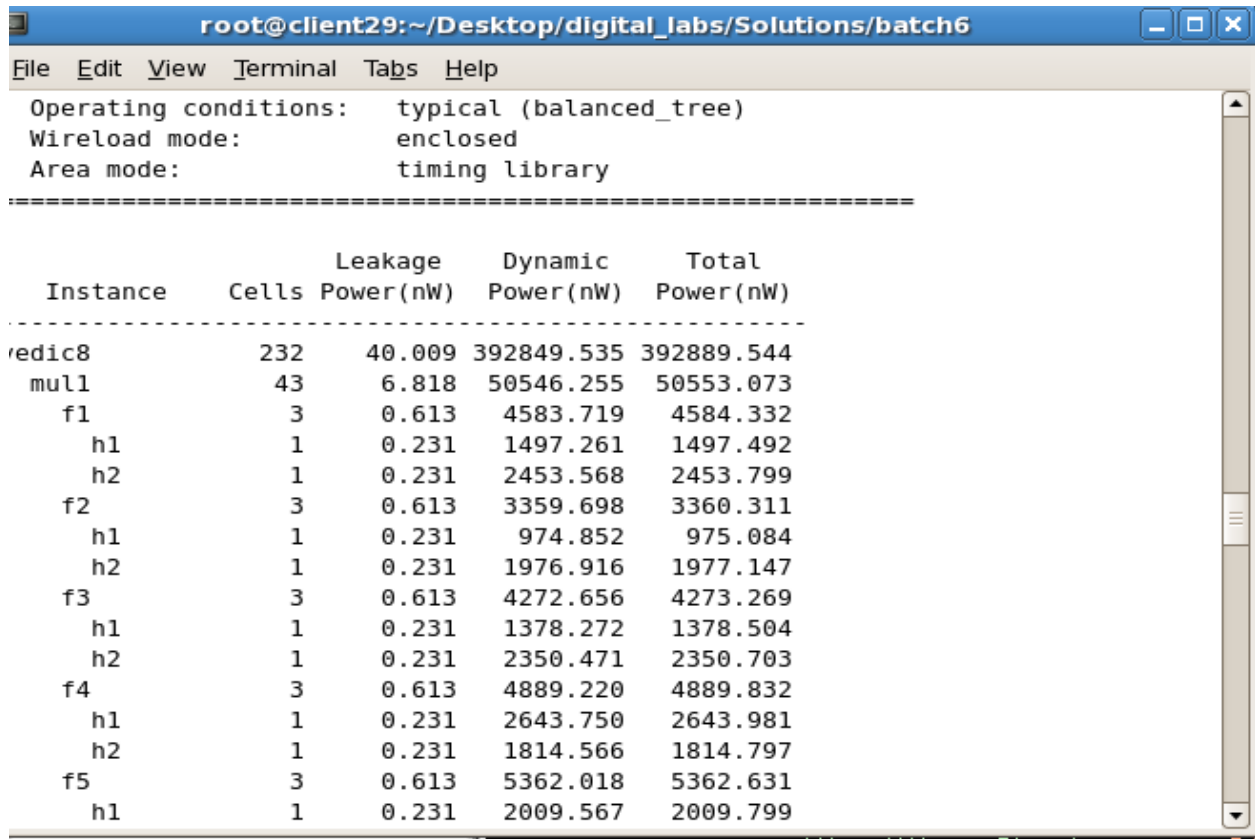
Figure6. Wallace 8 bit multiplier delay

```

root@client29:~/Desktop/digital_labs/Solutions/batch6
File Edit View Terminal Tabs Help
Operating conditions: typical (balanced_tree)
Wireload mode: enclosed
Area mode: timing library
=====
Instance      Cells  Cell Area  Net Area  Total Area  Wireload
-----
vedic8        232    5908       0         5908      <none> (D)
mul4          43    1015       0         1015      <none> (D)
  f8           3      86        0          86      <none> (D)
    h2          1     37        0          37      <none> (D)
    h1          1     37        0          37      <none> (D)
  f7           3      86        0          86      <none> (D)
    h2          1     37        0          37      <none> (D)
    h1          1     37        0          37      <none> (D)
  f6           3      86        0          86      <none> (D)
    h2          1     37        0          37      <none> (D)
    h1          1     37        0          37      <none> (D)
  f5           3      86        0          86      <none> (D)
    h2          1     37        0          37      <none> (D)
    h1          1     37        0          37      <none> (D)
  f4           3      86        0          86      <none> (D)
    h2          1     37        0          37      <none> (D)
    h1          1     37        0          37      <none> (D)

```

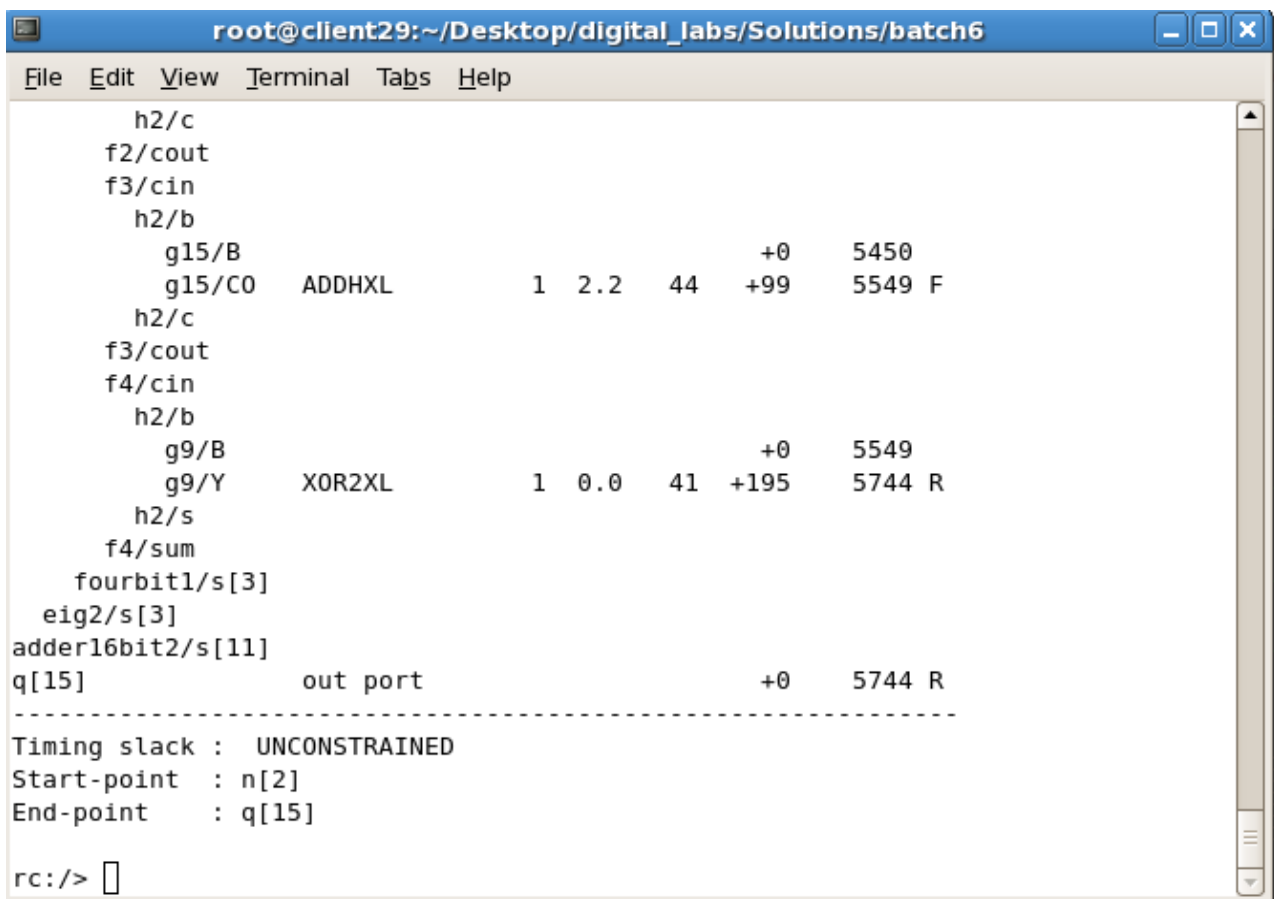
Figure 7.Vedic 8 bit multiplier area



Operating conditions: typical (balanced_tree)
Wireload mode: enclosed
Area mode: timing library

Instance	Cells	Leakage Power(nW)	Dynamic Power(nW)	Total Power(nW)
redic8	232	40.009	392849.535	392889.544
mul1	43	6.818	50546.255	50553.073
f1	3	0.613	4583.719	4584.332
h1	1	0.231	1497.261	1497.492
h2	1	0.231	2453.568	2453.799
f2	3	0.613	3359.698	3360.311
h1	1	0.231	974.852	975.084
h2	1	0.231	1976.916	1977.147
f3	3	0.613	4272.656	4273.269
h1	1	0.231	1378.272	1378.504
h2	1	0.231	2350.471	2350.703
f4	3	0.613	4889.220	4889.832
h1	1	0.231	2643.750	2643.981
h2	1	0.231	1814.566	1814.797
f5	3	0.613	5362.018	5362.631
h1	1	0.231	2009.567	2009.799

Figure 8.Vedic 8 bit multiplier power

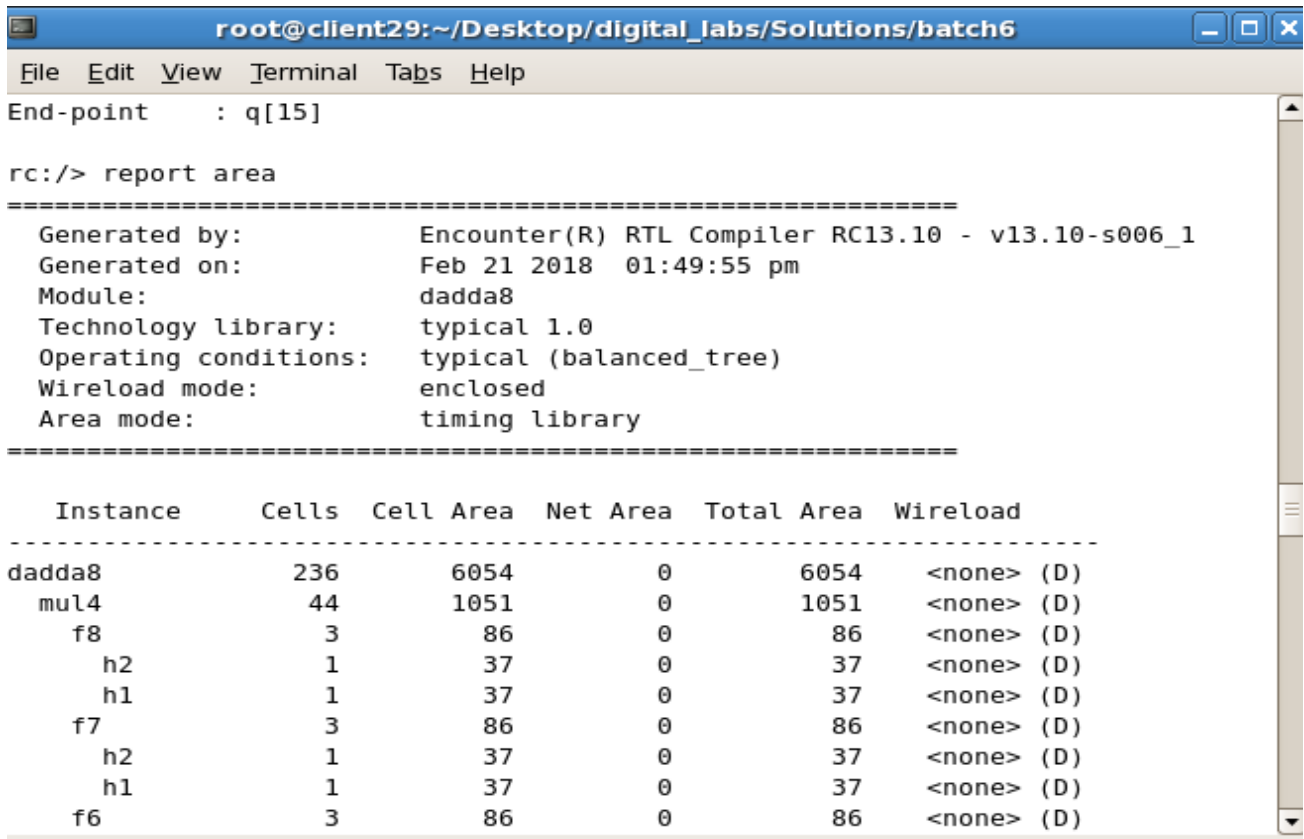


h2/c
f2/cout
f3/cin
h2/b
 g15/B +0 5450
 g15/C0 ADDHXL 1 2.2 44 +99 5549 F
h2/c
f3/cout
f4/cin
h2/b
 g9/B +0 5549
 g9/Y XOR2XL 1 0.0 41 +195 5744 R
h2/s
f4/sum
fourbit1/s[3]
eig2/s[3]
adder16bit2/s[11]
q[15] out port +0 5744 R

Timing slack : UNCONSTRAINED
Start-point : n[2]
End-point : q[15]

rc:/> █

Figure 9.Vedic 8 bit multiplier delay



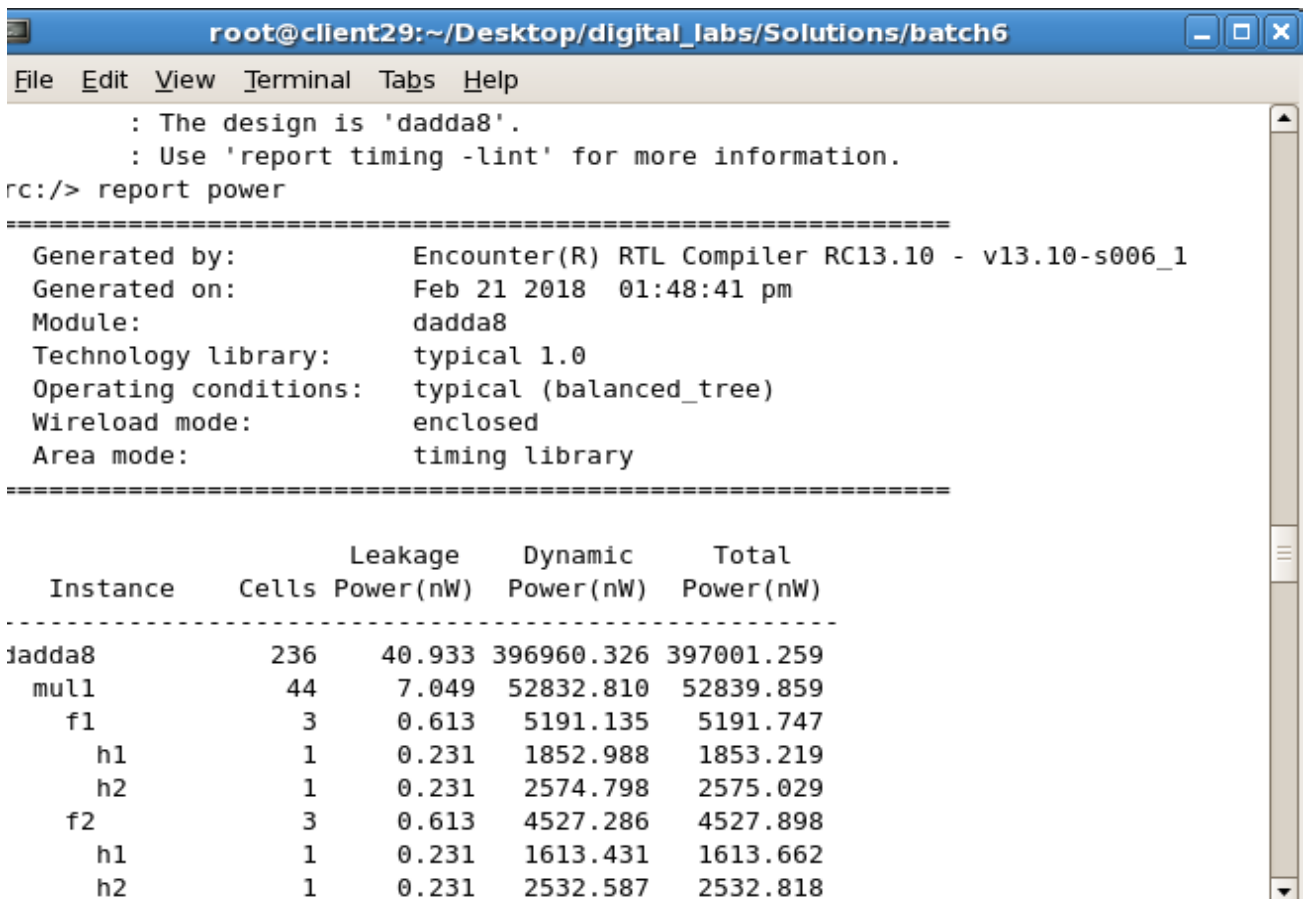
```

root@client29:~/Desktop/digital_labs/Solutions/batch6
File Edit View Terminal Tabs Help
End-point : q[15]

rc:/> report area
=====
Generated by:      Encounter(R) RTL Compiler RC13.10 - v13.10-s006_1
Generated on:      Feb 21 2018  01:49:55 pm
Module:            dadda8
Technology library: typical 1.0
Operating conditions: typical (balanced_tree)
Wireload mode:     enclosed
Area mode:         timing library
=====

  Instance      Cells  Cell Area  Net Area  Total Area  Wireload
-----
dadda8          236    6054      0         6054      <none> (D)
  mul4          44    1051      0         1051      <none> (D)
    f8           3     86       0          86      <none> (D)
      h2          1     37       0          37      <none> (D)
      h1          1     37       0          37      <none> (D)
    f7           3     86       0          86      <none> (D)
      h2          1     37       0          37      <none> (D)
      h1          1     37       0          37      <none> (D)
    f6           3     86       0          86      <none> (D)
  
```

Figure10. Dadda 8 bit multiplier area



```

root@client29:~/Desktop/digital_labs/Solutions/batch6
File Edit View Terminal Tabs Help
: The design is 'dadda8'.
: Use 'report timing -lint' for more information.
rc:/> report power
=====
Generated by:      Encounter(R) RTL Compiler RC13.10 - v13.10-s006_1
Generated on:      Feb 21 2018  01:48:41 pm
Module:            dadda8
Technology library: typical 1.0
Operating conditions: typical (balanced_tree)
Wireload mode:     enclosed
Area mode:         timing library
=====

  Instance      Cells  Leakage  Dynamic  Total
                Cells  Power(nW) Power(nW) Power(nW)
-----
dadda8          236    40.933  396960.326  397001.259
  mul1          44     7.049   52832.810   52839.859
    f1           3     0.613   5191.135   5191.747
      h1          1     0.231   1852.988   1853.219
      h2          1     0.231   2574.798   2575.029
    f2           3     0.613   4527.286   4527.898
      h1          1     0.231   1613.431   1613.662
      h2          1     0.231   2532.587   2532.818
  
```

Figure 11.Dadda 8 bit multiplier power

```

root@client29:~/Desktop/digital_labs/Solutions/batch6
File Edit View Terminal Tabs Help
: The design is 'dadda8'.
: Use 'report timing -lint' for more information.
rc:/> report power

=====
Generated by:      Encounter(R) RTL Compiler RC13.10 - v13.10-s006_1
Generated on:      Feb 21 2018  01:48:41 pm
Module:            dadda8
Technology library: typical 1.0
Operating conditions: typical (balanced_tree)
Wireload mode:     enclosed
Area mode:         timing library
=====

Instance          Cells Leakage Power(nW) Dynamic Power(nW) Total Power(nW)
-----
ladda8             236   40.933 396960.326 397001.259
  mull             44    7.049 52832.810 52839.859
    f1              3    0.613 5191.135 5191.747
      h1             1    0.231 1852.988 1853.219
      h2             1    0.231 2574.798 2575.029
    f2              3    0.613 4527.286 4527.898
      h1             1    0.231 1613.431 1613.662
      h2             1    0.231 2532.587 2532.818

rc:/> 

```

Figure 12.Dadda 8 bit multiplier delay

VI COMPARISON

Comparison of Wallace, Vedic and Dadda multipliers are shown below where parameters are area, power requirement and delay.

Parameters	Area(μm^2)	Delay(ps)	Power(μW)
Wallace	4704	4777	533.21
Vedic	5908	5744	392.88
Dadda	6054	6012	397.00

On the basis of simulation performed, it can be concluded that Area and delay required by Wallace multiplier is less i.e, $4704\mu\text{m}^2$ and 4777ps respectively and power requirement is less for Vedic multiplier. The above comparison is done using RCA(Ripple carry adder).