Proc. of Int. Conf. on Control System and Power Consumption , CSPC

High Speed Parallel Architectures for Implementing Novel & Efficient Frequency Domain SNR for 1024 Sampled Signals using Xilinx FPGAs

Ranganadh Narayanam
Assistant Professor, ECE department,Faculty of Science and Technology,The ICFAI Foundation for Higher Education, Hyderabad, (Deemed to be University under section 3 of UGC Act, 1956), India
ranganadh.narayanam@gmail.com

Abstract*—In plenty of real time applications of signal processing, wireless communications, radio communications, biomedical systems at the receiving end the signals have to be de-noised and their "Signal to Noise Ratio (SNR)" values have to be found. In this way in many advanced applications such as Cognitive radio receiver, Wireless BMI RF receiver, in many Digital Signal Processing systems either at the intermediary stages, or at the receiving end the noise performance is very critical. For noise performance SNR is a key parameter. In many of such DSP systems one of the most efficient approaches is to have a dedicated hardware. Keeping in mind of the critical role of SNR, in this research we have developed hardware for a Novel highly efficient SNR mechanism, which is a frequency domain SNR. As this Novel SNR is found to be highly efficient, and as it has never been implemented in hardware we have got motivated towards implementing this SNR mechanism onto Xilinx Field Programmable Gate Arrays, specifically to find its speed of operation and compatibility for the current ongoing technologies. So, as a first step we developed hardware for 64 sampled data and we found that the hardware is working at high *Ghz* rates and compatible for all current real time technologies in terms of sampling rates and system clock speeds and also useful for future technologies. As much of the current technologies use 1024 sampled data, so secondly we have extended this hardware for 1024 sampled signals. We have developed original parallel architectures for this SNR hardware implementation for 1024 sampled signals. We found some Novel results and in this paper we are presenting the implementation results of the parallel architectures for 1024 sampled signals and comparing them with that of 64 sampled signals in terms of speed of operation and power utilization. The parallel architectures developed are found to be highly efficient even in the case of 1024 sampled signals also, this hardware is working at high *Ghz* rates, but slower than 64 point hardware, and also compatible for future 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.

***Index Terms*— HDL, Architecture, FPGAs, SNR, DSP.**

I. INTRODUCTION

In digital signal processing real time applications, the Signal to Noise Ratio (SNR) is a measure that

Grenze ID: 02.CSPC.2018.2.501

compares the level of a desired signal to the level of background noise. All the signals get corrupt during the process of their transmission, so SNR value measurement is very important. It plays vital role in plenty of DSP based real time systems such as speech processing, acoustics signal processing, EEG signal data analysis, Wireless communications, Radio receivers, Cellular phone communications, Voice Activity Detection (VAD) in speech auditory Brainstem responses and so on for assessing the signal quality level.

In fisheries acoustic digital signal processing, by using SNR values the thresholds are developed for measurement process and data quality assurance [**6]**. The noise performance and hence the SNR is most important in Radio receiver. 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.

Signal to Noise Ratio can be calculated with time domain samples and also with frequency domain samples. For frequency domain samples we require the FFT spectral samples of the noisy signal. It is found that Frequency domain SNR having certain advantages than time domain SNR and found to be obtaining almost equal values as time domain SNR [1]. Here we are presenting a SNR mechanism which is frequency domain SNR found to be efficient in Voice Activity Detection for Auditory Brainstem Responses of EEG [2,4], but this we can apply in any of the DSP systems. In this research we did implementation of Digital Hardware for this mechanism.

The first 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.

Due to these motivational reasons we have developed hardware for the current Frequency domain SNR mechanism under consideration, for 64 sampled signals. For this we developed two different types of architectures (a) sequential architectures (b) parallel architectures. To improve the speed we developed these parallel architectures. We have implemented on Xilinx Artix-7 FPGAs using Xilinx Vivado 2015.2 Design suite. We used Verilog HDL as the programming language. With sequential architectures we reached a maximum speed of 48.9 GHz, and with parallel architectures we reached a maximum speed of 311.2 GHz [5]. Both architectures are working at high GHz rates than the today's all real time DSP systems, FFT systems clock speeds and can be used in these dedicated systems, and also useful for the future technologies [5].

The second motivational cause for the implementation is to implement 1024 sampled signals. In today's real time technologies almost all DSP systems use 1024 sampled signals. So, in this paper we have extended this research to develop parallel architectures for 1024 sampled signals. Then we have compared the implementation results of 1024 point parallel architectures with that of 64 point parallel architectures in terms of speed, device utilization summary and power utilization.

Section II introduces the Novel SNR, section III talks about the parallel architectures developed, and section IV result analysis, and section V concludes the research. The design considerations are described in section VI.

II. The Novel Frequency Domain Snr Mechanism

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 [2, 4]. 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

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

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

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

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

$$SNRPVD(X, S) = (A/B)/ (C/D)$$

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. Parallel Architectures

In this research we developed original parallel architectures for the implementation of SNRPVD hardware for 1024 sampled signals. In this there are number of signed multiplications and signed addition operations. We have implemented 32-bit Booth's signed multiplication hardware unit [4]. Then we have implemented 32-bit Ripple Carry Adder unit. In this we used two different types of Full adders given in Figure 1, to develop two different types of RCAs.

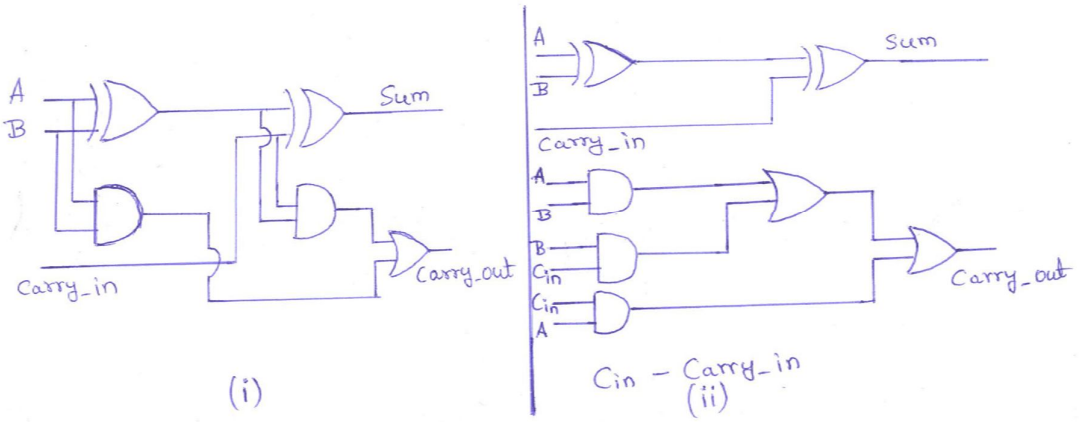

Figure 1. (i) Full Adder 1 (ii) Full Adder 2, for speeding up the addition process

Two different types of 32-bit RCAs are developed using both the above Full adders named as RCA using Full Adder 1 in Figure 1(i); and Faster RCA using Full Adder 2 in Figure 1(ii). Both of them are used as addition units in the parallel architectures and the parallel architectures are designed two times for different speeds.
Here for implementing parallel architectures, there are 4 parts to be implemented one parallel architecture for formula A, one for formula B, one for formula C and formula D. Then a single top module which consists of all the four modules integrated one, which takes the input spectral samples X(K) of the 1024 sampled signal and gives out the logarithmic value of SNR value using the Novel frequency domain SNR under consideration. As part of description we are describing here for the formulas A & B. C & D are almost same as A & B respectively.

A. Hardware for the Formula A

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

The parallel architecture for formula A is given in the **Figure 2.** Two "array registers memories" are used each of 1024 locations, each location is of 32-bit signed numbers. One 1024 memory array for the frequency spectral samples of the noisy signal X; second one for the vector S. On reset "rst" the S vector memory array is loaded with proper S vector values. The design timing events are based on the positive edge of the clock "clk". At the input there are two units: one is counter "counter1" which can count from 0-1023, one more is control logic "control logic1". At the input the control logic generates proper control signal to start loading the registers from the 1024 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

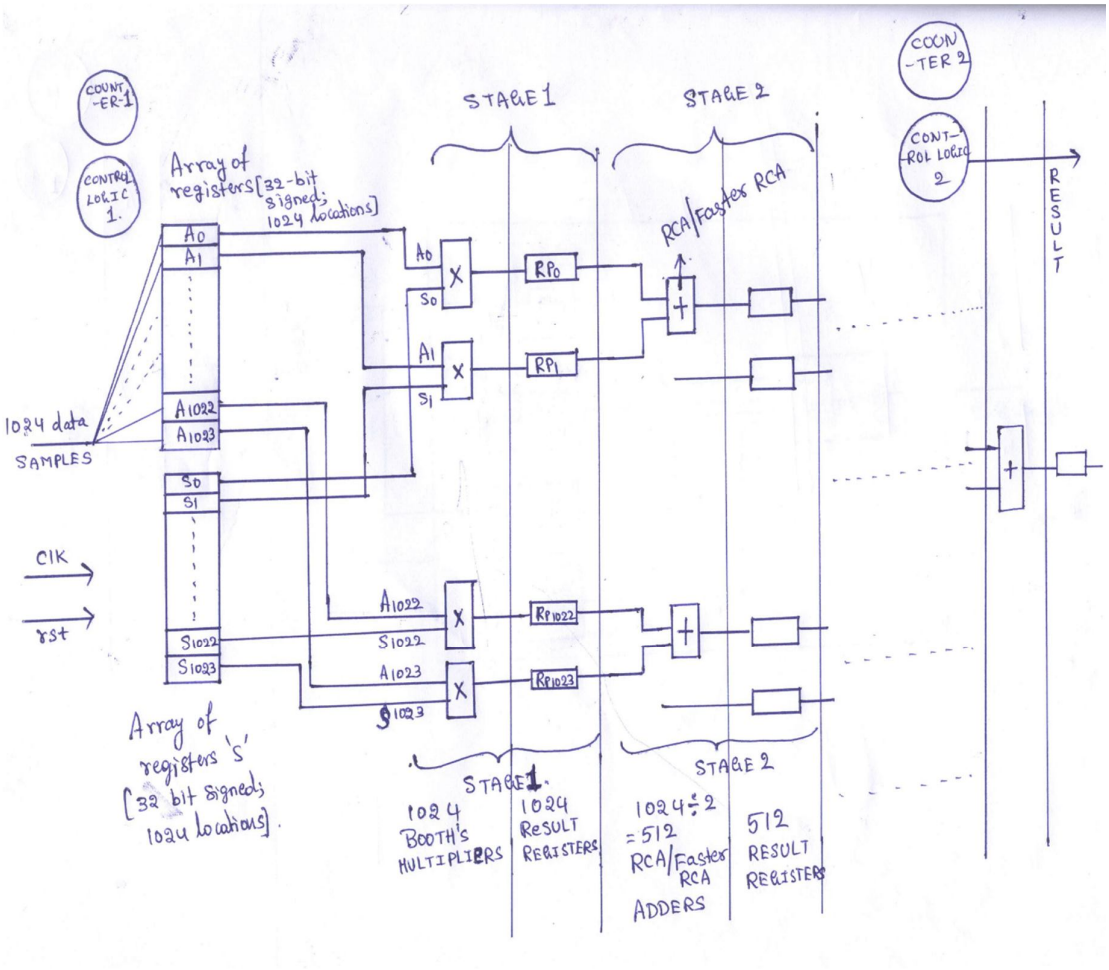

Figure 2. Parallel Architecture hardware block diagram for formula A

signal to start the next process for formula A. In this system, there are 11 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 1024 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 1024th 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 – 1024 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 512 adders. The first adder takes the

first two products and adds them; similarly all the 512 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 256 adder and registers, 4th stage 128 adders and 128 registers, 5th stage 64 adders and 64 registers, 6th stage 32 adders and 32 registers, 7th stage 16 adders and 16 registers, 8th stage 8 adders and 8 registers, 9th stage 4 adders and 4 registers, 10th stage 2 adders and 2 registers, and at the end 1 adder and 1 register. Then at the final stage – basing on 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 11th stage is loaded into 11th 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.

B. Hardware for Formula B

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

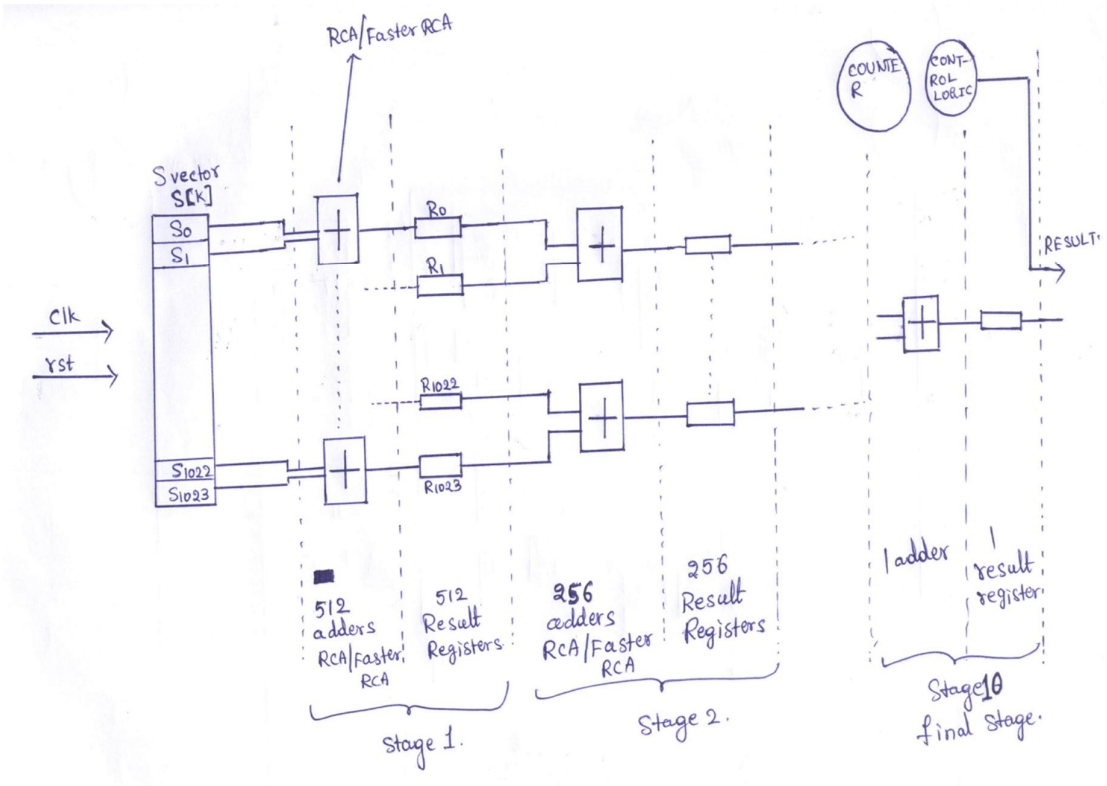

Figure 3. Parallel Architecture hardware block diagram for formula B

Figure 3 shows the parallel architecture for formula B. For storing the 1024 S vector values there is an "array of registers" memory, 1024 locations each of size 32 bits for 32 bit signed numbers. There is one 10 bit "counter"(for counting 10 stages), 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 10 stages. In the first stage there are 512 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 512 adders generate 512 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 256 adders, and 256 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.

C. Top Module

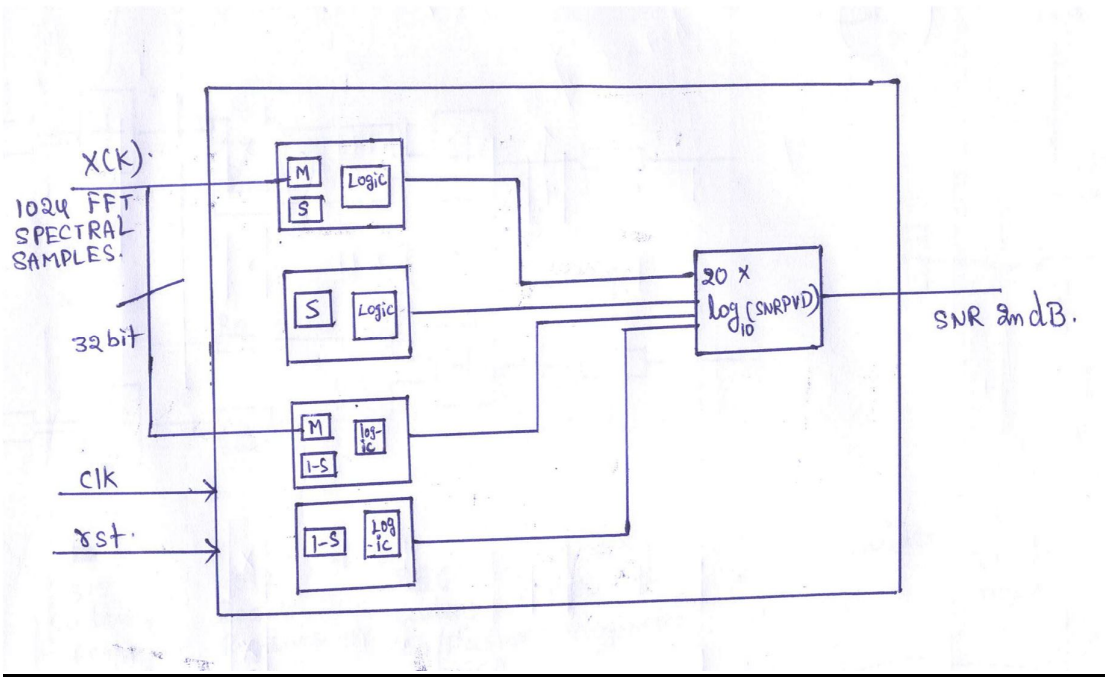

Figure 4. The block diagram of the top module of the SNR Hardware

For the above parallel architectures the overall top model block diagram is given in the **Figure 4.** It 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.

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

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

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

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.

IV. Result Analysis

Here in this we did implemented the top module onto the Xilinx Artix-7 FPGAs using Xilinx Vivado 2015.2 Design suite. Verilog Hardware Description Language (HDL) is used for programming this hardware. This Parallel architectures based implementation of 1024 point SNR we did for the purpose of knowing what is the speed of operation and whether it is applicable for current real time DSP system clock speeds, and also almost all DSP systems in current technologies are for 1024 sampled signals, and hence to make sure that if the parallel architectures are performing well or not for 1024 point SNR. Through this implementation we would also like to see if the clock speed is increasing or decreasing while we go up from 64 point SNR to 1024 SNR hardware. The results for 1024 Integrated Circuit are given in Table 1. Table 2 shows results of 64 point SNR hardware [5].

TABLE I. RESULT TABLE FOR 1024 POINT SNR PARALLEL ARCHITECTURES: USING RCA AND FASTER RCA

	Total Delay(ns)	**Clock Speed(GHz)**	**No. of SLICE LUTS**	**No. of SLICE Registers**	**Power (W)**
RCA	18.288	189.0	18192	28081	6.23
Faster RCA	11.698	291.8	20200	30129	6.71

TABLE II. RESULT TABLE FOR 64 POINT SNR PARALLEL ARCHITECTURES: USING RCA AND FASTER RCA [5]

	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

From the results obtained it is clear that the developed parallel architectures are highly useful for 1024 point SNR hardware also efficiently in the view that the clock speed is much higher than the current real time DSP systems and is useful in all of them. This High GHz clock speed is highly useful for future Technological systems speeds also. It is found that the clock speed reduced from 64 point to 1024 point SNR, but still working at high GHz rates.

V. Conclusion

In the research of this paper, High speed parallel architectures for the Highly Novel & Efficient Frequency domain SNR under consideration for 1024 point sampled signals. These 1024 point architectures are also working at High GHz speeds (but with less speed than 64 point architectures), and hence useful for current and future real time DSP systems. This confirms the efficiency, and high usefulness of the parallel architectures we developed, for current and future real time technologies. Using Faster RCA the 1024 point device is working 55% faster than RCA based 1024 point device. The total hardware (SLICE LUTs + Registers) required and the power utilized in the 1024 point device using Faster RCA is 9% & 8% more than the 1024 point device using RCA respectively.

VI. Design Considerations

1) 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.
2) 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].

3) 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.
4) 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.

REFERENCES

[1] Chaitanya, P.Rajalakshmi, U. B. Desai.Real "Time Hardware Implementable Spectrum Sensor for Cognitive Radio Applications" IITH website.

[2] R Narayanam "Robust detection of speech auditory brainstem responses using Voice Activity Detection (VAD) algorithms"; IEEE CAMAN international conference–2012, 2012.

[3] 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.

[4] 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.

[5] Ranganadh Narayanam, SSSP Rao" IMPLEMENTATION OF A HIGHLY EFFICIENT NOVEL FREQUENCY DOMAIN SNR HARDWARE USING XILINX FPGAs". researcherID-INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY, 2017 Dec.

[6] Robert Kieser, Pall Reynisson, and Timothy J. Mulligan, "Definition of signal-to-noise ratio and its critical role in split-beam measurements". Elsevier-ScienceDirect-ICES Journal of Marine Science, 62: 123e130 (2005).