IJIRSET

ISSN(Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 5, May 2015

Novel High-Performance High-Valency Ling Adders

P.Santhosha [1], M.Rajeswara Rao [2]

P.G. Student, Department of ECE, SIETK, Andhra Pradesh, India

Assistant Professor, Department of ECE, SIETK, Andhra Pradesh, India

ABSTRACT: Parallel prefix adders are used for economical VLSI implementation of binary variety additions. The proposed Ling design offers a quicker carry computation stage compared to the standard parallel prefix adders by projecting a replacement methodology to solve Ling adders, which helps to cut the complexity and also the delay of the adder. This paper discusses the look and implementation details for such lower complex, quick parallel prefix adders supported Ling theory of resolving. Specifically, valency or node indicates number of inputs given to a particular node in a carry tree. The proposed ling adder shows that the high-valency Ling adders have superior space area and delay characteristics over antecedent reportable adders for identical input size. Moreover, our 20-bit high valency adder features a higher space and delay measuring than the antecedent used 16-bit adders.

KEYWORDS: Parallel prefix adders, Area, Delay, Valency and Node.

I. INTRODUCTION

One of the elemental operations in electronic circuits is Binary Addition. Several fashionable circuits contain many adder units for applications like arithmetic logic unit; thus, there's a substantial interest to style less advanced and better speed and adder architectures. Within the past few decades, numerous architectures of adders are planned to optimize the delay of the adder, examples embrace, carry-look ahead, ripple carry and parallel prefix adder. The parallel prefix adder is one in all the foremost in style architectures and offers sensible compromise among power, space and speed. This sort of adder implements a logic performs to see whether or not every bit position kills the carry or propagates it or generates it. Then these generate and propagate/not kill functions are hierarchically combined to cipher the carry into every bit position forming a carry tree. The ultimate stage computes add at as position using exclusive or (XOR) gates [1]. The rest of paper is structured as follows. Section2 gives summary of distinction between parallel prefix adders and others, and section3 presents the structure of parallel prefix adders. Section4 describes types of parallel prefix adders, Section 6 describes proposed ling adder. Section6 gives the simulation results and section7 provides conclusion of the paper.

II. RELATED WORK

In order to calculate the carries earlier with less delay and less complexness we have a tendency to use parallel prefix adders [2]. Three operations are often outlined to explain standing of carry (a) Propagate: Previous carry is propagated to next bit (b) Generate: Generate a carry bit (c) Kill: Kill the previous carry

$$G_{i:i} \equiv G_i = A_i . B_i$$
$$P_{i:i} \equiv P_i = A_i \oplus B_i$$
$$K_{i:i} \equiv K_i = \overline{A_i + B_i}$$

The PPA's pre-computes generate and propagate signals. Using the basic carry operator (fco), these computed signals are combined in [3].The fundamental carry operator is denoted by the symbol "o",

$$(g_L, p_L) o (g_R, p_R) = (g_L + p_L \cdot g_R, p_L, p_R)$$

For example, 4 bit CLA carry equation is given by

ISSN(Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 5, May 2015

$$C_4 = (g_4 , p_4) o [(g_3 , p_3) o [(g_4 , p_4) o (g_3 , p_3)]]$$

Table-1. Propagate, Generate and Kill the Carry

A	B	P	G	K
0	0	0	0	1
0	1	1	0	0
1	0	1	0	0
1	1	0	1	0

For example, 4 bit PPA carry equation is given by

$$C_4 = [(g_4 , p_4) o (g_3 , p_3)] o [(g_4 , p_4) o (g_3 , p_3)]$$

From these Equations we are able to observe that, the carry look ahead adder takes three steps to come up with the carry; however the bit PPA takes a pair of steps to come up with the carry.

A. Structure of Parallel-Prefix Adder

PPA's primarily consists of 3 stages: 1. Pre computation.2 Prefix stage.3. Final computation

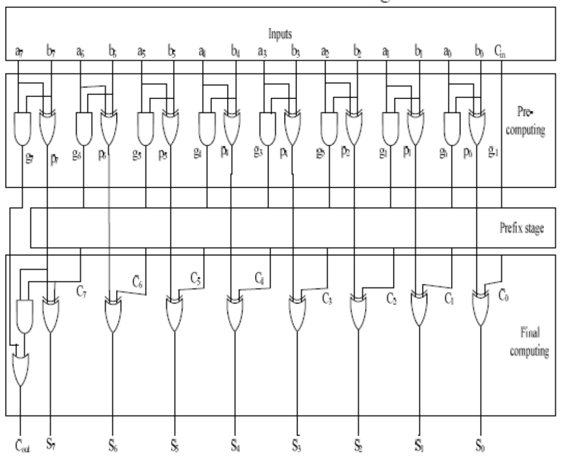


Fig.-1 Parallel-Prefix Structure

A. Pre computation

In pre computation stage, propagates and generates are computed for given inputs.

B. Prefix stage

In the prefix stage, group generate/propagate signals are computed at each bit using the given equations. The black cell (BC) generates the ordered pair, the gray cell (GC) generates only left signal.

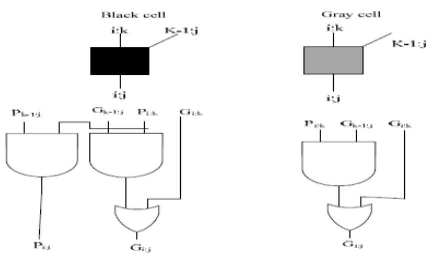


Fig.-2 Black Cell and Gray Cell Configurations

$$G_{i:k} = G_{i:j} + P_{i:j} . G_{j-1:k}$$
$$P_{i:k} = P_{i:j} . P_{j-1:k}$$

More practically, the equations can be expressed using a symbol "o "denoted by Brent and Kung. Its function is exactly the same as that of a black cell i.e.

$$G_{i:k} : P_{i:k} = (G_{i:j} : P_{i:j}) o (G_{j-1:k} : P_{j-1:k})$$

The "o" operation can facilitate create the foundations of building prefix structures.

C.Final computation

In the final computation, the total and carryout are the ultimate outputs

$$S_i = P_i . G_{i-1:-1}$$
$$C_{out} = G_{n:-1}$$

Where "-1" is that the position of carry-input.

III.TYPES OF PARALLEL PREFIX ADDERS

The sixteen bit SKA [6] uses black cells and grey cells similarly as full adder blocks too. This adder computes the carries using the BC's and GC's and terminates with four bit RCA's. Completely it uses sixteen full adders. The sixteen bit SKA is shown in figure-3.

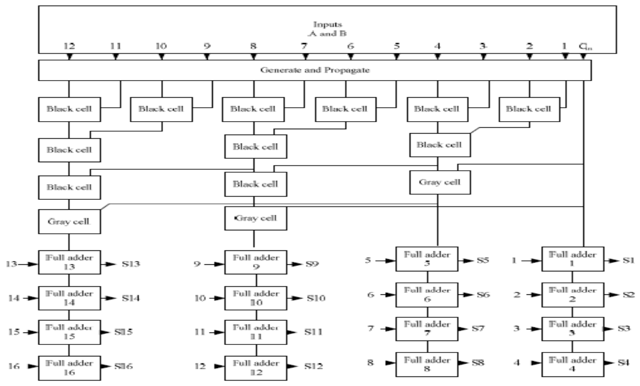


Fig.-3 16 Bit Sparse Kogge Stone Adder

During this adder, initial the input bits (a, b) are born-again as propagate and generate (p, g).Propagate and generate terms are given to BC's and GC's. The carries are propagated beforehand using these cells. Later this is given to full adder blocks.

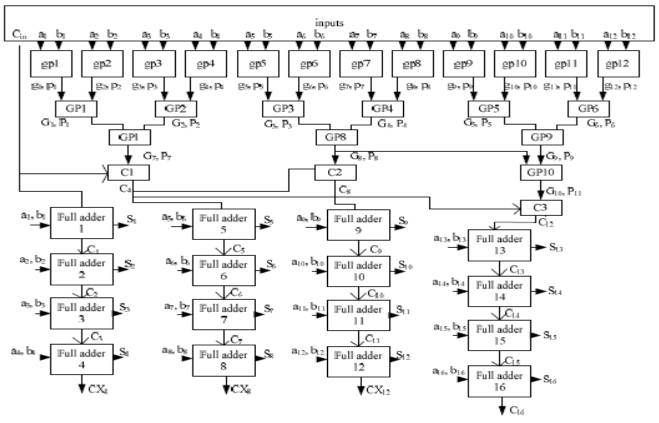


Fig-4 16 Bit Spanning Tree Adder

Another PPA is thought as STA [5] is just like the SKA; this adder additionally terminates with a RCA. It additionally uses the BC's and GC's and full adder blocks like SKA's however the distinction is that the interconnection between them .The sixteen bit STA is shown within the below figure-4.

KSA is another of prefix trees that use the fewest logic levels. A 16-bit KSA is shown in Figure-5. The sixteen bit kogge stone adder uses BC's and GC's and it won't use full adders. The sixteen bit KSA uses thirty six BC's and fifteen GC's. And this adder all operates on generates and propagates blocks. That the delay is a smaller amount when put next to the previous SKA and STA. The sixteen bit KSA is shown in figure five. In this KSA [7]; there aren't any full adder blocks like SKA and STA.

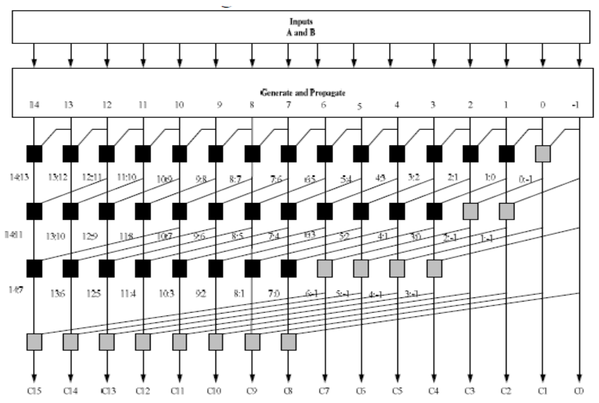


Fig- 5 16 Bit Kogge Stone Adder

Another carry tree called BKA that conjointly uses BC's and GC's however but the KSA. Therefore it takes less space to implement than KSA. The sixteen bit BKA uses fourteen BC's and eleven GC's however kogge stone uses thirty six BC's and fifteen GC's. Therefore BKA has less design and occupies less space than KSA. The sixteen bit BKA [8] is shown within the below figure-6.

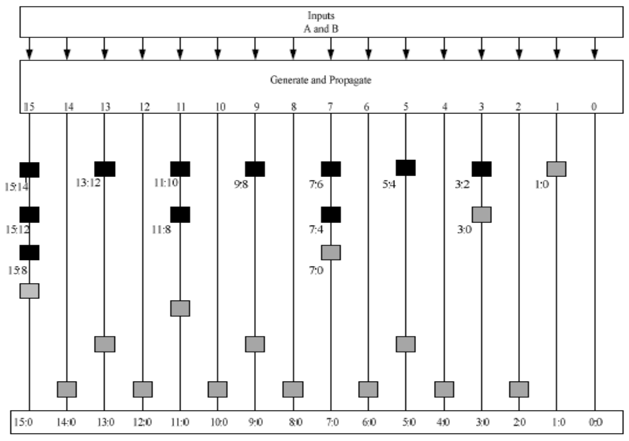


Fig.- 6 16 Bit Brent Kung Adder

BKA occupies less space than the opposite three adders known as SKA, KSA, and STA. This adder uses restricted range of propagate and generate cells than the opposite three adders. It takes less space to implement than the KSA and has less wiring congestion. The operation of the sixteen Brent Kung adders is given below. This adder uses less BC's and GC's than kogge stone adder and has the higher delay performance.

ISSN(Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 5, May 2015

IV. PROPOSED SYSTEM

The proposed ling adder technique is used to create the simplified group generate perform known as Pseudo carry. The factorization of 1 not kill bit makes the primary level of carry computation stage straightforward. And also the remainder of the logic of the carry tree is computed in an exceedingly similar manner because the standard theory of parallel prefix adder. The not kill term has got to be combined with the pseudo carry before computation of add at the top. As this involves no delay by the electronic device rather than X-OR circuit to reason adds, the ensuing adder becomes quicker.

A new replacement approach that the Ling adder will be simplified not solely within the initial stage, however conjointly within the subsequent stages. We showed that it's doable to use the Ling factorization to any or all the amount of the carry tree to create the reduced group generate term, that reduces complexness at every level. These pseudo carries will be hierarchically combined to create the carry tree so the ultimate add is computed using multiplexers.

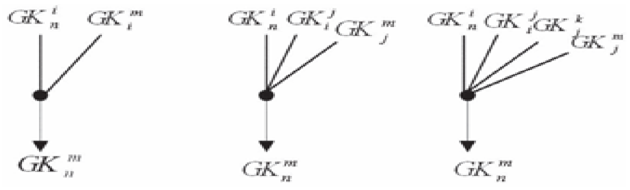


Fig. 7 Prefix Operator with valency 2, 3, & 4

The valency or base indicates the amount of inputs to one node within the carry tree. During this paper, we tend to use the terms low valency for 2 and 3 inputs, and high valency for four and 5 inputs. Using low valency within the logic circuit reduces the complexness at every level.

One 20-bit adder is meant use valency-5 .The designed adders are quicker than the traditional adders and plenty of the recently enforced ones. The new theory using high valency, evidenced economical than the traditional adders use of valency-2, in terms of space and speed. Experimental results show the comparison of recent adder styles and numerous varieties of typical parallel prefix adders styled within the same design flow and conjointly comparison of freshly designed adders with the recently revealed adders supported Ling theory.

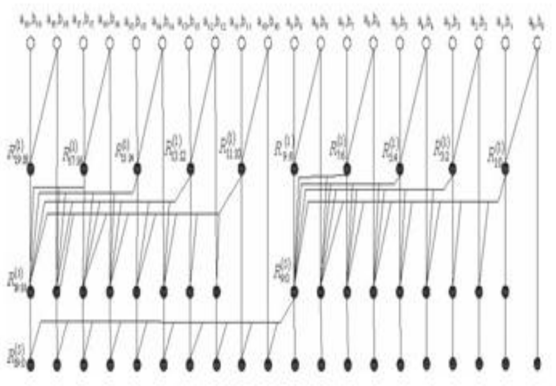


Fig. 8 Proposed System Architecture

The primary stage is created because the previous adders with valency-2 equations. The reduced group generates term misses one not kill bit that is denoted by superscript of the R term. The second stage a number of the positions mix five subgroups to make wider teams to work out the reduced cluster generate in 2 logic levels as no single cell with four inputs may be used. within the second stage for valency-3 and -4 one not kill term is factored out, creating the cluster generate equation straightforward and potential to work out in 2 logic levels in valency-4 and Fig. 4. 20-bit adder carry tree with valency 2*5*2 by single cell in valency-3. The valency-5 equation is extremely advanced during a typical adder. By resolving out 2 not kill terms, a lot of less complicated reduced generate equation is obtained. The third stage is created using valency-2.

V. SIMULATION RESULTS

The Design and Implementation of both existing and proposed systems are shown below.

A. Existing System Simulation Results

In Sparse kogge stone adder, we are using 16 bit inputs with Xilinx design suit for simulation and its delay is 19.976 ns.It consumes more area because of using more full adders.

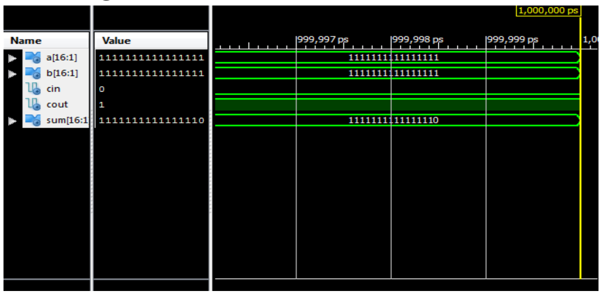

```
Name        Value
a[16:1]     1111111111111111
b[16:1]     1111111111111111
cin         0
cout        1
sum[16:1]   1111111111111110
```

Fig.-9 Simulation Results for Sparse Kogge Stone Adder

In spanning tree adder, we are using 16 bit inputs with Xilinx design suit for simulation and its delay is 18.930 ns. It consumes less area when compared to sparse kogge stone adder.

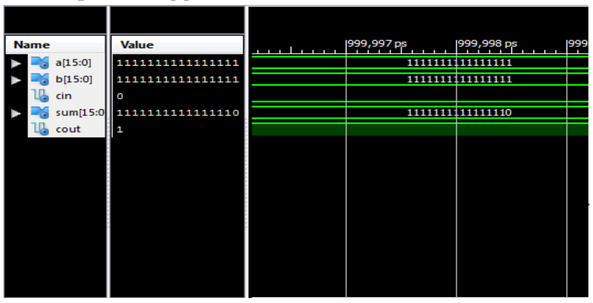

```
Name        Value
a[15:0]     1111111111111111
b[15:0]     1111111111111111
cin         0
sum[15:0    1111111111111110
cout        1
```

Fig-10 Simulation Results for Spanning Tree Adder

In kogge stone adder, we are using 16 bit inputs with Xilinx design suit for simulation and its delay is 14.277 ns. It has less delay when compared to sparse kogge stone adder and spaning tree adder because it does not use full adders.

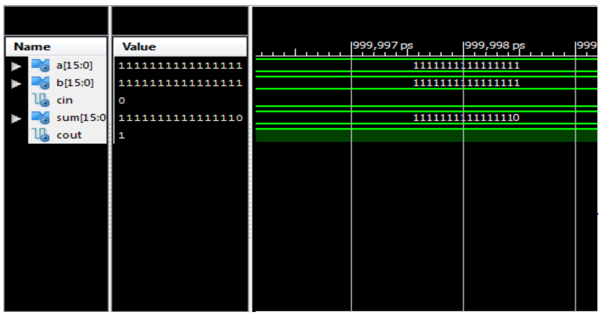
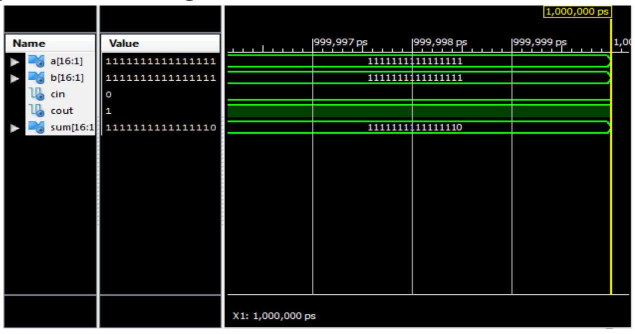

```
Name        Value
a[15:0]     1111111111111111
b[15:0]     1111111111111111
cin         0
sum[15:0    1111111111111110
cout        1
```

Fig-11 Simulation Results for kogge stone Adder

In Brent Kung adder, we are using 16 bit inputs with Xilinx design suit for simulation and its delay is 20.105 ns. It consumes less area when compared to remaining 3 adders.

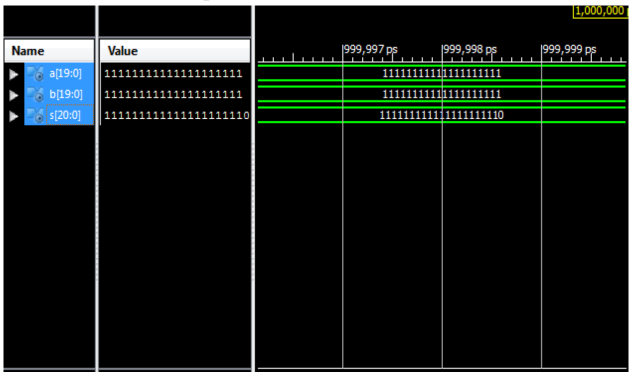


```
Name        Value
a[16:1]     1111111111111111
b[16:1]     1111111111111111
cin         0
cout        1
sum[16:1]   1111111111111110
```

Fig-12 Simulation Results for Brent Kung Adder.

B.Proposed system

The proposed 20-bit Ling adder has better Area and Delay characteristics when compared to existing Parallel Prefix Adder. The complexity is also reduced when compared to the other conventional adder.



```
Name      Value
a[19:0]   11111111111111111111
b[19:0]   11111111111111111111
s[20:0]   111111111111111111110
```

Fig-13 Simulation Results for proposed Ling adder

Ling design offers a quicker carry computation stage compared to the standard parallel prefix adders. This new technique to factorize Ling adders, helps to cut back the complexness also because the delay of the adder additional. Experimental results show that the high-valency Ling adders have superior space and delay characteristics over antecedent used adders. Moreover, our 20-bit high valency adder includes a higher space and delay measure than the antecedent revealed 16-bit adders.

Table-2. Comparison of Delay and Area for Adders

S.No	Adder Name	Delay in (ns)	Device Utilization (LUTs,IOBs) (1408,108)
1.	Sparse Kogge Stone Adder	19.976	40, 50
2.	Spanning Tree Adder	18.930	38, 50
3.	Kogge Stone Adder	14.277	92, 50
4.	Brent Kung Adder	20.105	38, 49
5.	Ling Adder	19.169	63, 61

VI. CONCLUSION

Ling factorization may be recursively applied any to any or all stages during a carry computation tree of an adder. This factorization reduces the complexness of the carry path that is mostly the vital path of the adder, creating it quicker. This makes another ways a lot of advanced, however if the complexness is correctly balanced the ensuing adder will work quicker. Plan of mixing a lot of simplified logic during a cluster while not buffers in between so drives another stage of logic, which is high valency implementation of carry stages. Because of using valency-4, 16-bit adders resulted in similar speed as that of the standard adders, enforced use of Ling factorization at the primary level, however space improvement is sizable. Using valency-5, 20-bit adders are higher in terms of each speed and space than the standard adders. Hence, it's shown that top valency Ling adders have superior space and delay characteristics over existing adders.

REFERENCES

[1] A. Weinberger and J. L. Smith, "A logic for high speed addition", National Bureau of Standards, Circulation 591, pp. 3-12, 1958.
[2] T. Lynch and E. Swartzlander, "A spanning tree carry look ahead adder", *IEEE Trans. on Computers*, vol. 41, no. 8, pp. 931-939, 1992.
[3] G. Yang, S. Jung, K. Baek, S. Kim, S. Kim, S. Kang, "A 32-bit carry lookahead adder using dual-path all-N logic", *IEEE Trans. on VLSI Systems*, vol. 13, no. 8, pp. 992-996, 2005.
[4] S. Hauck, M. Hosler and T. Fry, "High performance carry chains for FPGAs", *IEEE Trans. on VLSI Systems*, vol. 8, no. 2, pp. 138-147, 2000.
[5] C. Huang, J. Wang, C. Yeh and C. Fang, "The CMOS carry-forward adders", *IEEE Journal of Solid-State Circuits*, vol. 39, no. 2, pp. 327- 336, 2004.
[6] R.E. Ladner and M.J. Fischer, "Parallel prefix computation", *Journal of ACM*, vol. 27, no.4, pp.831-838, Oct. 1980.
[7] S. Knowles, "A family of adders", *Proc. 14th IEEE Symp. On Computer Arithmetic*, pp.30-34, 1999.
[8] P.M. Kogge and H.S. Stone, "A parallel algorithm for efficient solution of a general class of recurrence equations", *IEEE Trans. On Computers*, vol. C-22, no. 8, pp.786-793, Aug. 1973.
[9] H. Ling, "High speed binary adder", *IBM Journal of Research and Development*, vol. 25, no. 3, pp. 156-166, 1981.
[10] S. Das and S. P. Khatri, "A novel hybrid parallel-prefix adder architecture with efficient timing-area characteristic", *IEEE Trans. On VLSI Systems*, vol. 16, no. 3, pp. 326-331, 2008.

BIOGRPHY

P.Santhosha pursuing Mtech VLSI Design in the Department of Electronics and Communication Engineering in Siddharth Institute of Engineering and Technology, Puttur. She received her Bachelor Degree in Department of Electronics and Communication Engineering in Sri Padmavati Mahila University, Tirupati in Chittoor District.

M.Rajeswara Rao is working as Assistant Professor in the Department of Electronics and Communication Engineering, in the Siddharth Institute of Engineering and Technology. He received his Bachelor Degree in Department of Electronics and Communication Engineering from PCE College in Soollurupeta and Master Degree in Department of Electronics and Engineering in SITAMS, Hyderabad.