ISSN: 2277-9655
Impact Factor: 4.116
CODEN: IJESS7

IJESRT
INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

IMPLEMENTATION OF A HIGHLY EFFICIENT NOVEL FREQUENCY DOMAIN SNR HARDWARE USING XILINX FPGAs

Ranganadh Narayanam[*1] **& SSSP Rao**[2]
[*1]Assistant professor, EECE, ITS, IFHE – Hyderabad
[2]Adjunct professor, EECE, ITS, IFHE – Hyderabad

DOI: 10.5281/zenodo.1116789

ABSTRACT
Now a days at the receiving end in the applications of Cognitive Radio, Tele medicine, wireless networking, and in receiving system of multiple channels Electro Encephalo Gram (EEG) collected data, Wireless Brain Machine Interfacing Radio Frequency (RF) wireless data receiver system, and in RF Radar communication systems receiver's noise performance is very critical. For noise performance Signal to Noise Ratio (SNR) is a Key parameter. SNR has to be calculated right after the receiving of the data, or at the intermediary stages of the Digital Signal Processing of the system. In many of such systems one of the most important approaches is to have a dedicated hardware at the receiving end, where a DSP processor which processes the required data processing. In these Dedicated Digital Hardware systems SNR Hardware plays a critical role. In this research we have implemented Digital Hardware for a highly efficient Novel Signal to Noise Ratio (SNR) mechanism & it is a frequency domain SNR. Due to SNR hardware requirement in many advanced applications, and keeping in mind of the novel formula's efficiency, and as it has never been implemented in hardware we have got motivated towards implementing this SNR mechanism onto Xilinx Field Programmable Gate Arrays, especially to find its speed of operation and compatibility for the current ongoing technologies. In this research work we have developed two different types of architectures a) Sequential data processing architectures b) Parallel data processing architectures. During this hardware design we have developed a novel Full Adder (Faster Full Adder) for addition digital circuit, which makes the Ripple Carry Addition (RCA) process much faster than regular RCA process (Faster RCA). Parallel architectures are working faster than sequential architectures both in terms of total delay and clock speed, at the expense of somewhat more hardware and power utilization. Both architectures are working at ***high GHz*** rates than current real-time digital technologies sampling rates and system clock speeds, and hence are very useful for all ongoing real time technologies. These speeds are highly useful even for future real-time technologies. We did this implementation on Xilinx Artix-7 FPGAs using Xilinx Vivado 2015.2 Design suite. Verilog Hardware Description Language (HDL) is used for programming this hardware.

KEYWORDS: Digital Hardware, SNR, RCA, Artix-7, FPGA, Xilinx, Verilog HDL.

I. INTRODUCTION
Finding Signal-to-Noise Ratio (SNR) is most important in most DSP applications such as in RF communications, in Speech Processing applications of Voice Activity Detection (VAD) [13], in biomedical applications – EEG de-noising [10,11,12]; Voice Activity Detection (VAD) in Speech Auditory Brainstem Responses [10,11,12], Wireless communications applications, RF IC Design for Brain Machine Interfacing.

In EEG data collections, the noise is very high when compared to the signals of interest, in such cases SNR value of the signal has to be very high [1]. SNR calculations play essential role in telemedicine satellite systems [15], wireless sensor networks [14].

The noise performance and hence the SNR is most important in Radio receiver. SNR is of prime importance in simple broadcast receivers, in cellular/wireless communications, fixed or mobile radio communications, two way radio communication systems, satellite radio and plenty more applications [2, 4]. In radio receivers, the overall SNR performance depends on the front end RF stage. In this noise introduced by first RF amplifier will be added to the noise of the second amplifier and so on. So, in these stages finding SNR is most important to proceed further [2, 4].

In Real Time hardware implementations of Cognitive Radio applications also require SNR calculations [3]. In RF DSP in Digital down Conversion (DDC), DDC has advantages in applications with large, varying, IF frequencies and where a good signal-to-noise ratio is required, where we need to calculate SNR [3].

Like this SNR plays vital role in almost all DSP application systems. In this research we have implemented a Digital Hardware for an efficient SNR mechanism [5, 6], which is found to be most efficiently worked for Auditory EEG signals for finding Voice Detection (VAD), but this we can apply to any of the Digital Signal Processing (DSP) systems. This mechanism requires FFT frequency spectral samples of input noisy signal.

The motivational causes of this hardware implementation are: (a) This Novel SNR mechanism was never implemented in hardware, and to our best knowledge our design is the first time hardware implementation; (b) Novel SNR's efficient performance; (c) To know its speed of operation to find whether it is compatible to ongoing technologies speeds, if compatible, to which ongoing technologies it is compatible (d) Due to SNR essential requirement in many applications and so on are some important motivating causes.

In this hardware design we developed (a) Sequential data processing architectures (b) Parallel data processing architectures. We observed both architectures are working at ***high GHz*** speeds which are much faster than all the ongoing real-time technologies ADC speeds, and clock rates of systems; and also all FFT processor speeds [7, 8, 9]. So, our design is compatible for all ongoing real-time technologies and also useful for future technologies. We have implemented on Xilinx Artix-7 FPGAs using Xilinx Vivado 2015.2 Design suite. We used Verilog HDL as the programming language.

The description of this paper is organized in the following way:
Section I: Introduction
Section II: The Novel SNR mechanism of interest in the research of this paper
Section III: Digital Hardware architectures for RCA (our own Novel Faster RCA), Booth's multiplication
Section IV: Digital Hardware architectures sequential data processing
Section V: Digital Hardware architectures for parallel data processing
Section VI: Result Analysis
Section VII: Conclusion of this research.
Section VIII: Design considerations.
Section IX: Acknowledgements

II. NOVEL SNR MECHANISM: [5,6]

In this research we have designed and implemented a Novel SNR Mechanism. This is frequency domain SNR formula. This approach was found to be highly useful for Voice Activity Detection in Speech Auditory Brainstem Responses [5, 6, 12]. But it can be highly useful in plenty of different types of DSP applications. This SNR is based on Spectral subtraction method. This is formulated as follows

$$A = \sum_{k=0}^{n-1}(X[k] \times S[k])$$

$$B = \sum_{k=0}^{n-1} S[k]$$

$$C = \sum_{k=0}^{n-1}(X[k] \times (1 - S[k]))$$

$$D = \sum_{k=0}^{n-1}(1 - S[k])$$

$$\mathrm{SNRPVD}(X,\ S) = (A/B)/\ (C/D)$$

SNRPVD – Signal to Noise Ratio Peak Valley Difference (SNRPVD) Detection Ratio specifying the SNR value of the given signal. This has to be taken logarithm so that we get the result in dB values. In this formulation the X is the frequency spectrum samples of the given noisy signal, for example from FFT of the noisy signal. Then S is a Vector of same number of samples as X. This vector is designed in such a way that, the locations where spectral peaks of the X are expected are kept 1, and remaining locations we can keep small fractional values just higher than 0. In ideal conditions the S vector is the most ideal noise free signal frequency expected spectrum of X. So, we can keep all 0's in the remaining locations of the S vector. But in real time conditions, the noisy signal's spectral energy distributes to small extant into the surrounding locations, in addition to the original peak locations. So, we can keep some small factional values, for example values between "0 & 0.2", instead of suppressing the remaining locations entirely to "0". The S vector can be anything basing on the frequency spectrum of the signal. In reality this formulation finds the similarity between the idealistic expected signal spectrum (S vector); and the real time noisy signal spectrum (X). The SNR value of the real time noisy signal X can be found by using the above specified formula.

S vector frequency location calculation formula.

Frequency location = [(Sampling Frequency / Number of Samples) (frequency for which we need to find the location)] + 2.

III. HARDWARE DESIGN OF BASIC MULTIPLICATION AND ADDITION ARITHMETIC UNITS

Design methodology
We have designed & implemented this SNR formulation in Digital Hardware using Verilog Hardware Description Language, on Xilinx Artix-7 board, using Xilinx Vivado 2015.2 tool. For this we designed 32-bit signed Multiplication arithmetic unit – Booth's multiplication, 32-bit signed addition RCA unit. We have developed a novel Full adder which is faster than regular Full adder – Faster Full adder, using which we developed a much faster RCA – Faster RCA. Then we have designed sequential architectures for the formulas A,B,C,D using these 32-bit multiplication and 32-bit addition units. We have designed the entire sequential system of A,B,C, D as two separate systems one by using RCA, one more by using Faster RCA. To improve the speed of the design, we have developed parallel architectures for all the above A, B, C, D formulas using 32-Multiplication arithmetic units, 32-bit addition RCA, Faster RCA units. The architectures are being discussed below.

Multiplication Hardware

Booth's multiplication algorithm
Architecture: We have developed hardware architecture for Booth's algorithm for two 32-bit signed numbers Multiplication, in Figure 1. In this architecture there is a 32-bit Adder/Subtractor unit (ADD/SUB UNIT), a 65 bit shift register, control logic, and a 5-bit down counter. At the beginning on reset "rst" the Q-1 bit ie shift register [0] bit, and from bits 33 to 64 bits in the shift register and the 1-32 bits are cleared to zero. When "load1" signal is generated by the control logic for a clock cycle period of time, the M & Q registers are loaded into proper locations: the multiplier Q is stored in the shift register from shift register [1] to shift register [32] bits of the shift register; the multiplicand is stored in the register M. The down counter register is loaded with 31 on reset "rst" signal. The Control logic observes both the least significant bit of Q: Q0 and Q-1 of the shift register together, then if [0, 1] it generates the ADD signal to the ADD/SUB unit. The Multiplicand M and A part of the shift register are added (A+M) using the ADD/SUB unit ; or if [1, 0] it generates SUB signal and then the ADD/SUB unit subtracts M from A (A – M). The output result of ADD/SUB unit is saved in A part of the shift register, while "load2" signal is asserted by the control logic for a clock period of time. Then in the next clock cycle the control logic generates "shift_right" signal then the shift register shifts {A,Q,Q-1} together towards the right by 1 bit. If [Q0, Q-1] is [1,1] or [0,0] then a "shift_right" signal is generated in that clock cycle then {A,Q,Q-1} shifts towards right by 1 bit. At each step when the "shift_right" signal is generated the down counter is decremented by 1, and every time the counter value is verified and if not 0, it continues doing the above process. When the counter value is found to be 0, the next clock cycle in control logic generates "finish" signal (at the same time it clears all its control signals) indicating the multiplication product is ready. Then the 64 bit result can be found in the 64 bit shift register {A, Q}. The right shift operation is arithmetic signed shifting, maintaining the sign of the most significant bit as it is, even while shifting right also.

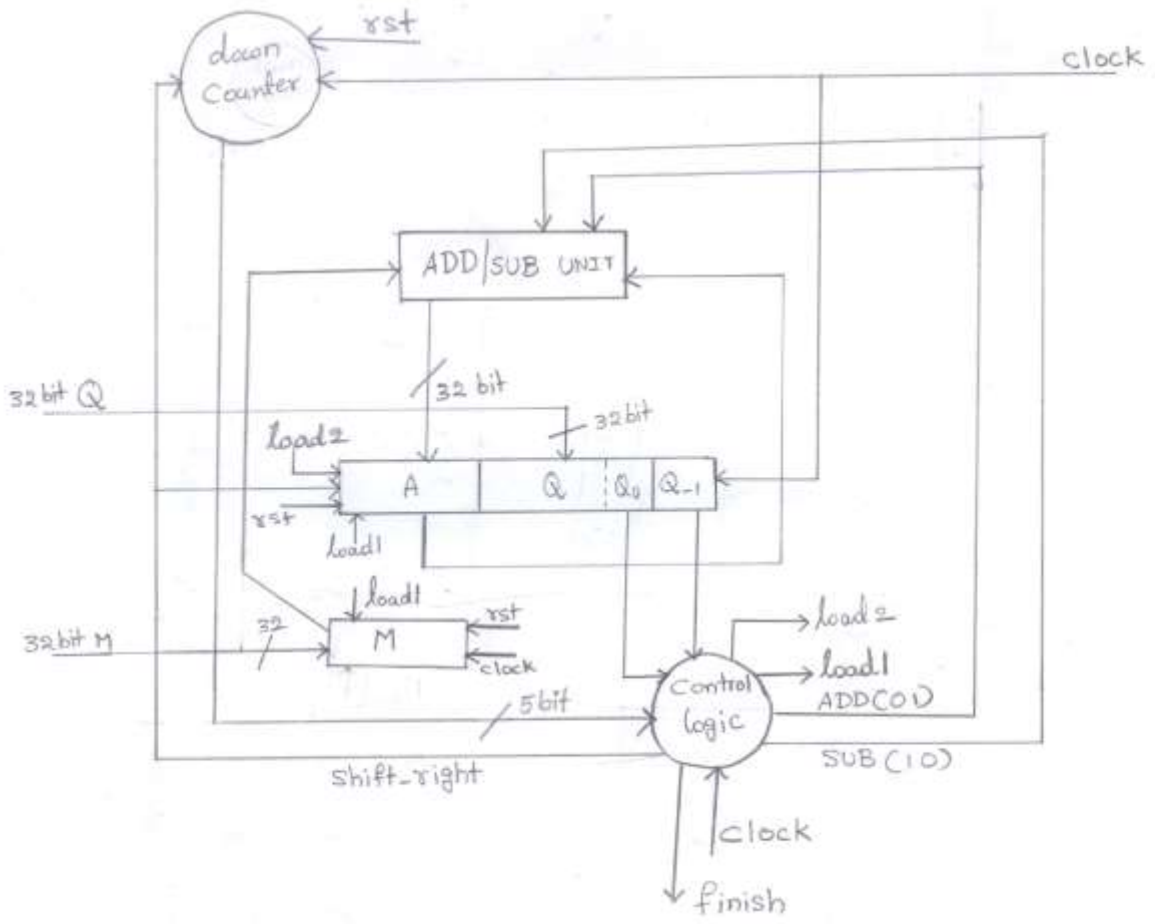

Figure 1 the architectural block diagram of the 32-bit Booth's multiplication algorithm

Addition Hardware

Design of a Novel RCA unit: Faster RCA
In our research we require 32-bit signed addition process. In this we have used Ripple Carry Adder (RCA). This RCA we have designed by using two approaches
a) U
sing regular full adders – 32 bit RCA.
b) U
sing a Novel Full Adder we developed, which speeds up the RCA process: Faster RCA.

Development of a Novel Full Adder – Faster Full Adder
The Novel full adder we have developed is given in the following Figure 2. When compared to the regular full adder, this faster full adder is taking less total delay for the operation and is faster but at the expense of additional hardware. We have designed RCA by using Full adder – "RCA"; and also by using Faster Full adder, which gave us much faster Ripple Carry Adder when compared to "RCA" – "Faster RCA".

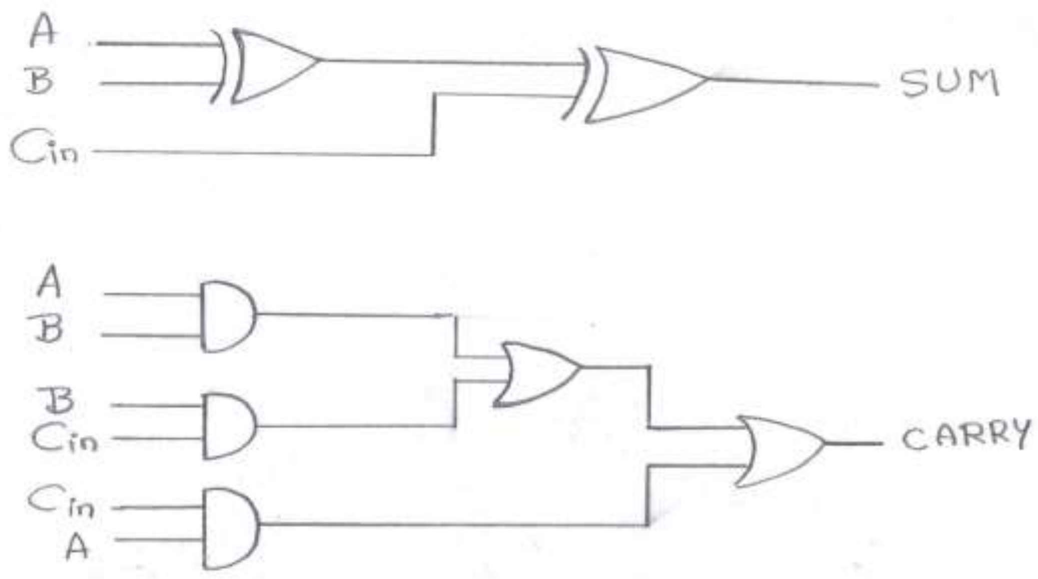

Figure 2 The Novel faster full adder

In the above figure, the A & B are input bits, and Cin is input carry bit; SUM is sum result bit of the inputs along with input carry; CARRY is output carry bit. A 32 bit signed RCA is designed using 32 regular full adders - RCA. One more 32-bit signed RCA is designed using 32 faster full adders – Faster RCA.

IV. SEQUENTIAL ARCHITECTURES

Here we are explaining sequential architectures for the formulas A & B, and the architectures for C & D are similar to A & B respectively.

Architecture For The Formula B: Repetitive Adder

$$B = \sum_{k=0}^{n-1} S[k]$$

The formula says it requires repetitive addition of n number of samples of S vector. The S vector is also of the same size of the FFT frequency spectrum of the noisy signal X. Here in this design we have designed using 64 samples. So, we have to use the 32-bit signed adder 64 number of times. In this we have designed this using both RCA and our novel "Faster RCA". This hardware unit for formula B, we called it as "Repetitive Adder".

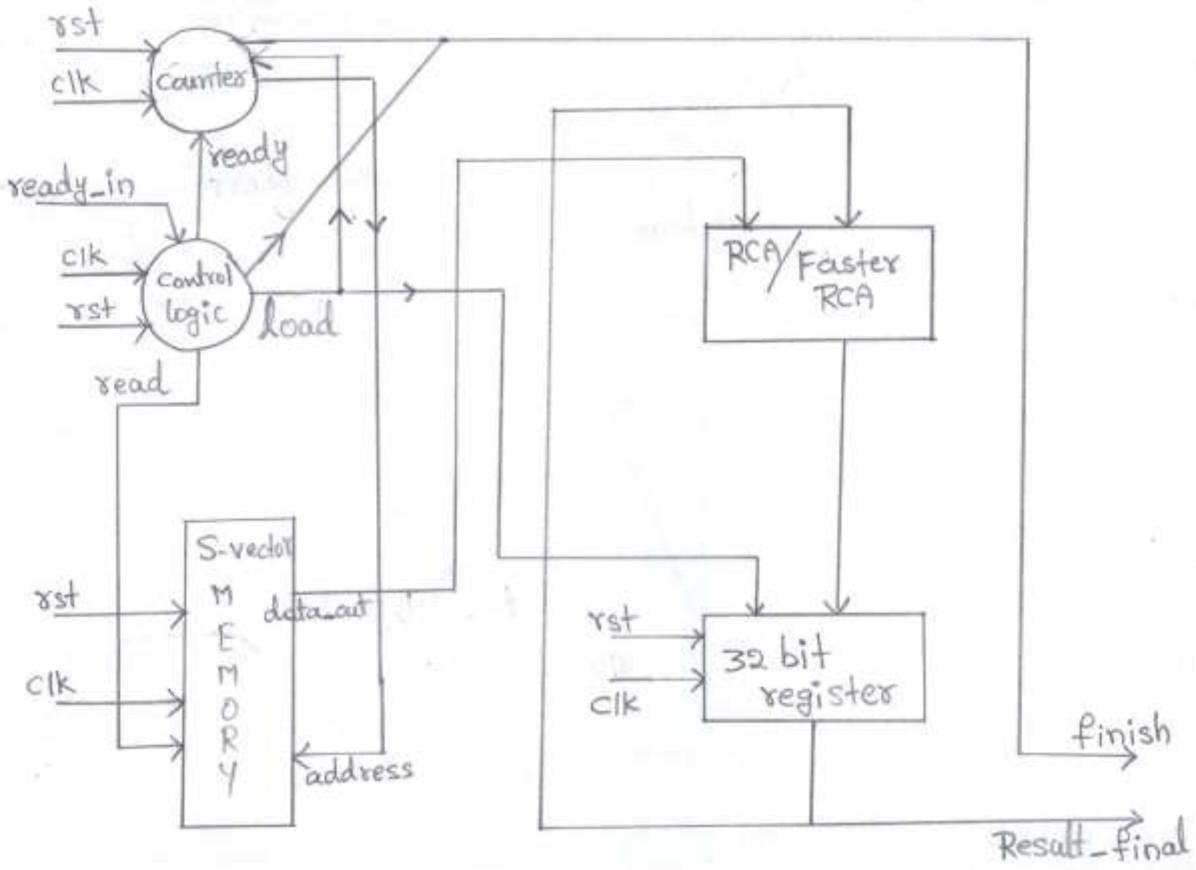

Figure 3 sequential repetitive adder architecture block diagram for the addition of 64 S vector samples of each 32 bits.

In this architecture, given in Figure 3, we are having

a) A control logic
b) A 6 bit up counter
c) A Memory block for 64 samples of S vector each of 32-bit size signed numbers
d) 32-bit signed RCA (we have evaluated both by keeping RCA, and faster RCA)
e) 32-bit register

The control logic generates the essential control signals such as read, ready, finish signals. A clock signal “clk” is common for all the system, which goes to control logic, counter, and the 32-bit register. On reset signal “rst”, the Memory block is loaded with proper data of 32-bit signed numbers of S vector, the counter & 32-bit register are loaded with 0, “ready”, “read”, “finish”, “load” signals are cleared to 0. When the external input signal “ready_in” is asserted, the control logic operates the system as long as it is “on” at positive edge of the clock signal “clk”, and generates the control signals basing on the conditions. The control logic generates the “ready” signal to the counter, and then “read” signal to the memory block. Then counter’s value is fed as address location to the memory from which to be read. Then in the clock cycle the output of the adder is ready, and in the next clock cycle the control logic generates “load” signal for a clock cycle period of time and the result is loaded into the 32-bit register. Then after each “load” signal the counter is incremented in the next clock cycle. So, at each alternative clock cycle after the first generation of “read” signal, and if as long as the “ready” signal is on at the positive edge of the clock, the “load” signal is generated and the output result of the adder is loaded into 32-bit register. The counter continues to count, and the data from the memory is read out basing on the counter value as “address” and the output of memory “data_out” is fed to the RCA as input. As long as the “ready” signal is asserted, the control logic continues to generate “load” signal for the 32-bit register. If this load signal is asserted at the positive edge of the clock, the output of the RCA is loaded into the 32-bit register. This register’s output is feedback to the RCA as second input. The control logic observes the counter value, and if observes the value of the counter as 63, and after “load” signal then all the addition process is finished, and it asserts the “finish” signal, and it clears “read”, “ready” , “load” signals; and counter values to 0. When the "finish" signal is asserted at the positive edge of the clock, and the result found in the 32-bit register is the final result "result_final", this finishes the value of B. We designed the formula B by using both RCA & Faster RCA.

Architecture For The Formula A: Multiplication-Products Repetitive Adder

$$A = \sum_{k=0}^{n-1}(X[k] \times S[k])$$

In this we require both "multiplication hardware unit" and "addition hardware (RCA/Faster RCA) unit" that we have discussed in the above sections. We called this hardware unit for formula A as "Multiplication-Products Repetitive Adder". The X[k] is samples of the FFT frequency spectrum of the noisy signal. Here we have taken 64 FFT frequency spectrum samples of the noisy signal. Each data sample is of 32-bit signed number. The architecture is given in the Figure 4.

The Figure 4 is showing the hardware architecture for the formula B. In this there are two memory blocks of 64 locations, each location is of 32-bit signed numbers. One memory block is for the FFT spectral samples of the noisy signal X, and the other one is for the S vector. On reset "rst" at positive edge of the clock "clk", the memory of S(K) is loaded with all the S vector values; and X memory is cleared to zeros. At "rst", on positive edge of "clk", all the registers, counters, enable signals are set to zero. This hardware requires one booth's multiplication unit, and one RCA/Faster RCA 32-bit signed number addition unit. There is control logic "control logic 1" at the Memory blocks. It takes a signal of "load" from an external unit which generates the "FFT spectral samples" X(K) of the noisy signal; and generates "start_count" signal to the counter, then the counter counts and the value of the counter is the address for the memory X(K); and at the same time the control signal generates the "write_en" signal to the memory to enable the writing of FFT samples into memory. Then the FFT frequency spectrum samples are loaded into the memory and when all the data samples are finished, the external unit sends a "finish_1" control signal to the control logic, and the control logic clears "start_count" signals and it consequently clears the counter. The control logic clears "write_en" memory signal also. Then in the next clock cycle, the control logic generates "read_en" signal to both memory blocks. The first data samples from the memory goes to booth's multiplication unit, and when the "start_mul" is generated by the control logic, then the Booth's hardware does the multiplication process and the product is generated. Then the booth's unit generates a "finish_mul" signal for a clock cycle time, then at the same time the control logic clears the "start_mul" signal. Whenever the "COUNTER1" observes the "finish_mul" signal at the positive edge of the clock it increments its count value as address to the memories. There is one more counter "COUNTER2" and control logic "control logic 2" on the right side, both take the "finish_mul" signal and using this signal the counter counts the number of multiplications finished; and the controller generates "load2" signal for a clock cycle period. The data from the booth's multiplication final product output (correct & final product is available when the "finish_mul" is generated), and the feed-back output of the 32-bit register, are the 2 inputs to the "RCA/Faster RCA" adder unit which generates 32-bit signed sum output. When the "load2" signal is generated the 32-bit register gets loaded with final correct sum output from the adder. This process of multiplication-product generation from booth's multiplication unit, and addition process repetitively goes on for repetitive sum of 64 product terms. When repetitive sum of 64 product terms finishes the control logic 2 generates the "finish_2" signal, which clears its counter. The generation of "finish_2" indicates the "A" value is ready to use, at the output "final-result_A" – the value in the 32-bit register.

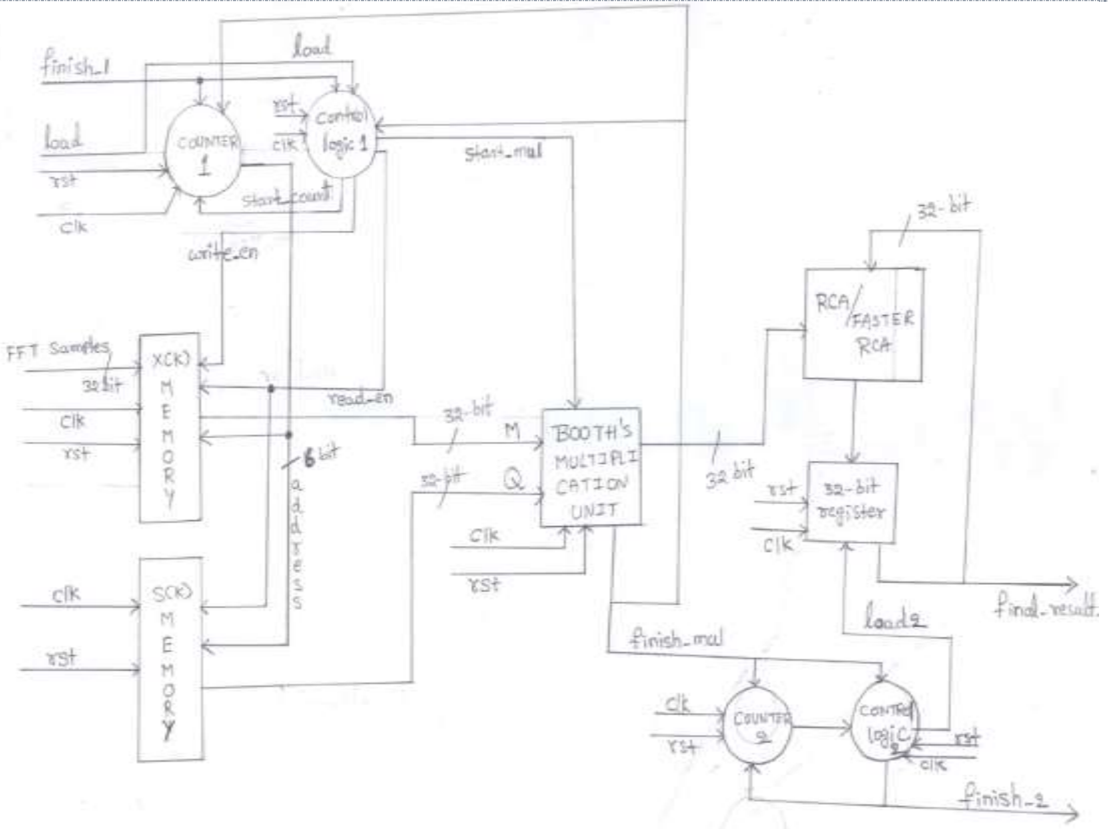

Figure 4 Sequential multiplication-products repetitive adder

Top Module

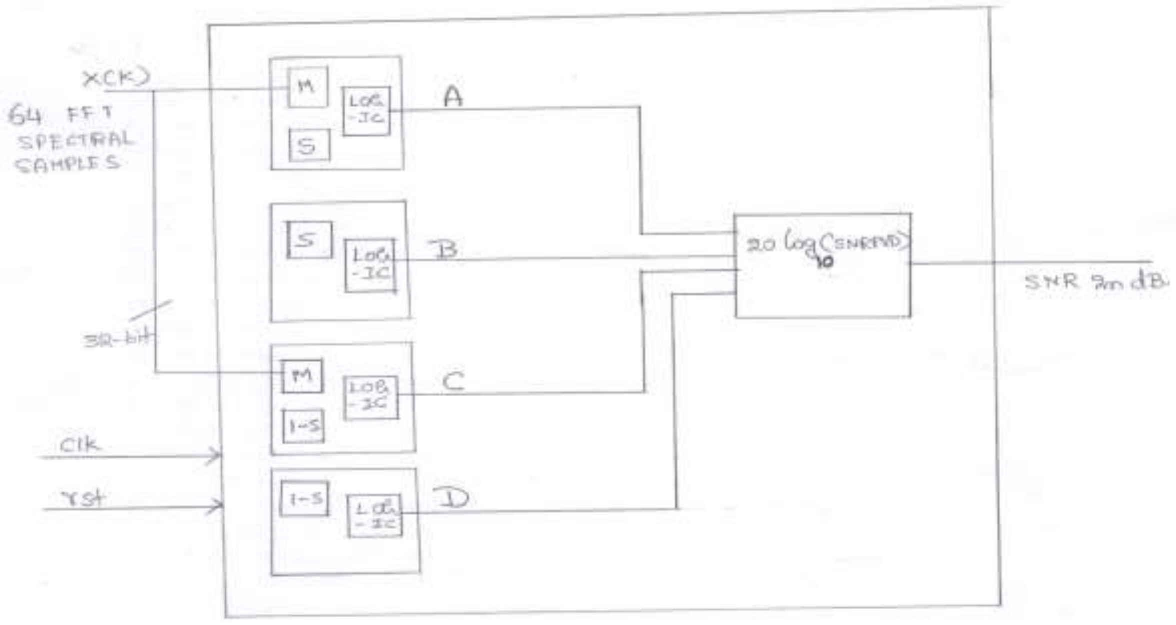

Figure 5 The block diagram of the top module of the SNR Hardware

For the above sequential (and also parallel architectures which we are going to explain in the following sections) the overall top model block diagram is given in the Figure 5. In contains 4 blocks – A, B, C, D. These blocks are those architectures described above for the formulas A, B. The design of C, and design of D are similar to A & B respectively, so we are not describing here. In this architecture, individual components of S and X block memories are there for each individual block of A, B, C, D, and all the 4 blocks A, B, C, D execute in parallel. It is inferred that system clock, and system reset are also there as inputs to the entire design. The input comes from the external module as integer transformed spectral samples of X [K]. When all the values of A, B, C, D are ready we can use the formula for calculating the SNR value.

$$\mathrm{SNRPVD}(X,\ S) = (A/B)/\ (C/D) \qquad \text{-------- (1)}$$

But actually we need logarithm of the value of it, so we can calculate it as

$$\mathrm{SNR} = 20^{*}\ (\log_{10} A + \log_{10} D - \log_{10} B - \log_{10} C) \qquad \text{----------------- (2)}$$

This gives us the final SNR value of the given signal in dB. The advantage over here is we do not need to do division again.

V. PARALLEL ARCHITECTURES

Hardware for the formula A

$$A = \sum_{k=0}^{n-1} (X[k] \times S[k])$$

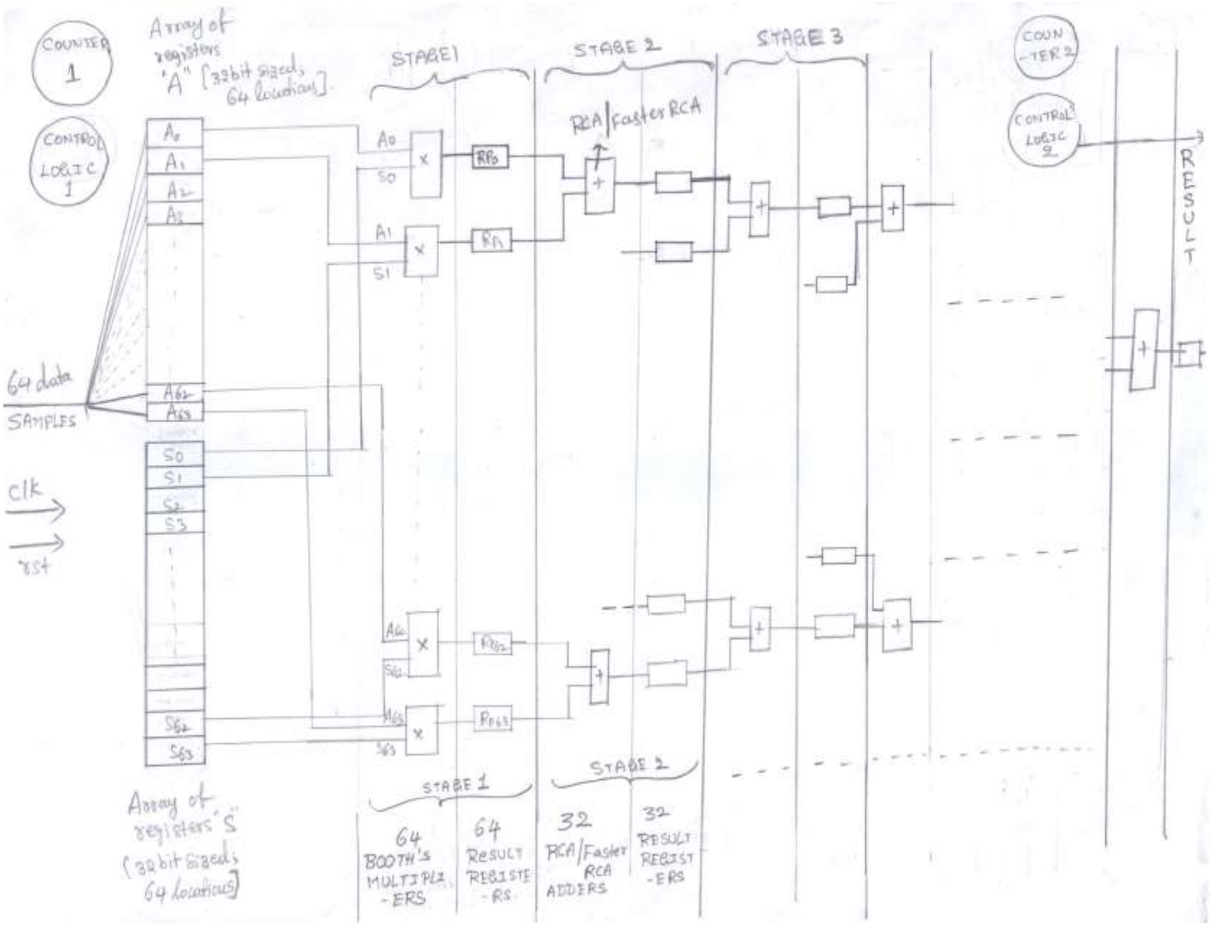

Figure 6 Parallel Architecture hardware block diagram for formula A

For improving the speed in the above sequential architectures we have developed parallel architectures for this system of A, B, C, D. The above Figure 6 shows the logical parallel architecture for the formula A. The design timing events are based on the positive edge of the clock "clk". In this there are two array registers memories, each of 64 locations, and 32-bit signed numbers; one for the frequency spectral samples of the noisy signal X; one more for the S vector S. The S vector array registers are loaded with proper S vector values at reset "rst" time. Here there is a counter "counter1" for counting 0-63, and a control logic "control logic1" at the input. At the input the control logic generates proper control signal to start loading the registers from the 64 data samples of the spectral samples of the input noisy signal, while there is a counter continues incrementing, whose value indicates how many data samples are already loaded. This counter value is being observed by the control logic, when all the data samples are loaded, then the control logic stops loading the registers and generates proper control signal to clear the counter and generates a signal to start the next process for formula A.

In this system, there are 7 stages. There is a second control logic and a second counter: "control logic2" and "counter2". The control logic generates control signal to increment the counter, and counter's value denotes the stage where we are at. First stage contains the 64 booth's multipliers working in parallel. In this first stage, first multiplier takes first sample of X[K] and first sample of S[K] and produces the product of them. The same way second, third, and so on 64th multiplier takes those corresponding S and X vector values and generate the product terms. All these product terms are generated in parallel. In the next clock cycle the second control logic generates a control signal for a clock period of time, which loads the outputs of the first stage multipliers into the registers which are at their outputs – 64 registered products of stage 1 – result registers, and the same counter 2 is incremented by one representing the processing is at next stage. Then in the stage 2 there are 32 adders. The first adder takes the first two products and adds them; similarly all the 32 adders generate the corresponding sum results. In the next clock cycle the control logic generates one more control signal for a clock period of time to load the corresponding sum results into the registers at the outputs of the second stage adders, and it also increments the counter 2. Similar process goes on in the 3rd stage with 16 adder and registers, 4th stage 8 adders and 8 registers, 5th stage 4 adders and 4 registers, 6th stage 2 adders and 2 registers, and at the end 1 adder and 1 register. Then at the final stage – from the value of the counter – the control logic generates the "RESULT" signal indicating the final result of the formula A, when the result of the 7th stage is loaded into 7th stage register. Then the same result signal clears the counter. Then at that time the result of the formula "A" is ready to use. For the adder stages, we have used both RCA and Faster RCA addition processes. We designed the entire system first by using RCA and then by using Faster RCA, and the results are compared.

Hardware For Formula B

$$B = \sum_{k=0}^{n-1} S[k]$$

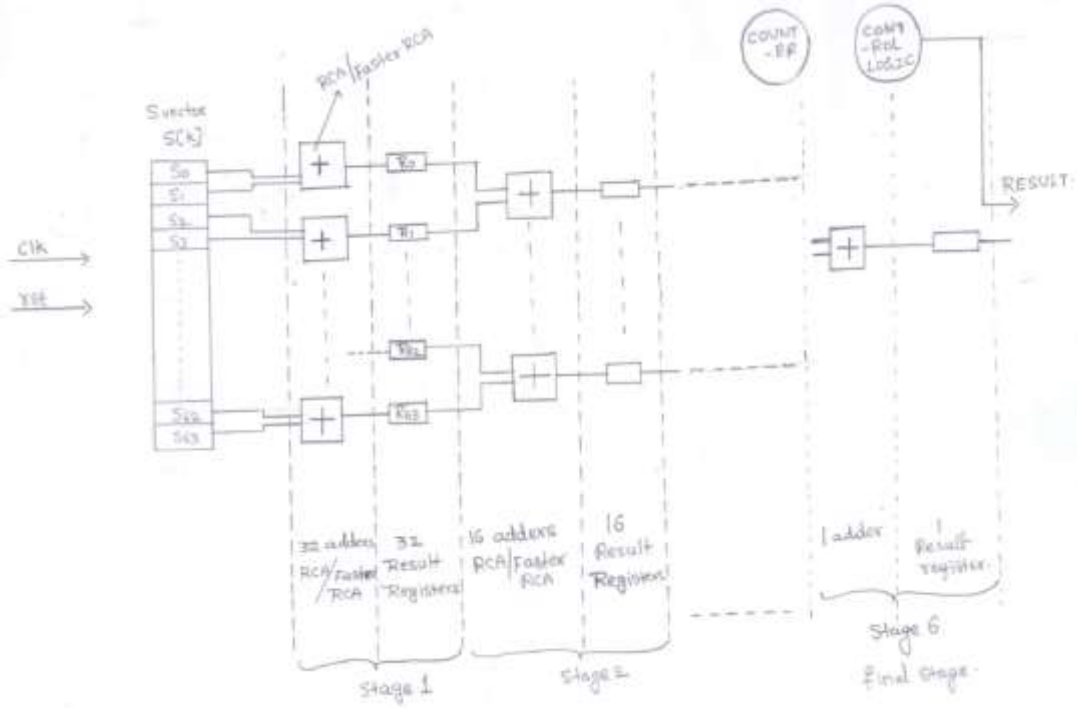

Figure 7 Parallel Architecture hardware block diagram for formula B

In this architecture of Figure 7 for formula B, there is an "array of registers" memory for storing the 64 values of S vector, each of size 32 bits signed number. There is one 3 bit "counter", and one "control logic". The array registers are loaded with S vector values on reset "rst" and also the counter and the control signals are cleared to "0" at the beginning. The timing events of this design are based on the positive edge of the clock "clk". In this parallel architecture there are 6 stages. In the first stage there are 32 adders, each of 32-bit signed numbers. The first adder takes the S[0] & S[1] and generates the addition result. In parallel, the second adder takes S[2] & S[3] values and generates addition result and so on all the 32 adders generate 32 result outputs. The Control logic generates control signal and when it is enabled (enabled state for a clock period of time), these output results are loaded in to the registers at the outputs of the adders. At the same time this enabled control signal increments the counter to represent the processing is at the 2nd stage. Then in the stage2 there are 16 adders, and 16 registers. The first adder adds the first 2 registered results; second adder adds next 2 registered results, and so on and these generate sum results. The control logic again generates one more control signal and enables for a clock period of time which loads the results into the second stage registers which are at the outputs of the adders, and the counter is incremented by 1. This process goes on until all the stages are finished. At the end stage, Basing on the value of counter, the control logic generates a control signal "result" signal to indicate the result of "B" is ready to use, and at the same time the counter and all control signals are cleared to 0. For the adder stages, we have used both RCA and Faster RCA addition processes. We designed the entire system first by using RCA and then by using Faster RCA, and the results are compared.

Top Module

For the above parallel architectures the overall top module block diagram is given in the Figure 5. It contains 4 blocks – A, B, C, D. These blocks are those architectures described above parallel architectures for the formulas A, B, C, D. The design of C, and design of D are similar to A & B respectively, so we did not specifically describe them. In both architectures, individual S and X block memories/array registered memories are there for each individual block of A, B, C, D, and all the 4 blocks execute in parallel. It is inferred that system clock, and system reset are also there as inputs to the entire design. As in the sequential architectures, the same logarithmic formulas here also work well for the SNR value in dB.

VI. RESULT ANALYSIS

In this we have done the hardware design and implementation of a novel SNR filter using Verilog programming in the Xilinx Vivado 2015.2 suite on Xilinx Artix-7 device and board. The implementation results are given in the tabular forms Table 1, 2, 3.

Table 1 Result table for Sequential architectures: using RCA and Faster RCA

	Total Delay(ns)	**Clock Speed(GHz)**	**No. of SLICE LUTS**	**No. of SLICE Registers**	**Power (W)**
RCA	78.48	36.6	8372	16376	2.55
Faster RCA	50	48.9	9100	16500	3.10

In the above Table 1, the implementation results of the SNRPVD Top module which encompasses sequential architectures are provided. The comparison is done between the designs with RCA and Faster RCA in the addition parts. It is evident that Design with Faster RCA is working much faster than the design with RCA and having less delay. But Design with Faster RCA is utilizing more resources and power.

Table 2 Result table for PARALLEL architectures: using RCA and Faster RCA

	Total Delay(ns)	**Clock Speed(GHz)**	**No. of SLICE LUTS**	**No. of SLICE Registers**	**Power (W)**
RCA	9.861	222.0	11151	18096	5.33
Faster RCA	5.521	311.2	13200	19300	5.80

The implementation results of Top module with parallel architectures are provided in Table 2. RCA and Faster RCA in the addition parts are imparted independently and compared in terms of speed, total delay, resource utilization, and power. With Faster RCA the design is working much faster than with RCA and having less delay, but with more resources and power utilization.

Table 3 Results table for Sequential & parallel architectures

	Using RCA		Using Faster RCA	
	Sequential	**Parallel**	**Sequential**	**Parallel**
Total Delay(ns)	78.48	9.861	50	5.521
Clock Speed(GHz)	36.6	222.0	48.9	311.2
Total hardware utilization	24748	29247	25600	32500
Power(W)	2.55	5.33	3.10	5.80

It is compared between the parallel and sequential architectures incorporated Top modules in Table 3. It is observed that parallel architectures are working much faster and with Faster RCA even much faster than the sequential architectures and with additional hardware and power utilization.

VII. CONCLUSION

In this work we could be able to implement an efficient Novel SNR digital hardware on Xilinx Artix-7 FPGAs by developing our own sequential and parallel digital hardware architectures. As per open literature this hardware had never been implemented. We did the design with basic architectures, and basic implementation considerations. From this we could be able to find that this hardware is working at high Giga hertz speeds of 36.6GHz & 48.9GHz (sequential) and 222.0 GHz & 311.2 GHz (parallel). These speeds are much higher than all ongoing technologies, and hence compatible to all ongoing technologies, and also can be useful for future technologies ie> in terms of speed compatibility. If we need higher sample sizes such as 512, 1024 parallel architectures are highly useful and must be working for future technologies. By optimizing the architectures, and taking into account of all implementation considerations we can attain higher speeds of operation, and less hardware (less area) and less power utilization, and can be useful even in very advanced future high speed technologies of wireless communications, computer networking and biomedical wireless devices.

VIII. DESIGN CONSIDERATIONS:

We considered all the design with integers. So, we considered the spectral values X[K] are all provided by a system from outside of our module in the form of integers. Usually all the spectral values are decimal numbers. So, different input data to FFT will have different decimal numbers [modulus of FFT spectral values]. So, normalization number will be different for different FFT values. So, for universalizing our design we considered an external hardware, which does the process that the numbers from FFT are normalized to [0,1] range, and multiplied with 10000, means considering a maximum of 4 digits after decimal point, and converted into integers, and then provided to our SNR hardware in 32 bit signed number format. So even after all A,B,C,D values are calculated, we have to multiply/divide with those ten thousands at the formulas (1),(2) in Top module, which all get cancel out in equation (1), and so directly we can use those integers values of A,B,C,D in equation (2) for calculating the SNR in dB. Similarly, the normalization factor also gets cancelled out in this formula.

In this design we have considered the numbers to be in the range of 16 bit signed numbers, even though the data we considered is 32-bit signed numbers. When it comes to multiplication the 16-bit signed numbers (in the 32-bit format) are multiplied the total result can be represented in 32-bit signed result (in the 64-bit format). But as per the total system we have considered 32-bit signed numbers. So, when the multiplication products after adding also can be represented in 32-bit signed numbers. This is also the case that we are considering the peaks of the specific locations where the required frequencies are expected, are maximum numbers. Remaining all numbers of the noisy peaks of the FFT spectrum, are suppressed while doing multiplication with S[K] vector. So, only the maximum number is at the location of the "+1" of the S[K] vector (maximum of +10000 after integer transformation of [0,1] numbers), where we do expect spectrum peak in the X[K].

As this is a dedicated hardware in FPGAs, All S vector values are in our hands. So, we can calculate both for S[K] and for 1-S[K] directly in the range of 16-bit integers (in 32-bit number format of our architecture) and load the S[K] values into memory.

As an important note, for this design we considered all the input numbers of X[K] & S[K] are within the range of 16-bit signed numbers represented as 32-bit numbers, so that the result can be fit within as 32-bit signed number even after multiplications and consequent additions in our case of SNR formula. For any application this constraint will suffice.

IX. ACKNOWLEDGEMENTS

In this research I would like to acknowledge the help of ECE Department Faculty Coordinator Prof. K. Kishore Kumar, ECE Department Adjunct Professor Prof. Dr. SSSP Rao, and Network Administrator Mr. Mahesh for the arrangement of necessary tools, kits and facilities.

X. REFERENCES

[1] Design of a 32-Channel EEG System for Brain Control Interface Applications, ching-sung-wang, journal of biomedicine and biotechnology, 2012
[2] Signal to Noise Ratio, SNR notes. http://www.radio-electronics.com/info/rf-technology-design/rf-noise-sensitivity/receiver-signal-to-noise-ratio.php
[3] Real Time Hardware Implementable Spectrum Sensor for Cognitive Radio Applications Chaitanya, P.Rajalakshmi, U. B. Desai. IITH website.
[4] Transceiver Design for Processing of 5G Cellular Signals J.SIVA SANKARI1, T.SUDHARCHANA2, G.SUNDARAVALLI3, M.THEIVA MEENA4. International Journal of Advanced Information Science and Technology (IJAIST) ISSN: 2319:2682 Vol.4, No.2, February 2015.
[5] AN EFFICIENT PEAK VALLEY DETECTION BASED VAD ALGORITHM FOR ROBUST DETECTION OF SPEECH AUDITORY BRAINSTEM RESPONSES, ranganadh narayanam, CS&IT-CSCP – CCSIT-SIPP – AIRCCJ – 2013. Indexed in Stanford university library.
[6] Robust detection of speech auditory brainstem responses using Voice Activity Detection (VAD) algorithms R Narayanam - IEEE CAMAN international conference–2012, 2012.

[7] High Speed FFT based Pulse Detection for a Digital ESM Receiver for Airborne Applications, Priya Suresh N, Abhijit Kulkarni, Virendra K. Jaiswal, Dr U.K.Revankar, Vikas Akalwadi, Sudipto Patra. Coreel technologies website. Indexed in Semanticscholar – 2009.

[8] Design and Implantation of High Speed FFT Processor for OFDMA System Using FPGA G.Purna Chandra Rao1, B.Ashok2, B.Saritha, Assoc. Prof , ECE,SVSIT,Warangal,India. International Journal of Advanced Research in Computer and Communication Engineering Vol. 2, Issue 7, July 2013.

[9] A high-speed FFT processor for OFDM systems, Circuits and Systems, 2002. ISCAS 2002. IEEE International Symposium on, B.S. Son ; B.G. Jo ; M.H. Sunwoo ; Yong Serk Kim

[10] ranganadh narayanam, "Brain-Activity-Filters: Efficient performance of Translation-Invariant (TI) Wavelets approach for Speech-Auditory Brainstem Responses of human subjects", International Journal of Scientific and Research Publications, Volume 4, Issue 9, September 2014.

[11] ranganadh narayanam, "Efficient De-noising Performance of a Combined Algorithm of Translation Invariant (TI) Wavelets and Independent Component Analysis over TI Wavelets for Speech-Auditory Brainstem Responses", Elsevier international journal Procedia Computer Science Volume 54, 2015, Pages 829-837, Sciencedirect.

[12] Ranganadh Narayanam, "Developing 'standard novel 'VAD' technique' and 'noise free signals' for speech auditory brainstem responses for human subjects", Thomson Reuters ENDNOTE – ResearcherID, IJESRT international journal, 5(6), June 2016.

[13] Rabiner, L. R., and Schafer, R. W., Digital Processing of Speech Signals, Englewood Cliffs, New Jersey, Prentice Hall, 512-ISBN-13:9780132136037, 1978.

[14] Fei Qin, Xuewu Dai, John E. Mitchell, "Effective-SNR estimation for wireless sensor network using Kalman filter", Elsevier, International journal of ad hoc networks volume 11, issue 3, may 2013.

[15] G Dumermuth, R Dinkelmann, "EEG data acquisition and preprocessing by microcomputer satellite system", Elsevier international journal of Computer Programs in Biomedicine, Volume 10, Issue 3, December 1979, Pages 197-208.

CITE AN ARTICLE

Narayanam, R., & Rao, S. (n.d.). IMPLEMENTATION OF A HIGHLY EFFICIENT NOVEL FREQUENCY DOMAIN SNR HARDWARE USING XILINX FPGAs. *INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY*, *6*(12), 413-426.