International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622
NATIONAL CONFERENCE on Developments, Advances & Trends in Engineering Sciences
(NCDATES- 09th & 10th January 2015)

RESEARCH ARTICLE OPEN ACCESS

Accumulator- Based- 3 Weight Pattern Generation

[1] G.Santoshi Vijaya Lakshmi, [2] N.Sai Akhila[3] M.Hima Bindu
Department of ECE, DVR College of Engineering and Technology, Sangareddy,Telangana.
M.Tech(ECE), Assistant Professor, DVR College of Engineering and Technology, Sangareddy,Telangana.
M.Tech(ECE), Assistant Professor, DVR College of Engineering and Technology, Sangareddy,Telangana.

Abstract:
Weighted pseudorandom built-in self test (BIST) schemes have been utilized in order to drive down the number of vectors to achieve complete fault coverage in BIST applications. Weighted sets comprising three weights, namely 0, 1, and 0.5 have been successfully utilized so far for test pattern generation, since they result in both low testing time and low consumed power. In this paper an accumulator-based 3-weight test pattern generation scheme is presented; the proposed scheme generates set of patterns with weights 0, 0.5, and 1. Since accumulators are commonly found in current VLSI chips, this scheme can be efficiently utilized to drive down the hardware of BIST pattern generation, as well. Comparisons with previously presented schemes indicate that the proposed scheme compares favorably with respect to the required hardware.

Keywords: Built-in self test (BIST), test per clock, VLSI testing, weighted test pattern generation.

I. INTRODUCTION

Pseudorandom built-in self test (BIST) generators have been widely utilized to test integrated circuits and systems. The arsenal of pseudo-random generators includes, among others, linear feedback shift registers (LFSRs) [1], cellular automata [2], and accumulators driven by a constant value [3]. For circuits with hard-to-detect faults, a large number of random patterns have to be generated before high fault coverage is achieved. Therefore, weighted pseudorandom techniques have been proposed where inputs are biased by changing the probability of a "0" or a "1" on a given input from 0.5 (for pure pseudorandom tests) to some other value [10], [15].

Weighted random pattern generation methods relying on a single weight assignment usually fail to achieve complete fault coverage using a reasonable number of test patterns since, although the weights are computed to be suitable for most faults, some faults may require long test sequences to be detected with these weight assignments if they do not match their activation and propagation requirements. Multiple weight assignments have been suggested for the case that different faults require different biases of the input combinations applied to the circuit, to ensure that a relatively small number of patterns can detect all faults [4]. Approaches to derive weight assignments for given deterministic tests are attractive since they have the potential to allow complete coverage with a significantly smaller number of test patterns [10].

In order to minimize the hardware implementation cost, other schemes based on multiple weight assignments utilized weights 0, 1, and 0.5. This approach boils down to keeping some outputs of the generator steady (to either 0 or 1) and letting the remaining outputs change values (pseudo-) randomly (weight 0.5). This approach, apart from reducing the hardware overhead has beneficial effect on the consumed power, since some of the circuit under test (CUT) inputs (those having weight 0 or 1) remain steady during the specific test session [30]. Pomeranz and Reddy [5] proposed a 3-weight pattern generation scheme relying on weights 0, 1, and 0.5. The choice of weights 0, 1, and 0.5 was done in order to minimize the hardware implementation cost. Wang [8], [13] proposed a 3-weight random pattern generator based on scan chains utilizing weights 0, 1, and 0.5, in a way similar to [5]. Recently, Zhang *et al.* [9] renovated the interest in the 3-weight pattern generation schemes, proposing an efficient compaction scheme for the 3-weight patterns 0, 1, and 0.5. From the above we can conclude that 3-weight pattern generation based on weights 0, 1, and 0.5 has practical interest since it combines low implementation cost with low test time

Current VLSI circuits, e.g., data path architectures, or digital signal processing chips commonly contain arithmetic modules [accumulators or arithmetic logic units (ALUs)]. This has fired the idea of arithmetic BIST (ABIST) [6]. The basic idea of ABIST is to utilize accumulators for built-in testing (compression of the CUT responses, or generation of test patterns) and has been shown to result in low hardware over-head and low impact on the circuit normal operating speed [22]–[27]. In [22], Manich *et al.* presented an accumulator-based test pattern generation scheme

that compares favorably to previously proposed schemes. In [7], it was proved that the test vectors generated by an accumulator whose inputs are driven by a constant pattern can have acceptable pseudorandom characteristics, if the input pattern is properly selected. However, modules containing hard-to-detect faults still require extra test hardware either by inserting test points into the mission logic or by storing additional deterministic test patterns [24], [25].

In order to overcome this problem, an accumulator-based weighted pattern generation scheme was proposed in [11]. The scheme generates test patterns having one of three weights, namely 0, 1, and 0.5 therefore it can be utilized to drastically reduce the test application time in accumulator-based test pattern generation. However, the scheme proposed in [11] possesses three major drawbacks: 1) it can be utilized only in the case that the adder of the accumulator is a ripple carry adder; 2) it requires redesigning the accumulator; this modification, apart from being costly, requires redesign of the core of the data path, a practice that is generally discouraged in current BIST schemes; and 3) it in-creases delay, since it affects the normal operating speed of the adder.

In this paper, a novel scheme for accumulator-based 3-weight generation is presented. The proposed scheme copes with the inherent draw-backs of the scheme proposed in [11]. More precisely: 1) it does not impose any requirements about the design of the adder (i.e., it can be implemented using any adder design); 2) it does not require any modification of the adder; and hence, 3) does not affect the operating speed of the adder. Furthermore, the proposed scheme compares favorably to the scheme proposed in [11] and [22] in terms of the required hardware overhead.

This paper is organized as follows. In Section II, the idea underlying the accumulator-based 3-weight generation is presented. In Section III, the design methodology to generate the 3-weight patterns util izing an accumulator is presented. In Section IV, the proposed scheme is com-pared to the previously proposed ones. Finally, Section V, concludes this paper.

II. ACCUMULATOR – BASED - 3 WEIGHT PATTERN GENERATION

We shall illustrate the idea of an accumulator-based 3-weight pattern generation by means of an example. Let us consider the test set for the c17 ISCAS benchmark [12], [31] given in Table I.

Starting from this deterministic test set, in order to apply the 3-weight pattern generation scheme, one of the schemes proposed in [5], [8], and [9] can be utilized. According to these schemes, a typical weight assignment procedure would involve separating the test set into two subsets, S1 and S2 as follows:

TABLE I
TEST SET FOR THE C17 BENCHMARK CIRCUIT

Test vector	Inputs A[4:0]
T1	00101
T2	01010
T3	10010
T4	11111

TABLE II
TRUTH TABLE OF THE FULL ADDER

#	Cin	A[i]	B[i]	S[i]	C_{out}	Comment
1	0	0	0	0	0	
2	**0**	0	1	1	**0**	$C_{out} = C_{in}$
3	**0**	1	0	1	**0**	$C_{out} = C_{in}$
4	0	1	1	0	1	
5	1	0	0	1	0	
6	**1**	0	1	0	**1**	$C_{out} = C_{in}$
7	**1**	1	0	0	**1**	$C_{out} = C_{in}$
8	1	1	1	1	1	

S1 = {T1, T4} and S2 = {T2, T3} The weight assignments for these subsets is W(S1) = (_, _, 1, _, 1) and W(S2) = (_, _, 0, 1, 0) where a "_" denotes a weight assignment of 0.5, a "1" indicates that the input is constantly driven by the logic "1" value, and "0" indicates that the input is driven by the logic "0" value. In the first assignment, inputs A[2] and A[0] are constantly driven by "1", while inputs A[4], A[3], A[1] are pseudo randomly generated (i.e., have weights 0.5). Similarly, in the second weight assignment (subset S2), inputs A[2] and A[0] are constantly driven by "0", input A[1] is driven by "1" and inputs A[4] and A[3] are pseudo randomly generated.

The above reasoning calls for a configuration of the accumulator, where the following conditions are met: 1) an accumulator output can be constantly driven by "1" or "0" and 2) an accumulator cell with its output constantly driven to "1" or "0" allows the carry input of the stage to transfer to its carry output unchanged. This latter condition is required in order to effectively generate pseudorandom patterns in the accumulator outputs whose weight assignment is "0".

III. DESIGN METHODOLOGY

The implementation of the weighted-pattern generation scheme is based on the full adder truth table, presented in Table II. From Table II we can

see that in lines #2, #3, #6, and #7 of the truth table, $C_{out} = C_{in}$. Therefore, in order to transfer the carry input to the carry output, it is enough to set A[i] = NOT(B[i]) . The proposed scheme is based on this observation.

The implementation of the proposed weighted pattern generation scheme is based on the accumulator cell presented in Fig. 1, which consists of a Full Adder (FA) cell and a D-type flip-flop with asynchronous set and reset inputs whose output is also driven to one of the full adder inputs. In Fig. 1, we assume, without loss of generality, that the set and reset are active high signals. In the same figure the respective cell of the driving register B[i] is also shown. For this accumulator cell, one out of three configurations can be utilized, as shown in Fig. 2.

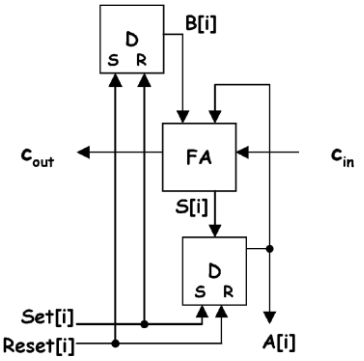

Fig1 Accumulator Cell

In Fig.2(a) we present the configuration that drives the CUT inputs when A[i] =" 1" is required. Set[i] = 1 and Reset[i] = 0 and hence A[i] = 1 and B[i] = 0. Then the output is equal to 1 and C_{in} is transferred to C_{out}.

In Fig.2(b) we present the configuration that drives the CUT inputs when A[i] =" 0" is required. Set[i] = 0 and Reset[i] = 1 and hence A[i] = 0 and B[i] = 1. Then the output is equal to 0 and C_{in} is transferred to C_{out}.

In Fig.2(c) we present the configuration that drives the CUT inputs when A[i] = "_" is required. Set[i] = 0 and Reset[i] = 0 and hence A[i] = 1 and B[i] = 0. The D input of the flip-flop of register B is driven by either 1 or 0, depending on the value that will be added to the accumulator inputs in order to generate satisfactorily random patterns to the inputs of the CUT.

In Fig. 3, the general configuration of the proposed scheme is presented. The Logic module provides the Set[n-1:0] and Reset[n-1:0] signals that drive the S and R inputs of the Register A and Register B inputs. Note that the signals that drive the S inputs of the flip-flops of Register A, also drive the R inputs of the flip-flops of Register B and vice versa.

IV. COMPARISONS

In this section, we shall perform comparisons in three directions. In Section 4.1, we shall compare the proposed scheme with the accumulator-based 3-weight generation scheme that has been proposed in [11]. In Section 4.2, we shall compare the proposed scheme with the 3-weight scan schemes that have been proposed in [5] and [8]. In Section 4.3, in order to demonstrate the applicability of the pro-posed scheme we shall compare the proposed scheme with the accumulator-based test pattern generation scheme proposed in [22].

4.1. Comparisons With [11]

The number of test patterns applied by [11] and the proposed scheme is the same, since the test application algorithms that have been invented and applied by previous researchers, e.g., [5], [8], [9] can be equally well applied with both implementations. Therefore, the comparison will be performed with respect to: 1) the hardware overhead and 2) the impact on the timing characteristics of the adder of the accumulator.

Both schemes require a session counter in order to alter among the different weight sessions; the session counter consists of bits, where is the number of test sessions (i.e., weight assignments) of the weighted test set. The scheme proposed in [11] requires the redesign of the adder; more precisely, two NAND gates are inserted in each cell of the ripple-carry adder. In order to provide the inputs to the set and reset inputs of the flip flops, decoding logic is implemented, similar to that in [8]. For the proposed scheme, no modification is imposed on the adder of the accumulator. Therefore, there is no impact on the data path timing characteristics

In Table III we present comparison results for some of the ISCAS'85 benchmarks. In the first column of Table III, we present the benchmark name; in the second and third columns we present the hardware over-head of the accumulator-based scheme proposed in [11] and in this work, respectively; in the fourth column we present the decrease of the proposed scheme over [11]. In the fifth through the seventh columns, we present the delay of the adder in terms of number of gates that the carry signal has to traverse, from the input of the adder (lower stage full adder cell) to the output (higher stage full adder cell), as well as the respective decrease obtained by the proposed scheme.

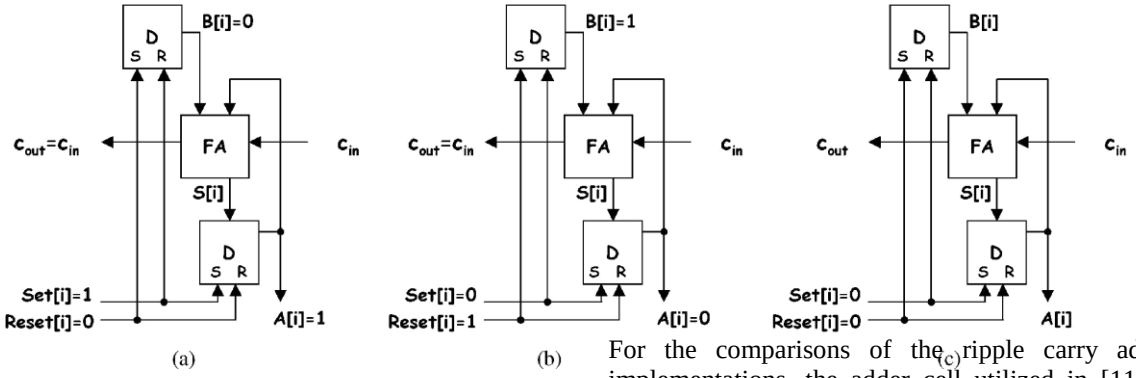

Fig.2 Configurations of the accumulator cell of Fig.1

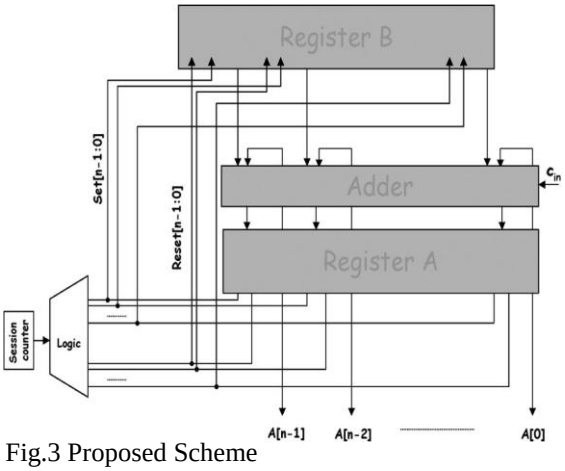

Fig.3 Proposed Scheme

In Table III, the hardware overheads are calculated in gate equivalents, where an -input NAND or NOR accounts for 0.5 gates and an inverter accounts for 0.5 gates, as proposed in [8]. For the calculation of the delay in the adder operation (columns under heading "#gates from C_{in} to C_{out}") we have considered both ripple carry and prefix adder implementations.

TABLE III
COMPARISONS WITH [11]

circuit	hardware overhead			delay from c_{in} to c_{out}				
					(ripple)		(prefix)	
name	[11]	pr op.	de cr.	[11]	pr op.	de cr.	pr op	de cr.
c880	41%	8%	81%	240	180		24	90%
c1355	28%	7%	74%	164	123		22	87%
c1908	13%	3%	77%	132	99		21	84%
c2670	34%	8%	75%	932	699	20%	32	97%
c3540	11%	5%	57%	200	150		23	89%
c5315	17%	2%	90%	712	534		30	96%
c7552	17%	4%	75%	828	621		31	96%

For the comparisons of the ripple carry adder implementations, the adder cell utilized in [11] is considered; in the cell presented in [11], initially the delay from the to of the adder cell is two NAND gates and one XOR gate; in the modified cell proposed in [11], the delay is increased to three NAND and one XOR gate; we have considered that the delay of a NAND gate is one gate equivalent, while the delay of an XOR gate is two gate equivalents. Since the implementation of the proposed scheme does not rely on a specific adder design, the utilization of a prefix adder can result in impressive results. For the calculation of the delay of prefix adders, the formula obtained by [29] is utilized, where the delay is of the order 2 , where is the number of the adder stages. From Table III, we can see that the proposed scheme results in 57%–90% decrease in hardware overhead, while at the same time achieving a decrease in operational delay over-head that ranges from 84% to 97% for the considered benchmarks.

4.2. Comparisons With Scan-Based Schemes [5], [8]

Since the test application algorithms that have been invented and applied by [5], [8], and [9] can be equally well applied with the proposed scheme, test application time is similar to that reported there. There-fore, the comparison will be performed with respect to hardware over-head.

In the 3-weight pattern generation scheme proposed by Pomeranz and Reddy in [5] the scan chain is driven by the output of a linear feed-back shift register (LFSR). Logic is inserted between the scan chain and the CUT inputs to fix the outputs to the required weight (0, 0.5, or 1). In order to implement the scheme [5], a scan-structure is assumed. Furthermore, an LFSR required to feed the pseudorandom inputs to the scan inputs is implemented (the number of LFSR stages is , where is the number of scan cells), as well as a scan counter, common to all scan schemes. A number of 3-gate modules is required for every required weighted input (in [5, Table V], the hardware overhead is calculated for the ISCAS'85 benchmarks).

Wang [8] proposed a low-overhead 3-weight random BIST scheme, again based on scan chains. He proposed two schemes, namely serial fixing BIST and parallel fixing BIST. Serial fixing scheme is shown to be more costly [8]; therefore we shall concentrate our comparisons to the parallel fixing BIST scheme. For an -input CUT and, assuming the availability of the scan chain, the hardware overhead, apart from the LFSR to generate the pseudorandom inputs and the scan counter, includes a decoding logic. The hardware overhead of the decoding logic for some of the ISCAS benchmarks is calculated in [8, Table I]. All schemes require the application of the session counter, required to alter among the different weight sessions. Schemes proposed in [5] and [8] are test per scan schemes, and, of course, assume the existence of scan capability of the latches of the design.

In Table IV, we have presented arithmetic results for some of the ISCAS'85 benchmarks. For the calculations in Table IV, we have assumed that schemes [5] and [8] are applied to a circuit with scan capability; therefore, the hardware overhead to transform the latches into scan latches is not taken into account. For the scheme proposed in [5], the total hardware overhead comprises the hardware for the weighting gates (second column), the scan counter and the LFSR implementation. In order to calculate the numbers in the second column, we utilized the data found in in [5, Table V]. For the scheme in [8], the LFSR, the scan counter and the decoding logic are required; the hardware overhead for the decoding logic has been quoted from [8, Table I].

TABLE IV
COMPARISONS WITH THE SCAN SCHEMES PROPOSED IN [5] AND [8]

CUT	Pomeranz [5]				Wang [8]				Proposed
	weighting gates + scan counter + LFSR = Total				LFSR + decoding logic + scan counter = Total				
c880	5	47	47	**99**	47	6	47	**100**	**27**
c1355	1	43	43	**87**	43	6	43	**92**	**38**
c1908	0	40	40	**80**	40	2	40	**82**	**23**
c2670	542	63	63	**668**	63	39	63	**165**	**101**
c3540	2	45	45	**92**	45	24	45	**114**	**73**
c5315	0	60	60	**120**	60	7	60	**127**	**39**
c7552	1134	62	62	**1258**	62	66	62	**190**	**139**

TABLE V
COMPARISON WITH THE SCHEME PROPOSED IN [22]

benchmark			[22]		proposed	
circuit	h/w	#inp	tests	h/w	#tests	h/w
c880	383	60	1112	36	768	27
c1355	546	41	1409	26	1024	38
c1908	880	33	3198	23	1536	23
c2670	1193	157	1962	1386	4096	101
c3540	1669	50	2167	31	1536	73
c5315	2307	178	1453	189	2048	39
c7552	3512	206	2918	1090	4608	139
s5378	1004	214	2078	791	5120	47
s9234	2027	247	14803	4763	11264	181
s13207	2573	700	14476	7497	12288	61
s15850	3448	611	14902	18438	21504	159
s38584	11448	1464	8449	26235	16384	82
Average values			**5744**	**5042**	**6848**	**81**

4.3. Comparisons With [22]

In [22], Manich *et al.* proposed a methodology to reduce the total test time using an accumulator-based scheme. The scheme operates in test sessions based on triplets of the form (S, I, L), where S is the starting value of the accumulator, I is the increment, and L is he number of cycles the increment is applied before going to the next session. For the comparisons we have utilized the data from [22, Table I], and have considered that the seeds are stored in a ROM; for the hardware calculation we have considered that a ROM bit is equivalent to 1/4 gates, as has been also considered in [20] and [32]. The comparison data for some of the ISCAS'85 and ISCAS'89 benchmarks are presented in Table V, where the same fault coverage, i.e., 100% is targeted.

In Table V, in the first three columns we present the benchmark characteristics (name, hardware overhead in gates, and number of inputs). In the two columns to follow, we present the number of tests required for the scheme in [22], and the respective hardware overhead in gate equivalents. Next, we present the number of test patterns and hardware overhead for the proposed scheme. From Table V it is trivial to see that the proposed scheme presents an important decrease in the hard-ware overhead, while the number of tests is comparable, while in some cases it also outperforms the scheme in [22]. It is interesting to note that the hardware overhead (with respect to the hardware overhead of the benchmarks) is practical, in contrast to [22] which sometimes exceeds the benchmark hardware (c2670, s5378, s9234, s13207, s15850, s38584). It is also interesting to note the average values presented in the last line of the Table. The average increase in the number of tests is 19%, while the average decrease in hardware overhead is 98%.

V. CONCLUSION

We have presented an accumulator-based 3-weight (0, 0.5, and 1) test-per-clock generation scheme, which can be utilized to efficiently generate weighted patterns without altering the structure of the adder.

Comparisons with a previously proposed accumulator-based 3-weight pattern generation technique [11] indicate that the hardware overhead

of the proposed scheme is lower (75%), while at the same time no redesign of the accumulator is imposed, thus resulting in reduction of 20%–95% in test application time. Comparisons with scan based schemes [5], [8] show that the proposed schemes results in lower hardware overhead. Finally, comparisons with the accumulator-based scheme proposed in [22] reveal that the proposed scheme results in significant decrease (98%) in hardware overhead.

REFERENCES

[1] P. Bardell, W. McAnney, and J. Savir, *Built-In Test For VLSI: Pseudo-random Techniques*. New York: Wiley, 1987.

[2] P. Hortensius, R. McLeod, W. Pries, M. Miller, and H. Card, "Cel-lular automata-based pseudorandom generators for built-in self test," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol. 8, no. 8*, pp. 842–859, Aug. 1989.

[3] A. Stroele, "A self test approach using accumulators as test pattern generators," in *Proc. Int. Symp. Circuits Syst.*, 1995, pp. 2120–2123.

[4] H. J. Wunderlich, "Multiple distributions for biased random test pat-terns," in *Proc. IEEE Int. Test Conf.*, 1988, pp. 236–244.

[5] I. Pomeranz and S. M. Reddy, "3 weight pseudo-random test genera-tion based on a deterministic test set for combinational and sequential circuits," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol. 12, no. 7*, pp. 1050–1058, Jul. 1993.

[6] K. Radecka, J. Rajski, and J. Tyszer, "Arithmetic built-in self-test for DSP cores," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol. 16, no. 11*, pp. 1358–1369, Nov. 1997.

[7] J. Rajski and J. Tyszer, *Arithmetic Built-In Self Test For Embedded Systems*. Upper Saddle River, NJ: Prentice Hall PTR, 1998.

[8] S. Wang, "Low hardware overhead scan based 3-weight weighted random BIST," in *Proc. IEEE Int. Test Conf.*, 2001, pp. 868–877.

[9] S. Zhang, S. C. Seth, and B. B. Bhattacharya, "Efficient test compaction for pseudo-random testing," in *Proc. 14th Asian Test Symp.*, 2005, pp. 337–342.

[10] J. Savir, "Distributed generation of weighted random patterns," *IEEE Trans. Comput., vol. 48, no. 12*, pp. 1364–1368, Dec. 1999.

[11] I. Voyiatzis, D. Gizopoulos, and A. Paschalis, "Accumulator-based weighted pattern generation," presented at the IEEE Int. Line Test Symp., Saint Raphael, French Riviera, France, Jul. 2005.

Mrs. G. Santoshi Vijaya Lakshmi is presently is working towards a Master of Technology in E.C.E from JNTU Hyderabad at prestigious DVR Engineering College Hyderabad, India.

Miss. N. Sai Akhila is presently working as an Assistant Professor of Electronics and Communication Engineering in DVR Engineering College Hyderabad, India has done M.Tech from JNTU Hyderabad

Mrs. N. Hima Bindu is presently working as an Assistant Professor of Electronics and Communication Engineering in DVR Engineering College Hyderabad, India has done M.Tech from JNTU Hyderabad