Survey on Hardware Implementation of Random Number Generators on FPGA: Theories and Experiments Analysis

Mohammed Bakiri[1] and Christophe Guyeux[2]

[1] Center for Development of Advanced Technologies, Baba Hassan, Alger, Algeria
[2] FEMTO-ST Institute, UMR CNRS 6174, University of Franche-Comte, 25000, France.
mbakiri@cdta.dz,Christophe.Guyeux@femto-st.fr

Abstract. this paper introduce a survey on PRNG on FPGA.. . .

Keywords: Random Number Generator, PRNG, TRNG, Chaotic PRNG, Cryptography, Security, FPGA

1 Introduction

1

2 Terminologies and Basic Recall

Some definitions and Symbols resume

2.1 Definitions

Definition: A random bit generator is a device or algorithm which outputs a sequence of statistically independent and unbiased binary digits

3 Random Number Generator: Theories and Classification

We can find many implementation of RNG in Software and Hardware but all can be classifies generally as PRNG, TRNG and Hybrid, but new concept has been introduce this last year's defined by parallel and chaotic generator.

3.1 PRNG

Some of the algorithms that generate pseudo random numbers are Blum Blum Shub, Inversive congruential generator, ISAAC (cipher), Lagged Fibonacci generator, Linear congruential generator, Linear feedback shift register, Mersenne twister, generalized feedback shift register (GFSR), twisted GFSR (TGFSR), Multiply-with-carry, Well-Equidistributed-Long-period Linear, Xorshift and Cellular Automata. As well as separately implementation of these structures, with use one or several of them, hybrid PRNGs can be designed. An important property of all these generators is that they are special cases of a general class of generators whose state evolves according to a (matrix) linear recurrence modulo 2 and the bits that form the output are also determined by a linear transformation modulo 2 applied to the state.

Linear Feedback Shift Register

lfsr

Blum Blum Shub

In 2012 [BBS-1] Khushboo Sewak et all, The main purpose of this paper is to study the FPGA implementation of two 16 bit PN sequence generator namely Linear Feedback Shift Register (LFSR) and Blum-Blum-Shub (BBS). The use of feedback shift register permits very fast generatio PN sequence whereas BBS method requires a number of time consuming arithmetic operation as it is based on quadratic Congruential equation. Also the performance is better in 4-bit LFSR as compared to 16-bit BBS as the time required is much more in BBS

In 2010 [BBS-2] Pedro Peris-Lopez and Enrique San Millan, Our analysis looks at the feasibility of RFID tags for supporting Cryptographically Secure Pseudorandom Number Generators (CS-PRNG) on their limited chip. Specifically, we study the implementation of the Blum-Blum-Shub (BBS) pseudorandom number generator for security levels 232 (160 bits) and 264 (512 bits) respectively, This paper makes a detailed analysis of the feasibility of implementing the BBS generator for 160 bits and 512 bits length. The implementation of the BBS algorithm is based on modular squaring, which can be computed by modular multiplication. Specifically, we study the hardware implementations. Different algorithms have been proposed to obtain the modular multiplication of two numbers (A x B mod M) [14]. for the following algorithms: 1) Classical combinational multiplier, with Barrett's reduction; 2) Classical shift and add multiplication, with Barrett's reduction; 3) Karatsuba's multiplication, with Barrett's reduction; 4) Montgomery's direct modular multiplication, iterative architecture. In particular, we analyze different architecture implementations of the BBS generator, considering the different modular multiplications discussed in the previous section. As the delay is not a differentiated factor, the choice of algorithm will generally depend on the area they occupy. In this respect, we can see that the Montgomery iterative approach produces the best results. *Mastrine Tewister*

In 2013 [MT-1] Shengfei Wu et al, In this paper, a hardware architecture for the generation of parallel long-period random numbers using MT19937 method was proposed. Most hardware implementations of MT19937 are straightforward

non-parallelized implementations of the original C-code. The Mersenne Twister Method, which is a pseudorandom number algorithm based on a matrix linear recurrence over F2, is developed by Makoto Matsumoto in 1997. In order to get the long period and good equidistribution, the Mersenne Twister is cascaded with a tempering transform to compensate for the reduced dimensionality of equidistribution, the temper is defined in the case of Mersenne Twister. We use dual-port BRAMs in FPGA for the implementation and 3 degrees parallelization will be introduced as an example.

In 2013 [MT-2] Pedro Echeverra et al, In this work we present two complete MT URNGs based on the same set of parameters, MT19937, but now designed to fulfill the above mentioned features: 1. All in hardware. 2. Capable of generating one sample per cycle. 3. Highly efficient in area and performance. The other key element in the hardware architecture is the storage element needed for the n w work area of the linear recurrence. There are two suitable options: a storage table, Fig. 2a, which is the solution adopted in previous works, and a circular buffer, the solution proposed in this work.In an FPGA, this three port table has a direct translation into two dual port tables implemented by embedded Block-RAMs. A second option is the use of a circular buffer (CB) of registers taking advantage of the fixed relationship between the indexes of the words and considering that each step of the recurrence xk is replaced by xk+n in the work area. This way, the linear recurrence (L. R. in Fig. 2b) and the buffer of registers can be considered as a circular buffer where the linear recurrence is carried out by some combinational logic between the input and the output of the buffer. Hence the architecture is simplified as no logic for the table indexes is needed.

in 2013 [MT-3] Yuan Li et al, We propose a hardware architecture for the MT19937 algorithm which achieves a throughput of 1 sample per cycle. We design a dedicated 3R/1W RAM structure for MT19937, which is the key component for the entire system to achieve the expected throughput, with little resource overhead. The overview of our proposed hardware architecture for MT19937 algorithm is illustrated in Fig. 2. It is composed of five components, i.e. the Address Unit, the Transform Unit, the Temper Unit, the Control Unit and a 3R/1W RAM. The Transform Unit and the Temper Unit correspond to the Transform and Temper processes described in (1) and (4), respectively. These two components can be fully pipelined to maximize the clock speed. The Address Unit produces the proper addresses for the ports of the 3R/1W RAM. It implements the address generation algorithm described in Fig.3. The 3R/1W RAM is the key component of the system. It is capable of providing 3 Reads and 1 Write concurrently in a single cycle. We will present its structure in details in the next sub-section. The Control Unit is responsible for generating control signals to coordinate the entire system.

In 2009 [MT-4] Xiang Tian and Khaled Benkrid, This paper presents the design and implementation of a parallel Mersenne Twister PRNG on FPGAs. It also compares the FPGA implementation to equivalent software implementations running on a multi-core CPU, together with a GPU. Generating the tempering matrix for each computing core based on the given word length, size

of working area, and process ID; Initializing the generator based on the given seed number; Generating the untempered numbers; Tempering. In our hardware implementation, step 1 and 2 are performed in software while step 3 and 4 are performed in hardware. two block RAMs are needed: one is for xl, and the other is for x. However, FPGAs are much more energy efficient. Indeed, using the same amount of energy, FPGAs can generate 37x and 35x more Mersenne Twister random samples than the CPU and the GPU, respectively.

in 2008 [MT-5] Shrutisagar Chandrasekaran and Abbes Amira, The three key functional blocks in the MT19937 architecture are the Initialise Generator (IG), Generate Numbers (GN) and Extract Number (EN). All three blocks are executed in parallel. To enable parallelisation without increasing hardware overhead, multiported block RAMs are used rather than arrays built from distributed logic. the design has beenperformed using Handel C and prototyped on the Celoxica RC1000 board.

Cellular Automata

In 2003 [CA-1] Tkacik proposed a hardware random number generator implemented on a custom IC which combines the outputs of a CA with an LFSR. A hybrid 90/150 rules an 37 bit CA was combined with a 43-bit LFSR to produce a maximal length RNG, and it was found that the LFSR and CA must be clocked at different frequencies to create a sequence of numbers that can pass all the DIEHARD tests. Then, in 2012 [CA-2] Juan C.Cerda and al, notice in Tkcide [2003] [CA-1] the combination must be clocked at different frequency to pass all NIST test, and present another combination PRNG using HCA using 90/150 and LFSR to solve this problem. The trick is XORing the last bit of HCA with the last bit of LSFR to generate 1-bit per clock cycle, and they found the best combination for a high quality of PRNG is 37-bit LFSR with 16-bit CA. then 2012 [CA-3], they compare they preview work LFSR/HCA with a SPCA 2004 [CA-4] that use 90/156 rules, and they find that even SPCA fail in one statistic test but give a better throughput than the LFSR/HCA.

In 2006 [CA-5] Leonidas Kotoulas et al, The proposed 1-d CA is based on the real time clock sequence and used for stream cipher. The authors show that by using rules based computer times sequence as year, months to seconds can generate initial state and the length of CA cells. the initial state configuration and simultaneously the length of CA cells the product of all the above numbers, namely day, month, year, hour, min and seconds was calculated. The execution time was decided to be t = x(60 - x). The first rule arises from the product of minutes by seconds. The second rule is the number of minutes divided by the number of seconds multiplied by a constant.

In 2007 [CA-6] Petre Anghelescu et al, propose a combination of two logic combinational circuit of Hybrid HCA where PRNG and block cipher for an encryption system, and the first HCA-1 use two rules 90/150 as a real-time key stream generator and the second HCA-1 use 51/153/195 rules. To select witch rules will be used by the block cipher HCA-2, the PRNG or HCA-1 generate an encrypting rules to switch the rules and that provide each cell has is own rules.

In 2010 [CA-7] Ioana Dogaru and Radu Dogaru, create an automatic software tool based on Algebraic Normal Form (ANF) representation to generate an RTL code of any hybrid CA depending on ID rules. the results show by using ANF representation $y = [K_0 xor K_1(U_1) xor K_2(U_2) xor K_3(U_3) xor K_4(U_1 * U_2) xor ... K_7(U_1 * U_2 * U_3)] xor mask$ with ID=101 and 3 neighbor are identical to Matlab results.

In 2013 [CA-8] Lakshman Raut and David H. K. Hoe, show us another stream cipher design and combination of CA and LFSR but they introduce NLFSR block based on A2U2 design to resist more for a various forms of cryptanalysis, such as correlation attacks and algebraic attacks. Where CA and NFSR are both has feedback for each other and use a LFSR counter as input random, then the key bit stream will pass to a mixer mechanism to increase the complexity for decryption.

In 2014 [CA-9] Dogaru Ioana and Dogaru Radu, introduce a comparison between two implementation of HCA as PRNG to maximize efficiency/throughput, where the first is a basic 63-bit HCA and the second is a chain of HCA(2 HCA) and they demonstrate a high ration of frequency/ares and cryptography by using a chain of HCA instead of single HCA.

3.2 TRNG

Phase-Locked Loop In 2002 [TR-1] Viktor Fischer and Milos Drutarovsky, propose a analysis about extracting randomness from the jitter of the PLL implemented on Altera FPLD. Their studies is based on detecting the jitter by the sampling of the reference clock signal (F_{CLK}) using a correlated signal synthesized in the PLL (F_{CLG}) where $F_{CLG} = F_{CLK}(K_M/K_D)$, and the maximum distance between the two clock (CLK,CLG) must be minimum $MAX(\Delta T_{min}) < \sigma_{jit}$. However they confirm in ideal environment condition and without jitter the sampled output or random is deterministic under a period of $TQ = K_D T_{CLK} = K_m T_{CLG}$, then they conclude in a real condition $\sigma_{jit} \neq 0$ the randomness is not deterministic and depending on jitter distribution where the $MAX(\Delta T_{min}) = T_{CLK} * GCD(2K_M, K_D)/4K_M$.

In 2006 [TR-2] Martin Simka et al, The authors demonstrate by taking 2002 [TR-1] as model that by combined more than one PLL even parallel or series, can increase significantly sensitivity on the jitter $S = F_{CLK} MAX(\Delta T_{min})$ and the output-bit of the generator compared to the use of one PLL. The configuration of multiple PLL are based on input/output length, CVO frequency and MUL/DIV factors (K_M/K_D).In 2010 [TR-3], 2011 [TR-4] Martin Simka et al test the impact of the the change on operation condition environment as temperature of an PLL and illustrate that with low bandwidth of PFF cause a higher number of the critical samples, decreases the output jitter and thus increase the tracking jitter. As in application, in 2008 [TR-5] Michal Varchola et al explore embedded system application of TRNG based PLL to extract randomness from the jitter and propose two version where the slower 40kbps can pass the tests.

Inverter ring oxialltor In 2004 [TR-6] Paul Kohlbrenner et Kris Gaj and In 2009 [TR-7] Cristian Klein et al, propose a TRNG based on two ring oscillators

clocked by different clock generated by an internal PLL on FPGA, and with low area implement by only one CLB slice. the authors also extract the jitter of the 2 RO using a simpler and eliminate any correlation between successive bits.

In 2003 [TR-8] K.H.Tsoi et al, propose a Hybrid implementation on FPGA of TRNG based on RO and PRNG based on BBS generators and with high operation frequency of 400Mhz. However they generate an off-chip low frequency based resistor and capacitors and it was notice they implement BBS using ALU structure that satisfies the mathematical model as squaring and modulo operation which will perform the clock cycle of each operation by $(4.5 * n^2 + n)$.

FIGARO ring oxialltor IN 2007 [TR-9] Markus Dichtl et Jovan Dj.Golic, propose a new approach that can replace RO based on inverters and prove higher randomness using XORE combination between Fibonacci (FIRO) and Galois ring oscillators (GARO). the main key consists of a number, r, of inverters connected in a cascade together with a number of XOR logic gates forming a feedback in an analogous way where the feedback polynomial form is $f(x) = (1 + x)h(x)$ where $h(1) = 1$ and the result show withe the new method can achieve a stable state less than classical RO.

Self-timed ring STR In [2013] [2011] [2013] [TR-10,11,12]Abdelkarim Cherkaoui et al, the authors propose another alternative more robust to environment (power, temperature) than RO based inverter and based on Self-Timed Ring (STR). The SRT approach consist of a ripple of L stage of FIFO as a ring $(C_i)_{1 \leq i \leq L}$ with a phase of $\Delta\varphi = T/2L$, and extract jitter of each oxillator stage using two asynchronous handshaking protocol as even that can be (taken or bubble). However and first, to have the randomness bits, that outputs $(S_i)_{1 \leq i \leq L}$ events will be samples using a flip-flop by the main clock and the result will be combined by a XOR operation $\psi = s_1 \oplus s_2 \oplus ... \oplus s_L$. Secondly, the authors suggest that to avoid the limitation frequency of the STR by the long period delay, the maximum frequency is achieve when the propagation delay (forward and reverse static delay) is near to ring accuracy (N of token and bubble) and $N_T/N_B \cong D_{ff}/_D rr \simeq 1$.

Metastability In [2008] [TR-13] Ihor Vasyltsov et al, the authors present a studies of using Metastability phenomen as a entropy source generated by 5 IRO stage. They claim that by implementing the inverter as loop ring and using a Control Clock Generator to switch the connectivity between the IRO stages flowing two mode (MS, Generation), the output voltage converges to metastability level and stays longer than using bi-stable circuit (Flio-Flop) causing a high entropy. However, the authors wan to estimate the robustness of the system after applying the sampling process in a different process and environment variation modes using CMOS process and FPGA, and they find that it must added another stage for a higher quality output as decreasing the operation rate, applying a Von-Neumann post-processing and influences the loads (RC parasitic) of the last inverter, when it was noted that just post-processing is used in FPGA .

In 2011 [TR-14] Mehrdad Majzoobi et al, The authors propose a TRNG using the metastability of the flip-flop when there is a violation in setup/hold time. the system is based on closed-loop feedback mechanism for auto-adjustment on delay Δ controlled by the Programmable delay lines (PDLs) stage based on

LUT to avoid violation and maintain the metastability,, however the system use at-speed monitor to keep tracking the output bit probability and proportional-integral (PI) controller to decides to add/subtract the delay difference ($\Delta \rightarrow 0$). The probability of the output is $ProbOut = 1 = Q(\Delta/sigma)$ where $Q(x) = 1/\sqrt{4\Pi}\int_x{}^{\infty} expr(-u^2/2)du$, where the updated/corrected delay difference is the difference between the bias/skew caused by the routing asymmetric with the delay induce by the environment condition and the correct delay injected by the PDL ($\Delta = \Delta_p + \Delta_b - \Delta_f$). A revision version proposed by Donggeon Lee [2014] [TR-15], to analyze the probability and maintain metastability state for a long period to avoid the deterministic state. however they use an extrat hardware resource as memory for storing the outputs and use Hamming weight to calculate the probability bits histories.

3.3 Chaotic Rundom Number Generator

Logistic Based on earlier study in In 2011 [CH-1] Pawel Dabal and Ryszard Pelka, implement two version of generator using XSG tool based on logistic mapping equation but distribute the equation $X_{t+1} = r * X_n * (1 - X_n)$. Then in 2012 [CA-2] propose a studies of HW implementation in FPGA of different chaos system as logistic and Hnon mapping and Frequency depended negative resistor FNDR.

In 2014 [CH-3] Pawel Dabal and Ryszard Pelka, The authors propose a study of an fast Pipeline PRNG based on chaotic logistic map,and to solve the problem of the short cycle, distortion and correlation, the authors implement two version of PRNG based on simple logistic map equation using pipeline processing to have a a high operation frequency and the ability to generate sequence at different start point. The first is logLUT based on on LUT blocs and high-speed carry line of the FPGA and the second is Log DSP that use directly DSP of FPGA, however they add delays to ensure parallel sequence generation and a complex initial sequence used for a better NIST test results. As results, many configuration and tests based on delays and precision of PRNG applied to have the most combination of area, throutput and chaotic random outputs.

In 2013 [CH-4] Lahcene Merah and all, demonstrate by mixing to chaotic map they can increase the security against plaintex attacks, by coupling a chaotic encryption system (ENS) based on 2-D Hnon map used to generate the chaotic sequence, and control system (CRS) based on 1-D logistic map to control a multiplexer to choose the output of ENS according to the value generated by the logistic map by XORing the MSB of 32-bit of CSR with his it neighbor LSB. The results verified a good autocorrelation, sensitivity to initial parameters. However to increase NIST test and to resist more for the attacks, they process all the outputs sequence of the system in to a logic circuit that Xor the output CSR with the output system sequence following some rules.

In 2013 [CH-5] Hariprasad and NagaDeepa, the authors demonstrating using the reseeding technique to avoid non-lineair chaotic PRNG as short-period problems, and that by mixing Reseeding module with a non-linear chaotic logistic map (CLM) using a vector mixer module based on auxiliary linear generator

ALG. For each fixed point condition by comparing the X_t and X_{t+1} sequence of the CLM it increase the reseeding period until it reached, and then pass the result X_{t+1} sequence to vector mixing to generate the final output bu XORing it with the outputs Y_{t+1} of ALG , $OUT_{t+1} = (X_{t+1}[1:31])XOR(Y_{t+1}[1:31])$.

Non-Linear Chaotic Dynamic Du to degeneration phenomenon and for a high quality of chaotic of PRNG, In 2010 [CH-6] Ziqi Zhu and Hanping Hu present a new high efficiency dynamic nonlinear transform arithmetic DNT. A chaos system based on three DNT process in a parallel structure that transform input 3 times and improve a high cycle-length and distribution output sequence, and that of each DNT module initiate his 2x256 code book and obtain the binary input sequence, then transformed and look up it using inputs as parameters $B_{i+1} = Bi(C(w)R(q))$.

Spationtemporal Chaos In 2009 [CH-7] Yaobin Mao et all, implement a new parallel PRNG based on digitized spatio-temporal chaos map using coupled mapped lattice (CML) model. To achieve a high operation speed, they first deal with continuous domain with digitazed all operand to be suited for HW implementation by using and modifying a bi-directional coupled chaotic map lattice to a finite integer set. then second and to avoid finite precision chaotic map problem, they compute only the significant bit is subject to output and randomly select initial value for each lattice.

Fibonacci post-processing In 2013 [CH-8] Abhinav S. Mansingka et al, present a post processing based on Fibonacci series based on LFSR for non-autonmous signum hyperchaotic PRNG. This paper presents a hardware implementation of a robust non-autonomous 4-D hyperchaotic-based PRNG driven by a 256-bit LFSR. The post processing is based on two loop feedback using as first a fixed 1-bit static rotation to suppress the short-term predictability, then the second is based on a variable rotation controlled using Fibonacci series of K-bit to enhances differential sensitivity if there is a change at any bit when the other bit propagate during M Cycle, $M = (i : K_i < 64, K_{i+1}, iEZ^+)$.

4 Rundom Number Generator:Statistic Test

DieHard
NIST
FIPS
TestU01
AIS

5 Rundom Number Generator: Cryptography Secure

Cryptography Secure

RNG	Algorithm	DieHard	NIST	FIPS
PRNG	LFSR	[1]	[2]	[3]
	BBS	[1]	[2]	[3]
	MT	[1]	[2]	[3]
	CA	[1]	[2]	[3]
TRNG	LFSR	[1]	[2]	[3]
	IRO	[1]	[2]	[3]
	FIGARO	[1]	[2]	[3]
	STR	[1]	[2]	[3]
	MS	[1]	[2]	[3]
Chaotic	Logistic	[1]	[2]	[3]
	DNT	[1]	[2]	[3]
	Spationtemporal	[1]	[2]	[3]
	Fibonacci	[1]	[2]	[3]
	CI	[1]	[2]	[3]

6 Rundom Number Generator: Hardware Implementation

Hardware Implementation

7 Conclusion

Conclusion

8 References

9 Reference

References

1. [CH-1] Pawel Dabal, Ryszard Pelka. : A Chaos-Based Pseudo-Random Bit Generator Implemented in FPGA Device. In: (2011)
2. [CH-2] Pawel Dabal et all.:FPGA Implementation of Chaotic Pseudo-Random Bit Generators. In: (2012)
3. [CH-3] Pawel Dabal, Ryszard Pelka. :A Study on Fast Pipelined Pseudo-Random Number Generator Based on Chaotic Logistic Map. In: 2014)
4. [CH-4] Lahcene Merah et all, :Coupling Two Chaotic Systems in Order to Increasing the Security of a Communication System - Study and Real Time FPGA Implementation. In (2013)
5. [CH-5] Hariprasad NagaDeepa. Ch. :FPGA Implementation of A Cryptography Technology Using Pseudo Random Number Generator. In: (2013)
6. [CH-6] Ziqi Zhu and Hanping Hu. :A Dynamic Nonlinear Transform Arithmetic for Improving the Properties Chaos-based PRNG*. In (2010)
7. [CH-7] Yaobin Mao et all. :Design and FPGA Implementation of a Pseudo-Random Bit Sequence Generator Using Spatiotemporal Chaos. In: (2009)

8. [CH-8] Abhinav S. Mansingka. :Fibonacci-based Hardware Post-Processing for Non-Autonomous Signum Hyperchaotic System. In: (2013)
9. Viktor Fischer and Milos Drutarovsky. [TR-1] (2002)
10. Martin Simka et al,.:In: (2006) [TR-2]
11. Martin Simka et al,.:In: (2010) [TR-3]
12. Martin Simka et al,.:In: (2011) [TR-4]
13. Michal Varchola et al,.:In: (2008) [TR-5]
14. Paul Kohlbrenner et Kris Gaj .:In (2004) [TR-6]
15. Cristian Klein et al.:In: (2009) [TR-7] ,
16. K.H.Tsoi et al,.:In: (2003) [TR-8]
17. Markus Dichtl et Jovan Dj.Golic,.:In: (2007) [TR-9]
18. Abdelkarim Cherkaoui et al,.:In: (2013) [TR-10]
19. Abdelkarim Cherkaoui et al,.:In: (2011) [TR-11]
20. Abdelkarim Cherkaoui et al,.:In: (2013) [TR-12]
21. Ihor Vasyltsov et al.:In: (2008) [TR-13]
22. Mehrdad Majzoobi et al.:In: (2011) [TR-14]
23. Donggeon Lee,.In: (2014) [TR-15]
24. Tkacik.:In: (2003) [CA-1]
25. Juan C.Cerda and al.:in: (2012) [CA-2]
26. Juan C.Cerda and al.:.In: (2012) [CA-3]
27. Guan et al.:.In: (2004) [CA-4]
28. Leonidas Kotoulas et al,:.In: (2006) [CA-5]
29. Petre Anghelescu et al,:.In: (2007) [CA-6]
30. Ioana Dogaru and Radu Dogaru,:.In: (2010) [CA-7]
31. Lakshman Raut and David H. K. Hoe,:.In: (2013) [CA-8]
32. Dogaru Ioana and Dogaru Radu. :. In: (2014) [CA-9]
33. Shengfei Wu et al .In: (2013) [MT-1]
34. Pedro Echeverra et al.In: (2013) [MT-2]
35. Yuan Li et al.in: (2013) [MT-3]
36. Xiang Tian and Khaled Benkrid.In: (2009) [MT-4]
37. Shrutisagar Chandrasekaran and Abbes Amira.in: (2008) [MT-5]
38. Khushboo Sewak et all. In: (2012) [BBS-1]
39. Pedro Peris-Lopez and Enrique San Millan. In: (2010) [BBS-2]