JOURNAL OF MECHANICS OF CONTINUA AND MATHEMATICAL SCIENCES
www.journalimcms.org

ISSN (Online) : 2454 -7190 Vol.-14, No.-3, May-June (2019) pp 112-123 ISSN (Print) 0973-8975

An Implementation of Area Optimized Low Power MAC

P. Ashok Babu
Professor, Department of Electronics and Communication Engineering,
Institute of Aeronautical Engineering, Hyderabad, India.

E-mail: ashokbabup2@gmail.com

https://doi.org/10.26782/jmcms.2019.06.00009

Abstract

The objective of the paper is to develop an Area optimized Low power digital circuit for MAC (Multiply and Accumulate) operation. We developed implementations of the MAC to avoid using multipliers and prefer to use the combinational circuits like multiplexers. We analyze all the MAC digital circuits to find out the best digital circuit which consumes minimum area and power. MAC is basic building block of many Digital Signal Processing Applications like Noise Cancellation Circuits, Speech Processing, Image Processing, Video Processing, Artificial Neural Networks etc. We also give some suggestions on the system level solutions based on the MAC. The digital circuit which is developed by us will be compatible to FPGAs, as it is developed by the industry standard Synthesis tool i.e. Synopsys Synlipy pro synthesis tool. The MAC which we are developing can be placed in the FPGA Fabric and it can be interfaced to any processors like Cortex M3, Cortex M0, 805 1etc. The overall throughput decreases due high latency and increase in the processing time. So, all the MAC operations must be performed in the hardware by the MAC block developed by us as it is low power, low area and fast hardware.

MAC, Digital Signal Processing,

Keywords : Digital Signl Processing, MAC

I. Introduction to MAC

In the modern world, devices are made use of RISC processor and Digital Signal Processing (DSP). [I] In DSP, one of the most intricate operation is the multiply accumulate operation. For high performance MAC unit is the basic element in applications like in convolution, inner products, and filters. DSP uses non-linear functions such as Discrete Cosine Transform (DCT) and Discrete Wavelet Transform (DWT). As MAC unit runs independent of the CPU, can process data separately. The basic concern of MAC design is to achieve increase in its speed. As speed and output

rate are always the main concerns of DSP systems. MAC unit performs many DSP applications involving multiplications and/or additions. The rate of the processor mainly depends on the speed of the MAC unit hence complexity of MAC unit design and power consumption is the major concern for real time processing applications. MAC operation will be those essential operation utilized within advanced plan about DSP applications, media processing, image transforming and different requisitions which oblige tedium multiplications What's more additions for example, such that fast Fourier transform (FFT), Finite impulse response (FIR). Will move forward the speed; MAC unit relies on the two primary sub-units. Those initial will be those duplication methodology and the other one is with amass. The essential capacity from claiming recommended MAC unit will be to increase those multiplier and more multiplicand Furthermore include the required item for the bring about shortages put away on a gatherer.

II. Construction of MAC

The MAC operation will be the principle computational part on Digital signal processing (DSP) architectures. [II] those MAC unit will be viewed as Concerning illustration a standout among those essential operations over DSP also it gets a essential part done Application-Specific-Integrated-Circuits (ASIC). The MAC unit determines the pace of the generally system; it generally lies in the discriminating way. Creating a secondary speed MAC will be essential for constant DSP provisions. Moreover, with the ever-increasing interest to WSN, a MAC unit with low force utilization might without a doubt lead the market. Huge numbers analysts need endeavored to plan MAC architectures with more computational execution also low control utilization. So as with enhance those speed of the MAC unit, there would two significant bottlenecks that requirement with make viewed as. The initially you quit offering on that one may be the incomplete items decrease organize that is utilized within the duplication piece and the second one is those gatherer. Both about these phases oblige s were as from claiming expansive operands that include long ways to convey proliferation.

Multiply-Accumulate is a common operation that computes the product of two numbers and adds that product to an accumulator. The multiplier A and multiplicand B are assumed to have N bits each and the addend Z has (2N+1) bits.

$$Z \leftarrow (AxB) + Z \, .$$

The MAC Unit is made up of a multiplier and an accumulator as shown in Fig. 1.

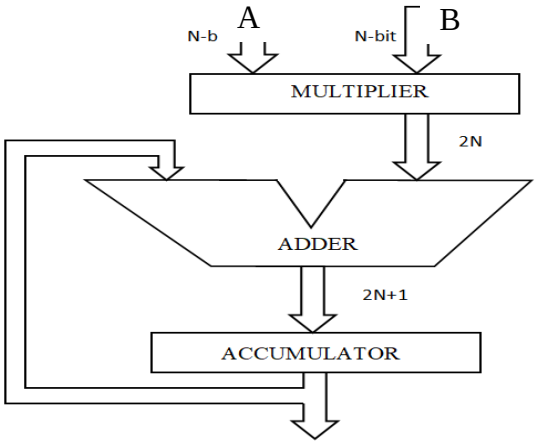

Fig .1 Block diagram of basic MAC Unit

That multiplier may be isolated under the fractional items generator, summation network, and last snake. Those summation organize speaks to those center of the MAC unit; it diminishes those number of incomplete items under two operands speaking to a whole of cash Also a convey. The summation organize involves in oak of the region Also expends the greater part of the circlet range Also delay. A few calculations and architectures would recommend in an endeavor with streamline the execution of the summation organize. Those last snake is then used to produce those duplication bring about shortages out for these two operands. The gatherer is used to perform a twofold precision expansion operation between those gathered operand Also duplication consequences. Because of the substantial operand size, that gatherer obliged a extensive snake.

.

III. Existing Methods of Multiplying and Accumulating (Adding) and their Analysis

There are many kinds of multipliers and adders which can be used to perform the MAC operation. Some of those methods are listed below:

Serial multiplier
Array multiplier
Booth multiplier
Wallace Tree multiplier
Twin Pipe Serial-Parallel multiplier

The numerical below are formulated in the form of a table by referring to previous analysis carried out by different researchers [3].

Table 1. Area, Delay, and Power Dissipation of Multipliers

Multiplier Topology	Area-Total CLB's	Maximum Delay(ns)	Power (mW)
Array Multiplier	1165	187.87	16.6506
Booth Multiplier	1292	139.41	23.136
Wallace Tree Multiplier	1659	101.141	30.95
Modified Booth-Wallace Tree Multiplier	1239	101.43	30.862
Twin-Pipe Serial Parallel Multiplier	133	22.58	2.089

We have many adders which can be employed in our implementation are listed below:

Ripple Carry Adder
Carry Save Adder
Carry Look-Ahead Adder
Carry Increment adder
Carry Skip Adder
Carry Bypass Adder
Carry Select Adder

The analysis of all these mentioned adders are not carried out as the part of this paper but these are referred from previous research papers [IV-V-VI].
These research papers have detailed analysis in them along with the comparison of various adders.

Table 2. Area, Delay, and Power Dissipation of Adders

Adder Topology	**PMOS**	**NMOS**	**Total**	**Power Dissipation (mW)**	**Area (μm²)**	**Delay (ns)**
RCA	144	144	288	0.206	2214	4.208
CSaA	288	288	576	1.082	5904	2.924
CLA	136	136	272	0.312	2160	3.1
CIA	171	171	342	0.261	2793	2.880
CSkA	194	194	388	0.603	3486	3.022
CByA	186	186	372	0.459	3116	3.01
CSelA	300	300	600	1.109	6201	2.75

IV. Proposed Methodology

Though there are many ways by means of which MAC unit has been implemented, out of which some are by using specific multipliers and adders.We have many good multipliers such as Wallace Tree Multiplier, Booth Multiplier, etc. Also we have many adders like Ripple carry Adder, Carry Look Ahead Adder, etc., but inspite of that we observed little flaws in their operations.

There is specifically no issue with the Adders, but the issue lies with the Multipliers. The multiplication operation in those cases are carried out by using asterisk "*", which is nothing but called a star operator used for multiplication.

Upon analysis we came across a fact that the asterisk "*" i.e., a star operator actually consumes much CPU resources like space (i.e., area on chip), time, efficiency. Apart from that it also consumes considerate amount of power.

In the streams like Digital Signal Processing (DSP), there are at every step almost many multiplication and addition operations that are carried out. The efficiency of the process will drastically get reduced once the system is loaded with huge number of inputs which requires huge computations.

Hence it is required to counter this big flaw. Hence we came up with an idea of making the Multiplier Less Algorithm (MLA) for carrying out the multiplication operation. As we know that any program will work extremely fast if it is incorporated with the processor level commands. We made the use of this fact and developed a unit MAC, which is actually multiplier less unit. The basic operation in most of the Digital Signal processing applications is MAC. The proposed method is on design and development of an area optimized MAC (Multiply and Accumulate Block).

IV.i. Proposed method involves :

Developing the traditional multiplier-based MAC, synthesis and area calculation. Developing a windows-based application to generate the multiplier less MAC equations. Developing the various multipliers less MAC blocks. (Targeting minimum 2 to 3 models). Synthesis of all the multiplier less MAC blocks. Summarizing the area utilization of all the multiplier less MAC blocks and comparing them with the traditional multiplier-based MAC.

Explain the importance of the proposed method in any of the real time applications like Video processing systems.

V. Implementation

In this section we present the algorithm of our implemented MAC serially for the better understanding.

In our paper we are dealing with a MAC unit which takes 4-bit coefficient and 8-bit variable as the input. This means that it can perform this multiplication between 8-bit

variable and 4-bit coefficient only. But this can also be increased efficiently to 8X8 or 16X16 or even 32X32!
Since this is a stating stage we implemented it successfully using 4-bit coefficient and 8-bit variable.

V.i. Proposed Algorithm:

Step1: Since we know that for now we are dealing with 4-bit coefficient and 8-bit variable, hence the width of the multiplication product has to be determined after multiplication.
Whenever 2 numbers are multiplied having widths as "a" and "b" then the resulting bit width will be "a+b".

Hence in our module the output width will be 12-bit.

Step2: After getting the initial basic requirement we moved on to the actual logic implementation. Here we have implemented the multiplier using the bitwise operators. It is known that the bitwise operators operate at the processor level and hence are way faster.
Now or the adder operation: We have already seen many kinds of adders in the previous section, but we are not employing any of such adders, because the adder will not have much significant effect on the area, power consumption as well as the speed of operation. Moreover we wanted to keep this project as simple as possible, hence we did not incorporate the use of such adders, and rather we have used a simple adder.
Step3: Now as logic which we are going to implement is known, we proceeded with coding. The coding has been done using the HFL language using the software "Mentor Graphics Model Sim Simulation Tool" and later is synthesized using "Synopsys Simplify Pro Synthesis Tool". These are the two software's which are used in our project and these are the licensed and paid software's. THESE ARE NOT OPEN SOURCE SOFTWARES.
Step4: Now having implemented the logic we synthesized and simulated it. The results what we obtained were verified theoretically and it came out to be genuine. The simulation and synthesis results are attached as in image in the results section.
Step5: The project module was implemented successfully!

VI. Results and Discussion

The code which was written was simulated and synthesized using the software Mentor Graphics Model Sim Synthesis Tool and Synopsys Synlipy Pro Synthesis Tool respectively.

VI.i. Implemented Multiplier Simulation and Synthesis Results

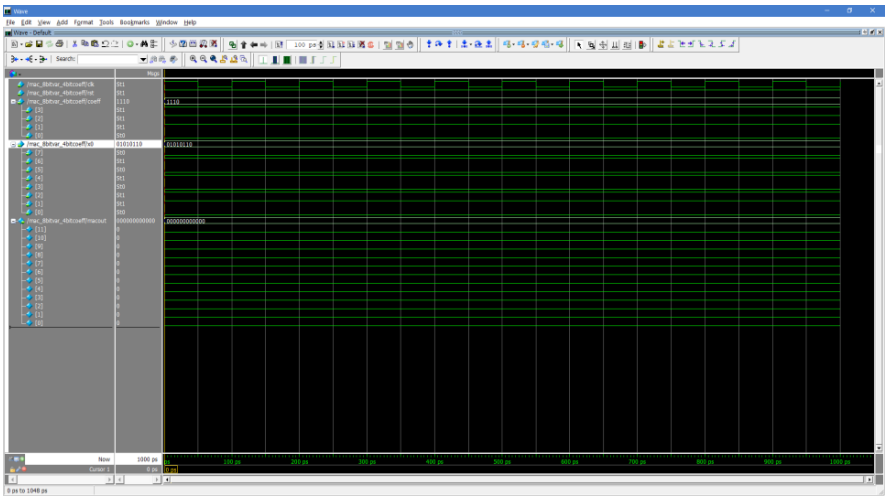

Fig 2. Simulation Waveform of implemented multiplier under reset

The input provided is 4-bit coeff as: “1110” whose value in decimal is “14” and 8-bit var as: “0101011110” whose value in decimal is “86”. Here it is observed clearly that when rst=1, then irrespective of any inputs the output is zero “000000000000” (12’b0).

In short the system is reset. Now let us see when rst= 0.

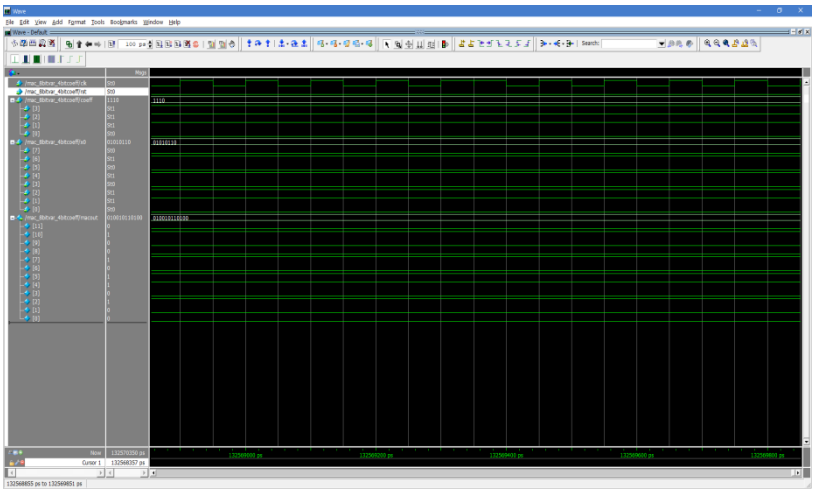

Fig 3: Simulation Waveform of implemented multiplier out of reset

Here we have made $rst = 0$, so basically now the system is out of reset state, hence the system will function as it was designed to function. Now it will produce the output which will be the multiplication of both the inputs.

Clearly the output obtained is macout= "010010110100" whose decimal value is "1204".

14 X 86 = 1204.

By this the functionality of our proposed idea is verified.

Below shown is the snippet which is the synthesis outcome of this particular algorithm using Synopsys Synlipy Pro Synthesis Tool.

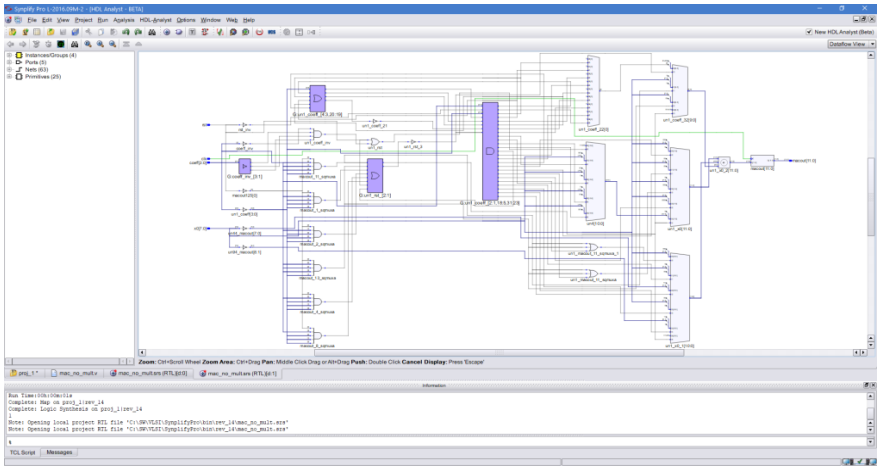

Fig 4: Synthesis outcome of implemented multiplier

no_mul_cellcount - Notepad

File Edit Format View Help

```
Report for cell mac_no_mult.verilog
Core Cell usage:
                  cell count       area count*area
                  AO13     1        1.0        1.0
                  AOI1     1        1.0        1.0
                   GND     1        0.0        0.0
                  MAJ3    16        1.0       16.0
                  MIN3     1        1.0        1.0
                  NOR2     1        1.0        1.0
                 NOR2B    34        1.0       34.0
                  NOR3     1        1.0        1.0
                 NOR3B     1        1.0        1.0
                   OA1     1        1.0        1.0
                   VCC     1        0.0        0.0
                   XA1     1        1.0        1.0
                  XA1B     9        1.0        9.0
                  XA1C     1        1.0        1.0
                  XOR2     8        1.0        8.0
                  XOR3    18        1.0       18.0


                  DFN1    12        1.0       12.0
                 -----                  ----------
                 TOTAL   108                 106.0


  IO Cell usage:
                  cell count
                CLKBUF     1
                 INBUF    13
                OUTBUF    12
                 -----
                 TOTAL    26
```

Fig 5: Report on resources utilized by the implemented multiplier

VI.ii. Implemented Adder Simulation and Synthesis Results:

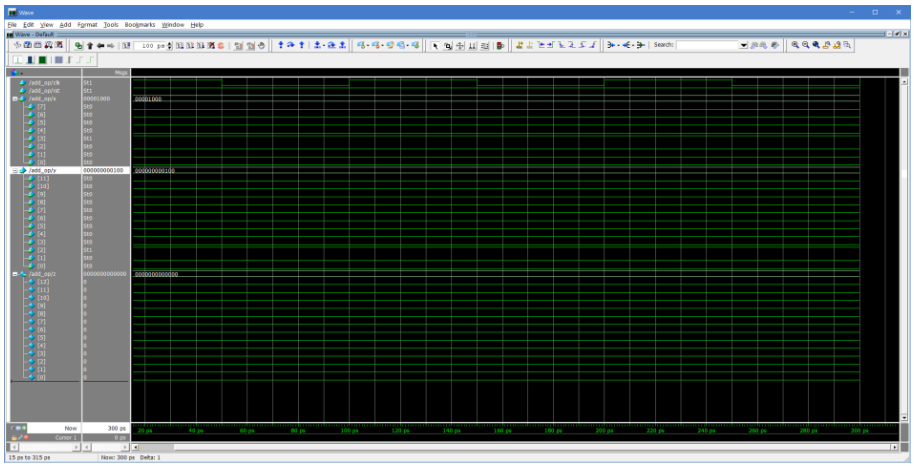

Fig 6: Simulation outcome of implemented adder under reset

Under this condition even though the inputs are non-zero, output will be zero.
In the following case 8-bit input x= 00001000 whose decimal value is “8” and 12-bit input Y=000000000100 whose decimal value is “4”.
As per the functionality the output should be z= 0000000001100 whose decimal value should be “12”.

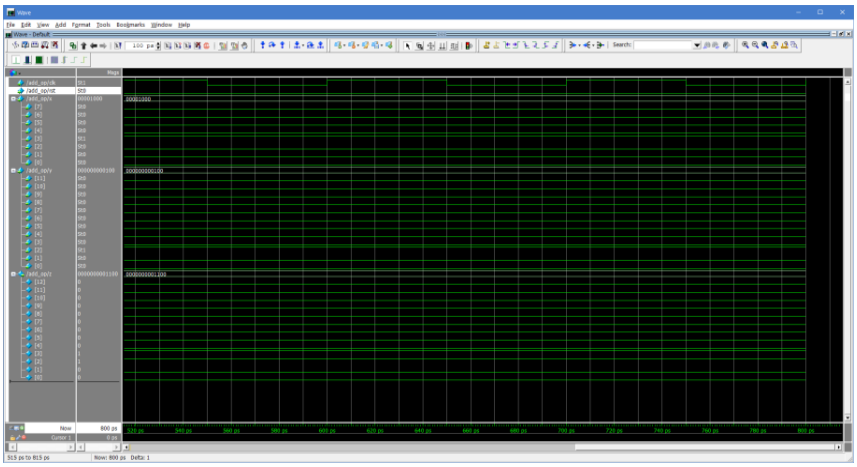

Fig 7: Simulation outcome of implemented adder out of reset

Here it can be clearly observed that the 13-bit output z= “0000000001100”. Hence the functionality of the circuit is verified.
Now let us synthesize the circuit using the Synopsys Synplify Pro Synthesis Tool. Below is the snippet attached.

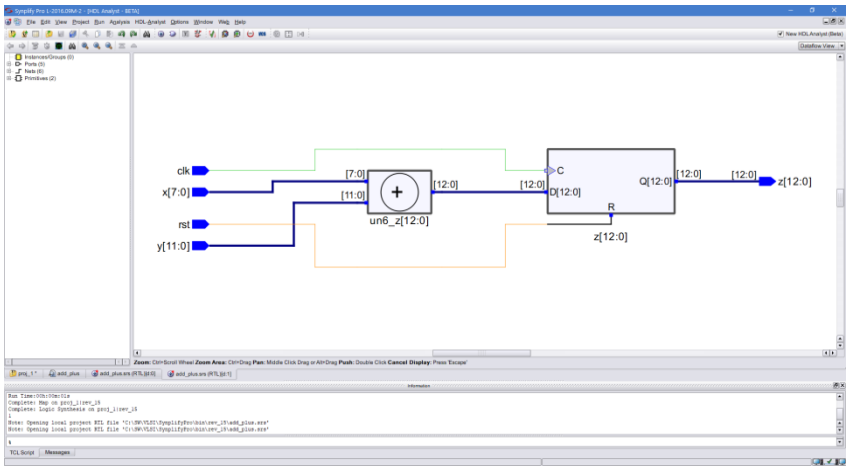

Fig 8: Synthesis of implemented adder

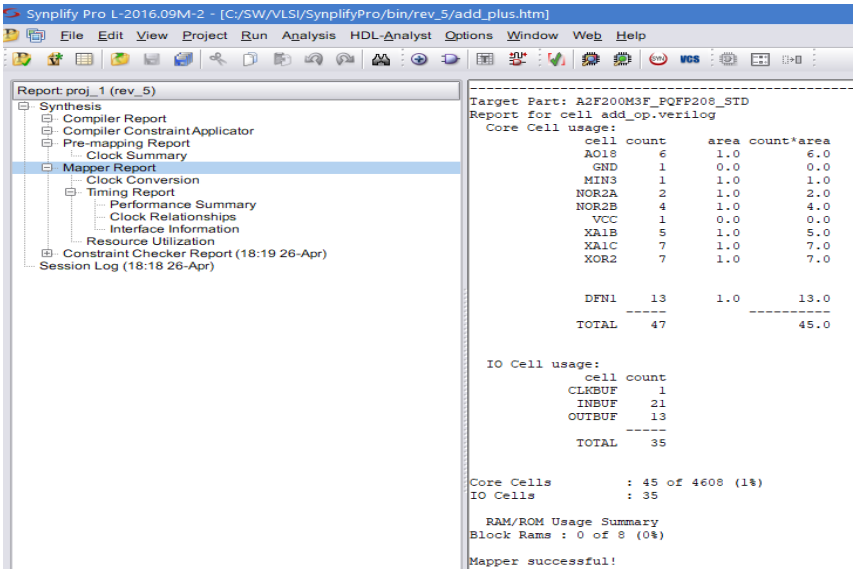

Fig 9: Report on resources utilized by implemented adder

VII. Conclusion

The electronics and semiconductor industry occupy a pivotal place in contemporary society by serving human needs and sub serving human happiness. Therefore, with the passage of time and growth in demand for electronic devices, the industry has witnessed fascinating growth. The development has been more pronounced in the arena of portable communication devices like Mobile phones, IPADS and notebooks. The existing high-performance systems used in the communication system utilizes Multiply Accumulate block which consumes large

area and power and are characterized by higher data throughput rate. But in the power starved society, there is an imperative need to discover systems that can reduce power and increase the speed of MAC system. In this context, several MAC with its area and power reduction techniques have been focused in this project work. Moreover, this analysis would help the future researchers to explore more in the field of low power, area and high speed multiply-accumulate unit.

VIII. Future Scope

The proposed methodology is designed keeping computational operations performed in Digital Signal Processing (DSP) applications in mind, but this can be further increased to other domains like Video Processing, and interestingly in Image Processing.

The main aim of MAC unit is actually the pace at which an operation is performed i.e., speed of operation. With further enhancements in the proposed methodology we can make MAC unit as more area efficient, faster, and minimize the power consumption even more.

They can be used in the implementation of finite impulse response filter design for DSP which are advantageous for low power applications. ASIC design for low power digital filter with low latency and power gating can be carried out. These arithmetic sub systems can be used in the implementation of arithmetic and logic units and multiply and accumulate units for DSP processor which improves performance of the system in different level of abstraction. The GDI based counter designs can be improved by using clock swing techniques in flip flops.

References

I. Abdelgawad, A. (2013). Low power multiply accumulate unit (MAC) for future Wireless Sensor Networks. 2013 IEEE Sensors Applications Symposium Proceedings.doi:10.1109/sas.2013.6493571.

II. Comparison among Different Adders Prof. Rashmi Rahul Kulkarni, IOSR Journal of VLSI and Signal Processing (IOSR-JVSP) Volume 5, Issue 6, Ver. I (Nov -Dec. 2015), PP 01-06 e-ISSN: 2319 – 4200, p-ISSN No. : 2319 – 4197

III. Jayanthi, A. N., &Ravichandran, C. S. (2013). Comparison of performance of high speed VLSI adders. 2013 International Conference on Current Trends in Engineering and Technology (ICCTET).doi:10.1109/icctet.2013.6675920

IV. Kapse, V., Jain, A., &Pattanaik, M. (2016). Design of an Area Efficient and Low Power MAC Unit.Smart Trends in Information Technology and Computer Communications, 276–284.doi:10.1007/978-981-10-3433-6_33.

V. Ramadass, Uma &Vijayan, Vidya&Mohanapriya, M & Paul, Sharon. (2012). Area, Delay and Power Comparison of Adder Topologies.International Journal of VLSI Design & Communication Systems.

VI. Shah, S., Al-Khalili, A. J., & Al-Khalili, D. (n.d.). Comparison of 32-bit multipliers for various performance measures.ICM 2000.Proceedings of the 12th International Conference on Microelectronics. (IEEE Cat. No.00EX453). doi:10.1109/icm.2000.916418