IOSR Journal of VLSI and Signal Processing (IOSR-JVSP)
Volume 7, Issue 4, Ver. II (Jul. - Aug. 2017), PP 56-65
e-ISSN: 2319 – 4200, p-ISSN No. : 2319 – 4197
www.iosrjournals.org

Design and Estimation of Power, Delay and Area for Parallel Adder in VLSI Circuits using 45nm Technology

*Sudhakar Alluri[1], M.Dasharatha[2], B.Rajendra Naik[3], N.S.S.Reddy[4]

[1]SUDHAKAR ALLURI is with the Electronics and Communication Engineering Department, Osmania University, Hyderabad, Telangana State, India,
[2]M.Dasharatha, is with the Electronics and Communication Engineering Department, Osmania University, Hyderabad, Telangana State, India,
[3]B.Rajendra Naik is with the Electronics and Communication Engineering Department, Osmania University, Hyderabad, Telangana State, India,
[4]N.S.S.REDDY is with the Electronics and Communication Engineering Department, VCE, Osmania University , Hyderabad, Telangana State, India,
Corresponding Author: Sudhakar Alluri

Abstract: *Adders are main component used in Digital signal processing (DSP) and are usually used in the digital integrated circuits. In Very-large-scale integration (VLSI) application delay, power and area are the necessary factors for any digital circuits. This paper presents 8 bit parallel adder mapped in Cadence Encounter(R) RTL Compiler Version v14.20-s013_1. By efficiently mapping in cadence tool, area, power and delay are decreased. The results of mapping are viewed using RTL synthesis tool in cadence VIRTUOSO at 45nm technology 0.7V. Based on digital signal processing (DSP) architectures, the code for low power is generated using 8 bit Carry Look ahead adder.*

Keywords: *High-Level Synthesis, Power Optimization, low Area, High Speed, low voltage, parallel adder, DSP, VLSI*

Date of Submission: 04-09-2017 Date of acceptance: 20-09-2017

I. Introduction

Integrated circuit (IC) by combining thousands of transistors into a single chip an (VLSI) Very-large-scale integration is the development of building. VLSI produced in the 1970s when complex semiconductor and communication technologies were being developed. The microprocessor is a VLSI device. Before the introduction of VLSI technology most ICs had a limited set of performs they could function. An electronic circuit might consist of a central processing unit (CPU), read-only memory(ROM), Random-access memory (RAM) and other glue logic. VLSI lets IC authors add all of these into one chip.

The system specifications are processor Intel (R) core (TM) i5-4570 CPU@3.20GHz.,3.20GHz.Installed memory (RAM) 4 GB (usable memory is 3.43GB) and system type: 32-bit operating system (OS). This paper are formed as follows Section II presents the literature review on 4 bit carry look ahead adder, ALU and DSP Section III presents the methodology for 4 bit carry look ahead adder and also discussed the low power analysis. Section IV shows the synthesis and simulation results and they are discussed clearly, finally the paper is concluded with Section V.

II. Literature Review

The issue show that Computation-in-memory (CIM) architecture in general and the Computation-in-memory (CIM) parallel adder in particularize have a high scalability. CIM parallel adder achieves at least two forms of magnitude improvement in energy and area in comparison with a multicore-based parallel adder. Moreover, due to a wide array of memristor design methods (such as Boolean logic), tradeoffs can be made between the area, delay and energy consumption[1]. We have designed and demonstrated two versions of an ERSFQ 8-bit parallel adder. ERSFQ is a resistor-free approach to dc biasing of Single Flux Quantum circuits that dissipates orders of magnitude less power than a traditional RSFQ logic while operating and has zero dissipation in inactive mode. The adders were designed for and fabricated with various fabrication processes, including HYPRES's 1.0-μm 4-layer 4.5 kA/cm2 process, HYPRES's 0.25-μm 4-layer 4.5 kA/cm2 process, HYPRES's 0.25-μm 6-layer 4.5 kA/cm2 planar zed process, and MIT Lincoln Lab's 0.25-μm4-layer 10 kA/cm2 process. These circuits serve as a good LSI fabrication process benchmark. We describe design and report on test results of all versions of the adder [2].Computer performance improvement has in the previous decades mostly been the result of CMOS downscaling [3]. In recent years, CMOS downscaling is reaching its end [4], [5] due to many challenges such as leakage power consumption [6], reliability [7], fabrication process and turnaround time [8], test complexity [9], cost for mask and design and yield [10].

As a result, increasing the clock frequency is no longer possible; performance gain has to be achieved through parallelism using multi core/many cores architectures. However, these architectures suffer from an inefficient programmability and high energy consumption [11] due to a gap between memory and processing unit speed, the so-called memory bottleneck [12], [13]. This is the core problem of the von Neumann store-program computer concept [14] used in today's computing systems, which leads to performance and energy inefficiency, especially for data intensive applications. Note that today supercomputers are used to deal with very limited number of data intensive (or compute intensive) applications; they are expensive, power hungry, and area inefficient [15],[16],[17]. Hence, there is a need for a novel architecture that significantly reduces the memory bottleneck, massively supports parallelism, and is energy efficient. To alleviate the memory bottleneck and provide practical and efficient solutions for data intensive applications, many architectural solutions have been proposed. They can be classified into three categories. First, processor-in-memory (PIM) was introduced as an architecture that consists of a host CPU, main memory, and a number of accelerators close to the main memory to prevent intensive communication with the CPU [18],[19],[20],[21]. Many implementations of this architecture have been proposed, e.g., EXECUBE [22], IRAM [23], Flex RAM [24], DIVA [25], and Gilgamesh [26]. However, the effectiveness of this architecture strongly depends on the technology to fabricate the accelerators and main memories, which is called merged-logic DRAM. Unfortunately, this merged technology still suffers from a high cost and low density [27], [28]. Second, near data architectures [29], [30] were proposed as a PIM architecture but using the emerging nonvolatile memory technology, either using traditional processor approach [31] or using novel neural computing approach [32].

III. Design Methodology

1. Parallel Adder

The speed limitation in the ripple adder arises from specifying *couti* as a function of the carry-out from the previous lower-order stage *couti* – 1. A considerable increase in speed can be realized by expressing the carry-out *couti* of any stage *i* as a function of the two operand bits, *ai* and *bi*, and the carry-in *cin*–1 to the low-order stage0 of the adder, where the adder is an *n*-bit adder *n*–1 *n*–2 . . . *n*1 *n*0. The Karnaugh map that represents the carry-out from stage*i* is shown in Figure 1 , which yields Equation 1.

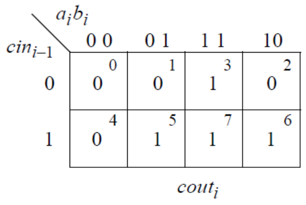


Figure 1 Karnaugh map for the carry-out of stage$_i$ of an n-bit adder

$$cout_i = a_i' b_i cin_{i-1} + a_i b_i' cin_{i-1} + a_i b_i cin_{i-1}' + a_i b_i cin_{i-1}$$

$$cout_i = a_i b_i + (a_i \oplus b_i) cin_{i-1} \qquad \text{.........................(1)}$$

Equation 1 states that a carry will be generated whenever *ai* = *bi* = 1, or when either *ai* = 1 or *bi* = 1 — but not both — with *ci* – 1 = 1. Note that if *ai* = *bi* = 1, then this represents a generate function, not a propagate function. Verilog requires a propagate function to be the exclusive-OR of *ai* and *bi*. A technique will now be presented that increases the speed of the carry propagation in a parallel adder. The carries entering all the bit positions of the adder can be generated simultaneously by a *carry lookahead* generator. This results in a constant addition time that is independent of the length of the adder. Two auxiliary functions are defined as follows:

$$Generate \rightarrow G_i = a_i' b_i$$

$$propagate \rightarrow P_i = a_i \oplus b_i$$

The carry *generate* function, *Gi*, reflects the condition where a carry is generated at the *i*th stage. The carry *propagate* function, *Pi*, is true when the *i*th stage will pass through (or propagate) the incoming carry *cini* – 1 to the next higher stage *i* + 1. Equation 1 can now be restated as Equation 2.

$$cout_i = a_i{}^{'}b_i + (a_i \oplus b_i)cin_{i-1}$$
$$cout_i = G_i + P_i cin_{i-1} \qquad \text{....................(2)}$$

Equation 2 indicates that the generate Gi and propagate Pi functions for any carry out couti can be obtained independently and in parallel when the operand inputs are applied to the n-bit adder. The equation can be applied recursively to obtain a set of carry-out equations in terms of the variables Gi, Pi, and cin–1 for a 4-bit adder, where cin–1 is the carry-in to the low-order stage 0 of the adder. The equations are shown in Equation 3.

$$cout_0 = G_0 + P_0 cin_{-1}$$
$$cout_1 = G_1 + P_1 cout_0$$
$$cout_1 = G_1 + P_1(G_0 + P_0 cin_{-1})$$
$$cout_1 = G_1 + P_1 G_0 + P_1 P_0 cin_{-1}$$
$$cout_2 = G_2 + P_2 cout_1$$
$$cout_2 = G_2 + P_2(G_1 + P_1 G_0 + P_1 P_0 cin_{-1})$$
$$cout_2 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 cin_{-1} \qquad \text{.......................... (3)}$$
$$cout_3 = G_3 + P_3 cout_2$$
$$cout_3 = G_3 + P_3(G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 cin_{-1})$$
$$cout_3 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 cin_{-1} \qquad \text{..........(3)}$$

Examples of the generate and propagate functions are shown in Figure 2 for two 8-bit operands in 2s complement notation.

$$\begin{array}{r|cccccccc|r}
A = & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & -51 \\
+)\ B = & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & +92 \\
 & G' & G & G' & G' & G & G & G' & G' & \\
 & P & P' & P' & P & P' & P' & P' & P & \\
 & & & & & & & & 0 & cin = 0 \\
\hline
cout = 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & +41
\end{array}$$

$$\begin{array}{r|cccccccc|r}
A = & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & +37 \\
+)\ B = & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & +79 \\
 & G' & G' & G' & G' & G' & G & G' & G & \\
 & P' & P & P & P' & P & P' & P & P' & \\
 & & & & & & & & 1 & cin = 1 \\
\hline
cout = 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & +117
\end{array}$$

Figure 2 Examples of generate and propagate functions.

Consider the expression for cout2 in Equation3 to further explain the generate and propagate functions to produce a carry-out. Each of the product terms shown below will produce a carry-out of 1 for cout2.

$$
\begin{array}{lcccccccc}
cout_2 = & G_2 & + & P_2\,G_1 & + & P_2P_1G_0 & + & P_2P_1P_0 & \\
 & 1 & & 0\ 1 & & 0\ 1\ 1 & & 1\ 0\ 0 & \\
 & 1 & & 1\ 1 & & 1\ 0\ 1 & & 0\ 1\ 1 & \\
 & & & & & & & \quad\ \ 1 \leftarrow cin_{-1} & \\
1 \leftarrow & 0 & 1 \leftarrow & 0\ 0 & 1 \leftarrow & 0\ 0\ 0 & 1 \leftarrow & 0\ 0\ 0 &
\end{array}
$$

It can be seen from Equation 3 that each carry is now an expression consisting of only three gate delays: one delay each for the generate and propagate functions, one delay to AND the generate and propagate functions, and one delay to OR all of the product terms. If a high-speed full adder is used in the implementation, then the sum bits can be generated with only two gate delays, providing a maximum of only five delays for an add operation. This technique provides an extremely fast addition of two *n*-bit operands. Equation 3 can be restated more compactly as shown in Equation 4.

$$Cout_i = G_i + \sum_{j=0}^{i-1}\left(\prod_{k=j+1}^{i} P_k\right)G_j + \prod_{k=0}^{i} P_k\, cin_{-1} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (4)$$

Group generate and propagate As *n* becomes large, the number of inputs to the high-order gates also becomes large, which may be a problem for some technologies. The problem can be alleviated to some degree by partitioning the adder stages into 4-bit groups. Additional auxiliary functions can then be defined for *group generate* and *group propagate*, as shown in Equation 5 for group *j*, which consists of individual adder stages *i* + 3 through *i*. In this method, each group of four adders is considered as a unit with its individual group carry sent to the next higher-order group.

$$Group \cdot generate \rightarrow GG_j = G_{i+3} + P_{i+3}G_{i+2} + P_{i+3}P_{i+3}G_{i+2} + P_{i+3}P_{i+2}P_{i+1}G_i$$

$$Group \cdot propagate \rightarrow GP_j = P_{i+3}P_{i+2}P_{i+1}G_i \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (5)$$

The group generate GGj signifies a carry that is generated out of the high-order (i + 3) bit position that originated from within the group. The group propagate GPj indicates that a carry was propagated through the group. The group carry can now be written in terms of the group generate and group propagate functions, as shown in Equation 6. The term GCj – 1 is the carry-in to the group from the previous lowerorder group. If groupj is the low-order group, then GCj – 1 = cin–1.

$$Group \cdot Carry \rightarrow GP_j = P_{i+3}P_{i+2}P_{i+1}G_i \ldots\ldots\ldots\ldots (6)$$

Section generate and propagate If the fan-in limitation is still a problem for very large operands, then the group generate and group propagate concept can be extended to partition four groups into one section. For a 64-bit adder, there would be four sections with four groups per section, with four full adders per group. Two additional auxiliary functions can now be defined as *section generate* and *section propagate* for section *k* , as shown in Equation7 .

$$Section \cdot generate \rightarrow SG_k = GG_{j+3} + GP_{j+3}GG_{j+2} + GP_{j+3}GP_{j+2}GG_{j+1} + GP_{j+3}GP_{j+2}GP_{j+1}GG_j$$

$$Section \cdot propagate \rightarrow SP_k = GP_{j+3}GP_{j+2}GP_{j+1}GG_j \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (7)$$

The section generate SGk signifies a carry that is generated out of the high-order (j + 3) position that originated from within the section. The section propagate SPk indicates that a carry was propagated through the section. The section carry can now be written in terms of the section generate and section propagate functions, as shown in Equation 8. The term SCk – 1 is the carry-in to the section from the previous lower-order section. If sectionk is the low-order section, then SCk – 1 = cin–1.

$$Section \cdot Carry \rightarrow SC_k = SG_k + SP_k SC_{K-1} \ldots\ldots\ldots\ldots(8)$$

The carry-out of the high-order section SCk + 3 is also the carry-out of the adder and can be written as coutn – 1. This section has presented a method to increase the speed of addition by partitioning the adder into sections and groups and developing carry lookahead logic within individual groups and within individual sections. Figure 3 shows a block diagram of a 64-bit adder consisting of four sections, with four groups per section, and four full adders per group.

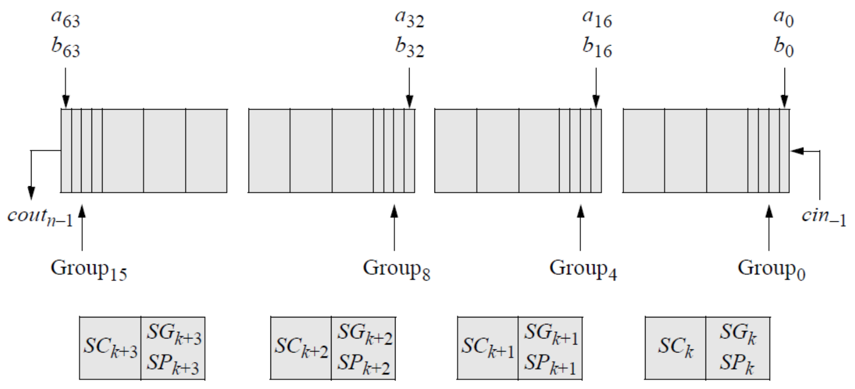


Figure 3 Block diagram of a 64-bit parallel adder.

An 8-bit parallel adder will be designed using dataflow modeling..

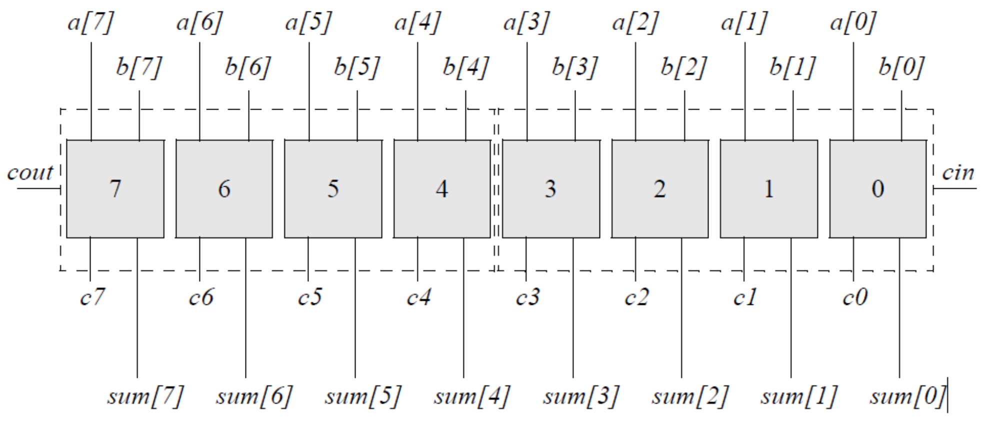


Figure 4 An 8-bit parallel adder.

The block diagram of the adder is shown in Figure 4. The adder has two groups of four bits per group. The high-order group has operands a[7:4] and b[7:4] that produce a sum of sum[7:4]; the low-order group has operands a[3:0] and b[3:0] that produce a sum of sum[3:0]. The carry-in is cin; the carry-out is cout, which is the carry-out of bit 7.

2. Low Power analysis

2.1 Dynamic Power

The total power of System on Chip design consists of dynamic power and static power [12].

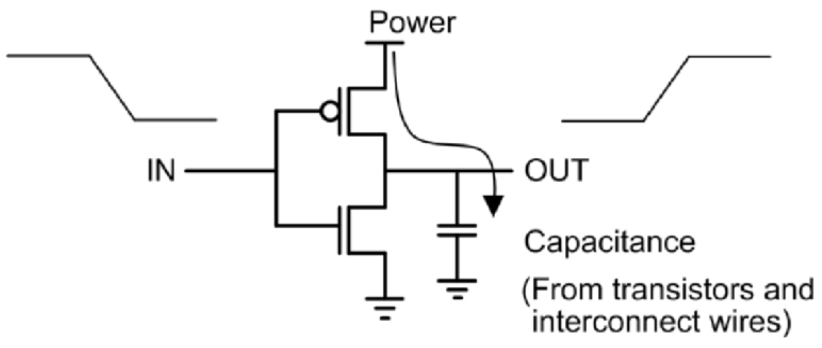


Figure 5: Dynamic Power [12].

Dynamic power is the power consumed when the material is in active mode. Whenever the device is in active mode the power dissipated in the device is called as Static power, but the signal values are unchanged.

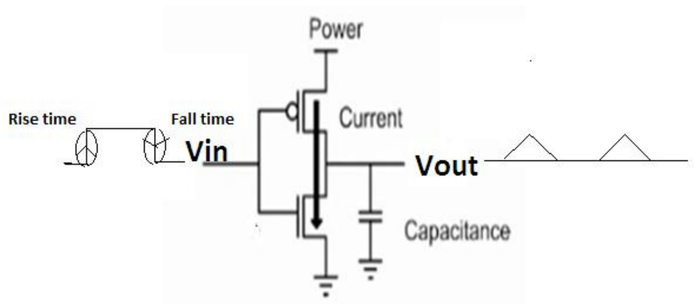


Figure 6: Short Circuit (Crow Bar) Power

$$P_{SC} = I_{SC} * v \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{(9)}$$

2.2 Static Power

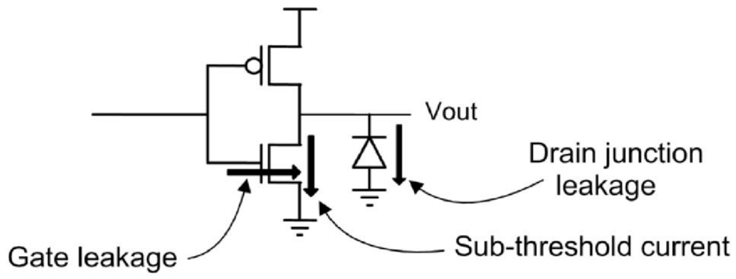


Figure 7: Leakage Power

Mainly the leakage currents are of four types. (Figure 7)

• Sub-threshold Leakage (ISUB): it is the current flows from drain to source which operates in inverse region.

• Gate Leakage (IGATE): it is the current flows from gate through oxide to the substrate .

• Gate Induced Drain Leakage (IGIDL): it is the current flows from drain to substrate caused by high voltage effect in MOSFET due to VDG.

• Reverse Bias Junction Leakage (IREV): it is the current caused because of minor drift and creation of electron hole pairs in the immobile region

$$P_{Static} = I_{Static} * v_{dd} \quad (10)$$

$$P_{Dynamic} = \alpha * c_L * v_{dd}^2 * f \quad (11)$$

$$P_{Shortcircuit} = I_{SC} * v \quad (12)$$

$$P_{Leakage} = V_{dd} * (I_S + I_G + I_D) \quad (13)$$

$$P_{Total} = P_{Dynamic} + P_{Leakage} \quad (14)$$

$$P_{Total} = (\alpha * c_l * v_{dd}^2 * f) + V_{dd} * (I_S + I_G + I_D) \quad (15)$$

Where α is a switching activity factor, c_L is a load capacitance, Vdd is a voltage (drain to drain), f is a frequency, IS (source current), IG (Gate current) and ID (Drain current) .

IV. Synthesis And Simulation Results

This paper presents 8 bit parallel adder is mapped in Cadence Encounter(R) RTL Compiler Version v14.20-s013_1.By efficiently mapping in cadence tool, area, power and delay are decreased. The results of mapping are viewed using RTL synthesis tool in cadence VIRTUOSO at 45nm technology and 0.7V. Based on DSP architectures, the code for low power is generated using 8bit parallel adder.

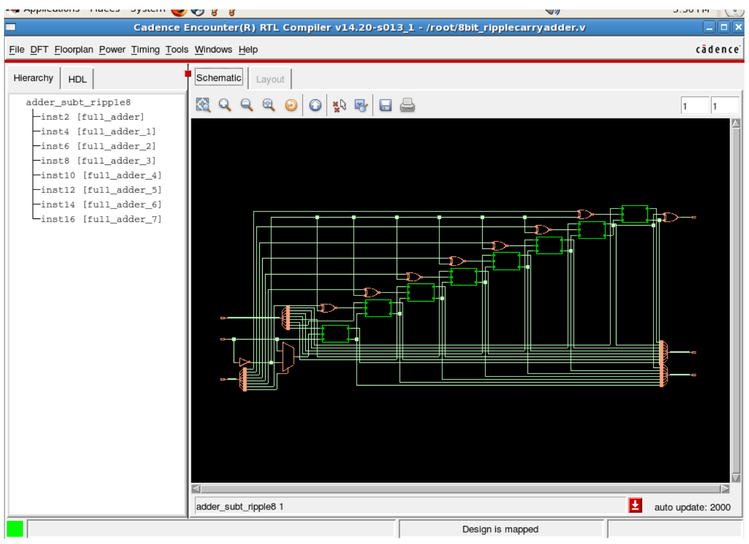
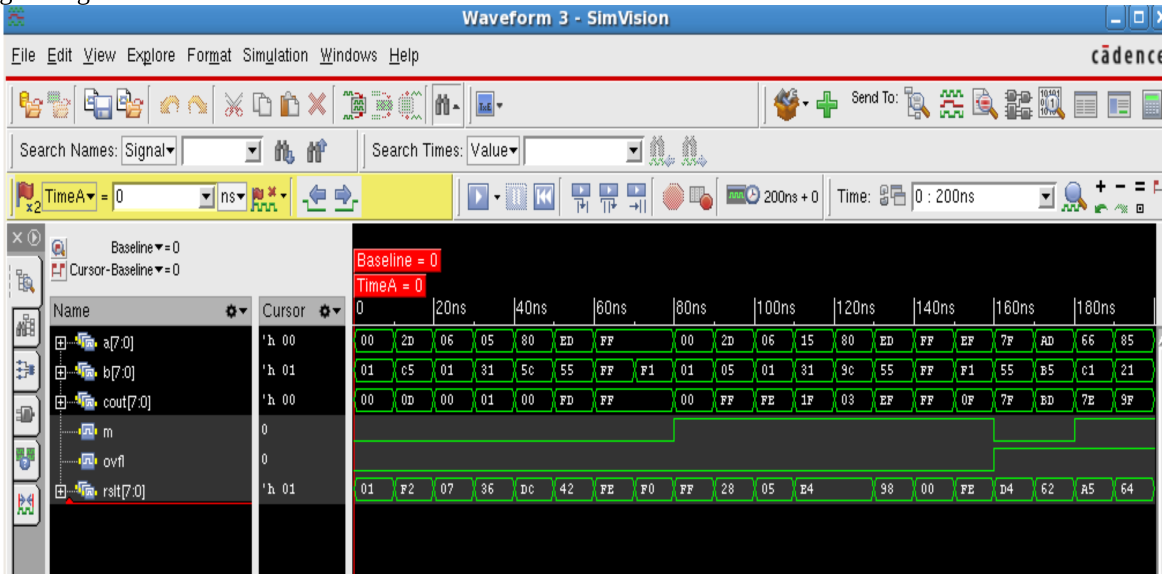


Figure 8 8bit parallel adder at 45nm technology

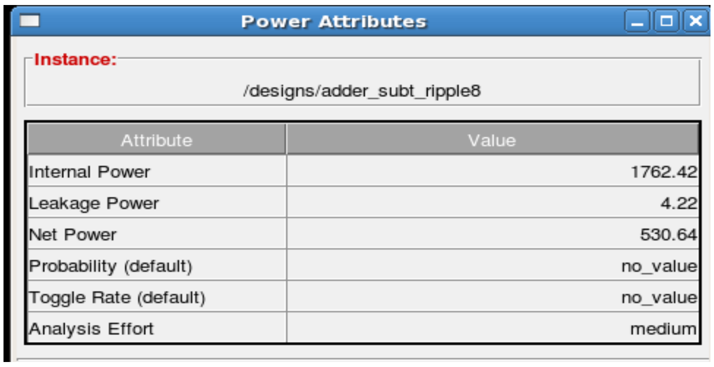
The proposed 8 bit parallel adder code is mapped in the cadence tool for simulation result of schematic diagram figure 8.



```
Name        Cursor   0   20ns  40ns  60ns  80ns  100ns  120ns  140ns  160ns  180ns
a[7:0]      'h 00    00 2D 06 05 80 ED FF    00 2D 06 15 80 ED FF EF 7F AD 66 85
b[7:0]      'h 01    01 c5 01 31 5c 55 FF F1 01 05 01 31 9c 55 FF F1 55 B5 c1 21
cout[7:0]   'h 00    00 0D 00 01 00 FD FF    00 FF FE 1F 03 EF FF 0F 7F BD 7E 9F
m           0
ovfl        0
rslt[7:0]   'h 01    01 F2 07 36 Dc 42 FE F0 FF 28 05 E4    98 00 FE D4 62 A5 64
```

Figure 9 8 bit parallel adder of Simulation Result at 45nm technology

In Figure 9, the design 8bit parallel adder simulation is observed after mapping into the cadence tool at 45nm technology. The output of the 8bit parallel adder waveform has the frequency of 0.2 MHz


Power Attributes

Instance:

/designs/adder_subt_ripple8

Attribute	Value
Internal Power	1762.42
Leakage Power	4.22
Net Power	530.64
Probability (default)	no_value
Toggle Rate (default)	no_value
Analysis Effort	medium

Figure 10 8 bit parallel adder of power attributes.

We proposed mapping style in cadence tool 45nm using 8 bit parallel adder by observing the figure 10 8 bit parallel adder of power attributes.

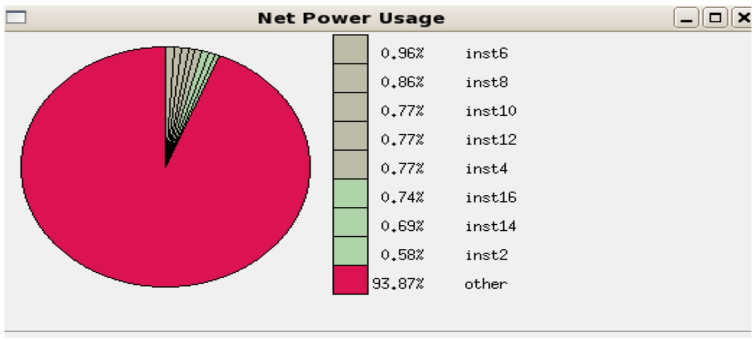


Figure 118 bit parallel adder of net power usages

We proposed mapping style in cadence tool 45nm using 8 bit parallel adder. By observing the figure 11, 8 bit parallel adder of net power usage.

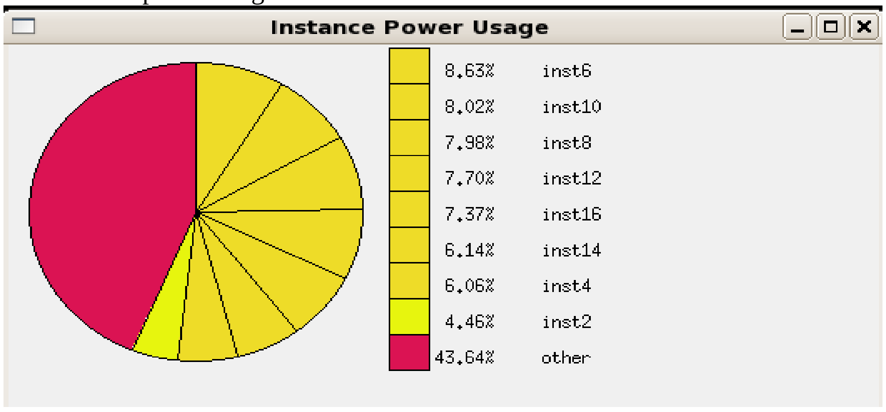


Figure 12 8 bit Parallel adder of instance power

We proposed mapping style in cadence tool 45nm using 8 bit parallel adder. By observing the figure 12, 8 bit parallel adder of instance power.

Table 1 Area of 8 bit parallel adder using cadence tool at 45nm technology

```
==============================================================
  Instance   Cells  Cell Area(nm²)  Net Area(nm²)  Total Area(nm²) Wireload
---------------------------------------------------------------------------
adder_subt_ripple8     18        64          0            64    <none> (D)
  inst8                 1         5          0             5    <none> (D)
  inst6                 1         5          0             5    <none> (D)
  inst4                 1         5          0             5    <none> (D)
  inst2                 1         5          0             5    <none> (D)
  inst16                1         5          0             5    <none> (D)
  inst14                1         5          0             5    <none> (D)
  inst12                1         5          0             5    <none> (D)
  inst10                1         5          0             5    <none> (D)
```

We proposed mapping style in cadence tool 45nm using 8 bit parallel adder. By observing the table 1, area of 8 bit parallel adder.

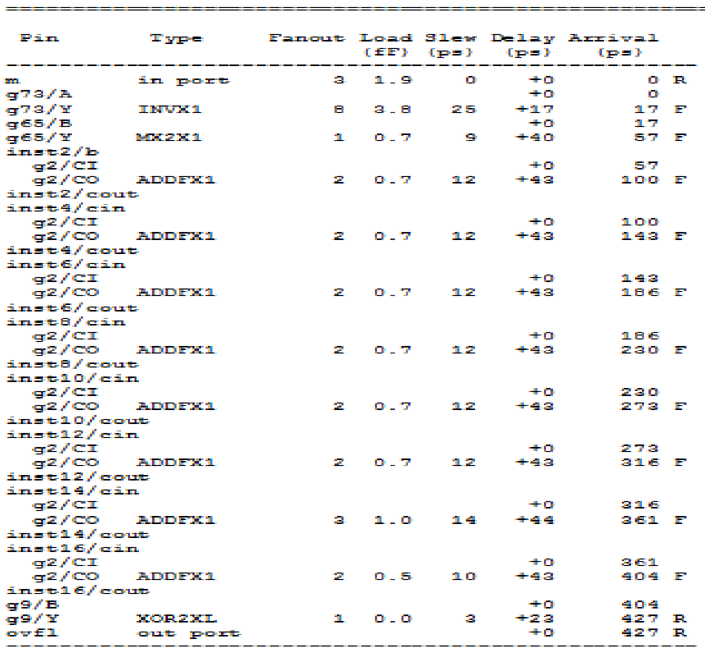
Table 2 Power Dissipation of 8 bit parallel adder using cadence tool 45nm Technology

```
=================================================================

                                  Leakage   Dynamic    Total
      Instance             Cells Power(nW) Power(nW) Power(nW)
-----------------------------------------------------------------
adder_subt_ripple8          18     4.216  2293.053  2297.269
  inst16                     1     0.326   185.896   186.223
  inst12                     1     0.326   194.243   194.569
  inst2                      1     0.325   115.308   115.633
  inst10                     1     0.324   201.614   201.938
  inst14                     1     0.324   156.532   156.856
  inst8                      1     0.323   202.949   203.272
  inst6                      1     0.322   219.989   220.311
  inst4                      1     0.321   156.526   156.847
```

We proposed mapping style in cadence tool 45nm using 8 bit parallel adder. By observing the table 2, power dissipation of 8 bit parallel adder.

Table 3 Delay of 8 bit parallel adder using cadence tool 45nm technology



```
===========================================================================

  Pin          Type        Fanout Load Slew Delay Arrival
                                  (fF) (ps)  (ps)   (ps)
-----------------------------------------------------------------------
m              in port        3   1.9    0    +0       0 R
g73/A                                         +0       0
g73/Y          INVX1          8   3.8   25   +17      17 F
g65/B                                         +0      17
g65/Y          MX2X1          1   0.7    9   +40      57 F
inst2/b
   g2/CI                                      +0      57
   g2/CO       ADDFX1         2   0.7   12   +43     100 F
inst2/cout
inst4/cin
   g2/CI                                      +0     100
   g2/CO       ADDFX1         2   0.7   12   +43     143 F
inst4/cout
inst6/cin
   g2/CI                                      +0     143
   g2/CO       ADDFX1         2   0.7   12   +43     186 F
inst6/cout
inst8/cin
   g2/CI                                      +0     186
   g2/CO       ADDFX1         2   0.7   12   +43     230 F
inst8/cout
inst10/cin
   g2/CI                                      +0     230
   g2/CO       ADDFX1         2   0.7   12   +43     273 F
inst10/cout
inst12/cin
   g2/CI                                      +0     273
   g2/CO       ADDFX1         2   0.7   12   +43     316 F
inst12/cout
inst14/cin
   g2/CI                                      +0     316
   g2/CO       ADDFX1         3   1.0   14   +44     361 F
inst14/cout
inst16/cin
   g2/CI                                      +0     361
   g2/CO       ADDFX1         2   0.5   10   +43     404 F
inst16/cout
g9/B                                          +0     404
g9/Y           XOR2XL         1   0.0    3   +23     427 R
ovfl           out port                       +0     427 R
-----------------------------------------------------------------------
```

We proposed mapping style in cadence tool 45nm using 8 bit parallel adder. By observing the table 3, delays of 8 bit parallel adder.

Table 4:Comparison of Power, Delay and Area of 8 Bit Parallel adder Using Cadence 45nm

S.NO	Leakage Power(nW)	Dynamic Power(nW)	Total Power(nW)	Delay(ps)	Area(nm^2)
8bit Parallel adder	4.216	2293.053	2297.269	44	64

We proposed mapping style in cadence tool 45nm using 8 bit parallel adder. By observing the table 4, Comparison of Power, Delay and Area of 8 Bit Parallel adder Using Cadence 45nm.

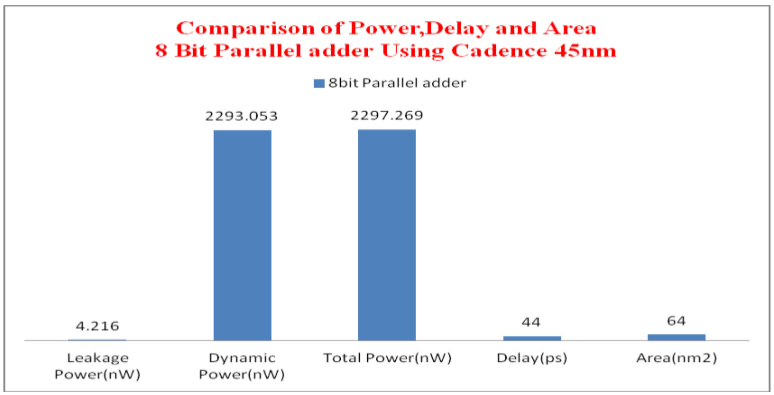


Figure 13 Comparison of Power, Delay and Area8 Bit Parallel adder Using Cadence 45nm Technology

We proposed mapping style in cadence tool 45nm using 8 bit parallel adder. By observing the figure 13, Comparison of Power, Delay and Area of 8 Bit Parallel adder Using Cadence 45nm Technology.

V. Conclusion

In this paper, we proposed mapping style in cadence tool using 8 Bit Parallel adder. Table 1 gives 8 Bit Parallel adder of area, table 2 represents 8 Bit Parallel adder of power dissipation, table 3 represents 8 Bit Parallel adder of delay, table 4 gives Comparison of Power, Delay and Area of 8 Bit Parallel adder Using Cadence 45nm Technology and power given is 0.7V. With the help of DSP architectures, the code is generated which results in low power.

Acknowledgements

We acknowledge the University Grants Commission (UGC) for its economic support in the form of Rajiv Gandhi National Fellowship (RGNF) for the year 2015-16. We thank all the faculties in the Department of (ECE) Electronics and Communications Engineering and the lab staff for their support.

References

[1] Hoang Anh Du Nguyen, Lei Xie,Mottaqiallah Taouil, Razvan Nane,Said Hamdioui, and Koen Bertels,"On the Implementation of Computation-in-Memory Parallel Adder",IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) YSTEMS,1063-8210 © 2017 IEEE, www.ieee.org.

[2] Alex F. Kirichenko, Igor V. Vernik,John A. Vivalda, Rick T. Hunt, and Daniel T. Yohannes,"ERSFQ 8-Bit Parallel Adders as a Process Benchmark",IEEE TRANSACTIONS ON APPLIED SUPERCONDUCTIVITY, VOL. 25, NO. 3, JUNE 2015,http://www.ieee.org.

[3] H. Esmaeilzadeh, E. Blem, R. S. Amant, K. Sankaralingam, and D. Burger, “Dark silicon and the end of multicore scaling,” in Proc. 38th Annu. Int. Symp. Comput. Archit. (ISCA), Jun. 2011, pp. 365–376.

[4] S. Kaxiras, Architecture at the End of Moore (Advances in Atom and Single Molecule Machines). Berlin, Germany: Springer-Verlag, 2013.

[5] Y. Taur, “CMOS design near the limit of scaling,” IBM J. Res. Develop., vol. 46, nos. 2–3, pp. 213–222, Mar. 2002.

[6] L. Rocks and R. P. Runyon, Chapter 20: The Energy Crisis (The Frontiers Collection). Berlin, Germany: Springer-Verlag, 2012.

[7] J. W. McPherson, “Reliability trends with advanced CMOS scaling and the implications for design,” in Proc. IEEE Custom Integr. Circuits Conf. (CICC), Sep. 2007, pp. 405–412.

[8] S. Borkar, “Design perspectives on 22 nm CMOS and beyond,” in Proc. Design Autom. Conf. (DAC), Jul. 2009, pp. 93–94.

[9] G. Gielen et al., “Emerging yield and reliability challenges in nanometer CMOS technologies,” in Proc. Conf. Design, Autom. Test Eur. (DATE), Mar. 2008, pp. 1322–1327.

[10] J. W. Janneck, “Computing in the age of parallelism: Challenges and opportunities,” Computer Science Department, Lund University, in Multicore Day, 23 Sep 2013.

[11] K. Lahiri and A. Raghunathan, “Power analysis of system-level onchip communication architectures,” in Proc. Int. Conf. Hardw./Softw. Codesign Syst. Synth. (CODES+ISSS), Sep. 2004, pp. 236–241.

[12] S. A. McKee, “Reflections on the memory wall,” in Proc. Comput. Front., 2004, pp. 1–6.

[13] A. W. Burks, H. H. Goldstine, and J. von Neumann, “Preliminary discussion of the logical design of an electronic computing instrument (1946),” in Perspectives on the Computer Revolution,Z. W. Pylyshyn and L. J. Bannon, Eds. Norwood, NJ, USA: Ablex Publishing Corp., 1989, pp. 39–48. [Online]. Available: http://dl.acm.org/citation.cfm?id=98326.98337

[14] W. Xue et al., “Enabling and scaling a global shallow-water atmospheric model on Tianhe-2,” in Proc. IEEE 28th Int. Parallel Distrib. Process. Symp., May 2014, pp. 745–754.

[15] X. Zhang et al., “Optimizing and scaling HPCG on tianhe-2: Early experience,” in Proc. 14th Int. Conf. Algorithms Archit. Parallel Process. (ICA3PP), Dalian, China, Aug. 2014, pp. 28–41.

[16] G. Bell and J. Gray, “What’s next in high-performance computing?” Commun. ACM, vol. 45, no. 2, pp. 91–95, 2002.

[17] D. G. Elliott, M. Stumm, W. M. Snelgrove, C. Cojocaru, and R. Mckenzie, “Computational RAM: Implementing processors in memory,” IEEE Des. Test Comput., vol. 16, no. 1, pp. 32–41, Jan./Mar. 1999.

[18] J. Draper et al., “A prototype processing-in-memory (PIM) chip for the data-intensive architecture (DIVA) system,” J. VLSI Signal Process. Syst. signal, Image Video Technol., vol. 40, no. 1, pp. 73–84, 2005.

[19] B. J. Jasionowski, M. K. Lay, and M. Margala, “A processor-in-memory architecture for multimedia compression,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 15, no. 4, pp. 478–483, Apr. 2007.

[20] E. Upchurch, T. Sterling, and J. B. Brockman, “Analysis and modeling of advanced PIM architecture design tradeoffs,” in Proc. Innov. Archit. Future Generat. High-Perform. Process. Syst., Jul. 2003, pp. 66–75.

[21] P. M. Kogge, “EXECUBE—A new architecture for scaleable MPPs,” in Proc. Int. Conf. Parallel Process. (ICPP), vol. 1. Aug. 1994, pp. 77–84.

[22] D. Patterson et al., “A case for intelligent RAM,” IEEE Micro, vol. 17, no. 2, pp. 34–44, Mar./Apr. 1997.

[23] J. Torrellas, “FlexRAM: Toward an advanced intelligent memory system: A retrospective paper,” in Proc. IEEE Int. Conf. Comput. Design (ICCD), Sep./Oct. 2012, pp. 3–4.

[24] J. Draper et al., “The architecture of the DIVA processing-in-memory chip,” in Proc. Int. Conf. Supercomput., 2002, pp. 1–12.

[25] T. L. Sterling and H. P. Zima, “Gilgamesh: A multithreaded processorin- memory architecture for petaflops computing,” in Proc. ACM/IEEE Conf. Supercomput., Nov. 2002, p. 48.

[26] D. Keitel-Schulz and N. Wehn, “Issues in embedded DRAM development and applications,” in Proc. 11th Int. Symp. Syst. Synth., Dec. 1998, pp. 23–28.

[27] D. Keitel-Schulz and N. Wehn, “Embedded DRAM development: Technology, physical design, and application issues,” IEEE Design Test Comput., vol. 18, no. 3, pp. 7–15, May 2001.

[28] A. Farmahini-Farahani, J. H. Ahn, K. Morrow, and N. S. Kim, “NDA: Near-DRAM acceleration architecture leveraging commodity DRAM devices and standard memory modules,” in Proc. IEEE 21st Int. Symp. High Perform. Comput. Archit. (HPCA), Feb. 2015, pp. 283–295.

[29] S. H. Pugsley et al., “NDC: Analyzing the impact of 3D-stacked memory+logic devices on MapReduce workloads,” in Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw. (ISPASS), Mar. 2014, pp. 190–200.

[30] R. Balasubramonian et al., “Near-data processing: Insights from a MICRO-46 workshop,” IEEE Micro, vol. 34, no. 4, pp. 36–42, Jul./Aug. 2014.

[31] A. Shafiee et al., “ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars,” in Proc. ISCA, Jun. 2016, pp. 14–26.

IOSR Journal of VLSI and Signal Processing (IOSR-JVSP) is UGC approved Journal with Sl. No. 5081, Journal no. 49363.

Sudhakar Alluri. “Design and Estimation of Power, Delay and Area for Parallel Adder in VLSI Circuits using 45nm Technology.” IOSR Journal of VLSI and Signal Processing (IOSR-JVSP) , vol. 7, no. 4, 2017, pp. 56–65.