

Optimized FPGA-Based Modified MFCC Architecture Using Time Division Multiplexed FFT Processing

M. Muthumari^{ib}, and R. S. Valarmathi^{ib}

Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology, Tamil Nadu, India

Corresponding author: M Muthumari. (e-mail: muthu2s94@gmail.com), **Author(s) Email:** R.S.Valarmathi (e-mail: drsrvalarmathi@veltech.edu.in)

Abstract Advances in real-time speech recognition have led to an increasing demand for compact as well as energy-efficient MFCC hardware capable of generating accurate and consistent speech features with low computational latency. Conventional MFCC solutions frequently necessitate specialized processing resources and several math operations, resulting in increased size, power consumption, and hardware complexity. To address these restrictions, this paper proposes a modified MFCC architecture that incorporates dual window operation, a Radix-2 Time Division Multiplexed FFT (TDM FFT), and an Approximate Radix-4 Distributed Arithmetic (DA) multiplier. The proposed framework utilizes shared computational abilities for speech framing, dual window operations, FFT calculation, magnitude prediction, Mel filter bank analysis, logarithmic conversion, as well as MFCC coefficient generation. A Xilinx KV260 Zynq UltraScale+ FPGA is used to implement the standard architecture, and 45 nm CMOS ASIC synthesis is used for further assessment. The MFCC coefficients produced by MATLAB are compared with the appropriate Verilog outputs to perform functional validation. In comparison to the standard architecture, experimental results show a 50% reduction in LUT/ALM utilization, a 60% reduction in DSP resource usage, and a 45% reduction in power consumption. ASIC synthesis achieves a roughly 50% reduction in core silicon area while further reducing processing latency from 1024 to 320 clock cycles, or a 69% reduction. Time-division multiplexing, approximate distributed arithmetic processing, and efficient computing resource sharing all contribute to these gains. The findings show that the suggested architecture strikes a good balance among MFCC feature characteristics, processing speed, power consumption, as well as hardware efficiency. Thus, for low-power, low-latency, along with resource constrained real-time voice recognition applications, the proposed architecture offers a feasible and scalable approach.

Keywords Approximate Radix-4 DA Multiplier, Feature extraction, low latency, MFCC, TDM FFT

1. Introduction

Smart automation, surveillance, adaptive systems, and human-machine interaction all make extensive use of speech recognition, which necessitates low-latency and energy-efficient processing. Speech processing research using FPGAs and ASICs has increasingly focused on attaining real-time operation while reducing computational along with hardware costs. Liu [1], Haq [2], Khamlich [3], and Bibin [4] illustrated hardware-driven speech recognition with Zynq/FPGA platforms, MFCC DTW, SVM/MLP, and then pipelined MFCC processing, respectively. Studies [5]-[8] examined speech feature characterization and computationally efficient MFCC extraction, whereas [9]-[12] investigated real-time FPGA interpretation, machine learning-driven speech analysis, including hardware software co-design.

Individual computing blocks were also improved, such as resource-constrained FFT processing [13], lightweight speech recognition processes [14], low power MFCC-driven detection [15], Booth encoding [18], approximate multiplication [19], programmable speech segmentation [20], lightweight hardware analysis [21], and hybrid neural networks [22]. Similar research has been done on FPGA-driven feature extraction [23], frame selection [24], approximation arithmetic [25], feature training and ranking [26], [27], robust voice recognition [28], standard imputation optimization [29], and efficient feature extraction [30]. While most of this research optimizes a particular algorithmic and architectural component, taken as a whole, they show significant advancements in speech processing technology. Also, some hardware designs [16], [17] require many resources or don't scale well, making them less useful for speech recognition.

Existing MFCC architectures [33-35] based on FPGAs [36], [37] have a number of drawbacks. While TDM FFT minimizes resource redundancy at the potential expense of scheduling and latency overhead, parallel FFT implementations offer great throughput but demand significant hardware resources. While approximation multipliers minimize area, latency, and power at the expense of numerical accuracy, distributed arithmetic minimizes the multiplication complexity but can increase memory needs. Therefore, rather than focusing on resource utilization, latency, power, and feature accuracy, previous techniques typically optimize separate components. Existing methods vary significantly in terms of FPGA/ASIC platform [38-40], FFT setup, arithmetic precision, and MFCC parameters, as Table 1 summarizes. Three main gaps remain in the literature. First, traditional parallel MFCC increases hardware consumption. Second, resource sharing is not paired with efficient pipelining. Third, although approximate multipliers can lower power consumption along with arithmetic complexity, little research has been done on how they integrate with TDM based MFCC processing and how this affects feature quality.

Table 1: Comparison of Hardware Architectures for MFCC Feature Extraction

Ref	Architecture / Approach	Platform	FFT / Bit Width	MFCC Configuration	Resource / Frequency / Latency	Power	Approximation	Major Limitation
[1]	FPGA based voice control framework	Zynq FPGA	—	MFCC based	Hardware accelerated	—	No	Focuses mainly on system level voice control
[2]	MFCC + DTW recognition	FPGA	—	MFCC + DTW	—	—	No	Limited arithmetic/hardware optimization
[4]	Pipelined MFCC extraction	FPGA	FFT based	MFCC	Low latency	—	No	Hardware resource optimization is limited
[13]	Single butterfly FFT	FPGA	FFT based	Applicable to MFCC	Reduced FFT resources	—	No	Does not optimize complete MFCC arithmetic
[15]	MFCC + Binary CNN	FPGA /ASIC	—	MFCC + CNN	Low power	Low	Binary computation	Recognition architecture rather than MFCC optimization
[18]	Booth multiplier optimization	FPGA	—	—	Reduced arithmetic resources	—	No	Limited to multiplier architecture
[19]	Approximate multiplier	FPGA /ASIC	—	DSP applications	Reduced latency	Reduced	Yes	Not specifically optimized for MFCC
[25]	Approximate multiplication	FPGA	—	—	Reduced complexity	Reduced	Yes	Accuracy complexity trade off not evaluated for MFCC
[29]	Optimized FPGA multiplier	FPGA	—	DSP applications	Reduced delay	—	—	Focuses on multiplier rather than complete MFCC pipeline
Proposed	TDM FFT + Approximate Radix-4 DA + dual window MFCC	FPGA	Radix-2 TDM FFT / fixed point	Dual window MFCC	Reduced resources and latency	Reduced	Yes	Integrated optimization of FFT, multiplication, memory reuse, and windowing

To address such constraints, this paper offers an FPGA-based modified MFCC architecture that includes a Radix-2 TDM FFT, an Approximate Radix-4 Distributed Arithmetic multiplier, with dual window processing. The suggested architecture for real-time speech processing utilizes computational reuse, improved arithmetic, and optimum memory utilization to attain a balanced improvement in hardware efficiency, latency, power consumption, along with MFCC feature quality. The remaining sections of this paper are organized as follows. The detailed construction and design of the suggested MFCC feature extraction framework are presented in Section 2. The simulation's outcomes and the hardware synthesis analysis are discussed in Section 3. The work is concluded in Section 4, which also discusses future research opportunities.

II. Method

The Mel Frequency Cepstral Coefficient (MFCC) method is a popular way to get speech features because it shows how the human ear perceives sound and gives an outline of the speech spectrum. Fig. 1 shows that the input speech signal goes through several steps of processing to get useful acoustic features, as follows.

A. Pre emphasis

A high-pass filter is used to improve and regulate the input signal to the spectrum in Eq. (1) [2].

$$A(n) = i(n) - a * i(n - 1) \quad (1)$$

Where $A(n)$ is amplified signal of input signal $i(n)$ of the pre-emphasis. Assuming $a = 0.95$, 95% of any given sample.

B. Framing

To ensure quasi-stationary, the amplified signal is segmented into small frames, usually lasting 20 to 30 milliseconds.

$$x_m(n) = x(mR + n), 0 \leq n \leq N_f \quad (2)$$

$$O = \left(1 - \frac{R}{N_f}\right) * 100\% \quad (3)$$

The continuous/discrete speech signal $x(n)$ is divided into short time frames by Eq. (2) [20], where m is the frame count, R is the frame shift or hop size, and N_f is the number of samples that exist in each frame. The percentage of overlap O among successive speech frames is determined by Eq. (3) [24]. For a given speech duration, a lower hop size results in more overlap and, hence, more frames. The FFT/MFCC engine gets fresh frames at a pace determined by the overlap.

C. Dual Windowing

Each frame is windowed to minimize spectral leakage using dual windowing. It has the option of choosing the windowing method based on user preference, as shown in Fig. 2. If user preference is 0, then the feature extraction process uses a Hamming window $w_H(n)$ as given in Eq. (5) [7], and if it's 1 then it will be processed with a Hanning window $w_{Ha}(n)$ as given in Eq. (6) [7].

$$y(n) = x(n).w(n) \quad (4)$$

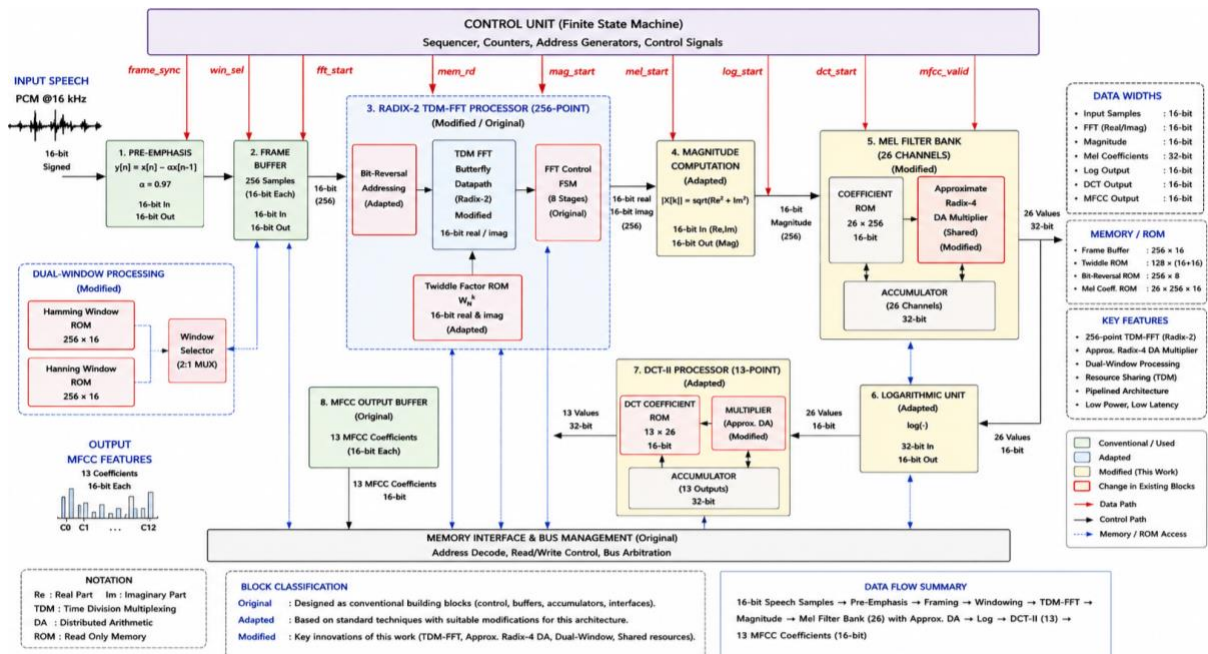


Fig. 1. Proposed MFCC hardware architecture integrating Radix-2 TDM FFT and an approximate Radix-4 distributed arithmetic multiplier

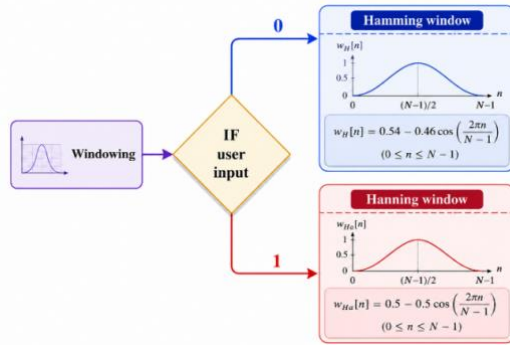


Fig. 2. Dual window selection flow between Hamming and Hanning window functions

$$w_H(n) = 0.54 - 0.46 * \cos(2\pi n / (N - 1)) ; 0 \leq n \leq N - 1 \quad (5)$$

$$w_{Ha}(n) = 0.5 - 0.5 * \cos(2\pi n / (N - 1)) ; 0 \leq n \leq N - 1 \quad (6)$$

$$w(n) \in \{w_H(n), w_{Ha}(n)\} \quad (7)$$

Here the $y(n)$ is output as given in Eq. (4) [2] after applying the windowing sequence $w(n)$ to the input sequence $x(n)$, n is the measure of the sample, and N is the total amount of samples in the window. At both ends, the Hanning window gradually tapers to zero, guaranteeing that $w(0)=w(n-1)=0$. By minimizing discontinuities at the signal borders, this feature lessens spectral leakage, which usually results from sudden boundary transitions. The dual windowing characteristic is expressed in Eq. (7) as the author's implementation. Depending on the chosen operating mode, the hardware can select between Hamming and Hanning coefficients. The equation should only be applied when both options are employed in the RTL. Different trade-offs between frequency selectivity and spectrum leakage are offered by the Hamming and Hanning windows. While the Hanning window has a wider main lobe with efficient leakage reduction and smooth spectral features, the Hamming window often offers lower side lobe levels, providing greater suppression of spectral leakage. The choice of window influences the practical separation and graphical representation of spectrum components. Both windows are included in the suggested architecture to offer flexibility in MFCC feature extraction under various speech signal conditions.

D. Fast Fourier Transform (FFT)

$$Y(K) = FFT(y(n)) \quad (8)$$

$$Y(K) = \sum_{n=0}^{N-1} y(n) e^{-i2\pi kn/N}, 0 \leq k < N - 1 \quad (9)$$

$$W_N^K = e^{-i2\pi k/N} = \cos\left(\frac{2\pi k}{N}\right) - j\sin\left(\frac{2\pi k}{N}\right) \quad (10)$$

FFT converts a time-domain signal $y(n)$ to frequency domain $Y(K)$ as given in Eq. (8) [2]. This discrete Fourier transform serves as the FFT's mathematical foundation in Eq. (9) [13]. Instead of doing the summation directly, the hardware TDM FFT performs the equivalent transformation using a series of Radix-2 butterfly operations. The standard FFT twiddle factor is W_N^K is given in Eq. (10) [13]. For every butterfly function, the complex input is rotated.

1. TDM FFT

Each speech frame is subjected to an FFT to extract spectral information for MFCC computation. For hardware implementations, a TDM FFT provides an extremely resource-efficient option, particularly on

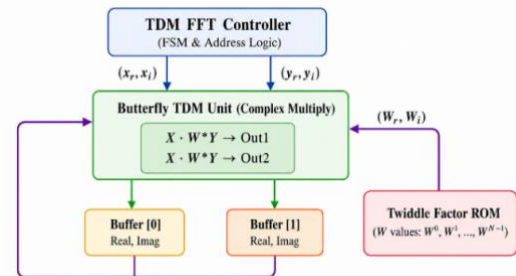


Fig. 3. Radix-2 TDM-FFT architecture using a shared butterfly processing unit

FPGA systems.

$$L = \log_2(N) = \log_2(256) = 8 \quad (11)$$

$$N_b = \left(\frac{N}{2}\right) \log_2(N) = 128 * 8 = 1024 \quad (12)$$

Eight steps L are needed for a 256-point Radix-2 FFT proposed formulations as given in Eq. (11) [13]. Until the full 256-point transform is achieved, each step splits the transform into smaller two-point combinations. Each of the eight phases of a 256-point Radix-2 FFT uses 128 butterflies, for a total of 1024 butterfly operations N_b proposed formulations as given in Eq. (12) [13]. By time-sharing these operations via a shared butterfly data path, the TDM architecture minimizes redundant arithmetic hardware. By reusing computational units over time, the TDM Radix-2 FFT architecture, depicted in Fig. 3, computes the FFT with minimal hardware overhead for a four-point transform. The architecture comprises the following components:

1. Input Register Bank: Temporarily holds incoming real or complex signal samples.
2. Radix-2 Butterfly Unit: Executes the core arithmetic operations required for FFT computation.
3. Twiddle Factor ROM or Generator: Provides the right complex exponentially varying weighting.

- Multiplexers: Use shared hardware to make it easier to dynamically route intermediate data among stages.
- Control FSM (Finite State Machine): Sets the order, timing, as well as memory addresses for the flow of data during the computation.

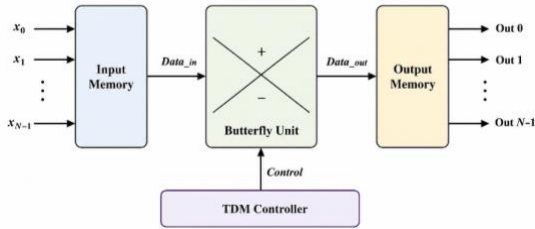


Fig. 4. Four point TDM FFT architecture illustrating sequential butterfly processing

The 4-point Time Division Multiplexed (TDM) FFT framework in Fig. 4 shows that processing begins with Stage 1, where input parameters are grouped into (x_0, x_2) and (x_1, x_3) to do two butterfly computations. The outputs of these butterfly operations are either stored in temporary registers or utilized again right away to save memory. In Stage 2, the outcomes from Stage 1 are integrated by adding the necessary twiddle factors. This lets butterfly calculations run one after the other, which saves a lot of hardware and power.

2. Approximate Distributed Arithmetic (DA) based Radix-4 multiplier

The TDMFFT layout, that utilizes an approximate Distributed Arithmetic (DA) based Radix-4 multiplier, as shown in Fig. 5, has resulted in a lot of important enhancements. As described in Eq. (13) [23], distributed arithmetic has a way to utilize the inner product $y(n)$.

$$y(n) = \sum_{i=0}^{N-1} w_i(n) \cdot x_i(n) \quad (13)$$

Within this formulation, $x_i(n)$ relates to the input samples, and this can change over time or stay the

same, and $w_i(n)$ represent the coefficients. This

Table 3: Numerical error characterization of the proposed approximate Radix-4 multiplier

Metric	Proposed Approximate Radix-4 Multiplier
Operand precision	16×16 bits
Product precision	32 bits
Radix	4
Approximation threshold	4 LSBs
Number of test cases	100,000
Erroneous cases	93,742
Error Rate	93.742%
Mean Error Distance	119,606
Normalized Error Distance	5.56957×10^{-5}
Worst Case Error	163,778

results in m memory lookups (implemented as LUTs) and m additions for every output sample computed. While this architecture avoids costly multipliers and simplifies hardware, it leads to increased memory access, higher power consumption, and greater delay and area requirements, especially when higher precision (larger m) is needed. Radix-4 Booth encoding groups three bits at a time, consisting of two newer bits and one overlapping bit from the already existing group, and encodes them into a single Booth digit as given in Table 2 where M denotes Multiplicand. The overlapping bit ensures accurate recoding of the multiplier segments. The first step in deriving the Booth-based Distributed Arithmetic (DA) equation is to define the most important parameters: let m is the entire number of bits in the coefficient, $w_i^j(n)$ denote the j^{th} bit of the coefficient $w_i(n)$ and express it in 2's complement form as given in Eq. (14) [23].

$$w_i(n) = -w_i^{m-1}(n) \cdot 2^{m-1} + \sum_{j=0}^{m-2} w_i^j(n) \cdot 2^j \quad (14)$$

This equation shows how to break down a signed number into its standard binary form. By applying radix-4 Booth encoding, every 2 bits of the coefficient are put together with 1 bit from the previous segment that overlaps, which lets us rewrite the coefficient as shown in Eq. (15) [23].

$$w_i(n) = \sum_{j=0}^{\left(\frac{m}{2}\right)-1} w_i^{1j}(n) \cdot 2^{2j} \quad (15)$$

where the recoded Booth digit $w_i^{1j}(n)$ is calculated by Eq. (16) [23]:

Table 2. Radix-4 Booth encoding and corresponding partial product operations

Bit groups (3bits)	Booth digit	Operation
000	0	$0 \times M$
001	+1	$+1 \times M$
010	+1	$+1 \times M$
011	+2	$+2 \times M$
100	-2	$-2 \times M$
101	-1	$-1 \times M$
110	-1	$-1 \times M$
111	0	$0 \times M$

$$w_i^{1j}(n) = -2w_i^{2j+1}(n) + w_i^{2j}(n) + w_i^{2j-1}(n) \quad (16)$$

This recoding decreases the number of sufficient partial products by around half, which makes the multiplication process in DA implementations more efficient. The coefficient bit $w_i^{-1}(n)$ is set to zero, as shown in Eq. (17) [23].

$$w_i^{-1}(n) = 0 \quad (17)$$

This makes sure that everything is lined up correctly during Booth recoding. Every single Booth recoded digit $w_i^{1j}(n)$ obtained from the radix-4 encoding process can take on whole numbers from the set as $w_i^{1j}(n) \in \{-2, -1, 0, 1, 2\}$. The Booth dependent DA Multiplier decreases the amount of partial products by almost half compared to regular multipliers, as shown in Fig. 6. To generate partial products, Booth digits have been multiplied using input samples, after that barrel shifters are used to scale them, and finally an adder tree or just accumulator adds them all together. Table 3 shows the numerical error characterization of the proposed multiplier with 100,000 randomly selected signed 16-bit operand combinations. Even though the approximate multiplier has a 93.742% erroneous case rate, the error rate by itself does not indicate application-level degradation; the measured MFCC level error metrics show the error magnitude and its propagation.

E. Mel Filter bank Processing

The power spectrum passes through a sequence of nonlinearly spaced triangular filters using the Mel scale. Next, the Mel for the selected frequency f in Hz $Mel(f)$ has been calculated using Eq. (18) [2]. Eq. (19) [17] is

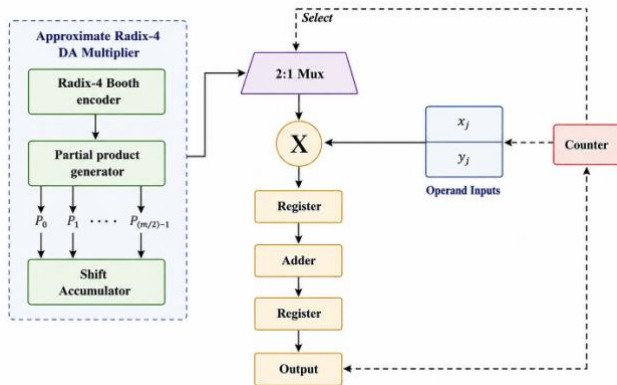


Fig. 5. FPGA architecture of the proposed approximate Radix-4 distributed arithmetic multiplier

the Mel transformation in reverse. The Mel filter exceed points are M_{max} , M_{min} uniformly distributed throughout the Mel domain by Eq. (20) [17]. The number of Mel filters is indicated by B .

Algorithm 1. Hardware Oriented MFCC Processing

Input: speech_sample[n], sample_valid, reset

Output: MFCC[0:12], mfcc_valid

1. Initialize registers, memories, counters, FSM, and valid flags
2. if sample_valid = 1 then
3. frame_memory[sample_count] = Quantize(speech_sample)
4. sample_count = sample_count + 1
5. if sample_count = 255 then
6. frame_valid = 1
7. sample_count = 0
8. end if
9. end if
10. if frame_valid = 1 then
11. for n := 0 to 255 do
12. windowed_data[n] = frame_memory[n] × Hamming_ROM[n]
13. FFT_memory[BitReverse(n)] = FixedPointScale(windowed_data[n])
14. end for
15. end if
16. for stage = 0 to 7 do
17. for butterfly = 0 to 127 do
18. Read A, B, and twiddle factor W
19. T = FixedPointMultiply(B, W)
20. X_A = FFT_Scale(A + T)
21. X_B = FFT_Scale(A - T)
22. Write X_A and X_B to FFT_memory
23. end for
24. end for
25. for k = 0 to 127 do
26. magnitude[k] = ComputeMagnitude(FFT_memory[k])
27. end for
28. for m = 0 to 25 do
29. mel_accumulator[m] = 0
30. for k = FilterStart[m] to FilterEnd[m] do
31. product = Approx_Radix4_Multiplier(magnitude[k], MelCoefficient[m][k])

$$Mel(f) = 2595 * \log_{10}(1 + \frac{f}{700}) \quad (18)$$

$$f = 700 * (10^{\frac{Mel}{2595}} - 1) \quad (19)$$

$$M_i = M_{min} + i * \left(\frac{M_{max} - M_{min}}{B+1} \right), i = 0, \dots, B+1 \quad (20)$$

F. Discrete Cosine Transform (DCT)

Changes the log mel spectrum towards the cepstral domain to separate the features, which gives a small set of coefficients (usually the first 12–13) that show the speech's spectral envelope. By compressing the dynamic range of the Mel filter energies E_i , the logarithm improves the representation's suitability L_i for speech feature extraction as given in Eq. (21) [2]. When the energy is zero, ϵ avoids an undefined logarithm. The DCT utilized to explicate the logarithmic Mel filter bank energies is defined by the Eq. (22) [2]. The MFCC feature vector is formed by the resultant coefficients C_q , Where $q = 0, 1, \dots, 12$.

```

32. mel_accumulator[m] =
    mel_accumulator[m] + product
33. end for
34. mel_energy[m]
    =FixedPointScale(mel_accumulator[m])
35. end for
36. for m = 0 to 25 do
37.   log_energy[m] =
    Log_Approximation(mel_energy[m])
38. end for
39. for c = 0 to 12 do
40.   DCT_accumulator = 0
41.   for m = 0 to 25 do
42.     DCT_accumulator = DCT_accumulator
    +FixedPointMultiply(log_energy[m],
    DCT_Coefficient[c][m])
43.   end for
44.   MFCC[c] =
    FixedPointScale(DCT_accumulator)
45. end for
46. Apply pipeline delay and valid signal
    propagation
47. mfcc_valid = 1
48. Output MFCC[0:12]
49. Update counters, memory addresses, FSM
    states, and prepare the next frame
50. End Algorithm

```

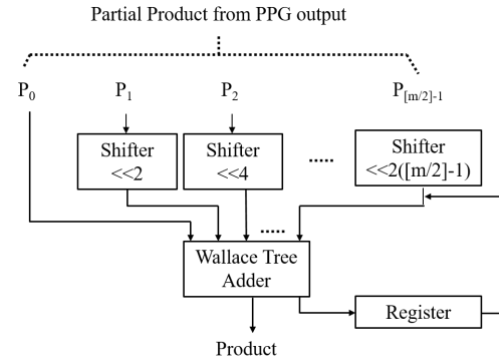


Fig. 6. Partial product generation and accumulation structure of the proposed Radix-4 multiplier

Table 4: Classification and contribution of individual components in the proposed MFCC hardware architecture

Component	Classification	Contribution in this paper
Pre emphasis	Adapted	Speech preprocessing
Frame buffering	Adapted/implementation	Stores 256 sample speech frames
TDM FFT	Adapted/modified	Time shared FFT computation for reduced hardware
256 point FFT architecture	Implementation specific	FPGA realization of the MFCC spectral analysis stage
Radix-4 multiplier	Adapted/modified	Used to reduce multiplication complexity
Mel filter bank	Adapted	26 channel spectral feature extraction
Log computation	Adapted/implementation	Hardware oriented logarithmic approximation
DCT II	Adapted	Generate 13 Coefficients
TDM FFT + Radix-4 multiplier integration	Main contribution	Hardware co-design of the two computational optimizations within one MFCC engine
Resource sharing/control organization	Original/modified architecture	Coordinates the TDM FFT and Radix-4 arithmetic while producing MFCC coefficients
Complete FPGA MFCC accelerator	System level contribution	End to end hardware implementation from speech frame to C0–C12

The suggested MFCC architecture is to make streaming processing more efficient while keeping the hardware simple and the contribution of the proposed work is given in Table 4. Algorithm 1 shows how data flows in a sequence and points out the hardware-oriented tasks that the architecture uses.

$$L_i = \ln(E_i + \varepsilon) \quad (21)$$

$$C_q = \sum_{i=0}^{B-1} L_i * \cos\left[\left(\frac{\pi q}{B}\right) * \left(i + \frac{1}{2}\right)\right] \quad (22)$$

III. Results

A. Implementation of Hardware MFCC Architecture

The proposed framework, which combines TDM FFT and MFCC processing, is being developed and tested on both Xilinx and Altera FPGA platforms to assess performance and portability. Experimental synthesis on

the Xilinx Virtex 6 XC6VLX240T device showed a significant reduction in resource utilization compared with traditional FFT MFCC implementations. The architecture was also synthesized on the Xilinx KV260 to show that it can be used in different situations. Fig. 7 shows the full data flow through the MFCC processing pipeline. The simulation waveform shows how the whole MFCC extraction pipeline works. As shown in Fig. 8, the RTL schematic presents a hardware-efficient implementation of the MFCC architecture. The conventional architecture is characterized in terms of an RTL based MFCC computational unit with conventional arithmetic operators. The proposed layout uses the same MFCC configuration along with arithmetic precision, but adds the Radix-2 TDM FFT data pipeline and Radix-4 multiplier to improve resource allocation and performance in computation.

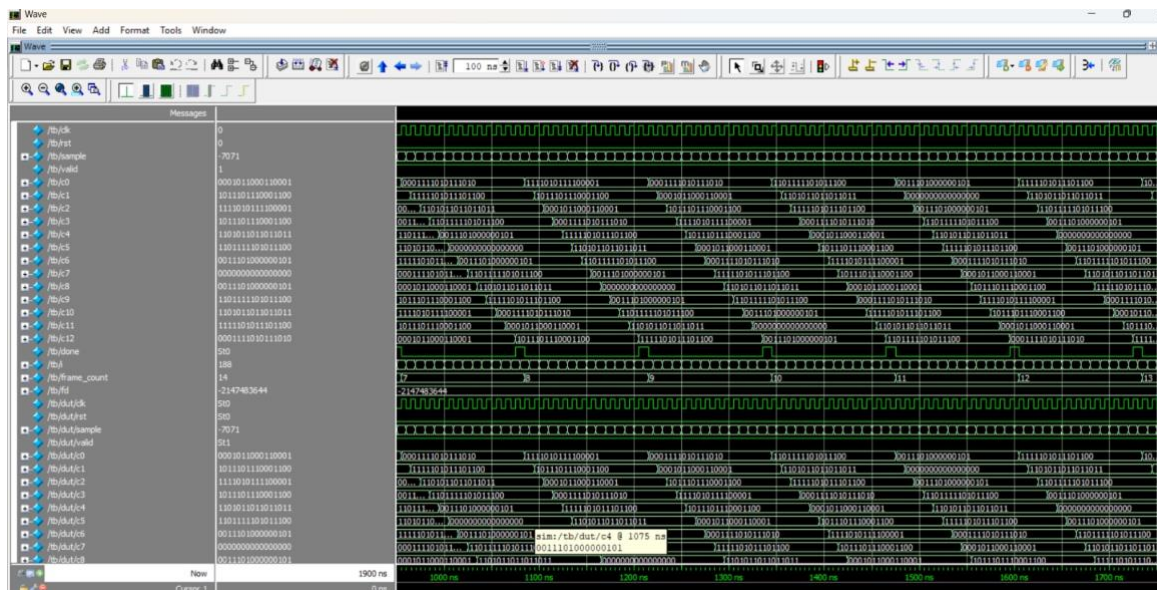


Fig. 7. RTL simulation waveform of the proposed MFCC processing architecture

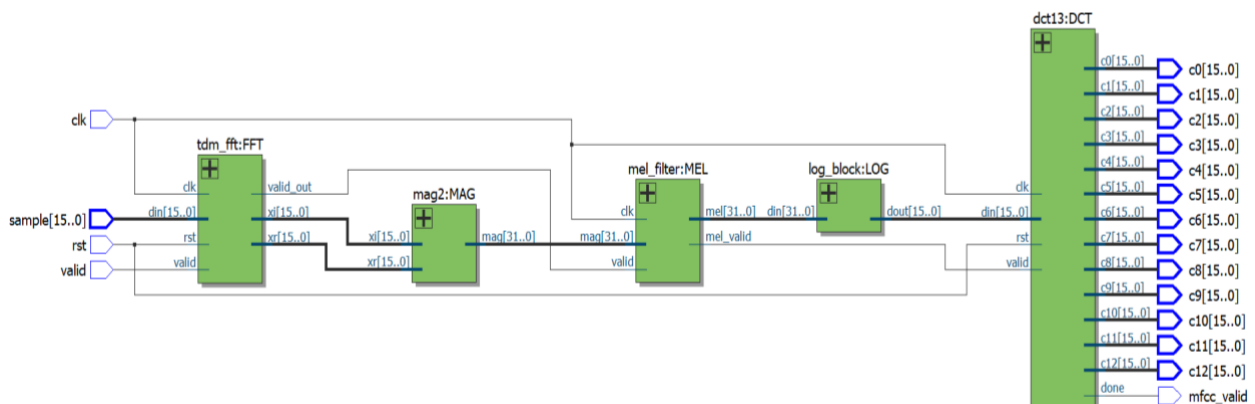


Fig. 8. RTL schematic of the proposed hardware MFCC processing pipeline

Table 5: ASIC synthesis comparison between conventional and proposed MFCC architectures using 45 nm CMOS technology

Parameter	Conventional MFCC	Proposed MFCC	Improvement
Technology	45 nm CMOS	45 nm CMOS	—
Area (mm ²)	1.85 mm ²	0.92 mm ²	↓ 50%
Gate Count	182 K	96 K	↓47%
Power	42.5 mW	21.3 mW	↓ 49%
Frequency	180 MHz	265 MHz	↑ 47%
Latency	1024 cycles	320 cycles	↓ 69%
Throughput	15 k samples/s	48 k samples/s	↑ 3.2×

A MATLAB model is made to test how well the suggested fixed-point MFCC framework works. Fig. 9 demonstrates high consistency across MATLAB and Verilog MFCC outputs, with $R^2=1.0000$ and practically zero frame-wise MAE. Minor coefficient differences are due to fixed-point quantization, truncation, plus pipeline effects, demonstrating the functional consistency with the proposed design. Even though there are some small differences, the overall deviation is still within acceptable limits, which shows that the proposed architecture is correct and reliable. ASIC and FPGA analyzes comprise the hardware evaluation. The AMD/Xilinx KV260 Zynq UltraScale+ FPGA implementation is shown in Table 6, the 45 nm CMOS ASIC synthesis is reported in Table 5, and a reference comparison with Virtex-6 FPGA implementations is given in Table 7. Results from multiple platforms are not directly integrated; improvement percentages are only computed for designs assessed using the same platform and configuration.

B. Speech Recognition Validation

To determine the efficiency of the proposed MFCC framework for speech recognition, the resulting MATLAB MFCC features have been compared to the NOIZEUS speech dataset by implementing an LSTM-based classifier. Speech signals were evaluated in both clean and noisy situations, with noisy signals handled using the suggested GQPSO [31] noise cancellation phase before MFCC extraction. 200 NOIZEUS audio samples across various speakers, including babbling noise at 5 dB SNR, were used for speech recognition evaluation. We used 20% of the dataset for testing and the remaining 80% for training. With the help of the Adam optimizer, a batch size of 1 as well as a maximum total of 200 training iterations, the collected MFCC

Table 6: FPGA implementation comparison of conventional and proposed MFCC architectures on the AMD/Xilinx KV260 Zynq UltraScale+ platform

Parameter	Conventional MFCC	Proposed MFCC
FPGA Device	AMD/Xilinx KV260 (Zynq UltraScale+)	AMD/Xilinx KV260 (Zynq UltraScale+)
FFT Size	256 point	256 point
FFT Architecture	Conventional Radix-2	Radix-2 TDM
Input Precision	16 bit	16 bit
MFCC Coefficients	13	13
Mel Filters	26	26
LUTs	18,500	9,200
FFs	22,300	11,800
DSP Slices	120	48
BRAM	32	18
Max Frequency	160 MHz	245 MHz
Power	3.8 W	2.1 W

features were transferred to an LSTM classifier featuring one hidden layer plus 128 hidden units. To provide a fair comparison, both MATLAB-generated and FPGA-generated MFCC features used the identical classifier architecture and test samples. The extracted MFCC features were fed into the trained LSTM-based classification algorithm [32], and the predicted speech labels were then compared to their corresponding ground truth labels.

IV. Discussion

Table 5 illustrates the findings of the ASIC synthesis, which compares the suggested MFCC architecture to a standard MFCC deployment that uses 45 nm CMOS technology. The standard MFCC design takes up 1.85 mm² of silicon and needs almost 182K logic gates. In contrast, the proposed architecture reduces the area to 0.92 mm², achieving nearly 50% area reduction. This improvement is obtained by employing a TDM FFT structure that enables hardware reuse and by using a DA-based radix-4 multiplier, which reduces the dependency on conventional multipliers. The hardware is even less complicated because the number of gates goes down from 182K to 96K, which is about a 47%

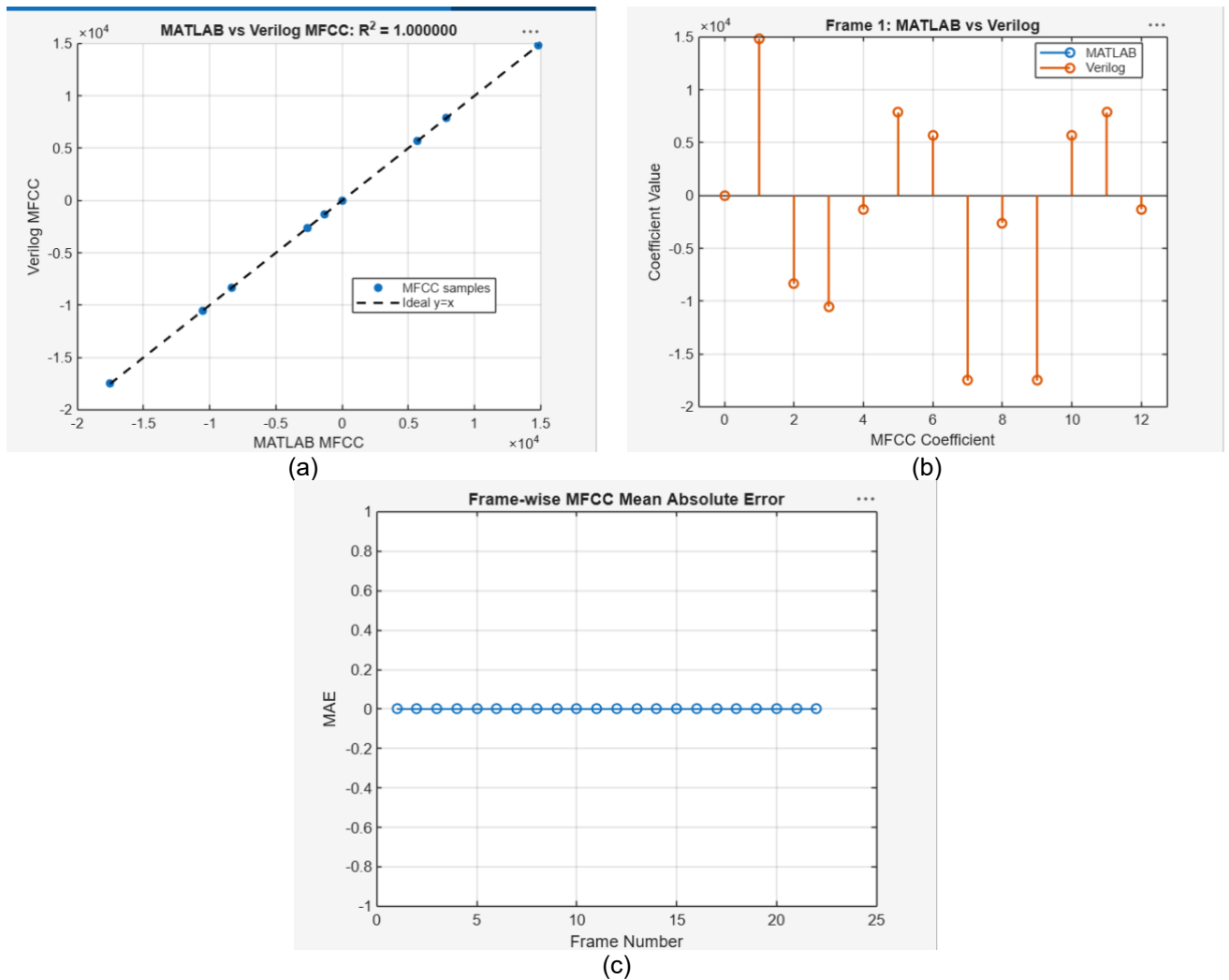


Fig. 9. MATLAB–Verilog validation of the proposed MFCC architecture: (a) correlation between MATLAB and Verilog MFCC outputs; (b) frame wise comparison of MFCC coefficients; and (c) frame wise mean absolute error/bit accurate validation

drop. The proposed design also uses a lot less power, going from 42.5 mW in the standard architecture to 21.3 mW, which is almost 50% less power. This decrease is mostly due to less switching activity, a decrease in active logic elements, and less need for DSP blocks because of the DA-based computation and TDM processing scheme. The proposed design has a highest frequency of 265 MHz in the ASIC synthesis platform, whereas the conventional architecture has a highest frequency of 180 MHz. This means that the proposed design is about 47% faster. The latency has also been reduced from 1024 clock cycles to 320 cycles, which is about a 69% cut. The whole framework throughput goes up from 15,000 samples per second to 48,000 samples per second, which means it can process almost 3.2 times more data. Table 6 indicates that the proposed MFCC architecture uses almost 50% less LUT and flip-flop space, about 60% less DSP slice space, and about 45% less power.

The architecture also raises the operating frequency by over 50%, which shows that it is good for FPGA-based real-world speech processing applications. The results show that the suggested MFCC architecture makes hardware more efficient for real-world speech processing.

Table 7 shows how the proposed MFCC architecture stacks up against a few other FPGA-based implementations. To make it fair, all of the comparisons are done on a normalized Xilinx platform. Wassi et al. (2015) [16] designed a pipelined MFCC that runs at 150 MHz and uses about 14,000 LUTs as well as 50 DSP blocks. Dao et al.'s [17] MFCC IP core performs at 120 MHz and utilizes about 15,000 LUTs and 64 DSP blocks, which shows how much it relies on multiplier-driven computations. Using LUTs, DSPs, frequency, latency, power, FFT size, precision, FPGA generation, and MFCC configuration, the suggested architecture is

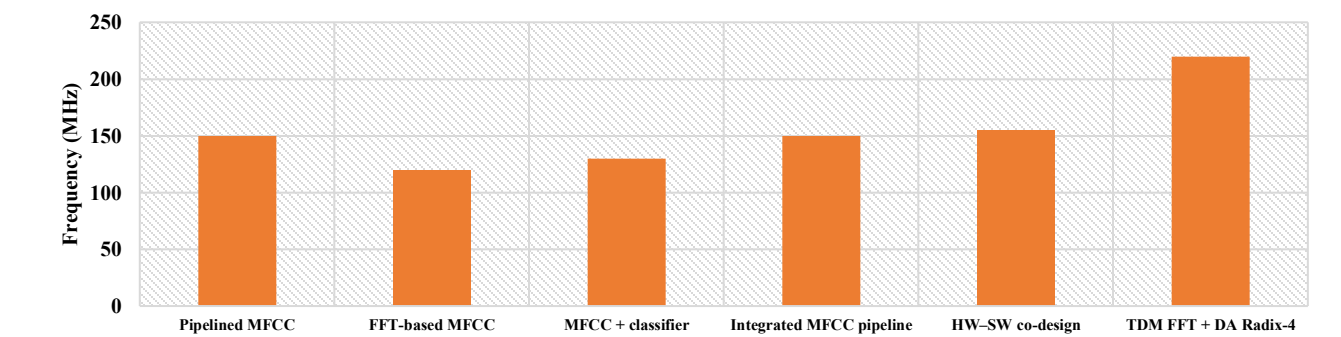


Fig. 10. Comparison of operating frequencies for the proposed and previously reported MFCC FPGA architectures on/normalized to the Xilinx Virtex-6 XC6VLX240T platform

contrasted with FPGA-based MFCC implementations documented in [3], [4], [9], [16], [17], and related research. However, these factors cannot be compared separately because they depend on the target FPGA, synthesis technique, and architectural structure. In order to reduce hardware duplication through resource reuse, the suggested architecture employs a 256-point Radix-2 TDM FFT with a shared butterfly datapath at the expense of more processing cycles. A speech system based on Zynq utilizes [1] about 13,000 LUTs along with 48 DSP blocks at a speed of about 130 MHz. It combines MFCC with classification; however, it still uses traditional

exhibits accurate and reliable classification performance with a level of accuracy of 97.06%, precision of 100%, and recall of 97.06%. The final F1-score of 98.51% attests to the excellent balance that exists between recall and accuracy. The proposed Time Division Multiplexed (TDM) MFCC architecture reduces both hardware utilization and processing latency by combining efficient resource sharing with pipelined execution. Although TDM typically serializes computations, the proposed scheduling strategy minimizes idle cycles and enables continuous data processing, allowing a single FFT processing unit to be

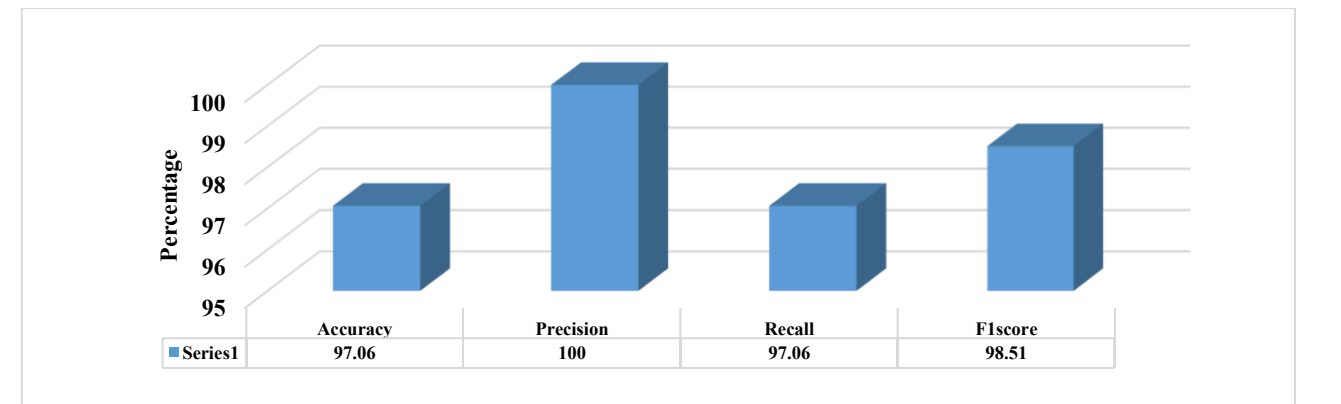


Fig. 11. Speech recognition performance of the LSTM classifier using MATLAB derived MFCC features

processing. Bibin et al. (2021) [4] use an integrated pipeline framework with about 12,000 LUTs as well as 48 DSP blocks to reach 150 MHz. Vreča et al. (2024) [11] use hardware-software co-design to cut resources down to about 8,000 LUTs and 46 DSP blocks, reaching 155 MHz. It only needs about 8,500 LUTs and 32 DSP blocks, but it performs at a higher frequency of 220 MHz, which shows that it is more efficient and faster than the previous one as given in Fig. 10. Recognition efficiency is evaluated through precision, recall, accuracy, and F1 score as shown in Fig. 11. The proposed approach

reused without significant performance degradation. The critical path mainly consists of the FFT butterfly computation, Approximate Radix-4 Distributed Arithmetic multiplier, and accumulation stages, where pipeline registers are inserted to reduce combinational delay and improve operating frequency.

Furthermore, shared memory buffers enable efficient reuse of intermediate data, reducing memory overhead and bandwidth requirements. While the approximate multiplier introduces a small computational error, its impact on MFCC feature quality is negligible, whereas

Table 7: FPGA synthesis comparison of the proposed MFCC architecture with previously reported implementations using the Xilinx Virtex-6 XC6VLX240T reference platform

Reference	LUTs	DSP	Frequency (MHz)	Architecture
Wassi et al. [16]	14000	50	150	Pipelined MFCC
Dao et al.[17]	15,000	64	120	FFT based MFCC
Zynq Speech System [1]	13,000	48	130	MFCC + classifier
Bibin et al. (2021)[4]	12,000	48	150	Integrated MFCC pipeline
Vreča et al. (2024) [11]	8000	46	155	HW SW co-design
Proposed Work	8,500	32	220	TDM FFT + DA Radix-4

the reductions in logic resources, DSP utilization, power consumption, and latency are significant. These architectural optimizations provide an effective tradeoff between throughput, hardware complexity, and computational accuracy, making the proposed design suitable for real-time FPGA-based speech processing applications.

V. Conclusion

In order to reduce hardware complexity while maintaining acceptable feature quality, this work provides a resource-efficient, VLSI-oriented MFCC architecture that integrates a 256-point Radix-2 TDM FFT and an approximate Radix-4 distributed arithmetic multiplier. FPGA synthesis on a Xilinx platform reduces LUTs by 29–50% when compared to architectures that use 12K to 15K LUTs and DSPs by 33 to 60% when compared to designs that use 46 to 64 DSP blocks. The maximum operating frequency is 220 MHz, which is up to 83% better than designs operating at 120 MHz and 42 to 47% better than 150 to 155 MHz architectures. By employing 45 nm CMOS for ASIC synthesis, silicon area is reduced from 1.85 mm² to 0.92 mm² (50%), gate count is reduced from 182 K to 96 K (47%), and power is reduced from 42.5 mW to 21.3 mW (50%). While delay drops from 1024 to 320 cycles (69%) and throughput increases from 15 ksamples/s to 48 ksamples/s (3.2×), operating frequency rises from 180 MHz to 265 MHz (47%). Overall, depending on the device and synthesis conditions, the design shows a workable tradeoff between hardware efficiency and numerical fidelity. Real-time hardware-in-the-loop validation, larger datasets, adaptive approximation, parallel/multicore architectures, low-power optimization, and a methodical comparison of feature quality and hardware efficiency are some of the upcoming projects. Despite the encouraging outcomes, there are a few things to be aware of. Cross-platform comparison is affected by differences in FPGA devices, FFT sizes, precision, synthesis tools, and implementation constraints;

therefore, reported resource, frequency, latency, and power values are not fully normalized. Speech recognition validation remains limited, and classification was performed using MATLAB-derived MFCC features. Furthermore, complete real-time end-to-end processing using live speech directly on the FPGA has not yet been demonstrated. These aspects are identified as important directions for future hardware-in-the-loop validation.

Funding

The authors declare that this research received no external funding.

Data Availability

The speech data used in this study were obtained from the publicly available NOIZEUS speech corpus. Processed data, MFCC feature vectors, and validation test vectors are available from the corresponding author upon reasonable request.

Code Availability Statement

The MATLAB scripts, Verilog/RTL implementation, synthesis configuration files, and supporting code used in this study are available from the corresponding author upon reasonable request.

Author Contribution

M. Muthumari: Conceptualization, Methodology, Hardware Design, Software, Validation, Data Analysis, Visualization, Writing Original Draft. R. S. Valarmathi: Supervision, Methodology, Formal Analysis, Validation, Writing Review and Editing. All authors have read and approved the final manuscript.

Declaration of Generative AI and AI-Assisted Technologies

The authors declare that no generative artificial intelligence or AI-assisted technologies were used in the preparation of this manuscript

Declarations

Ethical Approval

This research does not involve human participants, animals, or any clinical studies. Therefore, ethical approval was not required for this study.

Consent to Participate

Not applicable. This study did not involve human participants.

Consent for Publication Participants.

Not applicable. This study does not contain identifiable personal information or data from human participants requiring consent for publication

Competing Interests

The authors declare that there is no conflict of interest regarding the publication of this paper.

Acknowledgments

The authors have no acknowledgments to declare.

Declaration of Originality

The authors confirm that this manuscript is original, has not been published previously, and is not under consideration for publication elsewhere

Third Party Materials and Permissions

All figures and tables are original to this study unless otherwise indicated. Any adapted material has been appropriately cited and used in accordance with applicable permissions.

Dataset License / Usage Declaration

The publicly available dataset used in this study was accessed and used in accordance with the dataset provider's terms and conditions.

References

- [1] W. Liu, "Voice control system based on Zynq FPGA," *Journal of Physics: Conference Series*, vol. 1631, no. 1, p. 012177, 2020. doi: 10.1088/1742-6596/1631/1/012177
- [2] A. S. Haq, M. Nasrun, C. Setianingsih, and M. A. Murti, "Speech recognition implementation using MFCC and DTW algorithm for home automation," *Proceedings of the Electrical Engineering Computer Science and Informatics*, vol. 7, no. 2, pp. 78–85, 2020. doi: 10.11591/eecs.v7.2041
- [3] S. Khamlich, F. Khamlich, I. Atouf, and M. Benrabh, "Performance evaluation and implementations of MFCC, SVM and MLP algorithms in the FPGA board," *International Journal of Electrical and Computer Engineering Systems*, vol. 12, no. 3, pp. 139–153, 2021. doi: 10.32985/ijeces.12.3.3
- [4] S. P. S. Bibin, A. X. Glittas, and L. Gopalakrishnan, "A low latency modular level deeply integrated MFCC feature extraction architecture for speech recognition," *Integration*, vol. 76, pp. 69–75, 2021. doi: 10.1016/j.vlsi.2020.09.002
- [5] N. A. Al Hindawi, I. Shahin, and A. B. Nassif, "The exploitation of multiple feature extraction techniques for speaker identification in emotional states under disguised voices," in *Proc. 14th Int. Conf. Developments in eSystems Engineering (DeSE)*, 2021, pp. 269–273. doi: 10.1109/DeSE54285.2021.9719357
- [6] M. K. Gourisaria, R. Agrawal, M. Sahni, and P. K. Singh, "Comparative analysis of audio classification with MFCC and STFT features using machine learning techniques," *Discover Internet of Things*, vol. 4, no. 1, p. 1, 2024. doi: 10.1007/s43926-023-00049-y
- [7] H. Zhang, H. Li, W. Chen, and H. Han, "Feature extraction of speech signal based on MFCC," *Journal of Physics: Conference Series*, vol. 2584, no. 1, p. 012143, 2023. doi: 10.1088/1742-6596/2584/1/012143
- [8] Li Qin, Yuze Yang, Tianxiang Lan, Huifeng Zhu, Qi Wei, Fei Qiao, Xinjun Liu, and Huazhong Yang., "MSP MFCC: Energy efficient MFCC feature extraction method with mixed signal processing architecture for wearable speech recognition applications," *IEEE Access*, vol. 8, pp. 48720–48730, 2020. doi: 10.1109/ACCESS.2020.2979799
- [9] Deshmukh, Aditya, Arman Khan, Vedika Patil, and Kishor Bhargale. "Real Time FPGA Implementation for Speech Feature Extraction." In *2025 6th International Conference for Emerging Technology (INCET)*, pp. 1-7. IEEE, 2025. doi: 10.1109/INCET64471.2025.11140105
- [10] Abdusalomov, Akmalbek Bobomirzaevich, Furkat Safarov, Mekhriddin Rakhimov, Boburkhon Turaev, and Taeg Keun Whangbo., "Improved feature parameter extraction from speech signals using machine learning algorithm," *Sensors*, vol. 22, no. 21, p. 8122, 2022. doi: 10.3390/s22218122
- [11] J. Vreča, R. Pilipović, and A. Biasizzo, "Hardware–software co-design of an audio feature extraction pipeline for machine learning applications," *Electronics*, vol. 13, no. 5, p. 875, 2024. doi: 10.3390/electronics13050875
- [12] X. Mu and C. H. Min, "MFCC as features for speaker classification using machine learning," in *Proc. IEEE World AI IoT Congress (Allot)*, 2023, pp. 566–570. doi: 10.1109/Allot58121.2023.10174566
- [13] Hamouda, Salim, Samir Hamdani, Hamid Khelfi, Kamel Boudjit, Drid Mohamed Dhia Eddine, and

- Kassa Baghdouche., "Real time FPGA implementation of FFT architecture utilizing a single butterfly calculator: Application on rotor bars fault detection in induction motor," in *Proc. 12th Int. Conf. Systems and Control (ICSC)*, 2024, pp. 418–423. Doi: 10.1109/ICSC63929.2024.10928791
- [14] S. Bae, H. Kim, S. Lee, and Y. Jung, "FPGA implementation of keyword spotting system using depthwise separable binarized and ternarized neural networks," *Sensors*, vol. 23, no. 12, p. 5701, 2023. doi: 10.3390/s23125701
- [15] Shan, Weiwei, Minhao Yang, Tao Wang, Yicheng Lu, Hao Cai, Lixuan Zhu, Jiaming Xu, Chengjun Wu, Longxing Shi, and Jun Yang., "A 510 nW wake up keyword spotting chip using serial FFT based MFCC and binarized depthwise separable CNN in 28-nm CMOS," *IEEE Journal of Solid State Circuits*, vol. 56, no. 1, pp. 151–164, 2020. doi: 10.1109/JSSC.2020.3029097
- [16] Wassi, Guy, Sylvain Iloga, Olivier Romain, and Bertrand Granado. "FPGA based real time MFCC extraction for automatic audio indexing on FM broadcast data." In *2015 Conference on Design and Architectures for Signal and Image Processing (DASIP)*, pp. 1-6. IEEE, 2015. doi: 10.1109/DASIP.2015.7367253
- [17] Dao, V.-L., Nguyen, V.-D., Nguyen, H.-D., Hoang, V.-P. (2017). "Hardware Implementation of MFCC Feature Extraction for Speech Recognition on FPGA". In: Nguyen, N.T., et al. (eds) *Advances in Information and Communication Technology. AISC*, vol. 538, pp. 248–254. Springer, Cham. doi:10.1007/978-3-319-49073-1_27
- [18] Barma Venkata RamaLakshmi and Fazal Noorbasha. "FPGA Implementation of Optimized Radix 4 and Radix 8 Booth Algorithm" [*J. Int J Performability Eng*, 2021, 17(6): 552-558. doi: 10.23940/ijpe.21.06.p8.552558
- [19] Nagesh, Patnala Uma Mehar Sai, and Vella Satyanarayana. "Design of Radix-4 Booth (Radix 256 booth encoding) Multiplication Algorithm." In *2026 4th International Conference on Knowledge Engineering and Communication Systems (ICKECS)*, pp. 1-6. IEEE, 2026. doi: 10.1109/ICKECS70176.2026.11528088
- [20] Srinivas, Nettimi Satya Sai, Nagarajan Sukan, Lakshmi Sutha Kumar, Malaya Kumar Nath, and Aniruddha Kanhe. "FPGA Based Design of a Ready to Use and Configurable Soft IP Core for Frame Blocking Time Sampled Digital Speech Signals." *Electronics* 13, no. 21 (2024): 4180. doi: 10.3390/electronics13214180
- [21] Yang, Yucheng, Shukai Duan, and Lidan Wang. "An FPGA Accelerated Approach to Direct Waveform Processing for Low Power Keyword Spotting." In *Proceedings of the International Conference on Image Processing, Machine Learning and Pattern Recognition*, pp. 157-161. 2024. doi: 10.1145/3700906.3700931
- [22] Shihab, Md Shazzad Hossain, Shuvra Aditya, Jahangir Hossain Setu, K. M. Imtiaz Ud Din, and Md Iftekharul Alam Efet. "A hybrid GRU CNN feature extraction technique for speaker identification." In *2020 23rd International conference on computer and information technology (ICCIT)*, pp. 1-6. IEEE, 2020. doi: 10.1109/ICCIT51783.2020.9392734
- [23] Sun, Chang, Zhiqiang Que, Vladimir Loncar, Wayne Luk, and Maria Spiropulu. "da4ml: Distributed arithmetic for real-time neural networks on fpgas." *ACM Transactions on Reconfigurable Technology and Systems* 19, no. 1 (2026): 1-27. doi: 10.1145/3777387
- [24] Heriyanto, H., and D. A. Irawati. "Comparison of Mel Frequency Cepstral Coefficient (MFCC) Feature Extraction, With and Without Framing Feature Selection, to Test the Shahada Recitation." In *RSF Conference Series: Engineering and Technology*, vol. 1, no. 1, pp. 335-354. 2021. doi: 10.31098/cset.v1i1.395.
- [25] Mohanty, Basant Kumar. "Efficient approximate multiplier design based on hybrid higher radix booth encoding." *IEEE journal on emerging and selected topics in circuits and systems* 13, no. 1 (2022): 165-174. doi: 10.1109/JETCAS.2022.3229831
- [26] Aggarwal, Apeksha, Akshat Srivastava, Ajay Agarwal, Nidhi Chahal, Dilbag Singh, Abeer Ali Alnuaim, Aseel Alhadlaq, and Heung No Lee. "Two way feature extraction for speech emotion recognition using deep learning." *Sensors* 22, no. 6 (2022): 2378. doi: 10.3390/s22062378
- [27] Tripathi, Astha, and Poonam Rani. "Multilingual speech emotion recognition using IGRFXG–Ensemble feature selection approach." *Applied Acoustics* 240 (2025): 110905. doi: 10.1016/j.apacoust.2025.110905
- [28] Lu, Lu, Chi Liu, Duo Li, Xuexian Liu, Yubo Li, Lei Cheng, and Xu Zheng. "Fusion of Log Mel spectrogram and acoustic features for driver speech emotion recognition." *Expert Systems with Applications* (2025): p 131021 vol. 307. doi: 10.1016/j.eswa.2025.131021
- [29] Mokhtar, A. S., C. S. Ping, MF Md Din, N. Makmor, and M. Mahadi. "Implementation of Booth multiplier algorithm using Radix-4 in FPGA." *Jurnal Kejuruteraan* 1 (2021): pp 161-165. doi: 10.17576/jkukm-2021-si4(1)-20
- [30] Yu, Baoying, Mengjie Luo, Dingyi Wang, Xiaoqin

- Wang, and Shushan Qiao. "A low power and low latency speech feature extractor based on time domain filter bank." *IEEE Transactions on Circuits and Systems II: Express Briefs* 71, no. 3 (2023): pp 1421-1425. doi: 10.1109/TCSII.2023.3324337
- [31] Muthumari, M., and R. S. Valarmathi. "Optimal Realization of Parallel MAC Based Adaptive FIR Filter Using a Gaussian Mutation on Quantum Behaved Particle Swarm Optimization Algorithm." *Journal of Circuits, Systems and Computers* 34, no. 15 (2025): 2550330. doi: 10.1142/S021812662550330X
- [32] Choudhary, Tripti, Vishal Goyal, and Atul Bansal. "Enhanced lstm network with semi supervised learning and data augmentation for low resource ASR." *International Journal on Smart Sensing and Intelligent Systems* Vol. 18, Iss. 1, (2025). doi:10.2478/ijssis-2025-0009
- [33] D. Kale, G. Ganjre and R. Udani, "A Proposed Real Time FPGA System Using MFCC Features for Acoustic Anti Poaching Monitoring," 2026 *International Conference on Communication, Computing and Emerging Technologies (IC3ET)*, Vasai, India, 2026, pp. 215-220, doi: 10.1109/IC3ET64989.2026.11467226.
- [34] Bonet Solà, Daniel, and Rosa Ma Alsina Pagès. 2021. "A Comparative Survey of Feature Extraction and Machine Learning Methods in Diverse Acoustic Environments" *Sensors* 21, no. 4: 1274. <https://doi.org/10.3390/s21041274>
- [35] Li et al., "MSP MFCC: Energy Efficient MFCC Feature Extraction Method With Mixed Signal Processing Architecture for Wearable Speech Recognition Applications," in *IEEE Access*, vol. 8, pp. 48720-48730, 2020, doi: 10.1109/ACCESS.2020.2979799
- [36] Kia, Ali, Ajan Ahmed, and Masudul H. Imtiaz. 2026. "FPGA Based Implementations of Biometric Recognition: A Review" *Electronics* 15, no. 10: 2145. <https://doi.org/10.3390/electronics15102145>
- [37] Tsai, TH., Wang, CL. "GMM Based Speaker Verification System with Hardware MFCC in SoC Design". *Multimed Tools Appl* 83, 56991–57010 (2024). <https://doi.org/10.1007/s11042-023-17561-6>.
- [38] M. Kotari, Sharath, S. Kanabargi, S. Alori and Y. B, "A Review on AI Acceleration with FPGA and ASICs: A Comparative Study," 2025 *IEEE 4th International Conference on Data, Decision and Systems (ICDDS)*, Dharwad, India, 2025, pp. 212-215, doi: 10.1109/ICDDS67737.2025.11344680.
- [39] Konstantopoulou, Evangelia, George Athanasiou, and Nicolas Sklavos. "Review and analysis of FPGA and ASIC implementations of NIST lightweight cryptography finalists." *ACM Computing Surveys* 57, no. 10 (2025): 1-35. <https://doi.org/10.1145/3721122>.
- [40] Lou, Haozhe. "Comparative analysis and enhancement of FPGA and ASIC performance in targeted applications." In *International Conference on Artificial Intelligence and Communication Technology*, pp. 295-308. Singapore: Springer Nature Singapore, 2024. doi:10.1007/978-981-96-0096-0_23.

Author Biography



M Muthumari received the B.E. degree in Electronics and Communication Engineering from S. Veerasamy Engineering College and the M.E. degree in wireless technology from MIT, Anna University, Chromepet, India, in

2015 and 2017, respectively, and is currently pursuing the Ph.D. degree in signal processing at Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology. She has more than 6 years of teaching experience and 3 years of research experience. Her current research interests include speech signal processing, machine learning, Artificial intelligence, and VLSI signal processing. She has won the best paper award in IEEE conference. . She is a lifetime member of the International Association of Engineers.



Dr.R.S.Valarmathi B.E., M.E., Ph.D., SMIEEE. Dr. R. S. Valarmathi received the B.E. degree in Electronics and Communication Engineering and the M.E. degree in Applied Electronics from Bharathiar University,

Coimbatore, India, in 1989 and 2000, respectively, and the Ph.D. degree in Biometrics from Anna University, Chennai, India, in 2009. She is currently a Professor in the Department of Electronics and Communication Engineering and Dean of the School of Electrical and Communication at Vel Tech Dr. Rangarajan Dr. Sagunthala R&D Institute of Science and Technology, Chennai, India. She has more than 34 years of teaching experience and 18 years of research experience. Her research interests include biometrics, image processing, signal processing, machine learning, soft computing, VLSI, and computer networks. She is an Anna University-recognized research supervisor and has guided 18 Ph.D. scholars. She has published more than 270 research papers and has served as a reviewer for several journals and conferences, including IEEE, Springer, and Elsevier publications. She is a Lifetime Member of ISTE, Fellow

of IETE and the Institution of Engineers (India), Senior
Member of IEEE, and Member of IET.