Fpga Based Implementation of Bist Controller Using Different Approaches

Sachin D. Kohale and Ratnaprabha. W. Jasutkar

***Abstract*—Testing of integrated circuits (IC's) is of crucial importance to ensure a high level of quality in product functionality in both commercially and privately produced products. Due to complex systems, its very difficult to test it. One solution to this problem is to add logic to the IC so that it can test itself. This is referred to as "Built in self Test" (BIST). In this work, we are designing BIST controller which will detect and correct errors while computing greatest common divisor (gcd) of two non-negative integers using two approaches i.e Euclid's algorithm and Stein's algorithm and comparing the results of both approaches , that we are using in this work. The most efficient way that will come can use for the applications for finding gcd.**

***Index Terms*—Build-in-self-test(BIST), Euclid's algorithm, Stein's algorithm, VLSI testing.**

I. Introduction

As integrated circuits are produced with greater and greater levels of circuit density, efficient testing schemes that guarantee very high fault coverage while minimizing test costs and chip area overhead have become essential. As the complexity of circuits continues to increase, high fault coverage of several types of fault models becomes more difficult to achieve. In this project, we are designing BIST controller which helps for detecting and correcting errors, as it is very difficult to detect errors in such a complex digital systems now a days. Also, we are implementing two algorithms, Euclid's and Stein's which will helps for calculating greatest common divisor (gcd) of two non-negative integers. This algorithms will then be very useful for applications such as data security, cryptography etc. and so on.

A. *Built-In Self Test(BIST) Controller*

Built-In Self Test(BIST) [1] is a technique of integrating the functionality of an automatic test system onto a chip. It is a Design for Test technique in which testing (test generation and test application) is accomplished through built in hardware features.

On-line BIST [2] refers to testing which occurs during normal operation of the IC. Examples of this kind of BIST often have to do with functional testing such as Error Detecting/Error Correcting (ED/EC) codes or on chip electrical monitoring. *Off-line* operation occurs during a specified period when the CUT is idle. This operation occurs often over the period of multiple clock cycles and is usually intended to operate during a dedicated testing period.

The Basic block diagram of BIST architecture is shown below.

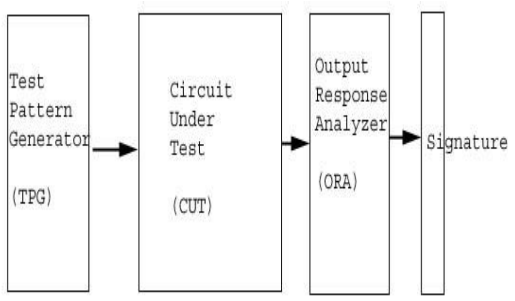

Fig. 1. Basic BIST architecture block diagram

II. Methodologies Used

A. *Euclid's Algorithm*

In mathematics, the Euclidean algorithm (also called Euclid's algorithm) [3] is an efficient method for computing the greatest common divisor (GCD) of two integers, also known as the greatest common factor (GCF) or highest common factor (HCF). In its simplest form, Euclid's algorithm starts with a pair of positive integers and forms a new pair that consists of the smaller number and the difference between the larger and smaller numbers. The process repeats until the numbers are equal. That number then is the greatest common divisor of the original pair. The GCD of two numbers is the largest number that divides both of them without leaving a remainder. The Euclidean algorithm (5) is based on the principle that the greatest common divisor of two numbers does not change if the smaller number is subtracted from the larger number. For if k, m, and n are integers, and k is a common factor of two integers A and B, then A = $(n \times k)$ and B = $(m \times k)$ implies A – B = $(n - m) \times k$, therefore k is also a common factor of the difference. That k may also represent the greatest common divisor is proven below. For example, 21 is the GCD of 252 and 105 (252 = 12 × 21;105 = 5 × 21)(4);since252 − 105 = (12-5) × 21 = 147, the GCD of 147 and 105 is also 21 [4].

Since the larger of the two numbers is reduced, repeating this process gives successively smaller numbers until one of them is zero. When that occurs, the GCD is the remaining nonzero number. By reversing the steps in the Euclidean algorithm, the GCD can be expressed as a sum of the two original numbers each multiplied by a positive or negative

Manuscript received December 14, 2012; revised February 17, 2013.

Sachin D. Kohale is a Post-Graduate Student with the Dept. of Computer Science and Engineering, G.H.Raisoni College of Engineering, Hingna Road, Nagpur, Maharashtra, India (e-mail: sdk_pz@yahoo.com).

Ratnaprabha. W. Jasutkar is a Assistant Professor with the Dept. of Computer Science and Engineering, G.H.Raisoni College of Engineering, Hingna Road, Nagpur, Maharashtra, India (e-mail: ratnaprabhajasutkar@gmail.com).

integer, e.g., 21 = [5 × 105] + [(−2) × 252]. This important property is known as Bézout's identity

Basically Euclid algorithm [3] can be described as

$$\gcd(a, 0) = a \tag{1}$$

$$\gcd(a, b) = \gcd(b, a \bmod b) \tag{2}$$

If arguments are both greater than zero, then

$$\gcd(a, a) = a \tag{3}$$

$$\gcd(a, b) = \gcd(a - b, b)\text{ ; if } b < a \tag{4}$$

$$\gcd(a, b) = \gcd(a, b - a)\text{ ; if } a < b \tag{5}$$

Ex. gcd(20, 20) is 20 (3). Also, gcd(20, 40) is same as calculating gcd(20 , (40–20)) (5) is again gcd(20, 20) (3).

B. *Stein's Algorithm*

The binary GCD algorithm, also known as Stein's algorithm [5], is an algorithm that computes the greatest common divisor of two nonnegative integers. It gains a measure of efficiency over the ancient Euclidean algorithm by replacing divisions and multiplications with shifts, which are cheaper when operating on the binary representation used by modern computers. This is particularly critical on embedded platforms that have no direct processor support for division. Basically Stein's algorithm can be described as

$$\gcd(0, v) = v \tag{6}$$

$$\gcd(u, 0) = u \tag{7}$$

$$\gcd(0, 0) = 0 \tag{8}$$

If u and v are both even, then

$$\gcd(u, v) = 2.\gcd(u/2, v/2) \tag{9}$$

If u is even and v is odd**,** then

$$\gcd(u, v) = \gcd(u/2, v) \tag{10}$$

Similarly u is odd and v is even then

$$\gcd(u, v) = \gcd(u, v/2) \tag{11}$$

If u and v are both odd and u is $\geq v$, then

$$\gcd(u, v) = \gcd((u - v)/2, v) \tag{12}$$

If both are odd and $u < v$, then

$$\gcd(u, v) = \gcd((v - u)/2, u) \tag{13}$$

For ex. gcd(0, 22) is 22 (6). Also, gcd(33, 0) is 33 (7). Similarly, gcd(21, 22) is same as gcd(21, 11) (11). Also, gcd(21, 41) is same as gcd((41-21) /2 , 21) is again same as gcd(10, 21)(13).

III. Simulation Results

A. *Euclid's Algorithm with BIST Features (4-bit Input)*

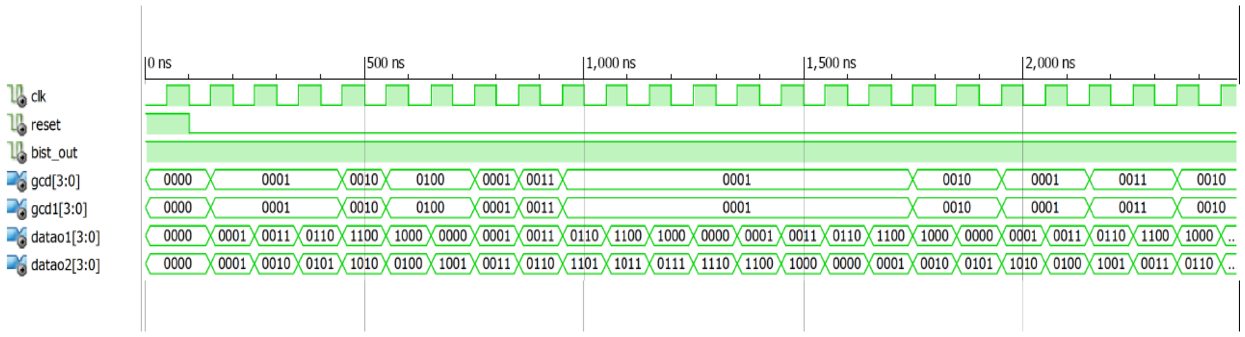

Fig. 2. Implementation of Euclid's Algorithm with BIST features with 4-bit input data

B. *Euclid's Algorithm with BIST Features(8-Bit Input)*

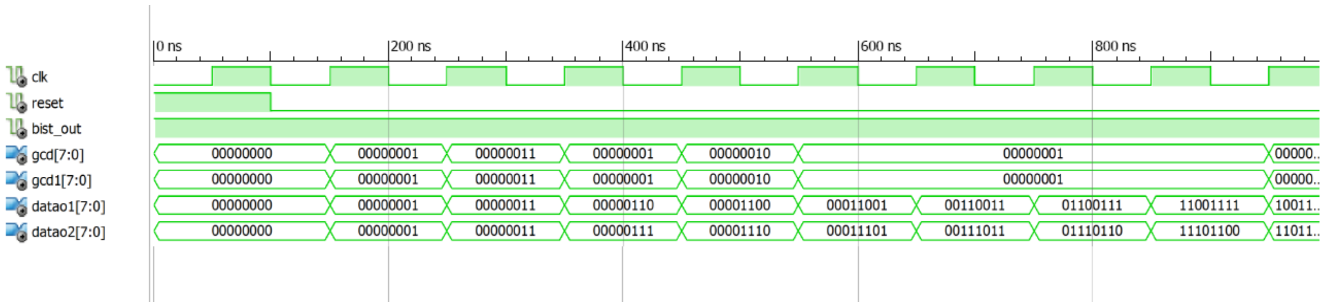

Fig. 3. Implementation of Euclid's Algorithm with BIST features with 8-bit input data

C. *Stein's Algorithm with BIST Features (4-Bit Input)*

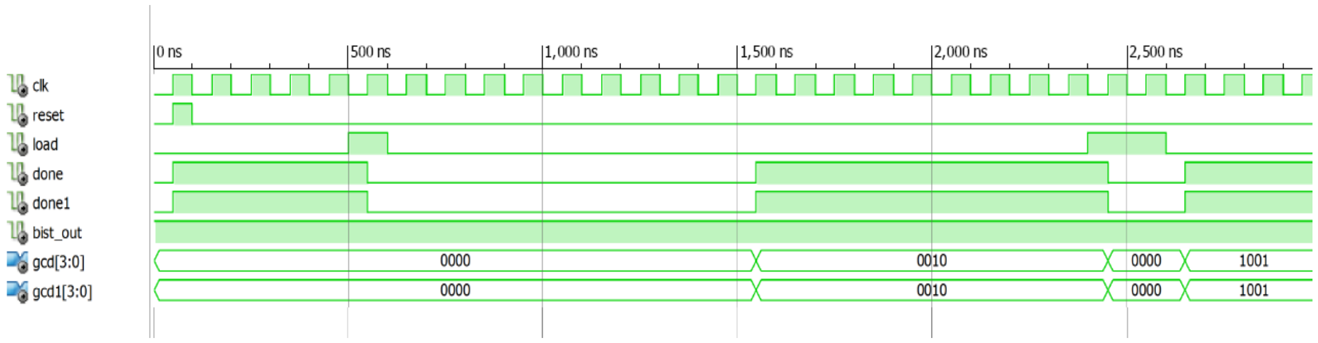

Fig. 4. Implementation of Stein's Algorithm with BIST features with 4-bit input data

D. *Stein's Algorithm with BIST Features (8-Bit Input)*

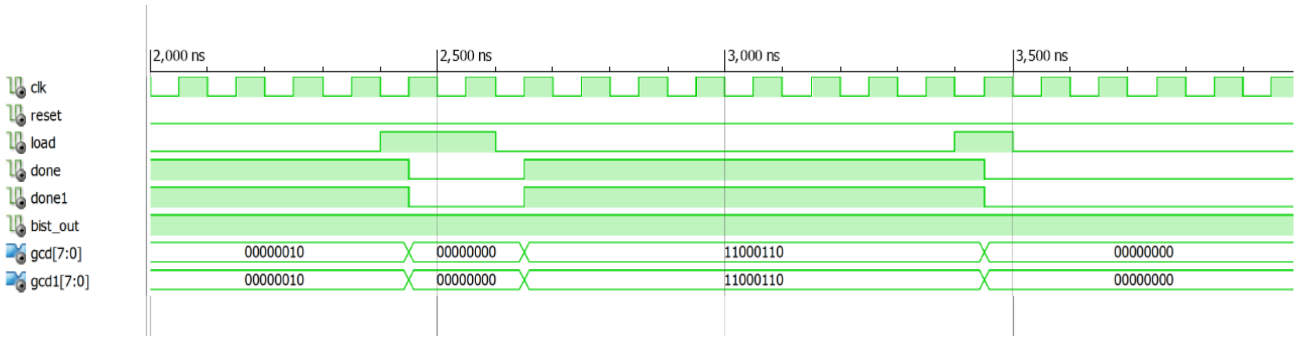

Fig. 5. Implementation of Stein's Algorithm with BIST features with 8-bit input data

Fig. 2 and Fig. 3 shown above shows the simulation results of finding gcd of two numbers taking 4-bit input and 8-bit input respectively using Euclid's algorithm with BIST technique [6]-[8].

The following are the signals that we are using in it is:

clk:- For counting how many clock cycles it is needed to get expected output.

reset:- It is for resetting the circuitry.

bist_out:- It is signal when at logic 1,which indicates that we are getting correct gcd output.

gcd:- It is the signal for gcd output which is considered to be as reference.

gcd1:- It is the signal for gcd output which is same as output of gcd signal, when bist_out = 1.

data1 and data2:- It is the signal which is having 16 bit data input for gcd calculations.

load:- It is the signal using for loading respective data input.

The data inputs are being generated continuously for infinite clock cycles. It is being achieved using Linear Feedback Shift Register(LFSR).

Fig. 4 and Fig. 5 shown above shows the simulation results of finding gcd of two numbers taking 4-bit input and 8-bit input using Stein's algorithm [5] with BIST technique.

load:- It is the signal when '1' , indicates instant at which data is being loaded and gcd of the respective loaded data's is being calculated.

done and done1:- It is the signal when logic '1' , indicates that gcd is being calculated and the output is being compared with referenced output.

Built In Self Test(BIST) is the technique which will test the circuit by itself. In both algorithms, we have taken reference processor assuming that it is giving exact expected results. Using that reference processor, we are now comparing the output results and performance of remaining processors. This indirectly saves testing time and also costs.

IV. Linear Feedback Shift Registers(LFSR)

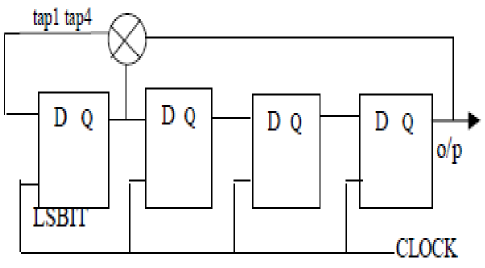

Fig. 6. Conventional LFSR

Linear Feedback Shift Register(LFSR) [4] is a circuit consisting of flip-flops connected in series with each other. The output of one flip-flop is connected to the input of the next flip flop and so on. The feedback polynomial which is also known as the characteristic polynomial is used to determine the feedback taps which in turn determines the length of the random pattern generation.

V. Euclid's Algorithm vs Stein's Algorithm

A. *Number of Look Up Tables(LUT's) Used for 4-Bit Input Data*

TABLE I: LUT'S USED FOR 4-BIT INPUT

Device	XC3S50	XC4VFX12	XC6SLX4
Euclid's without BIST	93	92	16
Euclid's with BIST	193	193	21
Stein's without BIST	37	36	34
Stein's with BIST	76	40	36

B. Number of Look Up Tables(LUT's) Used for 8-Bit Input Data

TABLE II: LUT'S USED FOR 8-BIT INPUT

Device	XC3S50	XC4VFX12	XC6SLX4
Euclid's without BIST	819	800	685
Euclid's with BIST	1613	1613	1397
Stein's without BIST	37	36	34
Stein's with BIST	76	40	36

VI. CONCLUSION

In this paper, GCD processor is being implemented using Euclid algorithm and Stein's algorithm with BIST features. In the next stage , we are comparing the performance of both algorithms in terms of no. of look up tables used separately for 4-bit and 8-bit input data and finding out the conclusion that which algorithm is working better while calculating greatest common divisor (gcd) of two non-negative integers.

After comparing performance of different devices like XC3S50 , SC4VFX12 and XC6SLX4, it can be noted that the number of look-up tables(LUT's) needed for 4-bit and 8-bit data input is less for XC6SLX4 device as compared with other devices. So, XC6SLX4 device is more efficient as compared with other devices and faster for computing greatest common divisor(gcd).

This technique is more useful in applications like Cryptography , Data security and sharing etc. and so on.

REFERENCES

[1] S. Jamuna and V. K. Agrawal, "VHDL Implementation of BIST Controller," in *Proc. of int. Conf. on Advances in Recent Technologies in Communication and Computin*, pp. 188-190, 2011.

[2] R. S. Oliveira, J. Semiao, I. C. Teixeira, M. B. Santos, and J. P. Teixeira, "On-line BIST for Performance Failure Prediction under Aging Effects in Automotive Safety-Critical Applications," *IEEE*, 2011.

[3] R. Devi, J. Singh, and M. Singh, "VHDL implementation of GCD Processor with Built in Self Test Feature," *International Journal of Computer Applications (0975 – 8887)*, vol. 25, no. 2, pp. 50-54, July 2011.

[4] P. Nayineni and S. K. Masthan, "Power optimization of BIST circuit using low power LFSR," *International Journal of Computer Trends and Technology*, vol. 2, Issue 2, pp. 5-8, 2011.

[5] G. Purdy, C. Purdy, and K. Vedantam, "Two Binary Algorithms for Calculating the Jacobi Symbol and a Fast Systolic Implementation in Hardware," *IEEE*, 2006.

[6] Z. H. Yu and L. H. Yun, "A BIST Scheme to Test Static Parameters of ADCs," *IEEE Symposium on Electrical & Electronics Engineering (EEESYM)*, 2012.

[7] N. Mukherjee, A. Pogiel, J. Rajski, and J. Tyszer, "BIST-Based Fault Diagnosis for Read-Only Memories," *IEEE transactions on computer-aided design of integrated circuits and systems*, vol. 30, no. 7, July 2011.

[8] K. Kobayashi, N. Takagi, and K. Takagi, "Fast inversion algorithm in GF(2m) suitable for implementation with a polynomial multiply instruction on GF(2)," *IET Computers & Digital Techniques Received on 25th January 2010 Revised on 1st February*, 2012.

Sachin D. Kohale received Bachelor of Engineering (B.E.) degree in Electronics from K. D. K. College of Engineering, Nagpur, Maharashtra, India in 2008. He is pursuing Master of Engineering (M.E.) in Embedded System and Computing (ESC) from G. H. Raisoni College of Engineering, Nagpur, Maharashtra, India. His research area includes designing gcd processor performance comparison of Algorithms used for designing gcd processor and also designing digital logic circuit.

Ratnaprabha W. Jasutkar received Master of Technology (M.Tech.) degree in VLSI Design from R. K. N. E. C., Nagpur, Maharashtra, India. Presently, she is pursuing Ph.D. in the field of Mixed Signal VLSI. She is also working as Assistant Professor in Computer Science and Engineering Dept. in G.H. Raisoni College of Engineering, Nagpur. Her research area includes Mixed Signal VLSI , VLSI Testing , Embedded System & Wireless Networks etc.