IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

# A High Performance and Low Power Adaptive FIR filter using Approximate arithmetic circuits

[1]ANISHA.K, [2]Dr.KALPANA.A.B,

[1]PG Student, [2]Associate Professor
[1]VLSI and Embedded Systems,
[1]Banglore Institute of Technology, Bengaluru, INDIA

*Abstract:* The real-world applications related to digital signal processing demand high performance computation. In such applications we use the various types of filters and one such type of filter is Adaptive Finite impulse Response filter. There are many architectures implementing the Adaptive Finite impulse Response filter. The architecture of Adaptive FIR is mainly composed of the error computing and weight updating modules. Because intensive computation is involved in practical applications, high efficiency and low-cost hardware is a must. Hence here a fixed-point finite impulse response adaptive filter is proposed using approximate distributed arithmetic (DA) circuits. In this design, the radix-8 booth algorithm is used to reduce the number of partial products in the DA architecture. In order to reduce the area, Kogge stone adders are used. As a result, the delay and power consumption of the proposed design are significantly reduced. Both the error computing schemes and weight updating schemes will be designed using Verilog.

***Index Terms* – Adaptive filter, approximate distributed arithmetic, radix-8 booth algorithm**

## I. Introduction

There is a high demand for the applications that involve the signal processing. Filters are one such devices that implement the algorithm for signal processing. These filters are used in many applications which are used in our day to the day life such as speech coding and transmission in mobile phones, medical imaging in MRI, speech recognition and processing in home automated systems etc. Designing and implementing these filters for real time applications poses a challenge. One such filter is Adaptive FIR filter. An Adaptive FIR filter is a system with a linear filter that has a transfer function controlled by variable parameters and a means to adjust those parameters according to an optimization algorithm. The closed loop adaptive filter uses feedback in the form of an error signal to refine its transfer function. It is composed of a FIR filter with variable coefficients (weights) and a weight update module. The weights are updates by an adaptive algorithm. Due to the closed loop-adaptive process and related algorithm, the hardware implementation of a direct form FIR filter is very complex. Moreover, the high-power consumption, large area and long critical path of the weighted sum operation in the linear filter significantly limit the throughput of such a digital processing system. Various architectures have come up to implement the adaptive filters, but they have some drawbacks which can pose the problems in real time applications Hence it is important to implement the architecture of an adaptive filter in an efficient way- that has high performance and consumes less area and power. Hence an efficient architecture is proposed to implement the adaptive FIR filter using Approximate arithmetic circuits.

## II. literature Review

- There are various types of computational schemes for adaptive filters. Some of them are perceptron based model [1],[2], the continuous spatiotemporal model [3], the higher order lead-lag compensator model [4]. These computational models have been proposed to explain and to mimic the cerebellar function for signal processing and motor control applications. But these models have high complexity and hence it becomes tedious to implement them in hardware.

- Adaptive filters are used in many real time applications involving the signal processing They are used in signal prediction, signal identification, image processing and echo suppression [5]. These applications require high performance and low power. Hence it is important to design them efficiently.

- Various architectures were proposed to implement the adaptive FIR filter. One such architecture has been proposed where adaptive filter was based on two distributed arithmetic (DA) [6] where weights were used as addresses to access the Look up tables (LUTs) storing the sums of the weighted delayed inputs. The memory requirement is reduced but the size of the LUTs exponentially increases as the order of the filter increases. Therefore, these designs are not suitable for adaptive filters with higher order.

- An efficient DA formulation has been presented for the block least mean square (BLMS) algorithm in a FIR adaptive filter [7]. Here the LUT is shared between the computations of the filter output and the weight increment; Only one column of LUTs is updated in each iteration by shifting the weight- vectors. Thus, figures of merits such as circuit area, power and timing are improved for the LUT updating process. But the size of the LUT is L times the size of the LUT [6] where L is the block size of the BLMS algorithm. Hence DA based FIR adaptive filters design using LUTs perform well for a low order filter.

- A novel shared-LUT design has been proposed to implement DA for a reconfigurable FIR filter [8]. In this design, an M-dimensional vector pair is decomposed into L P-dimensional small vector pairs (i.e., M = LP). A 2P-word LUT is shared by the bit slices (consisting of P bits) of different weightage. Totally, L partial product generators, L 2P-word LUTs, m (as the bit width of inputs) adder trees and a shift-add tree are required to compute the inner product. The contents in the LUTs are updated in parallel. This FIR filter achieves a significant reduction in energy compared with the systolic decomposition of a DA-based design.

- In order to improve the throughput of an adaptive filter, a pipeline structure can be used. But Least Mean Square (LMS) algorithm does not directly support the pipelining due to its recursive operation. Hence the LMS algorithm is modified into the so-called Differential least mean square (DLMS) [9]. DLMS significantly reduces the critical path delay of an adaptive filter by pipelining whereas the performance of convergence is degraded significantly due to the adaption delay [10].

- There is another DLMS based FIR adaptive filter proposed with low adaption delay [11] by using a novel partial product generator and an optimized balanced pipeline; a bit-level pruning of the adder tree is further employed to reduce the area and power consumption of the implementation. Synthesis and simulation have shown that this scheme consumes less power and requires less area than other DLMS adaptive filter designs. However, a large number of additional latches are used for the pipelined implementation of a DLMS adaptive filter and hence, overheads in area and power dissipation are incurred compared to an adaptive filter using the LMS algorithm.

- Many other techniques have been combined with DA to increase its efficiency. Factor sharing has been employed in a DA architecture to reduce the number of adders [12]. It reduces 44.5% of the adders in a multi-standard transform core design. This design has lower performance though it consumes lesser area.

- A result-biased circuit for DA has been used in the filter architectures for computing the discrete wavelet transform; it leads to a 20% to 25% reduction in hardware [13].

## III. Insights on Adaptive Fir Filter

An adaptive FIR filter consists of two modules. They are error computing module and weight updating module.

### 3.1 ERROR COMPUTING MODULE

Error computing module computes the error by comparing the desired signal with that of input signal. The input signal is fed to the multipliers and the delay elements. Multipliers multiply the input signal and its delayed versions with the appropriate weights and generates the partial products. These partial products are added together to generate the final sum. This sum is compared with the desired signal and whatever the difference obtained is given as an error.

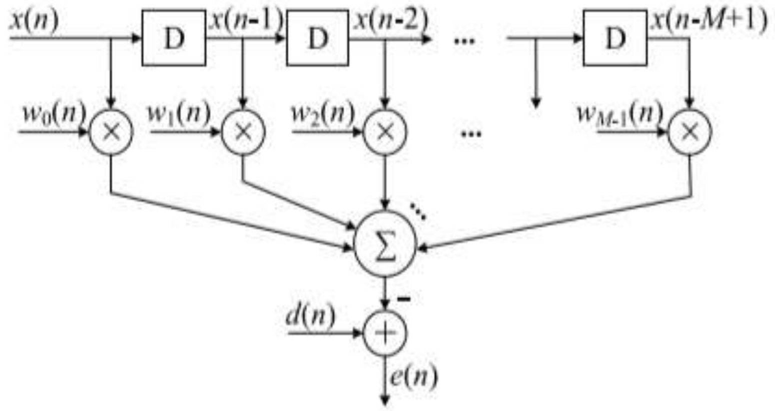


**Fig.3.1 Generic error computing module**

### 3.2 WEIGHT UPDATING MODULE

Weight updating module updates the weight according to the error generated by the error computing module. The input signal is fed to the delay elements and the multipliers. Multipliers multiply the input signal and its delayed versions with the error generated by the error computing module and generates the products. These products which represents the new weights will be compared with the older weights and the difference obtained will be the updated weights.

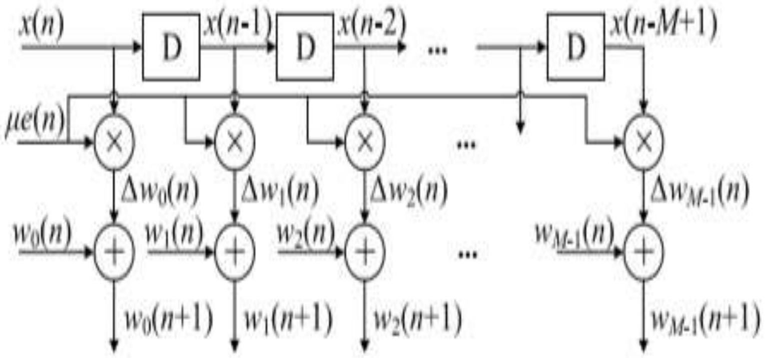


**Fig.3.2 Generic weight updating module**

## IV. Proposed Architecture And Methodology

Adaptive FIR filter is designed using the approximate arithmetic circuits. These are the circuits which uses an efficient technique to calculate the inner products or multiply and accumulate. The approximate arithmetic is performed to design bit level architectures for vector-vector multiplication with a direct application for the implementation of convolution which is necessary for digital filters. The following figures are the architectures of the proposed design of error computing scheme and weight updating scheme.

### 4.1 PROPOSED ERROR COMPUTING SCHEME

The error computing module comprises of D flipflop, Radix-8 Booth Encoder, Partial Product Generator, Kogge-stone adder and carry look ahead adder. There are three inputs given to the error computing module: Input sample, weights and the desired sample. The theory behind this methodology is that the architecture is designed step by step where after designing each functional module, their functional verification is checked. After the functional verification, all the modules are combined together to form the error computing. module. In the final step, the functionality of the error computing module is verified.

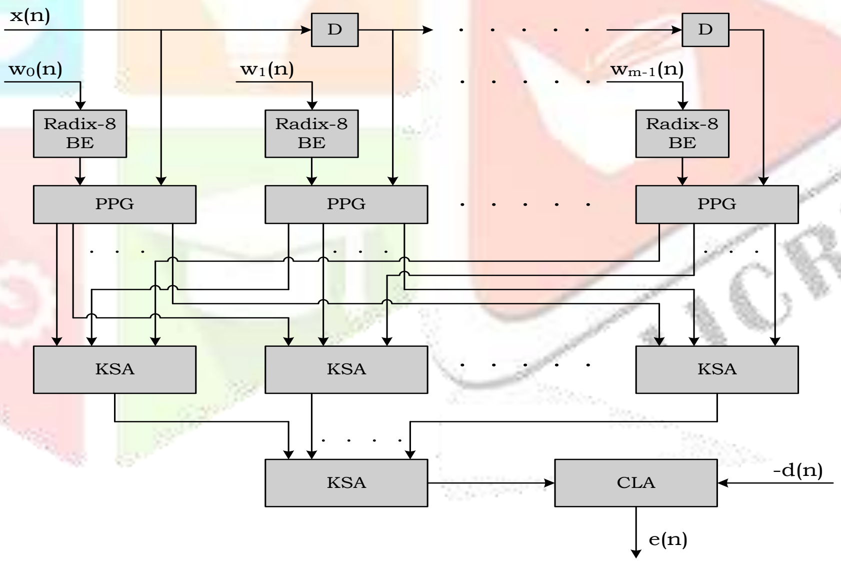


**Fig.4.1 Proposed error computing module**

### 4.2 PROPOSED WEIGHT UPDATING MODULE

The weight updating module comprises of D flipflop, Radix-8 Booth Encoder, Partial Product Generator and Kogge-stone adder. There are three inputs given to the weight updating module: Input sample, weights and the error. The error generated from the error computing module is given as input to the weight updating module to generate the new weights. This methodology also follows the same theory where the architecture is designed step by step where after designing each functional module, their functional verification is checked. After the functional verification, all the modules are combined together to form the weight updating module. In the final step, the functionality of the weight updating module is verified.

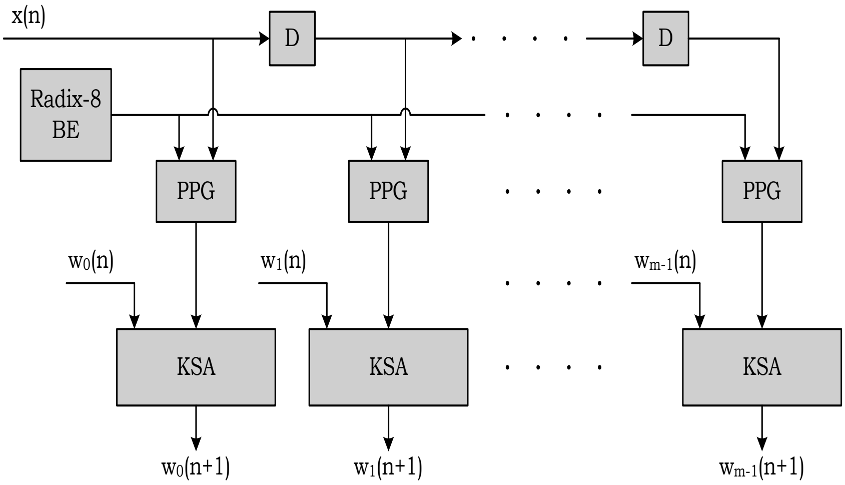


**Fig.4.2 Proposed weight updating module**

### 4.3 METHODOLOGY

The following steps elaborate the designing of error compute and weight update module. Each functional module is designed in Verilog using Xilinx ISE tool.

- The D flipflop is designed in first step. The flipflop is responsible to produce the delayed versions of input sample.

- The radix-8 booth encoder and partial product generator are treated as one module which is responsible for multiplication operation. The radix-8 Booth encoder takes the multiplicand and groups it to four. Each 4-bit value have an assigned operation. The partial product generator multiplies the multiplier and the encoded 4-bit value of multiplicand with an overlap of one bit into one number. This reduces the number of partial product numbers and thereby the area and hardware complexity. In error computing module, this module multiplies the input and its delayed versions by weights. In case of weight updating module, it multiplies the input and its delayed versions by error produced by the error computing module.

- Kogge-stone adder finds the sum of all the partial products generated by the partial product generator. This adder is chosen in order to attain the parallel fast operation. In weight updating module, the output of the Kogge-stone adder gives the new updated weights. These weights are updated according to the weight

- In case of error computing module, the carry look-ahead adder is designed to find the negative error output. This error is obtained by adding the output vectors of the kogge-stone adder with the desired sample. This adder reduces the propagation delay and increases the execution speed.

- In the final step, all these modules are combined together to form the error computing and weight updating module thereby forming the functionality of an adaptive FIR filter.

## V. Outcomes Of Project

The error computing and weight updating modules are verified for its functionality. This is done by applying the test cases. In error computing module, test case consists of 25 input samples, 3 weights and a desired input. The error computing module generates errors in accordance with the given test case. In weight updating module, test case consists of the error samples produced by the error computing module for the given input samples and weights. The weight updating module generates the updated weights in accordance with the error produced by the error computing module. Figure 5.1 and 5.2 gives the internal schematic of error computing module and weight updating module respectively. Figure 5.3 and 5.4 gives the simulated waveform of error computing module and weight updating module respectively. Figure 5.5 and 5.6 gives the power consumption of error computing module and weight updating module respectively.

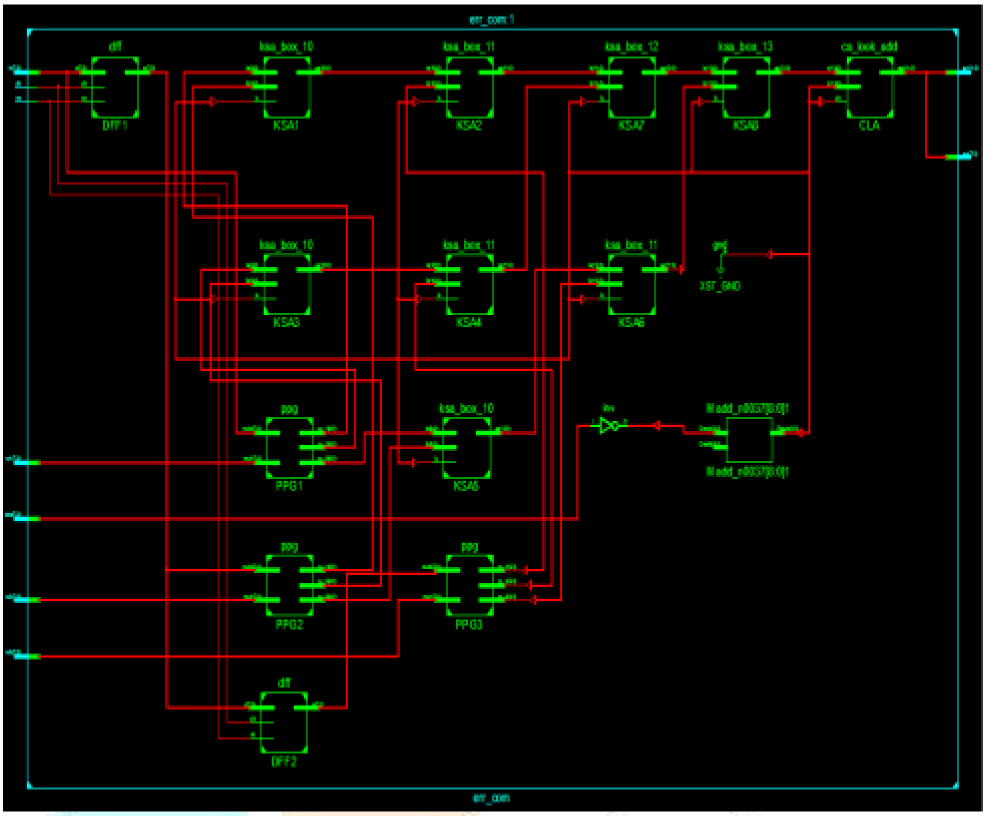


Fig.5.1 Internal schematic of error computing module

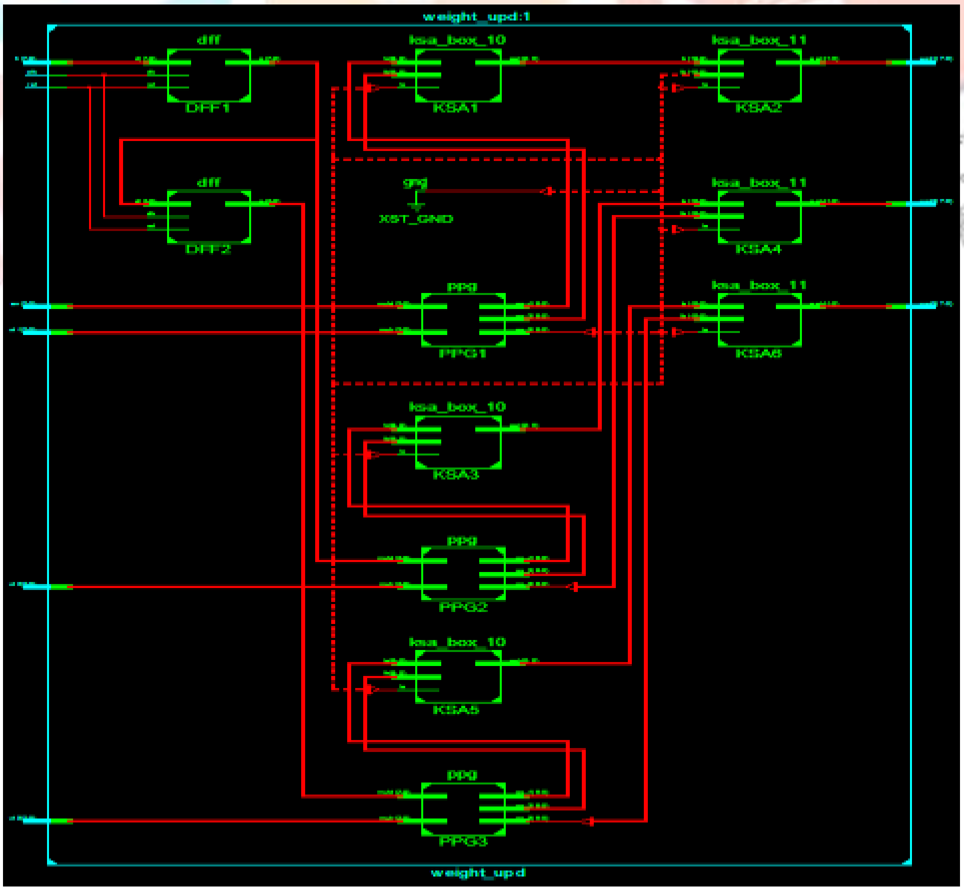


Fig.5.2 Internal schematic of weight updating module

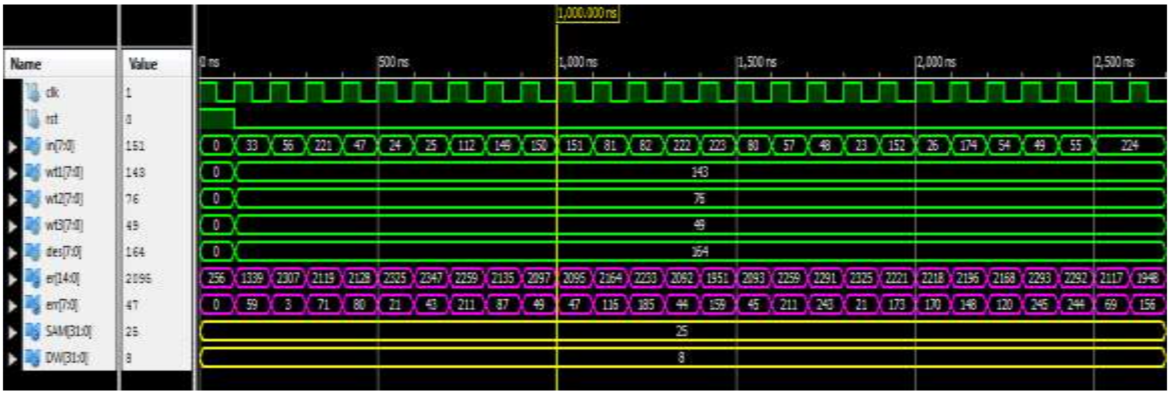


Fig.5.3 Simulated waveform of error computing module

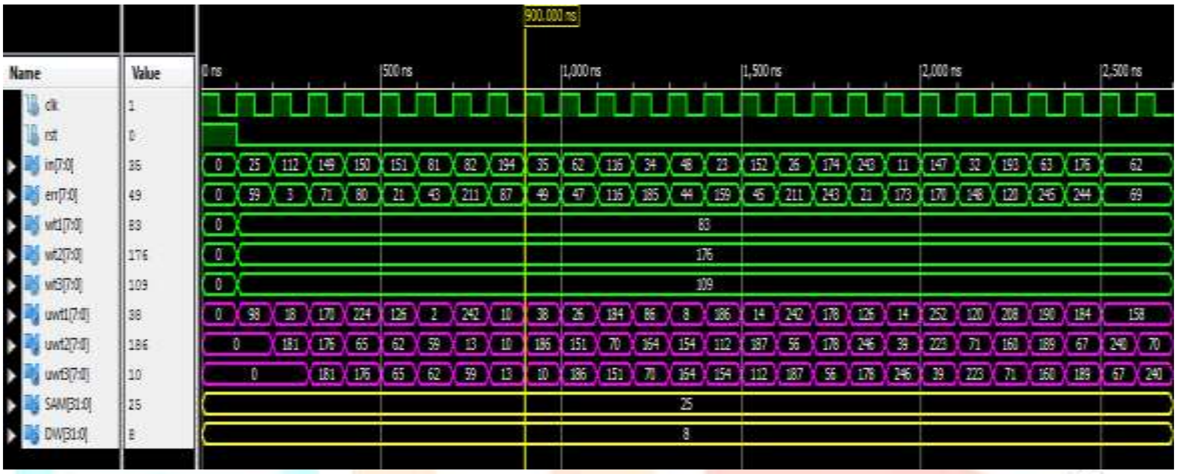
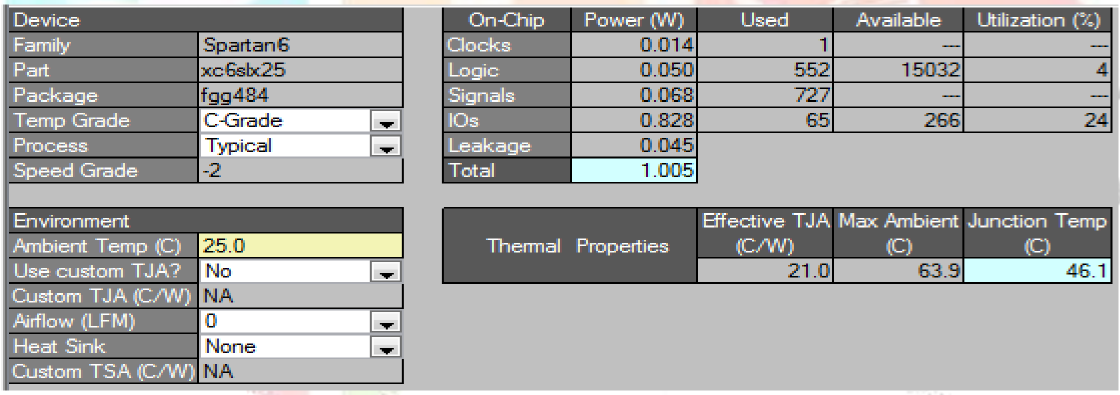


Fig.5.4 Simulated waveform of weight updating module



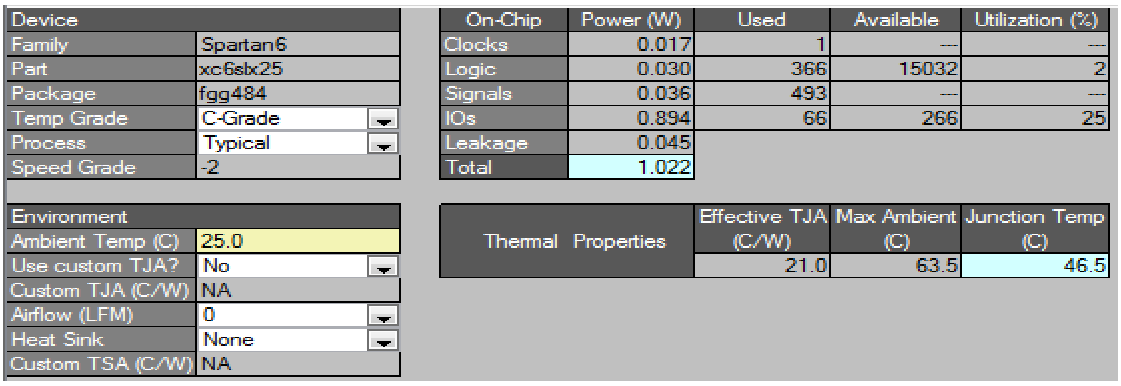
| Device | |
|---|---|
| Family | Spartan6 |
| Part | xc6slx25 |
| Package | fgg484 |
| Temp Grade | C-Grade |
| Process | Typical |
| Speed Grade | -2 |

| On-Chip | Power (W) | Used | Available | Utilization (%) |
|---|---|---|---|---|
| Clocks | 0.014 | 1 | --- | --- |
| Logic | 0.050 | 552 | 15032 | 4 |
| Signals | 0.068 | 727 | --- | --- |
| IOs | 0.828 | 65 | 266 | 24 |
| Leakage | 0.045 | | | |
| Total | 1.005 | | | |

| Environment | |
|---|---|
| Ambient Temp (C) | 25.0 |
| Use custom TJA? | No |
| Custom TJA (C/W) | NA |
| Airflow (LFM) | 0 |
| Heat Sink | None |
| Custom TSA (C/W) | NA |

| Thermal Properties | Effective TJA (C/W) | Max Ambient (C) | Junction Temp (C) |
|---|---|---|---|
| | 21.0 | 63.9 | 46.1 |

Fig.5.5 Power consumption of error computing module



| Device | |
|---|---|
| Family | Spartan6 |
| Part | xc6slx25 |
| Package | fgg484 |
| Temp Grade | C-Grade |
| Process | Typical |
| Speed Grade | -2 |

| On-Chip | Power (W) | Used | Available | Utilization (%) |
|---|---|---|---|---|
| Clocks | 0.017 | 1 | --- | --- |
| Logic | 0.030 | 366 | 15032 | 2 |
| Signals | 0.036 | 493 | --- | --- |
| IOs | 0.894 | 66 | 266 | 25 |
| Leakage | 0.045 | | | |
| Total | 1.022 | | | |

| Environment | |
|---|---|
| Ambient Temp (C) | 25.0 |
| Use custom TJA? | No |
| Custom TJA (C/W) | NA |
| Airflow (LFM) | 0 |
| Heat Sink | None |
| Custom TSA (C/W) | NA |

| Thermal Properties | Effective TJA (C/W) | Max Ambient (C) | Junction Temp (C) |
|---|---|---|---|
| | 21.0 | 63.5 | 46.5 |

Fig.5.6 Power consumption of weight updating module

## VI. Conclusions And Future Work

### 6.1 CONCLUSIONS

The proposed architecture of the Adaptive FIR filter using approximate arithmetic circuit achieves the required goal. The Radix-8 booth encoder reduces the partial products saves the delay in the multiplier design. Moreover, the use of the Kogge-stone adder in the proposed scheme makes it even faster. Finally, the area and power consumption of the design are significantly reduced due to the approximation in partial product generation and accumulation.

### 6.2 FUTURE SCOPE

This project work can be further extended by adopting different architectures for adaptive FIR filter. The adders and the multipliers used in the proposed approximate arithmetic circuits can be replaced by other efficient adders or multipliers to obtain efficient computation and performance. More efficient algorithm can be used in future to compute the error and update the weights.

## References

[1] D. Marr, “A theory of cerebellar cortex,” J. Physiol., vol. 202, no. 2, pp. 437–470, Jun. 1969.

[2] J. S. Albus, “A theory of cerebellar function,” Math. Biosci., vol. 10, nos. 1–2, pp. 25–61, Feb. 1971.

[3] T. W. Calvert and F. Meno, “Neural systems modeling applied to the cerebellum,” IEEE Trans. Syst., Man, Cybern., vol. SMC-2, no. 3, pp. 363–374, Jul. 1972.

[4] M. Hassul and P. D. Daniels, “Cerebellar dynamics: The mossy fiber input,” IEEE Trans. Biomed. Eng., vol. BME-24, no. 5, pp. 449–456, Sep. 1977.

[5] D. Comminiello, M. Scarpiniti, L. A. Azpicueta-Ruiz, J. Arenas-Garcia, and A. Uncini, “Functional link adaptive filters for nonlinear acoustic echo cancellation,” IEEE Trans. Audio, Speech, Language Process., vol. 21, no. 7, pp. 1502–1512, Jul. 2013.

[6] R. Guo and L. S. DeBrunner, “Two high-performance adaptive filter implementation schemes using distributed arithmetic,” IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 58, no. 9, pp. 600–604, Sep.

[7] B. K. Mohanty and P. K. Meher, “A high-performance energy-efficient architecture for FIR adaptive filter based on new distributed arithmetic formulation of block LMS algorithm,” IEEE Trans. Signal Processvol. 61, no. 4, pp. 921–932, Feb. 2013.

[8] S. Y. Park and P. K. Meher, “Efficient FPGA and ASIC realizations of a DA-based reconfigurable FIR digital filter,” IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 61, no. 7, pp. 511–515, Jul. 2014.

[9] G.-H. Long, F. Ling, and J. G. Proakis, “The LMS algorithm with delayed coefficient adaptation,” IEEE Trans. Acoust., Speech, Signal Process., vol. 37, no. 9, pp. 1397–1405, Sep. 1989.

[10] P. Kabal, “The stability of adaptive minimum mean square error equalizers using delayed adjustment,” IEEE Trans. Commun., vol. COM-31, no. 3, pp. 430–432, Mar. 1983.

[11] P. K. Meher and S. Y. Park, “Area-delay-power efficient fixed-point LMS adaptive filter with low adaptation-delay,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 22, no. 2, pp. 362–371, Feb. 2014.

[12] Y.-H. Chen, J.-N. Chen, T.-Y. Chang, and C.-W. Lu, “High-throughput multistandard transform core supporting MPEG/H.264/VC-1 using common sharing distributed arithmetic,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 22, no. 3, pp. 463–474, Mar. 2014.

[13] M. Martina, G. Masera, M. R. Roch, and G. Piccinini, “Result-biased distributed-arithmetic-based filter architectures for approximately computing the DWT,” IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 62, no. 8, pp. 2103–2113, Aug. 2015.