Generation of Random Number using Advanced Well Method Based on Dual Port Memory

P.Nishanthi[1] S.Sivaganesan[2]
[1]M.E. (VLSI design) [2]Assistant Professor
[1,2]Department of Electronics & Communication Engineering
[1,2]Kalaignar Karunanidhi Institute of Technology, Coimbatore, TN, India

Abstract— This paper presents a hardware architecture for efficient implementation of the well equidistributed long-period linear (WELL) algorithm. The design achieves a throughput of one sample per clock cycle and runs on a Xilinx FPGA device. The proposed advanced WELL architecture takes advantage of an Dual Port Ram (DPRAM) over Block Ram (BRAM). Our design is used to generate random numbers that are non-repeating in nature. The random numbers produced by our design are applied to the standard circuits to detect the faults in it. The use of Dual Port Ram (DPRAM) offers advantage like reduced access time, less occupancy of circuit board and lower power consumption. Comparative analysis between the two techniques has been made in respect to use of FPGA's internal resources, maximum operating frequency and power consumption. The simulation result obtained using VHDL coding is also presented.

Key words: Block Ram (BRAM), Dual Port Ram (DPRAM), Field Programmable Gate Array (FPGA), Well Equidistributed Long-period Linear (WELL) algorithm

I. INTRODUCTION

A. *Random Number Generator:*

A random number generator (RNG**)** is a computational or physical device designed to generate a sequence of numbers or symbols that lack any pattern, i.e. appear random. Random number generators are very useful in developing Monte Carlo-method simulations, as debugging is facilitated by the ability to run the same sequence of random numbers again by starting from the same random seed. They are also used in cryptography – so long as the seed is secret. Random numbers are useful for a variety of purposes, such as generating data encryption keys, simulating and modeling complex phenomena and for selecting random samples from larger data sets.

There are two main approaches to generate random numbers:

- Pseudo-Random Number Generators (PRNGs)
- True Random Number Generators (TRNGs).

The approaches have quite different characteristics and each has its pros and cons.

B. *Pseudo-Random Number Generator (PRNG):*

A pseudorandom number generator (PRNG), is an algorithm for generating a sequence of numbers that approximates the properties of random numbers. The sequence is not truly random in that it is completely determined by a relatively small set of initial values, called the PRNG's state. Although sequences that are closer to truly random can be generated using hardware random number generators, pseudorandom numbers are important in practice for simulations (e.g., of physical systems with the Monte Carlo method), and are central in the practice of cryptography and procedural generation. Common classes of these algorithms are linear congruential generators, Lagged Fibonacci generators, linear feedback shift registers, recent instances of pseudorandom algorithms include Blum Blum Shub, Fortuna, and the Mersenne twister.

PRNGs are efficient, meaning they can produce many numbers in a short time, and deterministic, meaning that a given sequence of numbers can be reproduced at a later date if the starting point in the sequence is known. Efficiency is a nice characteristic if your application needs many numbers, and determinism is handy if you need to replay the same sequence of numbers again at a later stage. PRNGs are typically also periodic, which means that the sequence will eventually repeat itself. While periodicity is hardly ever a desirable characteristic, modern PRNGs have a period that is so long that it can be ignored for most practical purposes.

C. *True Random Number Generator (TRNG):*

This method measures some physical phenomenon that is expected to be random and then compensates for possible biases in the measurement process. Example sources include measuring atmospheric noise, thermal noise, and other external electromagnetic and quantum phenomena. The physical phenomenon can be very simple, like the little variations in somebody's mouse movements or in the amount of time between keystrokes. Regardless of which physical phenomenon is used, the process of generating true random numbers involves identifying little, unpredictable changes in the data.

The characteristics of TRNGs are quite different from PRNGs. First, TRNGs are generally rather *inefficient* compared to PRNGs, taking considerably longer time to produce numbers. They are also *nondeterministic*, meaning that a given sequence of numbers cannot be reproduced, although the same sequence may of course occur several times by chance. TRNGs have no period. The poor efficiency and nondeterministic nature of TRNGs make them less suitable for simulation and modeling applications, which often require more data than it's feasible to generate with a TRNG.

II. EXISTING METHOD

A. *Mersenne Twister Method:*

HIGH quality random numbers are of critical importance to many scientific applications, particularly for Monte Carlo simulations. Given the advantages of high performance and reproducibility, pseudorandom number generators (PRNGs) based on linear recurrences over F2 are widely adopted in such simulations. One prevalent F2-linear PRNG is the Mersenne Twister (MT), which has very long period and good equidistribution. However, MT is also proved to have certain drawbacks. For example, one serious issue is that it is sensitive to poor initialization and can take a long time to recover from a zero-excess initial state.

B. *Uncorrelated PRNG IP Cores:*

It only achieves a throughput of one sample every two cycles and no structural details are revealed. The LFSR-160 and Tauss-113 behave particularly badly and fail multiple tests, including those do not depend on the linear structure of the generator. The integer generators MLFG and CMRG from HSPRNG (that are based on recurrence modulo a positive integer) pass all tests, but the decimal computations significantly slow them down, thus their performance/cost ratio is not attractive. The LUT-SR- 19937 also fails the two linear tests and is of about the same quality as WELL.

C. *Well Method:*

The well equidistributed long-period linear (WELL) algorithm is proposed to fix Mersenne twister method problem. Compared with MT, WELL has better equidistribution while retaining an equal period length. We propose a more resource-efficient structure that reduces the usage of BRAMs from four to two, while retaining the same throughput. The total resource used is also reduced as much as 50% compared with the original structure. We also design a software/hardware framework to parallelize its output stream based on the new structure.

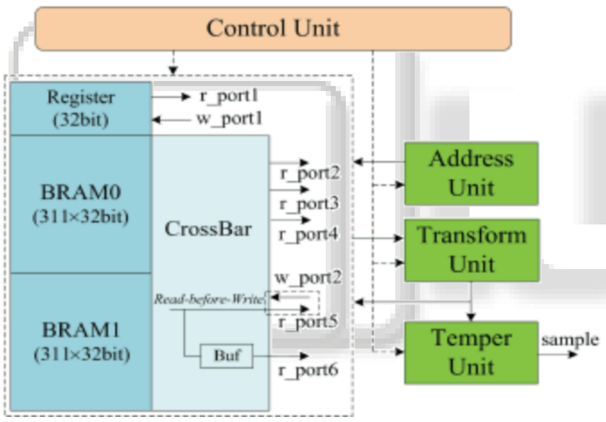


Fig. 1: Hardware architecture for WELL method

Fig. 1 shows our hardware architecture for WELL19937.It consists of five blocks: the Control Unit, the Address Unit, the Transform Unit, the Temper Unit, and a 6R/2W RAM. The core component is the RAM. The Address Unit generates appropriate R/W addresses for the RAM. The Transform Unit and the Temper Unit perform the Transform and Temper operations of the WELL algorithm, and can be fully pipelined. The Control Unit produces the control signals to coordinate the system.

1) *BRAM:*

Block RAMs are dedicated elements present inside FPGA. Usually they run parallel inside FPGA. its possible to have both single port as well as dual port block RAMs. In dual port block RAMs both the ports operate at different clock speeds.

The Block Ram (BRAM) is the embedded memory in FPGA device which use more resource in it. The BRAM can be either single port or dual port RAM. Similar to other Xilinx FPGA block RAMs, Write and Read are synchronous operations; the two ports are symmetrical and totally independent, sharing only the stored data. Each port can be configured in one of the available widths, independent of the other port. The signals connected to a block RAM primitive divide into four categories, as listed below.

1) Data Inputs and Outputs
2) Parity Inputs and Outputs, available when a data port is byte-wide or wider
3) Address inputs to select a specific memory location
4) Various control signals that manage read, write, or set/reset operations

a) Disadvantage:

The major disadvantage here is the usage of Block Ram (BRAM). The Block Ram (BRAM) is the embedded memory in FPGA which uses more resources in it. To overcome this we go for advanced WELL method where we use Dual Port Ram (DPRAM) instead of BRAM which reduce the resource usage.

III. PROPOSED METHOD

A. *Advanced Well Method:*

Here we present a hardware architecture for advanced WELL method which shown in fig 2. To overcome the disadvantage of existing method here we use the Dual Port Ram (DPRAM) instead of Block Ram (BRAM). The major advantage of Dual Port Ram (DPRAM) over Block Ram (BRAM) is that the usage of resources in FPGA is reduced which also takes less computation time and lower power consumption. The main aim of this method is to generate random numbers that are non-repeating in sequence which is to be applied in standard circuits to detect fault. The blocks used in proposed architecture are described below:

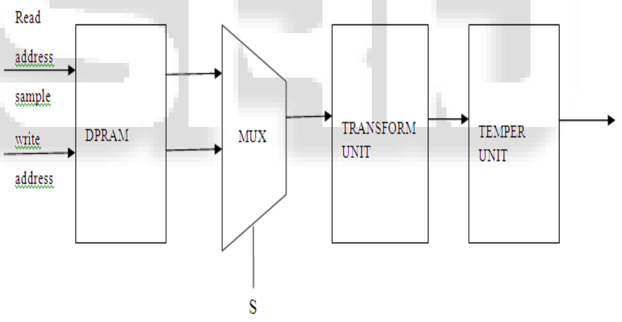


Fig. 2: Block diagram for advanced WELL method

B. *Block Functionality:*

1) *Address Unit:*

The address unit is used to generate the address for Dual Port Ram (DPRAM). The read address and write address is given to the Dual Port Ram (DPRAM) which reads the specific memory location of the RAM and will produce the data for their corresponding address.

2) *Dual Port Ram (DPRAM):*

The dual port ram (DPRAM) is implemented within the look up table (LUT). In our proposed method two BRAM is replaced by the single dual port ram (DPRAM) which have both synchronous read and write operation simultaneously. The power is given only to the used resources (LUT) in it which greatly consumes low power when compared to the BRAM.

3) *Multiplexer:*

In general mux performs the many to one operation. Depending upon the select signal it selects either read data or write data and given as input to the transform unit.

4) *Transform Unit:*

The transform unit will perform the transform operation. The structure of the transform unit consists of XOR gate and AND gate in it. The main operation of the transform unit is to perform row wise shifting in the input bit. The concat unit is used to concat the input bits. The output z4 is given as input to the temper unit. The internal structure of the transform unit is shown in fig 3.

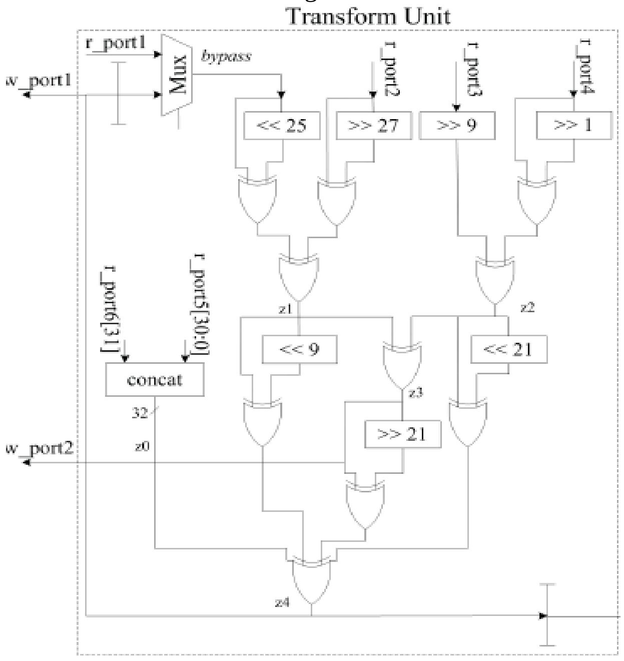


Fig. 3: Structure of the Transform Unit

5) *Temper Unit:*

The temper unit will perform the temper operation. The structure consists of XOR gate and AND gate. The main operation of the temper unit is to perform column wise shifting in the input bit(z4) and produce the final sample output. The internal structure of the temper unit is shown in fig 4.

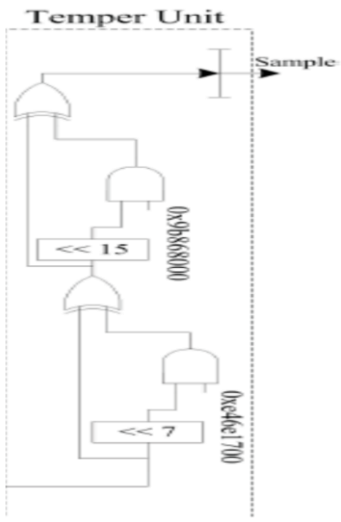
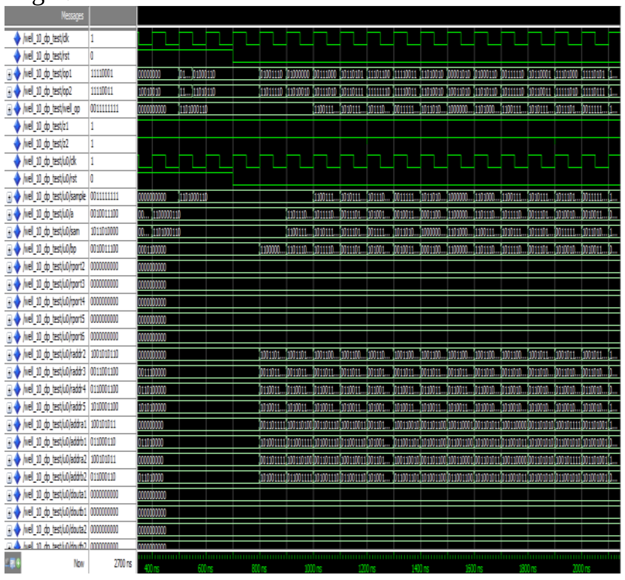


Fig. 4: Structure of the temper unit

IV. RESULT

The simulation result for the proposed architecture is shown in fig 5.The result obtained from the modelsim tool. The generated random numbers are applied to the benchmark circuits which is then performs testing to cover the maximum faults in it. The overall simulation result is shown in fig 5.



```
Messages
/well_10_dp_test/clk            1
/well_10_dp_test/rst            0
/well_10_dp_test/op1            11110001
/well_10_dp_test/op2            11110011
/well_10_dp_test/well_op        0011111111
/well_10_dp_test/z1             1
/well_10_dp_test/z2             1
/well_10_dp_test/u0/clk         1
/well_10_dp_test/u0/rst         0
/well_10_dp_test/u0/sample      0011111111
/well_10_dp_test/u0/a           0010011100
/well_10_dp_test/u0/sam         1011010000
/well_10_dp_test/u0/bp          0010011100
/well_10_dp_test/u0/rport2      0000000000
/well_10_dp_test/u0/rport3      0000000000
/well_10_dp_test/u0/rport4      0000000000
/well_10_dp_test/u0/rport5      0000000000
/well_10_dp_test/u0/rport6      0000000000
/well_10_dp_test/u0/raddr2      1001010110
/well_10_dp_test/u0/raddr3      0011001100
/well_10_dp_test/u0/raddr4      0110001100
/well_10_dp_test/u0/raddr5      1010001100
/well_10_dp_test/u0/addra1      100101011
/well_10_dp_test/u0/addrb1      011000110
/well_10_dp_test/u0/addra2      100101011
/well_10_dp_test/u0/addrb2      011000110
/well_10_dp_test/u0/douta1      0000000000
/well_10_dp_test/u0/doutb1      0000000000
/well_10_dp_test/u0/douta2      0000000000
/well_10_dp_test/u0/doutb2      0000000000
Now                             2700 ns
```

Fig. 5: simulation result of advanced WELL method

	Existing Method	**Proposed Method**
Time	8.570ns	5.187ns
Area	1,497	386
Power	206mw	136mw

Table 2: comparison table for existing and proposed method

V. CONCLUSION

Through our study we demonstrated that our proposed advanced WELL method generates random numbers that are non-repeating in sequence which achieves low area cost, low power consumption, and high quality output. Here the generated random sequences are then successfully applied to the benchmark circuits such as ISCAS85 C17 for combinational circuit testing and ISCAS89 S208 for sequential circuit testing for maximum fault coverage.

VI. FUTURE WORK

This paper is a further work of conference paper. We propose hardware architecture for advanced WELL method that includes interleaver address generator which uses bulk of circuitry to generate the address. This greatly reduces the complexity of the circuit and will achieve high performance output. The generated random sequences are then applied to the benchmark circuits for circuit testing.

REFERENCES

[1] D. B. Thomas and W. Luk, "High quality uniform random number generation using LUT optimised state-transition matrices," J. VLSI Signal Process, vol. 47, no. 1, pp. 77–92, 2012.

[2] D. B. Thomas and W. Luk, "The LUT-SR family of uniform random number generators for FPGA architectures," IEEE Trans. Very Large Scale Integrat. (VLSI) Syst., vol. 21, no. 4, pp. 761–770, Apr. 2013.

[3] F. Panneton, P. L'Ecuyer, and M. Matsumoto, "Improved long-period generators based on linear recurrences modulo 2," ACM Trans. Math.Softw., vol. 32, no. 1, pp. 1–16, Mar. 2006.

[4] H. Haramoto, M. Matsumoto, T. Nishimura, and P. L’Ecuyer, “Efficient jump ahead for F2-linear random number generators,” Inf. J. Comput., vol. 20, no. 3, pp. 385–390, 2008.
[5] I. L. Dalal and D. Stefan, “A hardware framework for the fast generation of multiple long-period random number streams,” in Proc. 16th ACM Int. Symp. FPGAs, Feb. 2008, pp. 245–254.
[6] P. L’Ecuyer and F. Panneton, “Fast random number generators based onlinear recurrences modulo 2: Overview and comparison,”in Proc. 37th Conf. Winter Simul., 2005, pp. 110–119.
[7] Uncorrelated Pseudo-Random Number Generator IP Cores, Ukalta Engineering Corporation, Edmonton, AB, Canada, 2009.
[8] Y. Li, P. Chow, J. Jiang, and M. Zhang, “Software/hardware framework for generating parallel long-period random numbers using the WELL method,” in Proc. 21st Int. Conf. Field Program. Logic Appl., Sep. 2011, pp. 110–115.