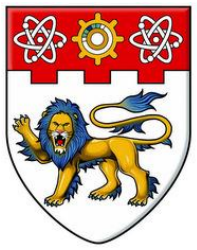
Ultra-low Power Signal Processor For Biomedical Applications Seyed Mohammad Ali Zeinolabedin 2017



NANYANG TECHNOLOGICAL UNIVERSITY

Ultra-low Power Signal Processor For Biomedical Applications

Seyed Mohammad Ali Zeinolabedin

SCHOOL OF ELECTRICAL AND ELECTRONIC ENGINEERING

2017

Ultra-low Power Signal Processor For Biomedical Applications

Seyed Mohammad Ali Zeinolabedin

School of Electrical & Electronic Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfilment of the requirement for the degree of
Doctor of Philosophy

2017

Acknowledgments

At this moment, I am taking my final steps toward the end of the PhD journey in Nanyang Technological University. This journey started with complex sets of emotions, feelings and thoughts like hopes and fears, certainty and uncertainty, bright and dim future and success and failure. I was able to see myself sometimes completely taken over by these feelings and thoughts and I was so frustrated to take the next steps. I should be really grateful from deep in my heart to come to know teachers from whom I have learnt how to control myself and progress in my daily life. I really thank to Mustafa Malekian, Eckhart Tolle, Abolhassan Banisadr and Rumi as four spiritual and philosophical teachers leading me to rationality and spirituality at the same time. I am sure without their help I was not able to tackle these destructive thoughts and feelings.

I would like to express my sincere gratitude to my supervisor, Prof. Tony Kim, who has continuously supported and given me freedom to show my capabilities in these four years. Personally, I have learned patience, politeness and hardworking and the way he looks into problems from him. I can remember that how deeply and precisely he put effort on my papers to scrutinize each sentence. I gratefully acknowledge his attempts to build up the connection with other researchers and groups to help the students to broaden their experiences and have a chance to be evaluated by professional researchers and scientists. I wish from my heart that Professor Kim and his lovely family become successful, happy and healthy in every moment of their life.

I express my deepest respect and appreciation to Dr. Jun Zhou, Dr. Anh Tuan Do and Dr. Liu Xin in Institute of Microelectronic (IME). Their technical supports, contributions and high level proficiency backed me in tough days of doing research. I had a great chance to do my first tape-out in IME under the supervision of Dr. Zhou and Dr. Liu Xin and they showed me how generous one can be in his life and how influential it is for a student toward his future career. I had also an opportunity to work with Dr. Do on my second project and learn how to look into problems in more efficient and precise way.

I thank all my friends in VIRTUS for their kind and generous help, especially during the tape-out. It turned out that the tape-out chance is not only a time to realize new ideas, but also it can be a way to find true friends. Last but not the least, I would like to thank my family and friends in my country who always encourage me to stand firmly against the difficulties and reach to the end of this journey.

Table of Contents

Abstract

Recent developments in integrated circuit (IC) design technology enable us to realize the complicated applications and algorithms which were mainly implemented in software-based platform. In addition, the rapid advancements in the world of digital signal and image processing have simultaneously brought the newer and more efficient algorithms into existence. For example, the frequency-based image processing that was first relied on Fourier transform, has been gradually looked for another powerful transforms. They not only unmask the frequency components, but also provide the position of the frequency components such as Laplacian and Gaussian Pyramids (LP and GP) and wavelet transform. Going to more than one dimensional signal processing necessitates having the transform providing more information of the signal such as directionality. For instance, curvelet (for continuous signals) and Contourlet (for two dimensional (2D) discrete signals) transforms have been developed to address those issues. By tracking the time passed from Fourier transform to Contourlet transform which is almost 130-year-old journey, we have been able to access to more information, though the complexity is considerably increasing. Now, there are various mathematically complex algorithms developed to achieve the desired outcome in the areas like biomedical and mobile applications, coding, robotics, three dimensional (3D) graphics and so on.

Many of the mentioned applications require to be implemented in hardware in order to have the real-time performance. Moreover, some needs to be ultra-low power to operate for a longer time, especially for biomedical applications in which both battery

life and thermal energy transferred to body tissues are of great concerns. For instance, the online spike sorting DSP has been developed to do sorting on the signals (spike) recorded from a brain in a real time. Two requirements that should be always met are the real-time performance and the power density. The latter one is also very essential because the hardware implanted in a brain must avoid damaging brain tissue. So, the main bottlenecks of achieving ultra-low power signal and image processors for biomedical and Internet of thing (IoT) applications are memory and computational complexity.

In the first part of this thesis, I propose architecture level contributions to achieve ultra-low power digital processor for biomedical and IoT applications. The proposed techniques reduce the power and area by addressing the area- and power-hungry components of such applications which are FIFOs and memory. The proposed techniques are 1- FIFO with error-reduced data compression (FERDC) and 2- FIFO with adaptive error reduced data compression (FAERDC). FERDC internal parameters are set in advance and it has less design complexity. Whereas in the latter one, the adaptive mechanism updates internal parameters with respect to input data pattern to reduce the output error and gain more power and area saving. FERDC and FAERDC can be generally applied to various digital signal and image processors and hardware accelerators because there is no assumption on which FERDC and FAERDC are proposed. FERDC has been extensively investigated using the various simulations to verify the functionality as well as power and area reduction. Then FAERDC along with a proposed extension method for filtering and a near-threshold operation have been proposed to design an ultra-low power

Laplacian Pyramid (LP) engine as a popular hardware accelerator. The LP has been fabricated in 180 nm CMOS technology and its functionality is verified at 0.5 V.

In the next part of this thesis, Contourlet transform (CT) which is a multiresolution image representation is studied. CT is one of the latest directional image transform whose hardware implementation has not been yet investigated. It consists of LP and directional filter bank (DFB). DFB is extracting the frequency components with respect to their directions. We have proposed the first hardware architecture of CT and studied the DFB in detail. The proposed architecture has been functionally verified through extensive simulation results. DFB is also a memory-intensive algorithm through which the whole input should be iteratively read, processed and stored back in the memory.

The last part of this thesis is devoted to a real-time multi-channel spike sorting chip. Various techniques are proposed to design a 128-channel real-time spike sorting DSP operating at near-threshold (NT) voltage. The main contributions are categorized as architecture and circuit levels. We have proposed a new spike detector, feature extraction and training algorithms to improve the accuracy of clustering and then reduce the memory requirement in training and clustering parts. In order to further reduce the area to accommodate 128 channels as well as leakage power, a full custom 8T-cell SRAM is designed to operate in NT. An auto-biased bit-line keeper provides a reliable sensing margin at near- and sub-threshold operation so that a single power supply can be used for both the digital core and SRAM. The measurement results verify the chip functionality at 0.54 V with 0.175 $\mu W/ch$.

List of Figures

List of Tables

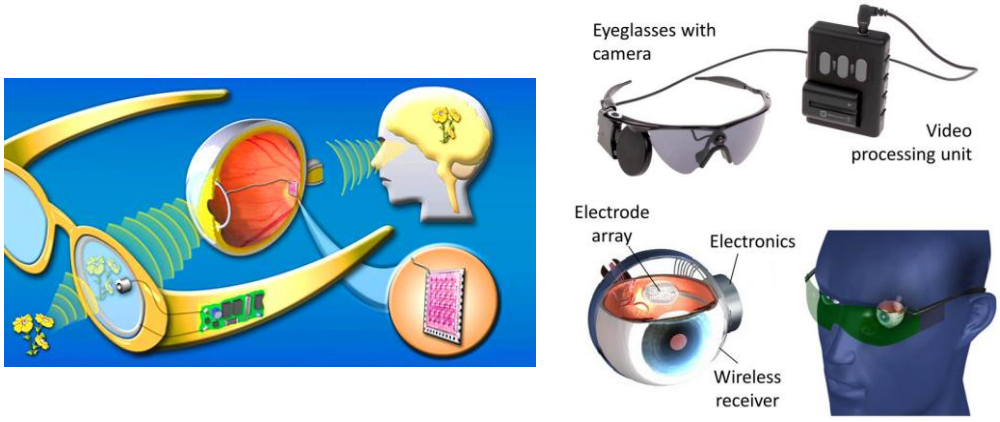
Chapter 1. Introduction

With the advancement of Integrated Circuit (IC) design technology, many biomedical and IoT (internet of thing) applications, requiring complex digital signal and image processing algorithms, can be implemented in hardware to have the real-time and ultra-low power operation. Bioelectronics and IoT are two popular interdisciplinary fields that can promote the life quality not only by making the life easier and safer but also by introducing the new approaches to diagnose and treat diseases and study the human body [1-5]. The latter one brings new hopes to patient with severe diseases and disabilities such as damaged retina cells, damaged inner ear and spinal cord injury. Besides, it has also made the new opportunities to neurophysiologists studying the human brain behaviour [6-9].

A typical biomedical system consists of digital, analog, mixed-signal and RF parts. Ultra-low power operation is a key element in such a biomedical system mainly because of the power management issue and heat dissipation to a nearby tissue. Digital parts are usually very power hungry due to the huge memory requirement as well as intensive computational complexity [10, 11]. Therefore, double efforts should be made to address these requirements in digital circuit and consequently improve the whole system performance. The new IC design technology makes us fulfil the complex hardware accelerator and system on chip (SOC) although incorporating all the desired parameters at the same time seems to be difficult and sometimes mutually exclusive.

1.1 Digital Signal/Image Processing in Biomedical Applications

As an state of the art example in biomedical applications, a retinal prosthesis system [12, 13], also known as the bionic eye or retinal implant, is a good candidate. It shows the role of the two dimensional (2D) digital image processing and its real-time hardware implementation in daily life and how important it is to bring new hopes to blind persons by restoring some basic vision. The conceptual architecture of retinal implant is illustrated in Figure 1-1(a). As shown, an electrode array will eventually replace a damaged retina by either epiretinal implants (on the retina) or subretinal implants (behind the retina). A miniature camera embedded in the glasses captures visual signals and transmits via wireless link to the multi-channel electrode array. The image processing unit is responsible for processing the taken image and sent the instruction to the implant. In more detail, an input image has to be decimated due to the limited number of intraocular implanted electrodes [1, 14]. Therefore, while doing decimation, the image crucial contents should be preserved as much as possible like edges as opposed to the texture. Figure 1-2(c) shows edge-detected binary image demonstrating the significant content of the image only by 1 bit per pixel as compared to 8-bit decimated image (Figure 1-2(b)). Interestingly, recently commercialized retinal implant named as Argus II [13] is already approved by FDA (U.S. Food and Drug Administration) in 2013 (Figure 1-1(b)), yet many steps left to be taken in different parts of the retinal implant to open the new window to the blind individuals. In the digital image processor, performing the many operations requires FIFO (first input, first output) which considerably contributes to a system power.



(a) (b)

Figure 1-1. (a) Conceptual design of retinal implant. (b) Argus-II retinal implant [12, 13].

(a) (b) (c)

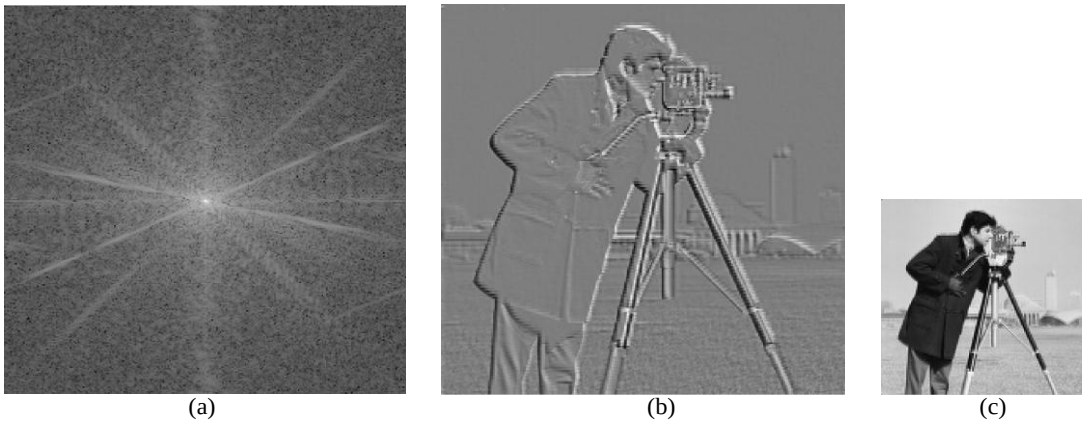
Figure 1-2. Conceptual procedure of image processing unit in retinal implant. (a) A 256x256 input image. (b) Decimated image (128x128). (c) Edge-detected binary image.

By looking into the world of digital image processing, we can figure out that the input image are processed in either spatial domain or frequency domain [15]. In the spatial domain, image pixels are directly manipulated whereas in the frequency domain an

input image is transferred into another domain to provide the meaningful information which is not visible in the image itself. Both domains are extensively applied in different image processing algorithms such as image enhancement, denoising, compression, feature extraction, object recognition and segmentation [16]. The frequency domain approach can be categorized as 1- 2D Fast Fourier transform (FFT) as the mainstay of this kind and 2- pyramidal transforms. The former one only gives what frequencies are in the input image. The latter one gives both what frequencies are available and where they occur, such as Gaussian and Laplacian Pyramid (GP and LP), Discrete Wavelet Transform (DWT) and Contourlet transform (CT) [17]. In addition, the pyramidal transforms incorporate the multiresolution processing in which the input image can be scaled down into different subbands providing the various resolutions of the input.

For example, the difference between 2D FFT and GP and LP can be figured out by applying them to the Figure 1-2(a). Figure 1-3(a) shows the output of 2D FFT, and pyramidal counterpart is shown in Figure 1-3(b)-(c). 2D FFT output does not include any spatial information of the frequency component whereas the low-pass and band-pass frequency responses have the spatial information. Since GP and LP outputs are more informative than that of 2D FFT, they are applied in many powerful image processing algorithms. Suppose a very simple compression scheme when the band-pass frequency components of the LP are simply set to zero (Figure 1-3(b) replaced by all-zero matrix) and the frequency components of 2D FFT lying in $\left(-\pi, -\frac{\pi}{2}\right]$ and $[\frac{\pi}{2}, \pi]$ change to zero. The reconstructed images displayed in Figure 1-4 show that the LP reconstructed image

is almost similar to input image, though PSNR (peak signal to noise ratio) is lower than 35 dB, whereas the FFT counterpart is nothing but a simple grey image.



(a) (b) (c)

Figure 1-3. (a) Fourier Spectrum of the image "cameraman". (b) LP approximation of the image "cameraman". (c) GP approximation of the image "cameraman".

(a) (b)

Figure 1-4. Compression efficiency of 2D FFT vs. LP in a simple lossy compression algorithm. (a) Reconstructed image by 2D FFT (PSNR ≅ 0). (b) Reconstructed image by the LP (PSNR ≅ 17 dB).

1.2 Ultra-low Power Digital Image Processors Design Challenges

Undoubtedly, memory is an essential element in performing the image processing algorithms [3, 10, 18] and its huge size is a key contributor to a power consumption (dynamic and static power components). For example, almost all image/video processing algorithms require digital filtering in frequency or spatial domains such as image enhancement, restoration, compression and segmentation. From the hardware view, filtering operations require storing a part of the input image or previous results temporarily in line buffer (i.e. FIFO). The size of the FIFO significantly affects the total area and power of the design. For instance, FIFOs consume more than 75% of the route area and power in [19] and more than 80% of power consumption in [20]. So in order to reach the ultra-low power design, the memory usage should be exactly investigated to figure out how we can improve the power and area without degrading the operating frequency and output accuracy. Intensive computational complexity is another important factor in designing the ultra-low power image processors. The algorithms should be precisely studied to apply different architectural techniques to reduce the complexity as well as area and power.

For instance, the Capsule Endoscopy (CE) [5] which is one of the examples of biomedical image processors is used to examine the digestive tract especially small intestine. The capsule shown in Figure 1-5(a)-(b) consists of a tiny camera, an image processor, RF transmitter and a battery. Once swallowed by the patient, it takes the picture of the gastrointestinal tract (Figure 1-5(c)) and transmit it to the outside station wireless-

ly in real-time. The image processor is so far intended to compress the captured image with the rich information of the alimentary canal and removing the rest. The major design challenges of digital part are huge required memory, a minimum PSNR of 35 dB for reconstructed image, up to 24 frames per second (fps) and the power consumption that should be less than 1 mW.

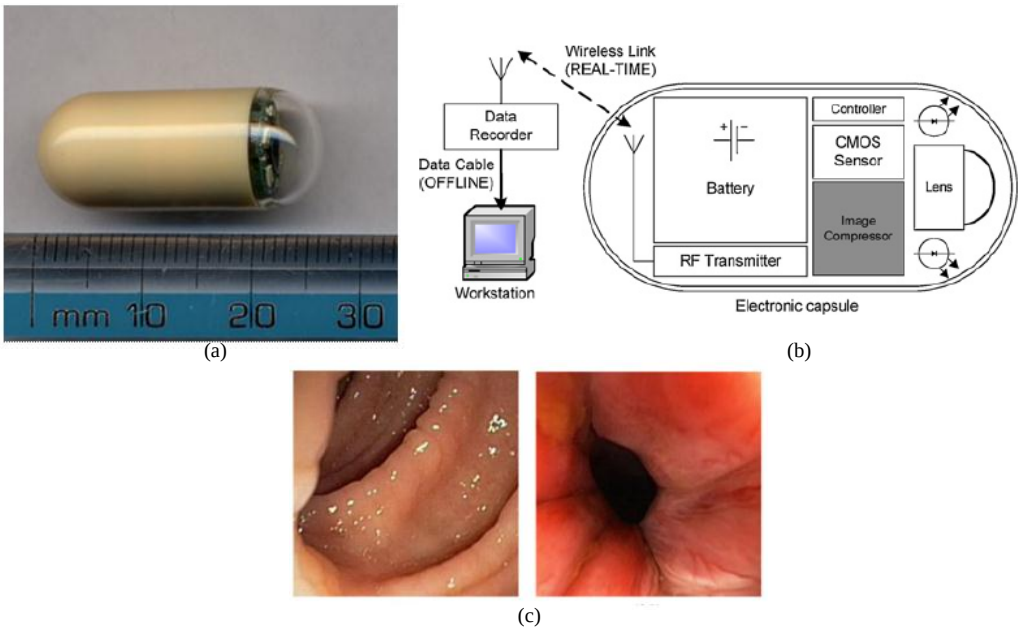


Figure 1-5. (a) A capsule endoscopy. (b) A capsule endoscopic system. (c) Some taken pictures [5].

1.3 Biomedical System on Chip for Brain Study

Brain study is another important biomedical application which has encouraged many researchers from various scientific fields to extensively work on that. Many attempts in IC design have been so far made to facilitate this process and to develop moni-

toring and stimulating brain systems to study the various brain functions and to treat some severe diseases. For example, spike sorting process is in fact one of the popular methods in neuroscience to investigate and study the brain behaviour [8]. In spike sorting, very small electrodes are implanted inside the objects' brains to monitor the activity of close by neurons named spikes or action potential (AP). Recorded signals combined of spikes and noise are later transferred to a computer to be studied by neurophysiologists/scientists. Each recorded signal includes 3 to 6 types of spikes from the adjacent neurons [9], given that, each neuron fires a particular spike. Therefore, spikes should be detected and later all similar ones should be classified in a single cluster. Each cluster in fact indicates an individual neuron.

In conventional recording system, the raw signals are transmitted to a nearby computer. However, after widely developing high-density arrays of microelectrodes [21, 22], this approach faces practical limitations such as very high data rate, real-time operation issue especially for neural prostheses applications, freedom of movement and infection on subjects. For instance, the data rate for 128 channels, 25kS/s recording system using 8 bit ADCs is 25.6 Mb/s. Therefore, practically transmitting this amount of data violates the power density requirement on implantable devices which is expected to be less than 277 $\mu W/mm^2$ [8].

An IC implementation of spike sorting has been in fact proved to be essential for multi-channel neural signal processing research. Integrating whole or a part of spike sorting algorithm on-chip leads to a significant reduction of the data to be transmitted to

the subsequent blocks [7, 9]. Moreover, it is much faster and more power-efficient than that in the software domain and especially makes it feasible for real-time multi-channel processing applications. Typical spike sorting algorithm as shown in Figure 1-6 consists of 1- a spike detector (SD) to detect and align spikes with respect to a common point; 2- a feature extractor and dimensionality reduction (FE and DR) to extract information-rich features from noisy data; 3- a classifier to assign the detected spike to a neuron ID; and 4- a training engine (TE) to train the chip and store cluster means to a memory to be used by the classifier. Prior to normal operation, on-chip spike sorting must be trained to identify mean values of clusters, each representing one neuron source. Once trained, the TE writes cluster means to a memory for subsequent clustering. During normal operation, TE is turned off and features of the detected spikes are compared with cluster means and assigned to a cluster with the minimum distance.

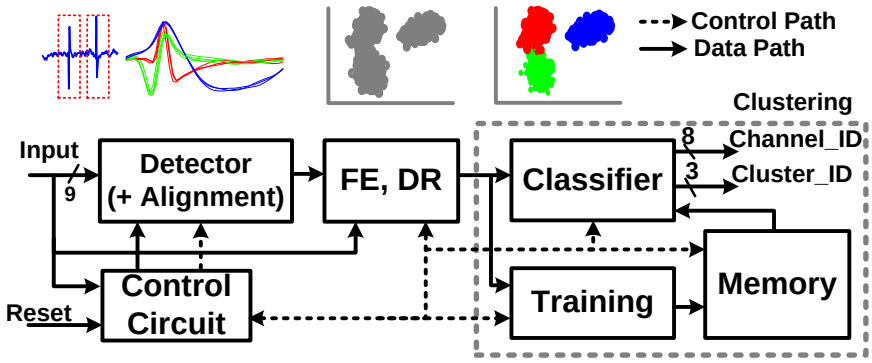


Figure 1-6. Typical spike sorting flow.

1.4 Summary of Thesis Contributions

This thesis makes several contributions and implements some digital signal and image processors. It first proposes an architecture level contribution to address the ultra-low power digital processor design issues for biomedical and IoT applications. It is intended to reduce the power and area and keep the mean square error small. The proposed techniques are 1-FIFO with error-reduced data compression (FERDC) and 2-FIFO architecture with adaptive error reduced data compression (FAERDC). In the latter one, it adaptively updates its internal parameters with respect to input data pattern to reduce the output error. However, the former one's internal parameters are set in advance and therefore it has less design complexity.

FERDC and FAERDC can be generally applied to almost all digital image processors and hardware accelerators since there is no assumption on the algorithms being operated. FERDC has been extensively investigated using the various simulations to verify the functionality as well as power and area reduction percentage. FAERDC along with a proposed extension method and a near-threshold operation have been applied in LP (including GP) as a sample popular hardware accelerator. The LP has been fabricated in 180 nm CMOS technology and its functionality is verified at 0.5 V. We have published these works in [23], [24], [25], [26].

In continuous of studying LP, we have proposed the first hardware architecture of CT. CT is one of the latest directional image transform whose hardware implementation has not been investigated yet. It consists of LP and directional filter bank (DFB).

DFB is responsible for extracting the frequency components with the respect to their directions. The proposed architecture has been functionally verified through extensive simulation results. DFB is also a memory-intensive algorithm through which the whole input should be iteratively read, processed and stored back in the memory in a very specific manner. Therefore a memory address generator is proposed to provide the appropriate data required by DFB. This work has been published in [27].

The last part of this thesis is devoted to a real-time multichannel spike sorting chip introduced in Section 1.3. Various techniques are proposed to design a 128-channel real-time spike sorting chip operating at near-threshold (NT) voltage. The main contributions are categorized as architecture and circuit levels. I have proposed new SD, FE and TE algorithms to first improve the accuracy of clustering and second reduce the memory requirement in clustering part. In order to accommodate 128 channels as well as reducing the leakage power dominated by the memory, a full custom memory is designed to operate in NT. The measurement results verify the chip functionality at 0.54 V. This work has been published in [28], [29], [30].

The organization of this thesis is as follows: Chapter 2 explains 2D filtering briefly and presents the proposed FERDC technique to reduce the power and area of the FIFO in image/video processing applications. Chapter 3 describes the proposed FAERDC technique to adaptively reduce the power and area and how LP is realised using this technique and NT operation. Chapter 4 discusses Contourlet transform and its hardware architecture. Chapter 5 studies a multi-channel spike sorting chip for 128

channels to real-time process the brain extracellular signals. Finally, Chapter 6 concludes this thesis.

Chapter 2. Proposed FERDC Technique

2.1 Introduction

Rapid advancement in VLSI systems has enabled the real-time hardware implementation of high computational complexity image/video processing algorithms. Some recent works requiring fast hardware implementation are video codec [3], computational photography [2], wavelet transform [31], laplacian pyramid [20] and biomedical applications [1]. Early image/video processors focused on improving the speed/throughput of the systems while the contemporary state-of-the-art image/video processors confront additional challenges such as low power consumption and real-time processing.

Many image/video processing algorithms require digital filtering like image enhancement, image restoration, image compression, image segmentation, and video coding [16], [32] in both spatial and frequency domains. Regarding the hardware implementation of image/video signal processing algorithms, many filtering operations require storing part of input image or part of previous stage results temporarily. However, it has not been comprehensively investigated to reduce the size of the temporary pixel buffers (i.e. FIFO). The size of the FIFO significantly affects the total area and power of a design. It has been reported that FIFOs consume more than 75% of the route area and power in [19] and more than 80% of power consumption in [20]. Various approaches have been developed to reduce the power consumption of FIFOs. However, most of them are focusing on the circuit-level techniques and memory-splitting schemes along with power

gating [19], [33]. In this chapter, I propose a new architecture-level technique to reduce the size of FIFO only with negligible degradation in the error metrics such as mean square error (MSE) and peak signal to noise ratio (PSNR) [34]. The reduced FIFO size through the proposed method also improves the dynamic and leakage power consumptions. It is worth mentioning that MSE and PSNR are two popular metrics to measure the output quality of various transforms. It is because they have low computational complexity and easy to realize it. The greater the PSNR is, the better the image quality is. Inversely, the lower the MSE is, the better the image quality is.

This chapter is organized as follows. Section 2.2 introduces the usage of FIFO in image/video processing. Section 2.3 reviews the existing design techniques for power- and area-efficient FIFO. Sections 2.4 and 2.5 presents the details of the proposed area and energy efficient FIFO using error-reduced data compression and near-threshold operation. Section 2.6 shows the experimental results and comparison between the proposed FIFO design and conventional design. Finally, Section 2.7 concludes the chapter Chapter 2.

2.2 2D Filtering in Image/Video Signal Processing

Filtering is an inevitable function of image/video processing algorithms, which can be generalized as:

$$g(x,y) = T_{xy} * f(x,y) \qquad 2\text{-}1$$

where $f(x,y)$ is the input image, T_{xy} is the filter transfer function and $g(x,y)$ is the output. The output is defined as the convolution of the pixel values located in a given filter window (T). The window is usually a square with $N \times N$ pixels where N is an odd number to have an exact centre pixel. T can be either a 2D filter where T_{xy} represents the filter coefficient or two one dimensional (1D) filters where t_{xy} is the coefficient of the $1 \times N$ filter. In the latter one, t is first applied to the horizontal rows of the input, followed by the vertical columns or vice versa. In either case, it requires $N-1$ FIFOs to store $(N-1)$ rows of W pixels (Figure 2-1(a) and Figure 2-1(b)) where W is the input image width. So the number of reads is N pixels during each clock cycle. wise, $N \times N$ pixels have to be read every clock cycle to fill the filter window [35]. As already mentioned, FIFOs have significant impact on the circuit area and power consumption, particularly when the size of the filter, N, is large. Table 2-1 shows the computational complexity of separable and inseparable 2D filtering and memory footprint size. Since memory size is quite significant ($W >> N$ for a practical application), the power is mainly contributed by memory.

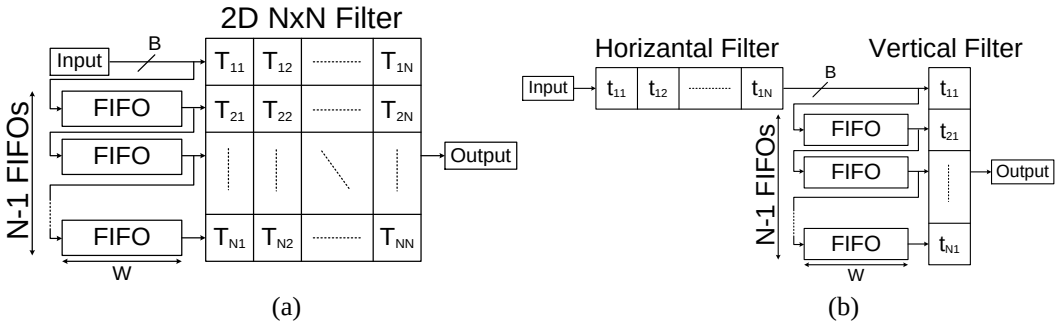


Figure 2-1: Sample filter configurations: (a) 2D filter and (b) Two 1D filters.

Table 2-1. Computational complexity of two 2D Filtering structures

Filter	Adder	Multiplier	Memory
Non-separable	N^2-1	N^2	$W(N-1)+N\times N$
separable	$2(N-1)$	$2N$	$W(N-1)+2\times N$

W: Image width, N: Filter size

2.3 Existing Design Techniques for Power- and Area-efficient FIFO

Various approaches have been proposed to reduce the size and power consumption of FIFO in image/video applications. They can be categorized as architecture-level and circuit-level techniques. The architecture-level techniques mainly focus on sharing FIFOs [36] or changing the original algorithms so that they require either less FIFOs [4] or less frequent FIFO access [37-40].

The H.265/HEVC (high efficiency video coding) decoder implemented in [36] shows an example of FIFO sharing technique. In conventional design, five FIFOs, occupying 75% of the total area, are needed in Intra Predictor (IP) and De-blocking Filter

(DF) blocks, respectively. In [36], first FIFO is shared between IP and DF by adding Share Above Line Buffer (SALB) block, leading to 20% area reduction in the price of 5% overhead in computational complexity. A 16-core graphical processing unit (GPU) is developed in [4] for mobile application. In order to drain less power, the approximated technique using wavelet is proposed. It replaces the lower scale of a flat texture adaptively by the higher one. Therefore, the average access to FIFO is reduced by 24.57% at the price of the quality degradation. The proposed DWT architecture [37, 38] uses tile-based image segmentation along with data interleaving to reduce the size of FIFO for low frequency spectrum so as to reduce both the area and power consumption. In [39], an image scaling processor is designed for up/downscaling of an image. *T*-model and inversed T-model filters are used to replace the conventional filter structures to reduce the storage requirement from two FIFOs to one with increase of MSE. The unified memory-centric architecture suitable for both separable and non-separable 2D FIR (finite impulse response) filter is proposed in [40] to address the memory issue. It uses fully-direct implementation of 2D FIR filter together with FIFO sharing to achieve area-delay-power-efficient architecture. To do so, the block-based approach applying the equally partitioned FIFOs provides the essential data for block processing. Therefore with the same amount of FIFO as conventional method, a block of input data will be processed at any time.

The circuit-level methods include latch-based FIFO [11], ultra-low power SRAMs [33], memory-splitting, and clock gating [19]. Regarding the circuit-level methods, in feature extraction accelerator [11], a new latch-based FIFO is introduced to re-

duce the area and power as compared to the conventional flipflop-based FIFO, which requires two latches per cell. The resultant FIFO is 49% smaller with 62% lower energy for a 16b 1k-entry FIFO compared to the flipflop-based FIFO. In [33], a decoupled 10-T SRAM cell with improved SNM (static noise margin), write margin and the bitline sensing margin is developed to achieve ultra-low voltage operation to reduce the power consumption in FIFO or other on-chip storage. A 24-core processor in [19] for multi-media and communication application uses a packet-switched network-on-chips (NOC) exploiting FIFOs to synchronize the on-chip data transfer. 29.3% power reduction is achieved by splitting the memory with clock gating.

2.4 Proposed Area and Energy Efficient FIFO with Error-reduced Data Compression (FERDC)

In this section, an area and energy efficient FIFO design is proposed. On architecture level, a technique named as FIFO with error-reduced data compression (FERDC) is proposed to reduce the FIFO size. This reduces both area and power consumption of the FIFO with negligible distortion. On circuit level, near-threshold operation is adopted to reduce the power consumption of the FIFO. We use MSE to evaluate the distortion introduced by the proposed technique [16]. In general cases, an acceptable MSE value is less than 20 [34]. Lower MSE values indicate that reconstructed images are closer to original ones.

Figure 2-2 illustrates the proposed FERDC technique to realize an 8x8 filter using the architecture described in Figure 2-1(b). Since there is an 8-tap vertical filter applied, 7 FIFOs are required to store the data in Figure 2-2. FERDC employs a concept of pixel prediction where every pixel can be predicted utilizing adjacent pixels and accordingly input data is horizontally decorrelated and then quantized. Finally the encoded data read from FIFO are decoded to retrieve the original input data. So, it consists of encoding and decoding parts along with the reduced-size FIFO. So in Figure 2-2, all FIFOs except for the first one are fed by the output of previous FIFOs. Therefore, one encoding part is required for the seven FIFOs while one decoding part is needed at the output of each FIFO.

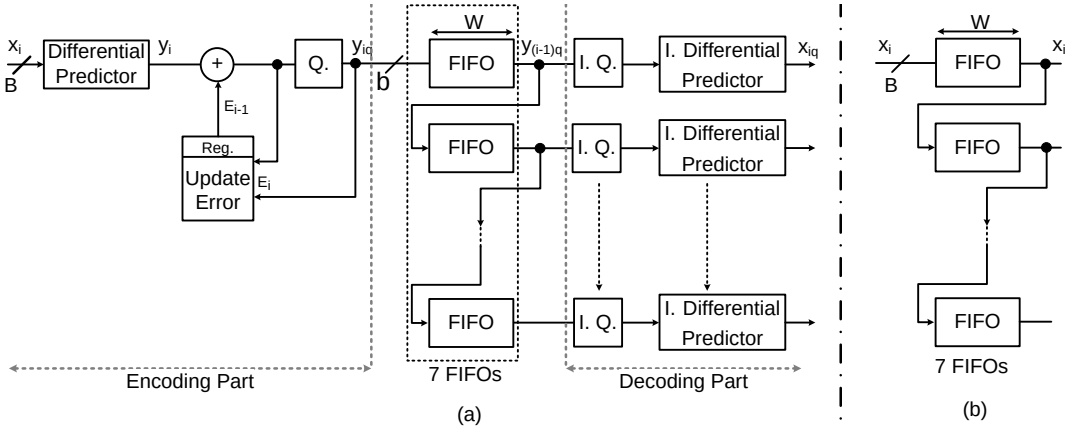


Figure 2-2. FIFO architecture for an 8-tab vertical filter: (a) Using proposed FERDC technique. (b) Using conventional FIFO.

2.4.1 Encoder Design

For encoding, input data, x_i is given to a differential predictor which calculates the difference between the consecutive inputs denoted as y_i. The differential predictor is to remove horizontal correlations between consecutive inputs. In that case, it ensures the energy of input correlated data, x_i (Figure 2-3(a)), is compacted around zero, y_i (Figure 2-3(b)) and only a few large numbers spread throughout the entire *B*-bit integer value range. A B-bit subtractor and adder along with a B-bit register are used for both differential predictor and inverse differential predictor as shown in Figure 2-4. Although applying the differential predictor helps to compact the energy of signal, the quantizer is required to reduce the data width as in detail explained in the next part. By doing so, the quantization error will get accumulated at the output of the design. So an update error block is proposed to cancel the error from getting accumulated at the output. Therefore, before sending the difference values to quantizer, the energy-compacted difference is updated by the quantization error of previous difference (E_{i-1}) to prevent the quantization errors from being accumulated at the output, x_{iq}. To obtain the error mathematical model of proposed FIFO, "update error" and "Q." (quantizer) blocks as shown in Figure 2-2 are substituted with the noise injection models as given in Figure 2-5. The output can be expressed by:

$$x_{iq} = y_{iq} + x_{(i-1)q}\ , \qquad for\ i = 1 : x_{1q} = x_1 \tag{2-2}$$

After solving Equation 2-2, the output, named as x_{iq_FERDC}, is:

$$x_{iq\_FERDC} = x_i + E_i - E_1 \ , \qquad E_{T\_FERDC} = E_i - E_1 \tag{2-3}$$

In order to compare FERDC output error with the architecture not including "update error" block (FDC) (Figure 2-6), similar calculation is performed on Figure 2-6. The output is derived as:

$$x_{iq\_FDC} = x_i + \sum_{j=2}^{i} E_j \ , \qquad E_{T\_FDC} = \sum_{j=2}^{i} E_j \tag{2-4}$$

As clearly seen in equation 2-4, the errors in the output are added together for any given i. It may cause the large accumulated error (E_T) if any large partial error (E_i) arises during the quantization and is not offset by next partial errors while processing the row i. Whereas the error component in Equation 2-3 is considerably reduced to error difference of the i^{th} input and the first input.

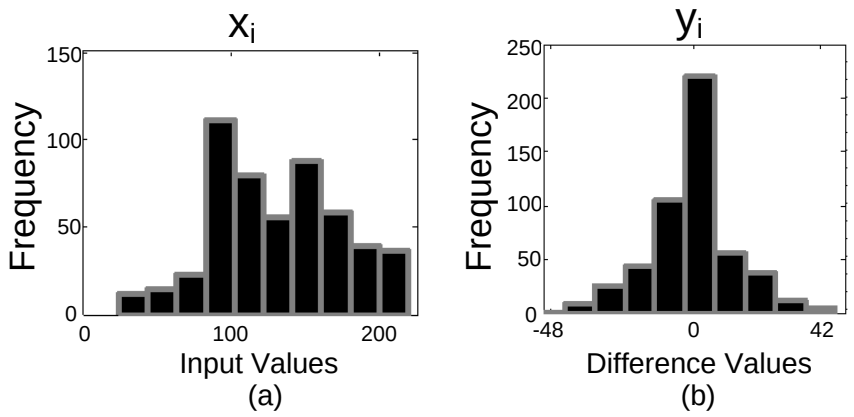


Figure 2-3. Histogram of "Barbara" (First row) for (a) $x_i$and (b) y_i.

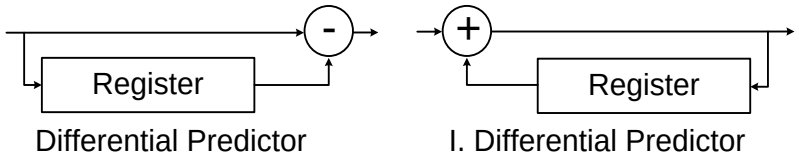


Figure 2-4. Differential and inverse differential predictors' architectures.

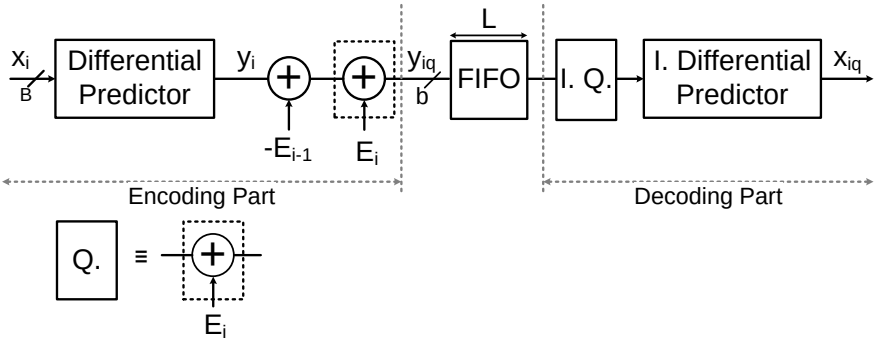


Figure 2-5. Error mathematical model of FERDC.

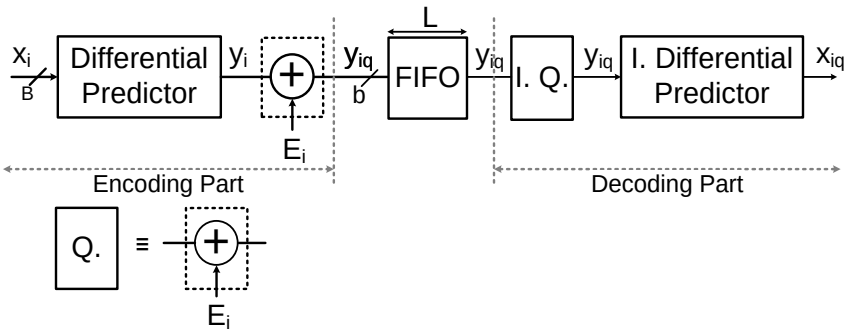


Figure 2-6. Error mathematical model of FERDC without applying the "error update" block (named as FDC).

For instance, the 310th row of horizontal low-pass filtered image "cameraman" is illustrated in Figure 2-7. The partial error, E_i and accumulated error E_{T_FDC} are drawn in Figure 2-8. As seen in Figure 2-8(a), there are some large errors such as points A, B and C due to quantization process. Note that in Figure 2-8(b) the accumulated error of point A is added to all next 68 values coming right after that, the same happens to other large error points shown in Figure 2-8(b). So it is observed that large error imposed on the output, x_{iq_FDC}, makes MSE very large. This situation happens in the positions known as

edge that is a pixel or pixels having the distinct brightness as compared to their neighbouring pixels (Figure 2-7). E_{T_FERDC} is almost similar to E_i as given in Equation 2-3.

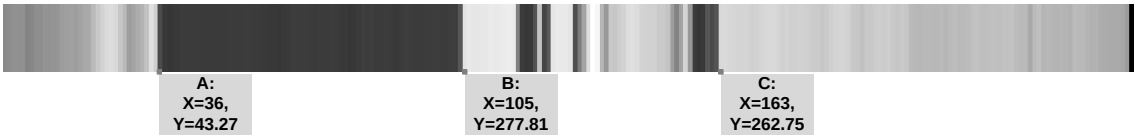
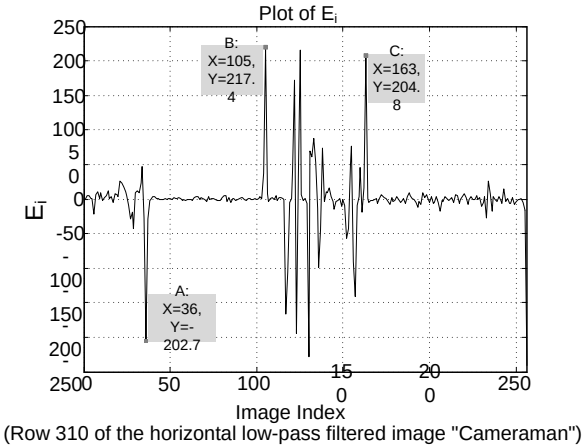


Figure 2-7. Row 310 of the horizontal low-pass filtered image "Cameraman".



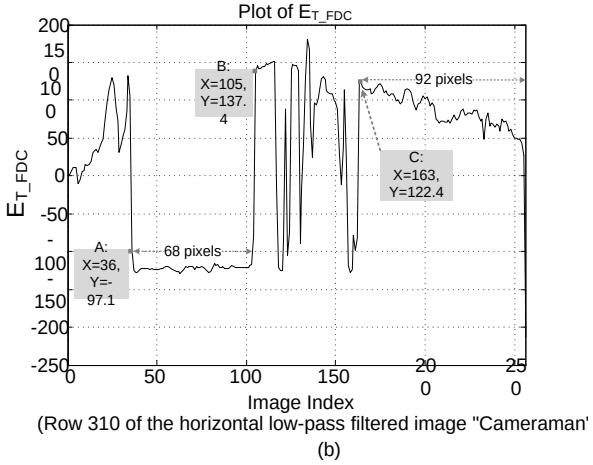


Figure 2-8. Error of the quantized difference values in Fig. 7: (a) Partial error, E_i. (b) Accumulated error, E_{T_FDC}.

As depicted in Figure 2-9, the output of FERDC (Figure 2-9(b)) compared to the output of FDC (Figure 2-9(c)) better resembles the original one (Figure 2-9(a)). MSE of Figure 2-9(b) is just 0.92 while MSE of Figure 2-9(c) is 65.98 times larger ($MSE_{\text{Figure 2-9(c)}} = 60.70$) because the error is getting accumulated in Figure 2-9(c).

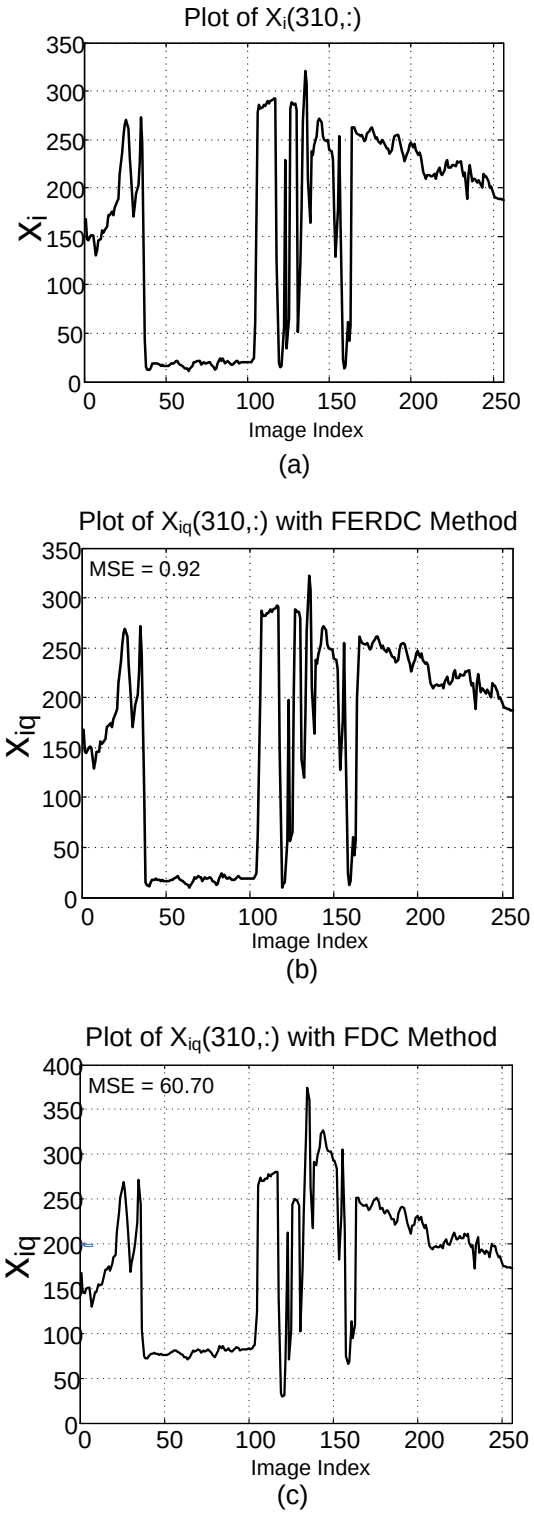


Figure 2-9. Comparison of outputs for FERDC and FDC methods with reference to original input. (a) Original input. (b) x_{iq_FERDC}. (c) x_{iq_FDC}.

Now the updated difference values can be quantized to b-bit integer values to reduce the data width and thus the FIFO size significantly. To implement a power- and complexity-effective quantizer, a non-uniformly distributed symmetric quantizer is proposed following the same distribution of its input data (Figure 2-3(b)). The quantizer comprises two parts named as Q_p and Q_n which are applied to positive and negative numbers. Q_p and Q_n have three segments determined by fL and sL as depicted in Figure 2-10.

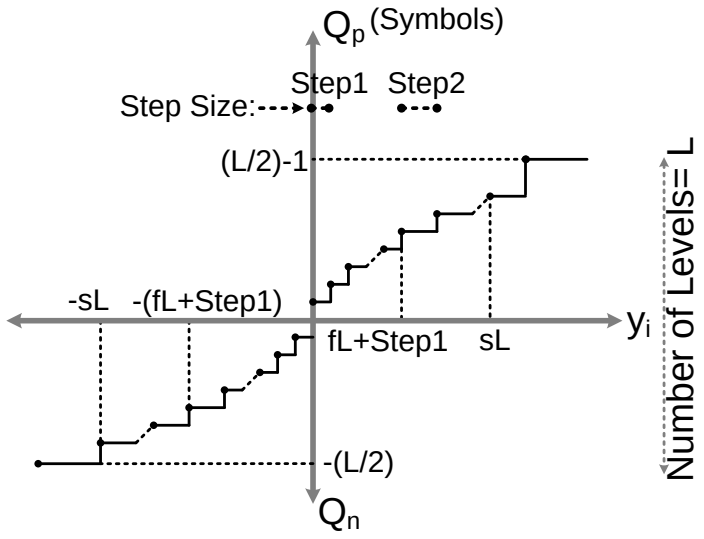


Figure 2-10. Quantizer used for positive (Q_p) and negative (Q_n) numbers.

In order to generalize the quantizer, the first two segments are considered to have the step sizes of $Step1$ and $Step2$, respectively. The output of quantizer is just a symbol which varies in the range of $[-L/2, L/2)$. If the positive number, i, lies between zero and $fL + Step1$, it is symbolized as $\lfloor i \rfloor$. If the positive number, i, lies between $fL +$

$Step1$ and sL, it is symbolized as $fL + Step1 + \lfloor (i - fL - Step1)/Step2 \rfloor$. Similarly the negative numbers in the corresponding ranges are symbolized as $-\lfloor i \rfloor$ and $-(fL + Step1) + \lfloor (i + fL + Step1)/Step2 \rfloor$. The last segment has a constant symbol which means that every value either larger than sL or less than $-sL$ is represented by one symbol. Figure 2-10 facilitates the calculation of the new FIFO cells width, b, by applying the total number of quantized levels denoted as L to Equation 2-5.

$$L = 2 * [fL\left(\frac{1}{Step1} - \frac{1}{Step2}\right) + sL\left(\frac{1}{Step2}\right) + 3 - \frac{Step1}{Step2}] \quad \text{2-5}$$

To calculate all the possible values for fL and sL with respect to $Step1$ and $Step2$, fL can be derived from Equation 2-6 as the function of $sL, Step1$ and $Step2$. The maximum value of fL is achieved as Equation 2-7 when $sL = 0$. Similarly, the maximum value of sL is also calculated when $fL = 0$ in Equation 2-8. Among all available combinations of fL and sL, those satisfy the condition of $sL > fL > 0$ are selected.

$$fL = \frac{\frac{L}{2} - sL\left(\frac{1}{Step2}\right) - 3 + \frac{Step1}{Step2}}{\left(\frac{1}{Step1} - \frac{1}{Step2}\right)} \quad \text{2-6}$$

$$fL_{max} = \frac{\frac{L}{2} - 3 + \frac{Step1}{Step2}}{\left(\frac{1}{Step1} - \frac{1}{Step2}\right)} \quad \text{2-7}$$

$$sL_{max} = \frac{\frac{L}{2} - 3 + \frac{Step1}{Step2}}{\left(\frac{1}{Step2}\right)} \quad \text{2-8}$$

For instance, if L is 256, b becomes 8 bits. Now for a certain value of b, the quantizer parameters fL and sL can be selected by monitoring the best values of MSE (or PSNR). Besides, since many image processing algorithms are implemented with fixed-point arithmetic, B can be measured at the output of the horizontal filter (Figure 2-1(b)) performed by the direct-form implementation as shown in Figure 2-11. In Figure 2-11, if the very input data is assumed to be represented by K bits and the filter coefficients (tap weight), t_{ij}, are quantized into M integer bits, the worst-case value of B is calculated as Equation 2-10, where +1 is for the sign bit. In order to compensate the M-bit integer representation of the filter coefficients, an M-bit right shifter is added in Figure 2-11.

$$C = \left\lceil log2\left(\sum_{j=1}^{N}\left|t_{1j}\right|\right)\right\rceil + \lceil log2(2^k - 1)\rceil \qquad 2\text{-}9$$

$$B = C - M + 1 \qquad 2\text{-}10$$

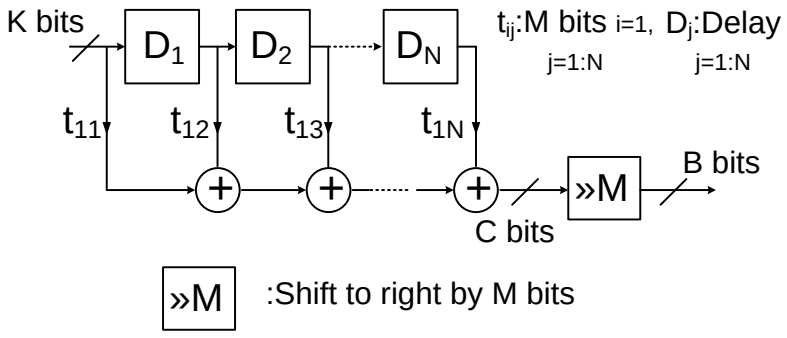


Figure 2-11. Direct-form Implementation of horizontal filters.

2.4.2 Decoder Design

In decoding part, the read data from the FIFOs are sent to the inverse quantizer (I.Q.) to retrieve the b-bit data corresponding to the symbol introduced by quantizer. The positive symbol, i, between zero and $fL + Step1$ is corresponding to value $\lfloor i \rfloor + 0.5$, the positive symbol i between $fL + Step1$ and sL is corresponding to value $fL + Step1 + \lfloor (i - fL - Step1)/Step2 \rfloor Step2 + Step2/2$ and finally symbol $(L/2) - 1$ is retrieved as a constant value, $sL + Step2/2 + 15$. Similarly, the negative symbol, i, between zero and $-(fL + Step1)$ is corresponding to value $-\lfloor i \rfloor + 0.5$, the negative symbol i between $-(fL + Step1)$ and $-sL$ is corresponding to value $-(fL + Step1) + \lfloor (i + fL + Step1)/Step2 \rfloor Step2 + Step2/2$ and finally symbol $-(L/2)$ is retrieved as a constant value, $-sL - Step2/2 - 15$. Then, the inverse differential predictor generates the output (x_{iq}) using the retrieved b-bit values (as depicted in Figure 2-4).

2.5 Near-threshold Operation Technique

Voltage scaling has been proved to be an effective way to reduce power consumption in digital circuits and systems [41]. In conventional voltage scaling, the supply voltage is scaled only to a voltage in super-threshold region, resulting in limited power reduction. Recently it has been shown that significant power reduction can be achieved by scaling the supply voltage to sub/near-threshold region [42-44]. This requires new circuit design methodology called near/sub-threshold design. Compared to sub-threshold design, near-threshold design can achieve much better performance with only slight

power increase. In the design of the proposed FIFO, near-threshold operation is adopted to achieve further power reduction.

According to the studies in the past, most of digital standard cells are able to operate in the near-threshold region except the cells with high fan-in which show functional failure or large delay variation. In this design, based on a standard cell library for nominal voltage operation, extensive simulations are performed to evaluate the performance of cells with high fan-in. The library is reconstructed by excluding the cells with fan-in more than 4. Following that, the reconstructed library is characterized at 0.5 V which is a near-threshold voltage in the selected process technology to obtain timing information for synthesis and back-end design [45]. For near-threshold operation, SRAM need to be redesigned as they have problems working at very low voltage. This usually causes large design effort [33]. In this design, however, as the FIFO size is significantly reduced, flip-flops can be used to implement FIFO. So the additional design effort is saved. Also, the flip-flop is usually more robust than SRAM in ultra-low voltage operation.

2.6 Experimental Results

To demonstrate the proposed FIFO design, it has been implemented using a 0.18 μm CMOS process technology. The implementation covers different FIFO length including 128, 256, 512 and 1024. For comparison, the FIFO using the conventional architecture as shown in Figure 2-2 is also implemented. Post-Synthesis simulation results

show that the proposed design is able to operate at 28.57 MHz at 0.5 V and 200 MHz at 1.8V. To provide realistic input data for testing, the commonly used CDF 9/7 biorthogonal filter [46] is used to implement 2D wavelet transform in MATLAB with the standard test images as its input. Then the output generated by horizontal wavelet is used as input to FIFOs.

2.6.1 Parameters Selection and MSE Results

To determine the parameters fL, sL, $Step1$ and $Step2$, first L is assumed to be 128 in Equation 2-5. In other words, the data input bit-width (B) which is 11 bits for the generated tested dataset shown in Figure 2-12, is compressed to 7-bit data before the input of FIFOs (b) in Figure 2-2(a). The tested dataset is extracted from popular standard test images in [47] to check the proposed design with the real data. Second, if the histogram of y_i (the output of differential predictor in Figure 2-2(a)) is monitored, it is observed that more than 68% of difference values lie in the range of (-74.6, 78.4) for given datasets. Figure 2-13 illustrates sL vs. fL for various combinations of $Step1$ and $Step2$. $Step1$ = 1 better fits to the variation of (-74.6, 78.4) because the maximum value of fL is equal to 61 while it is 122 for $Step1$ = 2 which is quite far from the range (-74.6, 78.4). However these values cannot be considered as the optimum values since they are a function of input data characteristics. The adaptive approach is proposed in Section 3.5.

In addition, because the average maximum value is around 615 (refer to ure 2-12) $Step2$ is selected to be 8. Hence, the maximum value of sL becomes 482

which is closer to 615 than that of $Step2$ when equals 16. The selected values for $Step1$ and $Step2$ in Figure 2-13 are the factors of two to achieve the multiplierless design.

Then average and standard deviation of MSE of dataset along with fL vs. sL are depicted in Figure 2-14. As shown in Figure 2-14, standard deviation of MSE is less than two for all fL values equal to or less than 29 which means that sensitivity of MSE to various datasets is negligible. So, finally, fL and sL leading to standard deviation of MSE and MSE less than 4 are chosen as general accepted values [34]. We selected 25 and 314 for fL and sL, respectively, which in average leads to MSE of 2.76 (Figure 2-14 and Figure 2-15).

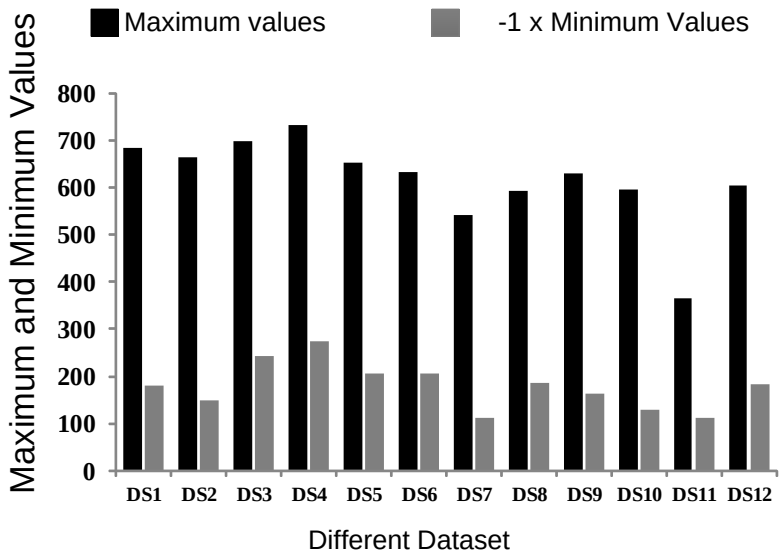


Figure 2-12. Maximum and absolute minimum values of datasets.

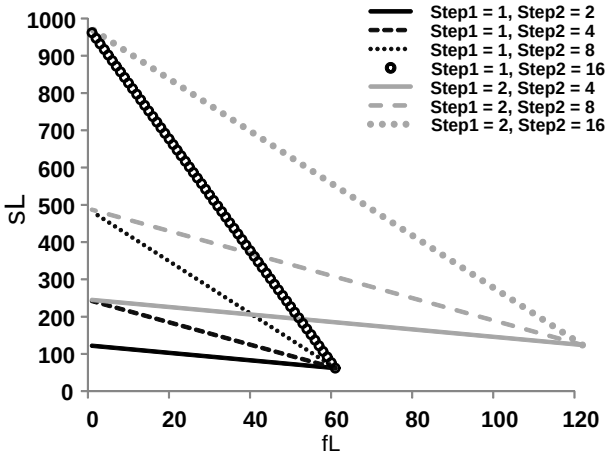


Figure 2-13. fL vs. sL for the various combinations of $Step1$ and $Step2$.

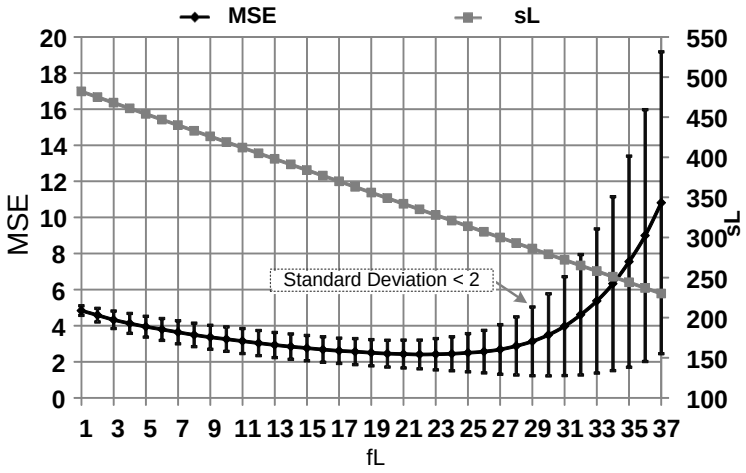


Figure 2-14. MSE and sLvs. fL for $Step1$=1 and $Step2$ = 8.

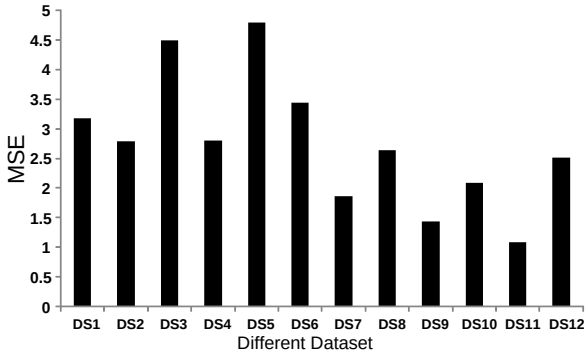


Figure 2-15. FERDC-based FIFO with L=128 (fL=25, sL=314. $Step1$=1 and $Step2$=8).

2.6.2 Area Reduction

Figure 2-16 shows the area for conventional FIFO and the proposed FIFO. The percentage of reduction increases from 25.14% to 34.91% when the width is increased from 128 to 1024. The area reduction is mainly from the reduced FIFO size due to proposed FERDC technique. Figure 2-17 shows that the overhead circuitry (including differential predictor, update error, quantizer, inverse quantizer and inverse differential predictor) varies between 17.37% (W = 128) and 2.6% (W = 1024) in the proposed FIFO which implies that area overhead is very small when W is larger than 256.

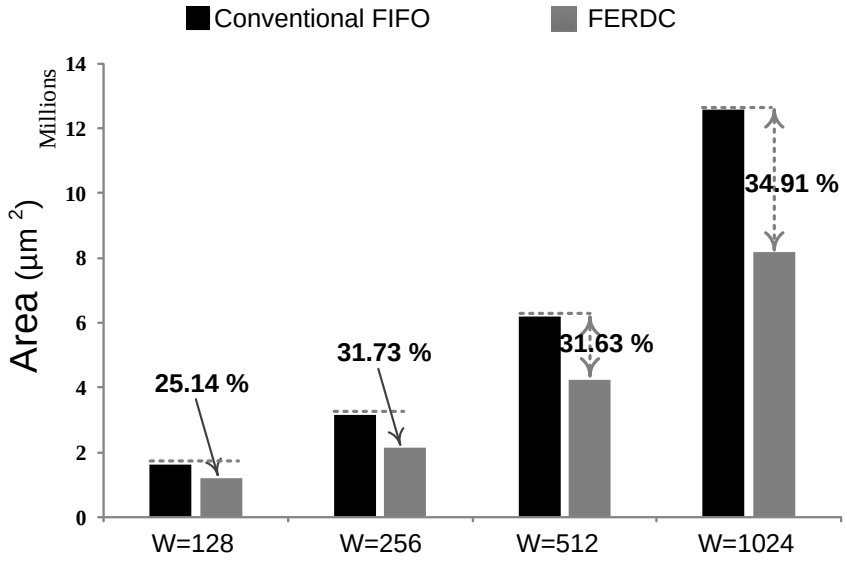


Figure 2-16. Area for the conventional FIFO and proposed FIFO.

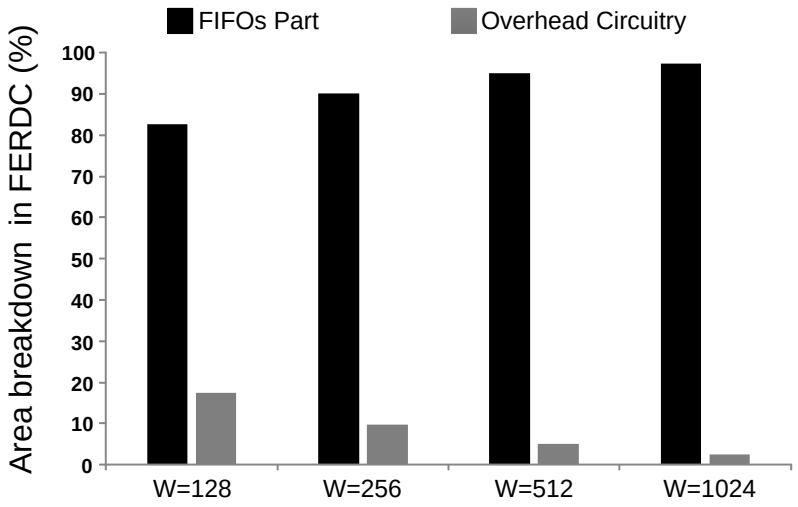


Figure 2-17. Area breakdown for the proposed FIFO.

2.6.3 Power and Energy Consumption Reduction

To estimate the power consumption, the generated tested data are applied to the conventional and the proposed FIFO. The comparison of power and energy consumption is illustrated in Figure 2-18. It shows that the total power and energy reduction is more than 90% between the conventional FIFO and the proposed FIFO with both FERDC and near-threshold operation. However the operating frequency of the proposed FIFO in near threshold region is degraded as compared to a nominal operation voltage.

Among them 14.86% ~ 28.84% (for various W) is from the FERDC and 70.47%~84.3% is from near-threshold operation. Figure 2-19 shows that the percentage of power consumption of the overhead circuitry varies from 24.65% (W = 128) to 3.79% (W = 1024) in the proposed FIFO. It is negligible for W larger than 256. The leakage power reduction of the proposed FIFO with FERDC and near-threshold operation is more than 75% compared with conventional FIFO as drawn in Figure 2-20. FERDC

contributes 17.65% to 32.73% of the leakage power reduction. The near-threshold operation exponentially reduces the leakage current which contributes 44.13%~54.9% of the leakage power reduction.

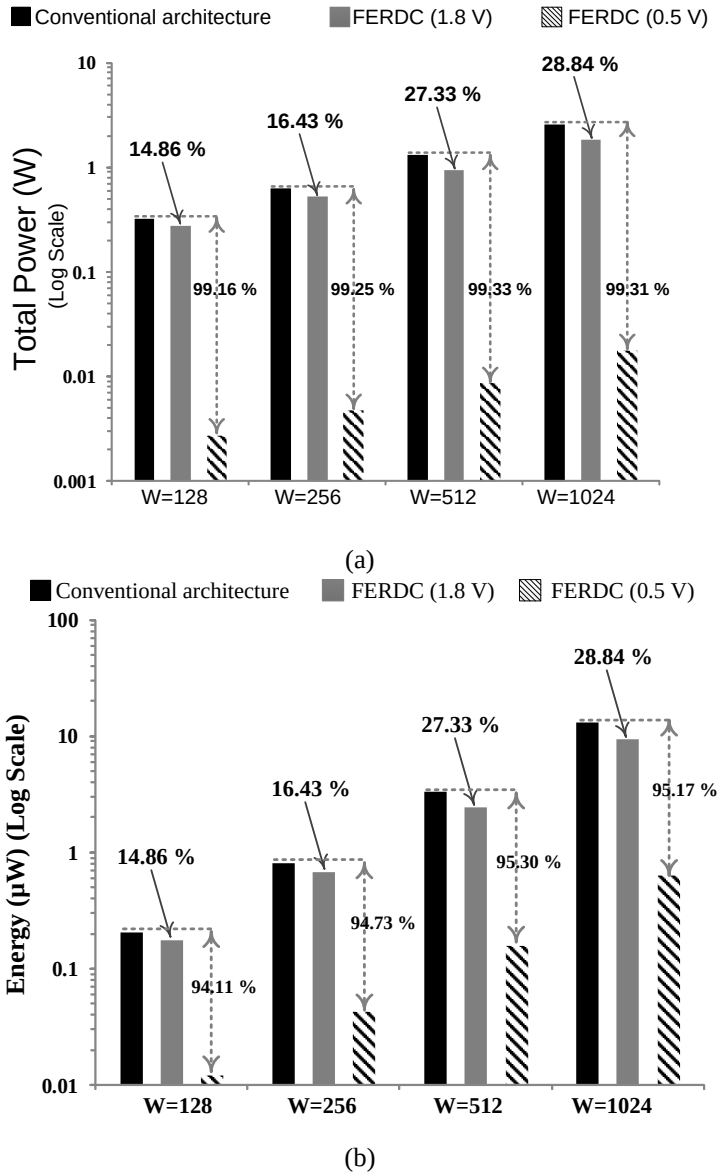


Figure 2-18. (a) Comparison between the total power of the conventional FIFO and proposed FIFO. (b) Comparison between the energy of the conventional FIFO and proposed FIFO.

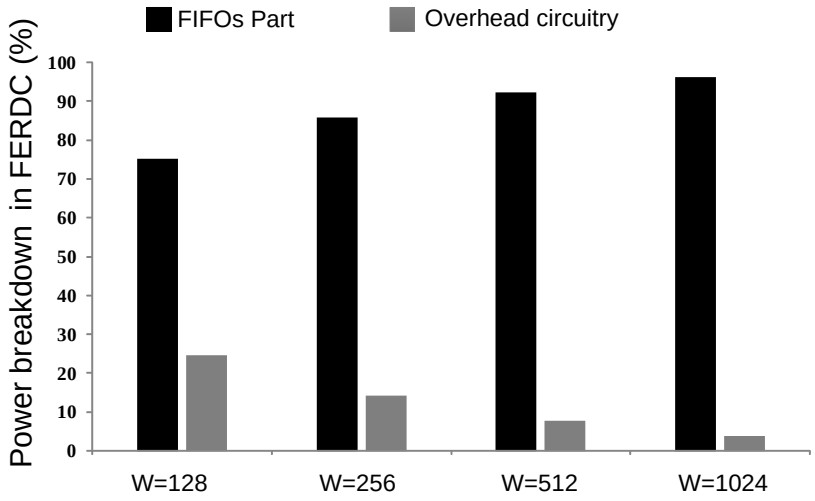


Figure 2-19. Power breakdown of the proposed FIFO for different W.

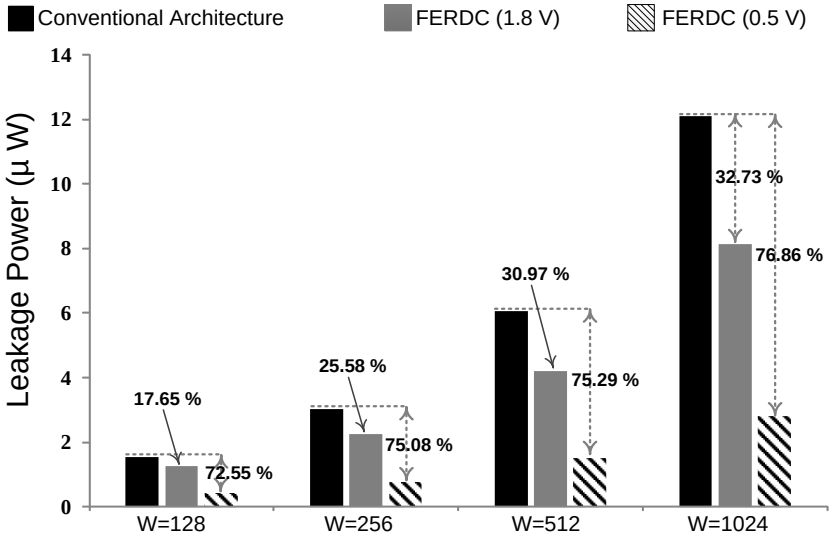


Figure 2-20. Leakage power comparison of the conventional FIFO and proposed FIFO.

2.7 Conclusion

Many image/video processing algorithms require FIFO for filtering. The FIFO size is proportional to the length of the filters and input data width, causing large area and power consumption. We have proposed an energy and area efficient FIFO design for image/video applications through error-reduced data compression and near-threshold operation. On architecture-level, FERDC technique is proposed to reduce the size and power consumption of the FIFO by utilizing the spatial correlation between neighbouring pixels and performing error-reduced data compression together with quantization to minimize the MSE. On circuit-level, near-threshold operation is adopted to achieve further power reduction while maintaining the required performance. To demonstrate the proposed FIFO, it has been implemented using a 0.18 μm CMOS process technology. The implementation covers different FIFO length including 128, 256, 512 and 1024. The experimental results show that the proposed FIFO operating at 0.5 V and 28.57 MHz achieves up to 99%, 65 % and 34.91% reduction in dynamic power, leakage power and area respectively with a small MSE of 2.76, compared to the conventional FIFO design. The proposed FIFO can be applied to a wide range of image/video signal processing applications to achieve high area and energy efficiency.

Chapter 3. Area- and Power-efficient Laplacian Pyramid Processing Engine Using FIFO with Adaptive Error Reduced Data Compression

3.1 Introduction

Recent developments in integrated circuits design enable us to realize complex algorithms on silicon. In the meanwhile, rapid advancements in image/video processing have introduced more powerful and efficient algorithms. For instance, LP, GP, Wavelet Transform (WT) and CT are extensively used for applications such as image compression, video coding, object recognition and 3D graphics [2-5]. Most of the image/video processors implemented in hardware require real-time processing performance [48-50]. In recent years, low power and miniaturized designs have become important as various portable and mobile image/video applications have been developed [10, 51, 52].

Laplacian Pyramid is one of the popular multi-resolution (multiscale) image representations in image/video processing applications such as image fusion, compression, feature extraction and object recognition [53-55]. Various LP hardware designs have been proposed [20, 56, 57]. However, they all focus on the performance rather than the power consumption. As portable applications become more and more popular, there is an increasing demand for system miniaturization and low power consumption, which is shifting the design focus from performance to area and power optimization.

As a heavily used component in LP processing engine (LPPE) as well as in other image/video processors, FIFO consumes significant portion of area and power consumption. Various techniques are available in literature for dealing with FIFOs as detailed in Section 2.3. Our work published in [24] and explained in Section 2.4 is a general architecture-level approach introducing the area and energy efficient FERDC. FERDC reduces dynamic and leakage power, and area by 28.84% and 32.73%, and 34.91% respectively. However, due to its non-adaptive nature, the MSE increases significantly over a wide range of input images. Moreover, the more FIFO size reduction was achieved at cost of large MSE, which is not applicable to practical implementations. In this chapter, the LPPE chip focusing on power and area optimization is proposed. The power and area optimization is performed utilizing a novel FIFO architecture with adaptive error reduced data compression (FAERDC), and near-threshold operation. In addition to the power and area optimization, a new extension method is proposed to reduce the output errors caused by boundary pixels of input image.

This chapter is organized as follows. Section 3.2 introduces the fundamental of LP and its application. Section 3.3 presents the existing hardware architectures of LP followed by hardware design considerations of LP in Section 3.4. Proposed FAERDC and semi-periodic extension techniques are described in Sections 3.5 and 3.6. Section 3.7 explains the detailed implementation of LPPE and Section 3.8 demonstrates the experimental and simulation results. Finally, Section 3.9 concludes the chapter.

3.2 Fundamental of Laplacian Pyramid and its Applications

LP comprises decimation and interpolation parts as shown in Figure 3-1. The decimation part, denoted as Gaussian pyramid (GP), produces low frequency coefficients (LFC) from an input image (C_0). The output of the decimation part is further processed by the interpolation part to calculate an estimated version of the input image. Subsequently, high frequency coefficients (HFC) are obtained by subtracting the estimated image from the original input image. The low frequency output can be fed back to the input to construct the hierarchical representation of the input image.

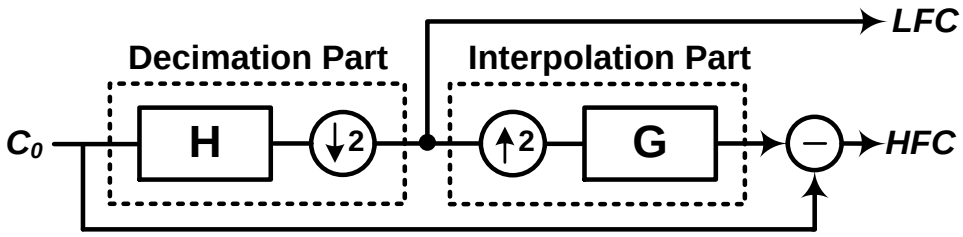


Figure 3-1. Laplacian Pyramid

Illustrative example in Figure 3-2 shows three resolutions of the image in which each new resolution is smaller in size and has a less amount of information as compared with previous ones.

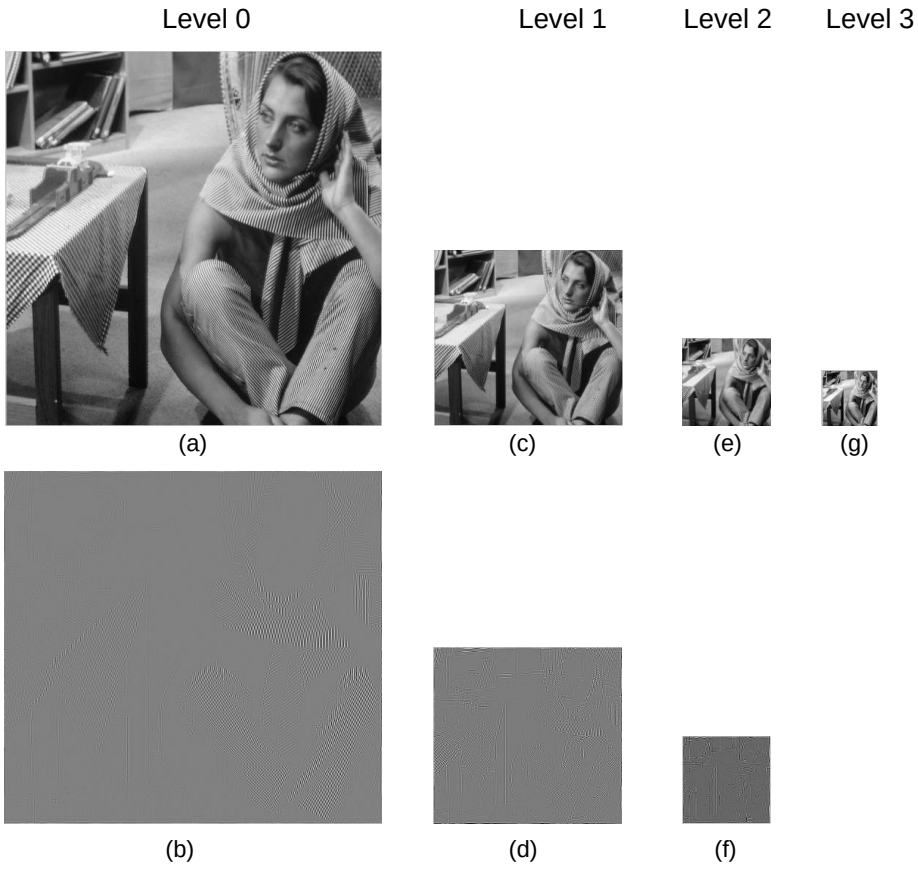


Figure 3-2. Hierarchical LP. (a) Input image, Barbara. (b) Scale 0 HFC. (c) Scale 1 LFC. (d) Scale 1 HFC. (e) Scale 2 LFC. (f) Scale 2 HFC. (g) Scale 3 LFC.

Laplacian pyramid was first proposed in [53] for image coding. Since LP only provides one bandpass signal [58], its output can be further fed to directional filter banks in order to do subband decomposition such as Contourlet transform [17]. LP is also used to realize the image fusion which requires real-time image processing and a smaller memory size [59]. Another popular application of LP is in optical flow [60] where motion estimation is approximated by visual displacement, and LP is applied to deal with the large displacement and noisy areas.

3.3 Existing Hardware Architectures of LP

First hardware-oriented architecture of the LP was introduced in [61]. It uses a so-called “segmented pipeline” to keep the processing element run at their maximum capacity. The first LP chip named as PYR chip is proposed in [62]. It functions at 15~20 MHz and processes a 512x480 image in 22.7 msec. In [63], a video processor utilizing three LP processors is realized on FPGA platform to reduce the processing time at the price of larger memory. In [64], A pattern-selective pyramidal image fusion based on GP and LP is implemented on FPGA. In [65], another real-time LP-based image fusion is realized on FPGA and can process dual channels 640x480 images at 25 frames per second. In [66], a color image fusion algorithm built upon the LP is implemented by FPGA and showed the real time performance for surveillance and security applications. A vision processor named as Acadia I in [67] is able to perform vision applications such as video stabilization, mosaicking, video fusion and enhancement. Acadia I operates at 100MHz and can process one 512x512 image within 2.75ms. The second generation of Acadia I chip is proposed in [68], where the pyramidal processing unit remains the same. In [69], LP is used to estimate the motion (optical flow) and disparity of color images. It supports real-time operation up to 36 fps for 640x480 input images. Real-time implementation of image blending in [57] is another LP application implementing four cascading LP blocks by FPGA to realize a dual video stream at 420 MHz. A new LP architecture presented in [20] tries to reduce the computational complexity by applying polyphase decomposition and noble identities. The existing LP hardware implementations mainly focus on the performance optimization. As portable applications become more

and more popular, there is an increasing demand for system miniaturization and low power consumption, which is shifting the design focus from performance to power and area.

3.4 Hardware Design Considerations of LP

Figure 3-1 shows the conventional LP architecture. In general, every other row and column of the output of H is discarded to generate the low-pass filtered output. After that, LFC is upsampled by adding a zero between subsequent samples and is convolved by a filter (G) to obtain the high frequency component of the input image. In the decimation part, 75% of the filter (H) outputs are redundant and should be removed after downsampling. A similar condition happens in the interpolation part where zeros added between neighboring pixels are involved in mathematical operations. Consequently, redundant arithmetic operations slow down the overall processing time as well as increase the power. All the existing designs except for [20] suffer from this issue. In [20], an improved LP architecture is presented. It uses the poly-phase representations and the noble identities [70] to reverse the order of resampling and filtering, and hence avoid redundant and zero-operand operations. In this LP architecture, filters (H and G) are realized by 1D filters as shown in Figure 3-3. The row and column blocks in Figure 3-3 can be replaced by the equivalent blocks illustrated in Figure 3-4(c) and Figure 3-4(f). The LP architecture after filter replacement is depicted in Figure 3-5. It is worth noticing that, contrary to Figure 3-3, two rows of an image are processed in Figure 3-5 (Row_{2i} and

Row_{2i-1}) because the equivalent column block in decimation part in Figure 3-3 necessitates its second input from the next row [20].

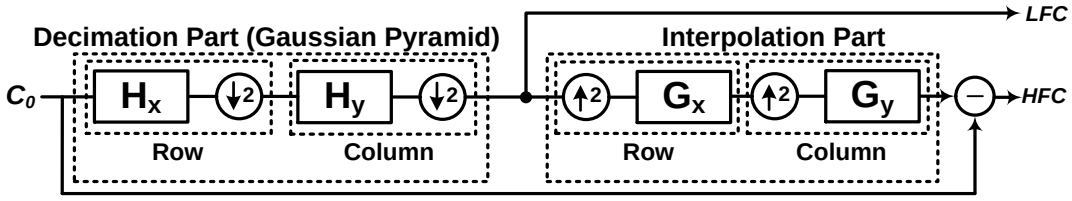


Figure 3-3. LP realized by 1D filters.

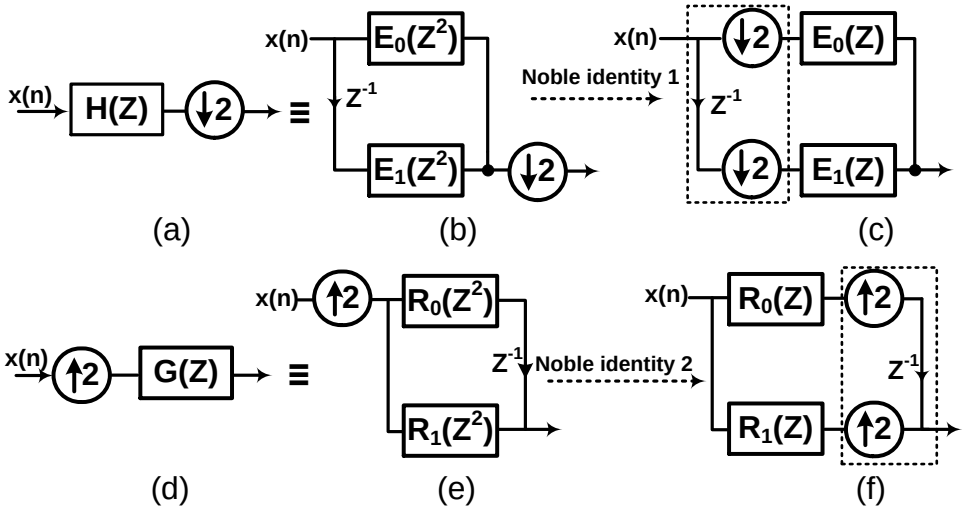


Figure 3-4. Equivalent block for the combinations of filter and resampling.(a) Filter H with downsampling. (b) Polyphase decomposition of filter H. (c) Reversing the polyphase components of filter H with downsampling after performing the noble identity 1. (d) Filter G with upsampling. (e) Polyphase decomposition of filter G. (f) Reversing the polyphase components of filter G with upsampling after performing the noble identity 2.

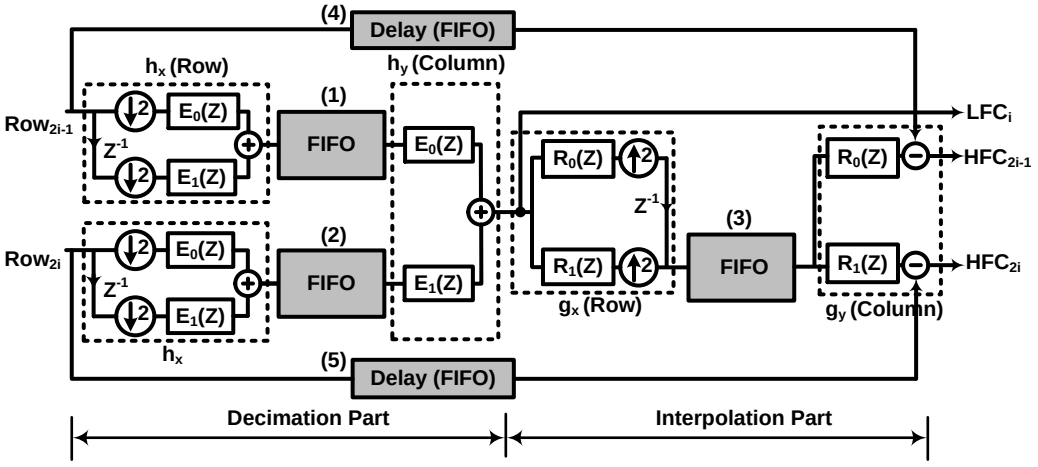


Figure 3-5. LP architecture in [20].

One of the main bottlenecks of achieving high area and power efficiency lies in the optimization of FIFO [71]. In LP, FIFO is the major component consuming power and area. Each column block (Figure 3-4) requires a big FIFO to keep the essential number of data, generated by the row block. Its size is the function of the filter length, the input image width and the input data bit-width. A FIFO block for a 5th-order filter requires 4 FIFOs, keeping the necessary amount of data to start filtering. Moreover, since the Delay blocks in Figure 3-4 synchronize the original input data with the outputs of "g_y", they are also realized by the FIFOs. Their size is defined by the input data bit-width and the total delay introduced by decimation and interpolation parts. The aforementioned delay is the function of the extension method and filters lengths applied in decimation and interpolation parts.

As we discussed in Chapter 2, various techniques to optimize FIFOs power and area can be categorized as architecture-level and circuit-level techniques. On architecture level, most of the techniques focus on algorithm dependent optimizations [39, 40]. The circuit level techniques mainly include latch-based FIFO [11], ultra-low power SRAMs [33], memory-splitting and clock gating techniques [19]. We have shown a general approach [23, 24] for area and power optimization of FIFO with error-reduced data compression in Section 2.4. However, due to its non-adaptive nature, the MSE increases significantly over a wide range of input images and therefore further power and area improvement is not achievable because the more compression on FIFO cells deteriorates the output MSE considerably. In the next Section, we will show how the FIFO area and power can be reduced by the proposed adaptive data compression technique.

Another design issue of the existing LP hardware architectures is caused by the boundary pixels where a filter window is not completely covered by the input image [35]. Methods such as constant extension, duplication extension and mirroring extension are typically applied to address this issue as described in Figure 3-6. The constant extension method basically assigns a predefined constant value to the image boundary. The duplication and mirroring methods extend the image across the boundary by deploying available pixels outside of the image. However, they all give rise to undesired artifacts within the image boundary, which also deteriorates the MSE of the low and high frequency outputs.

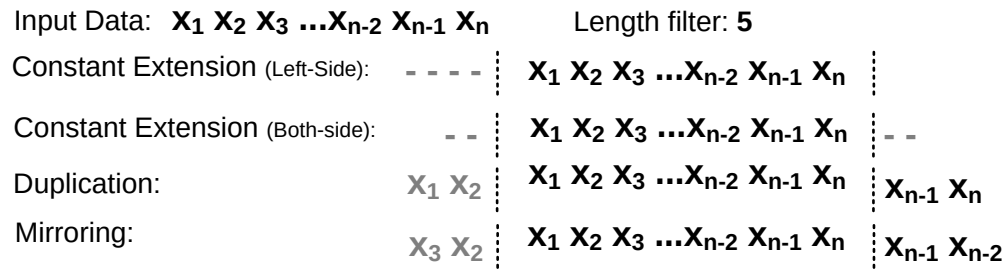


Figure 3-6. Conventional extension methods.

3.5 Proposed FAERDC Principles

3.5.1 FERDC Technique and its Potential Drawbacks

Figure 3-7(a) shows the conceptual block diagram of FERDC for a single FIFO [37] which is fully explained in Chapter 2. It consists of encoding and decoding parts. The input data (x_i) is first horizontally decorrelated by the differential predictor

(Figure 3-7(a)) and quantizer (Q.) (Figure 3-7) to reduce B bits to b bits where $b < B$. In addition, the update error block is applied to avoid adding the quantizer errors to very final output (x_{iq}). The quantizer (Q.) has three segments for both positive and negative input values which are represented by fL and sL. The first and second segments have the step size of $Step1$ and $Step2$, respectively and the last segment is represented by only one symbol. These parameters in [24] are constant and determined based on the simulation of some images and therefore they would not be updated with respect to the input data. The output of the quantizer is a b-bit symbol that is stored in FIFO whose word width is b bits which is less than B bits. The FIFO is implemented by a FF-based register bank to further reduce the power as compared to a conventional FIFO in which FFs connected in series. Similarly, in decoding part, the b-bit symbol from the FIFO is passed through I.Q. and real data is retrieved and then decoded by inverse deferential predictor (Figure 3-7(b)) to obtain the original input data.

Further studies on FERDC show that there are some input images having significant error at the output. It is because all the quantizer parameters of FERDC are constant and is not updated according to input data. In other words, when quantizer input ($Y_{i\varepsilon}$) faces different numbers of large values for various images, it is required to have variable quantizer parameters to reduce the error. However the FERDC quantizer is constant and equally behaves to all sets of input values which generate a big error when its input has large values. This situation becomes unacceptable when the large values located on either of the last segments of Figure 3-7(c). Since the last segments are only represented by one symbol, all input data are mapped similarly in encoding part and then they are all

retrieved to the fixed values meaning $\frac{L}{2}-1$ or $\frac{-L}{2}$ in decoding part. Therefore, a huge accumulated error appears in the final output. The MSE of LP outputs using FERDC are tabulated in Table 3-1 for some input images. It is seen that the MSE of LFC and HFC are very big due to the large accumulated errors introduced during the encoding and decoding phases.

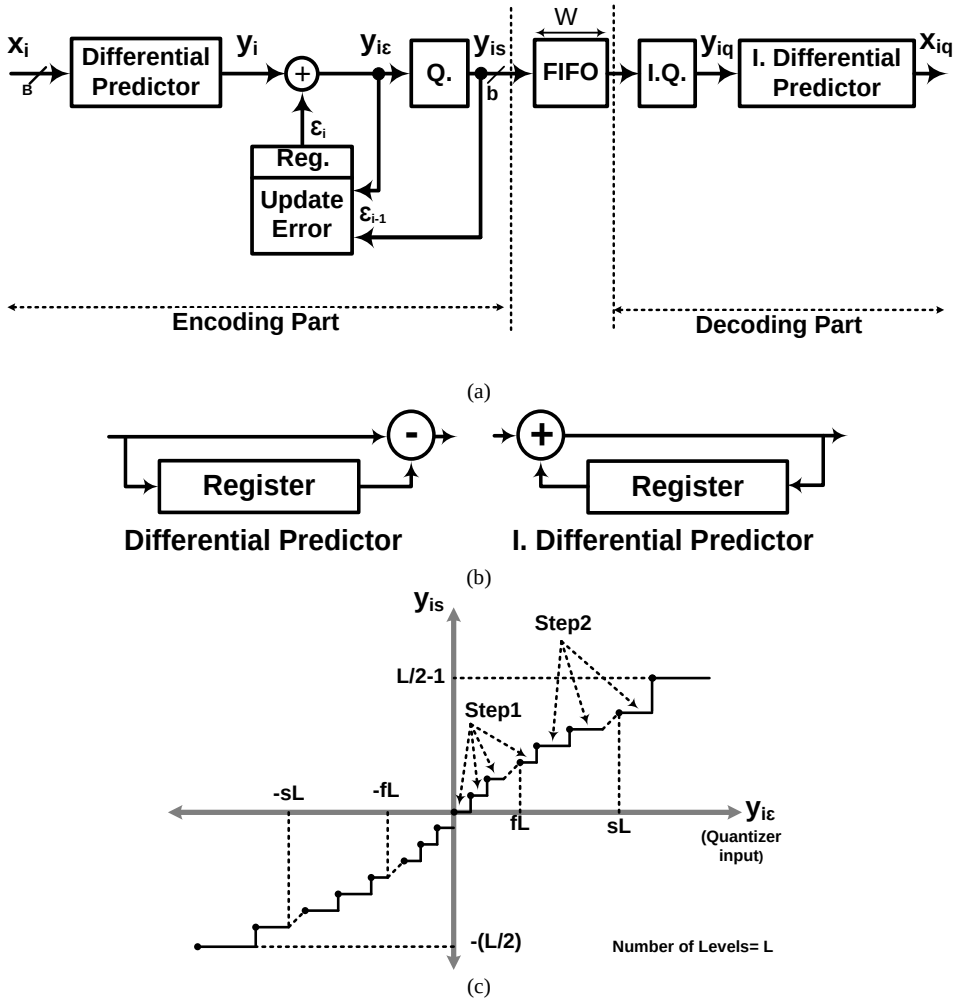


Figure 3-7. (a) A block diagram of FERDC. (b) Differential and I. Differential Predictors. (c) Quantizer for positive and negative numbers.

Table 3-1 MSE comparison between FERDC and FAERDC

Image	FERDC		FAERDC		Improvement (%)	
	LFC	HFC	LFC	HFC	LFC	HFC
Case1	53.9	13.5	1.5	0.9	97.2	93.3
Case2	82	4.4	1.4	0.9	98.3	79.6
Case3	119.1	29.5	1.9	1.8	98.4	93.9
Case3	198.7	49.8	2.7	1.4	98.6	97.2

3.5.2 FAERDC Technique

To address the above issues, I propose a FIFO with adaptive error reduced data compression technique (FAERDC) shown in Figure 3-8. Quantizer Tuner block consists of two parts. First, it adaptively updates the quantizer parameters ($fL,\ sL,\ Step1$ and $Step2$) for the next row by a function of the present row to encompass the input data spectrum within $[-sL, sL]$. Second, it reduces the error of the input data falling outside of $[-sL, sL]$ while processing the next row. To see how it updates the quantizer parameters, the total number of quantizer levels is denoted as L and is written in Equation 3-1. fL and sL are accordingly calculated as Equations 3-2 and 3-3. If L is set to 128 (meaning b is 7 bits) (Figure 3-7), sL vs. fL can be drawn for various combinations of $Step1$ and $Step2$ as illustrated in Figure 3-9(a). If $Step1$ and $Step2$ are adaptively chosen as (1, 4) or (1, 8), sL can vary from 242 to 482 which is quite enough for many applications. Then, fL and sL should be adaptively calculated with respect to Equations 3-2 or 3-3 which are derived from Equation 3-1. $Step1$ is selected as 1 in both cases because the quantizer input is densely populated in $[-fL, fL]$, and therefore it is necessary to

have a finer resolution in this region. In order to select a combinations of $Step1$ and $Step2$ for the next row, the maximum value of y_i (named as $Max(y_i)$) from the present row needs to be calculated. If $Max(y_i)$ is less than 242, it is assigned to sL and fL is calculated with Equation 3-2. Otherwise $Max(y_i)$ is larger than 242 and assigned to sL and then fL is calculated with Equation 3-3. $Step2$ is selected as a power of two values to minimize the hardware complexity.

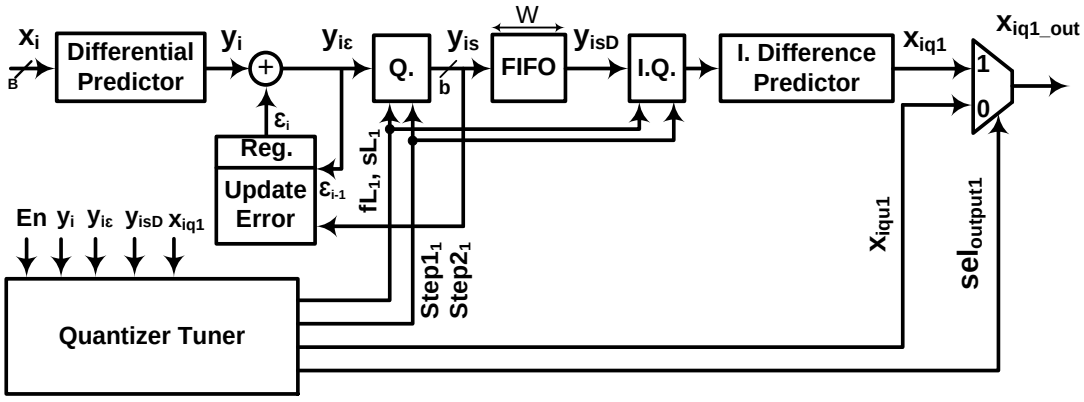
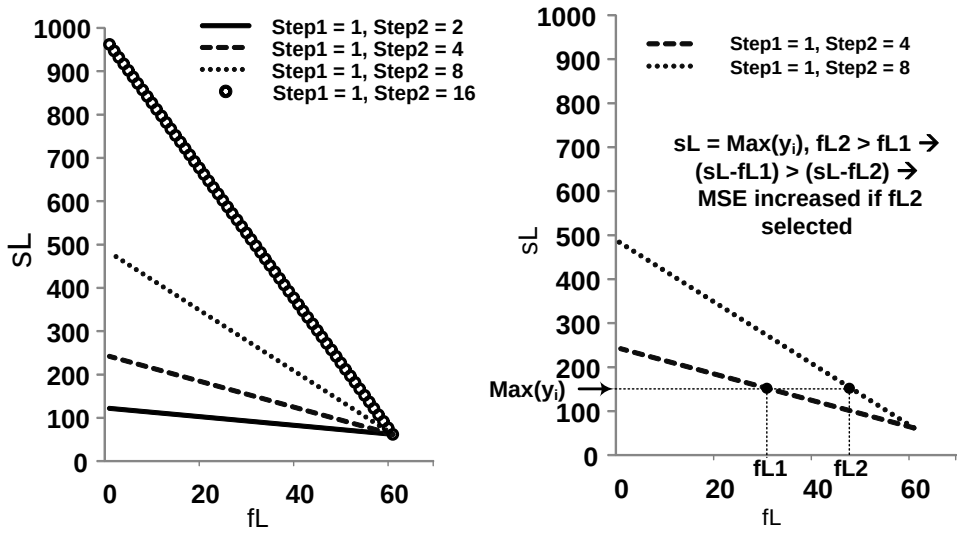


Figure 3-8. Proposed FIFO with adaptive error reduced data compression technique (FAERDC).



(a) (b)

Figure 3-9. (a) fLvs. sL for the various combinations of $Step1$ and $Step2$. (b) The reason why at least two ($Step1, Step2$) combinations are required.

$$L = 2 * [fL\left(\frac{1}{Step1} - \frac{1}{Step2}\right) + sL\left(\frac{1}{Step2}\right) + 3 - \frac{Step1}{Step2}] \quad 3\text{-}1$$

$$fL = \frac{245 - sL}{3} \qquad for\ Step1 = 1\ and\ Step2 = 4 \quad 3\text{-}2$$

$$fL = \frac{489 - sL}{7} \qquad for\ Step1 = 1\ and\ Step2 = 8 \quad 3\text{-}3$$

For $Max(y_i)$ smaller than 242 if $fL2$ is selected instead, as shown in ure 3-9(b), the $Step2$ region becomes contracted. Therefore the number of quantizer levels in this region is reduced and added to the $Step1$ region. In this case, overall MSE is degraded because more errors are contributed to the outputs due to less quantizer levels in the $Step2$ region. In other words, when the maximum value of the input row is smaller than 242, it is not required to allocate more quantizer levels to the $Step1$ region.

Besides, if there are some cases in which values larger than 500 are generated $(Step1, Step2)$ equal to (1,16) can be included in decision-making. As a result, this technique in fact exploits the vertical correlation between the subsequent rows and is hereafter referred as Quantizer Parameters Update block.

Although the present row is used to calculate the quantizer parameters of the incoming row, while processing the next row, there may be some values fall outside $[-sL, sL]$ (last segments of quantizer in Figure 3-7(c)). Since all those values are represented just by one symbol, the quantizer error is large. To suppress this error, all similarly generated errors in the encoding part are averaged and used later to be added to the

retrieved data (x_{iq1}) for improving MSE. This is referred as Average Update block. For other retrieved data, x_{iq1} is directly sent to output. The last two columns in Table 3-1 show the MSE improvement achieved by proposed FAERDC, which is more than 90%.

3.6 Proposed Semi-periodic Extension Method

The original extension method of LP introduced in [17] is periodic and only suitable for non-casual software implementation when the input image is completely stored in the memory. A new extension method named as semi-periodic extension is proposed and shown in Figure 3-10. For instance, for a 5th-order filter, first two data of each row is added to the end of the row and two last data of each row is added to the beginning of the next row. To do the column filtering, the mirroring extension method is used. The LP simulation result using the proposed extension method over 100 standard test images in Figure 3-11 reveals that the MSE of LFC is improved by 41.2% and 76.6% compared with mirroring and constant extension methods, respectively. MSE of HFC is also improved by 45.1% and 86.2% compared with mirroring and constant extension methods, respectively.

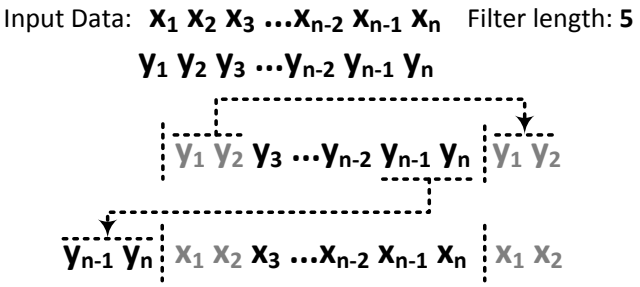


Figure 3-10. Proposed semi-periodic extension method for row filtering.

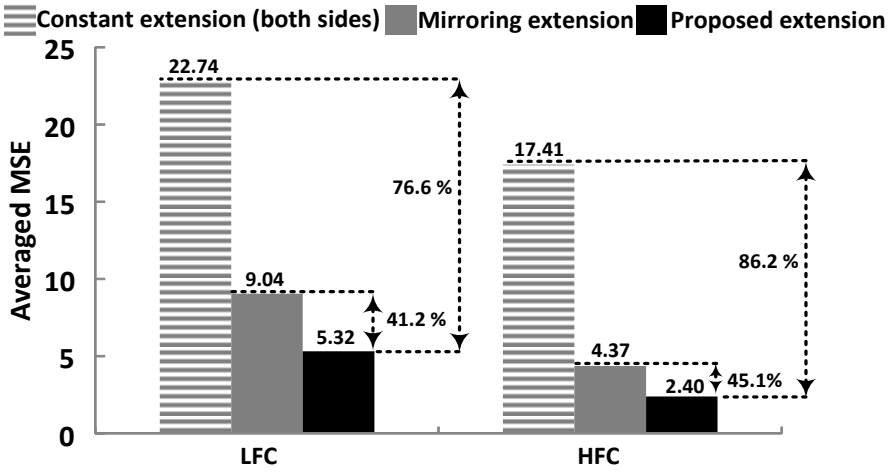


Figure 3-11. MSE comparison between the proposed extension and other extension methods over 100 standard test images.

3.7 Proposed LP Processing Engine (LPPE) Implementation

This section will describe the hardware implementation of the LPPE utilizing the proposed FAERDC and the semi-periodic extension method.

3.7.1 FIFO with FAERDC

Proposed LPPE employs FAERDC in the column filtering and data synchronization blocks to reduce the area and power. The '9/7' filters introduced in [46] are selected as H and G filters in Figure 3-1. After applying polyphase decompositions and noble identities, the E_0, E_1, R_0 and R_1 blocks (Figure 3-5) are 5th-, 4th-, 4th- and 3th-order filters, respectively. Figure 3-12 illustrates the proposed FIFO architecture for a 5th-order filter. In Figure 3-12, the output of each FIFO, named as y_{isjD} where j varies from 1 to 4, is connected to the input of the next FIFO core. So, only one encoding part is required. Moreover, the quantizer tuner calculates only the quantizer parameters for FIFO Core$_1$. The calculated parameters are stored to be used in the subsequent FIFO cores. However, each row needs a separate decoding part to generate the desired inputs for the filter.

In data synchronization application whose input-output delay is larger than input image width (W), the proposed FIFO architecture can be utilized. For instance, if it is needed to have a four-row ($4W$) delay between the input and output, the data synchronization in Figure 3-13 is proposed. Since there is only one output, one decoding part for the last FIFO is required. So, in general, MW-data synchronization, where M is larger than 1, can be designed by adding the M numbers of FIFO between encoding part and decoding part.

Five FIFO blocks in Figure 3-5 are numbered from one to five and a set of parameters is denoted as (B, L, N). They determine the FIFO block specification (FBS), where B is the input bit-width, L is the FIFO length and N represents the number of FIFOs in the FIFO block. Since there are five FIFO blocks, the FBSs are (12, $W/2$, 4), (12, $W/2$, 3), (12, W, 3), (9, W, 4) and (9, W, 4), respectively, where W is the input image width. For instance, the FIFO block, numbered as 1 and preceded E_0 (Figure 3-5), requires four FIFOs with length of $W/2$ because E_0 is a 5^{th}-order filter. Regarding the delay blocks, numbered as 4 and 5 in Figure 3-5, the number of FIFOs is 4 because the delay from input to output is the function of filter extension method which is measured as $4W$. For FIFO block 1, the FIFO architecture in Figure 3-13 can be exactly used, while for FIFO blocks, numbered as 2 and 3, the number of FIFO cores in Figure 3-12 should be reduced to 3. For FIFO blocks, numbered as 4 and 5, the FIFO architecture in Figure 3-13 is applied without further modification. In all proposed FIFO architecture, the total numbers of quantizer levels, L, is set to 128, meaning that all FIFOs in FIFO blocks are 7-bit wide (b in Figure 3-12 and Figure 3-13)

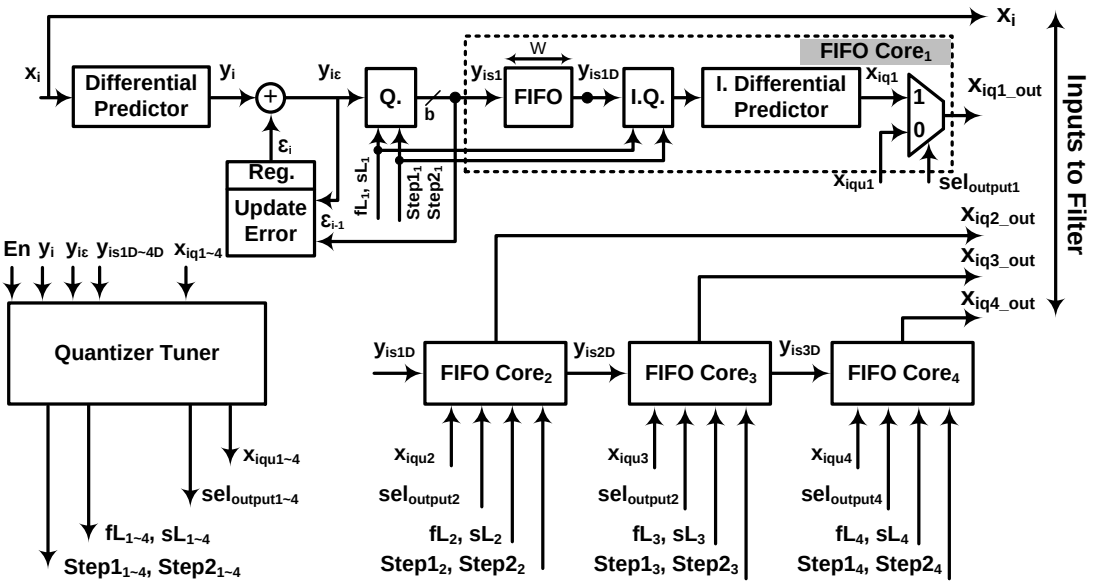


Figure 3-12. Proposed FIFOs architecture for a 5th-order filter.

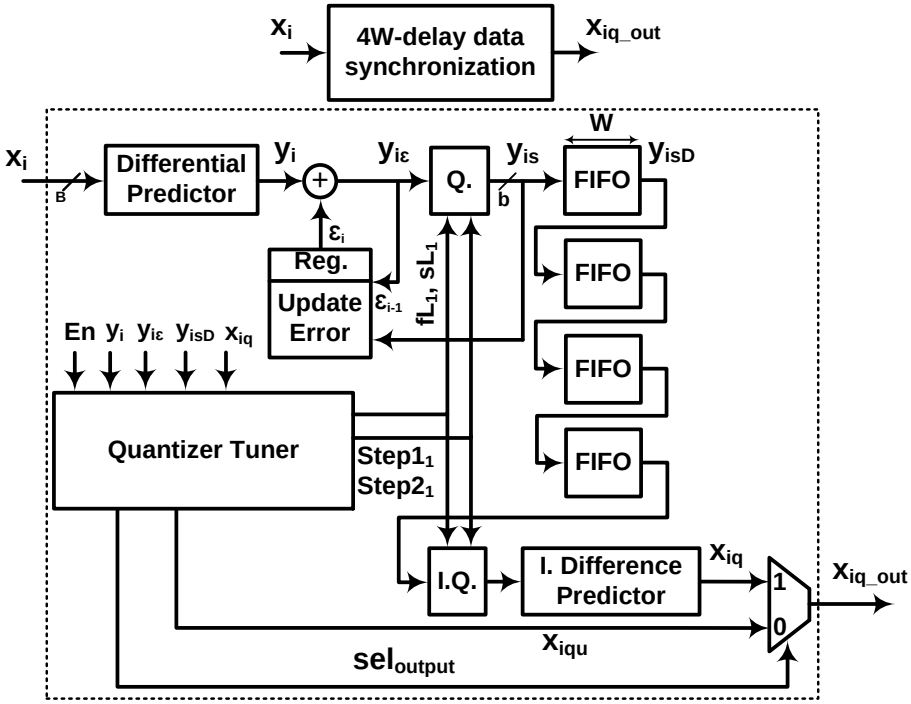


Figure 3-13. Proposed 4*W*-delay data synchronization. It has only one decoding part compared to Figure 3-12.

The “Differential Predictor” and “Update Error” are implemented as shown in Figure 3-14. The error of the previous input is generated by taking the difference between $y_{i\varepsilon}$ and y_{iq} where y_{iq} is calculated in “Q.” block. Then the output, $y_{i\varepsilon}$, is sent to “Q.” formulated as Equation 3-4.

$$y_{is} = \begin{cases} \begin{cases} \lfloor y_{i\varepsilon} \rfloor & 0 \le y_{i\varepsilon} < A \\ A + \left\lfloor \frac{y_{i\varepsilon} - A}{Step2} \right\rfloor & A \le y_{i\varepsilon} < B \\ 63 & B \le y_{i\varepsilon} \end{cases} \\ \begin{cases} \lfloor y_{i\varepsilon} \rfloor & -A \le y_{i\varepsilon} < 0 \\ -A + \left\lfloor \frac{y_{i\varepsilon} + A}{Step2} \right\rfloor & -B \le y_{i\varepsilon} < -A \\ -64 & y_{i\varepsilon} \le -B \end{cases} \end{cases} \qquad 3\text{-}4$$

$$A = fL + Step1$$

$$B = (fL + Step1) + \left(\left\lfloor \frac{sL - fL}{Step2} \right\rfloor + 1\right) \times Step2$$

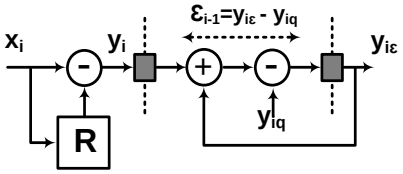
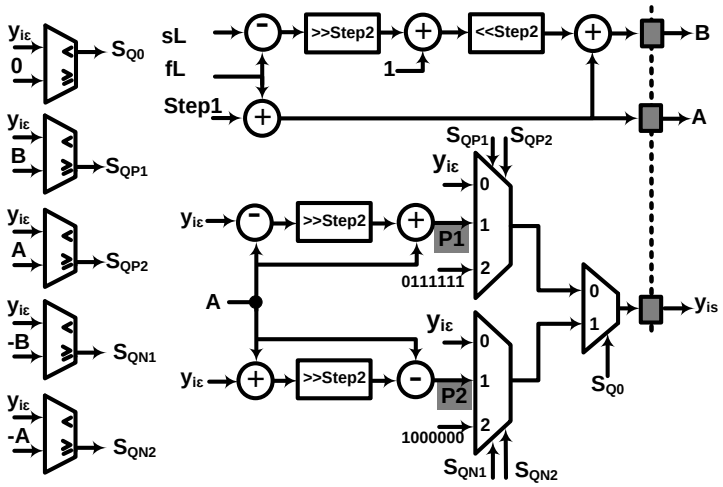


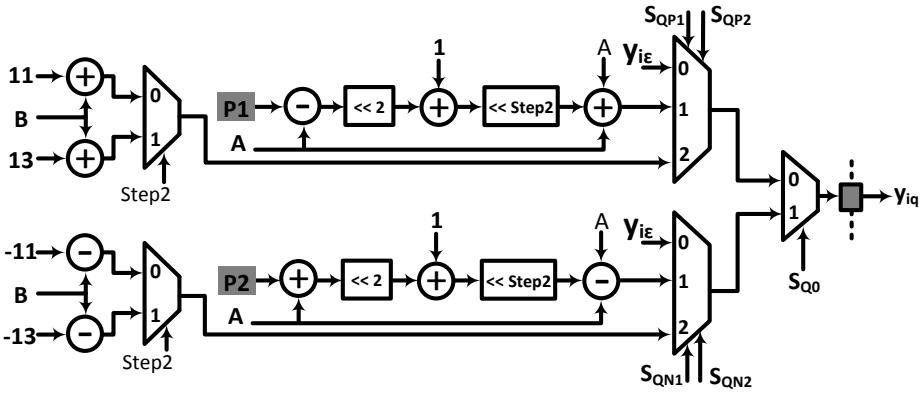
Figure 3-14. Differential Predictor and Update Error hardware implementation.

The hardware implementation of “Q.” is shown in Figure 3-15(a)-(b) where A and B specify the quantizer different areas. Figure 3-15(a) calculates 7-bit y_{is} (Figure 3-7(c)) which is the corresponding symbol of $y_{i\varepsilon}$. Similarly, Figure 3-15(b) calculates the y_{iq} expressed in Equation 3-5 which is applied to Figure 3-14 to generate $y_{i\varepsilon}$. After calculating y_{is}, it is stored in FIFOs which is realized by the conventional FF-based FIFO. Once the FIFOs are full, the output of FIFOs block is then passed through “I.Q” and inverse differential predictor shown in Figure 3-16 and Figure 3-17.

$$y_{iq} = \begin{cases} \begin{cases} \lfloor y_{is} \rfloor + 0.5 & 0 \le y_{is} < A \\ A + (2(y_{is} - A) + 1) \times \dfrac{Step2}{2} & A \le y_{is} < 63 \\ B - \dfrac{Step2}{2} + 15 & y_{is} = 63 \end{cases} \\ \begin{cases} -\lfloor y_{is} \rfloor & -A \le y_{i\varepsilon} < 0 \\ -A + (2(y_{is} + A) + 1) \times \dfrac{Step2}{2} & -63 \le y_{is} < -A \\ -\left(B - \dfrac{Step2}{2} + 15\right) & y_{i\varepsilon} = -64 \end{cases} \end{cases} \quad 3\text{-}5$$



(a)



(b)

Figure 3-15. Quantizer hardware implementation. (a) Calculating y_{is} (b) Calculating y_{iq}.

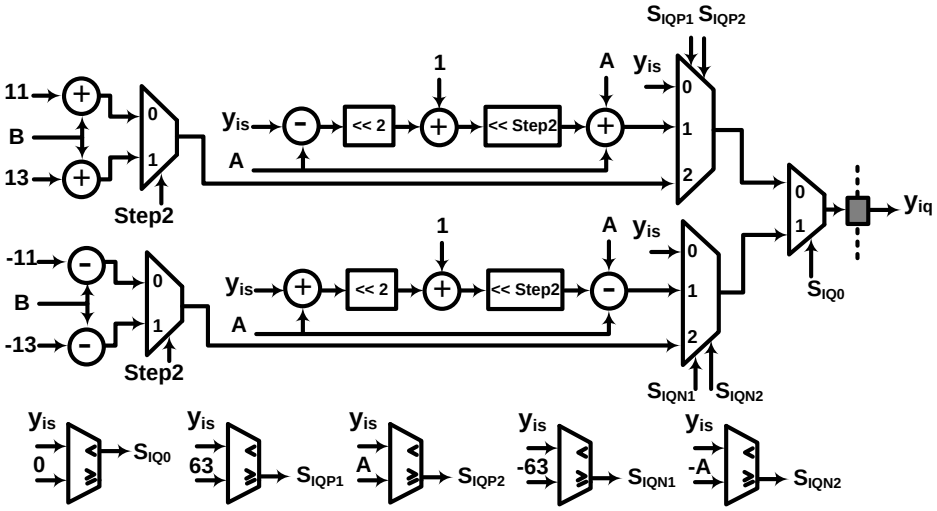


Figure 3-16. I.Quantizer hardware implementation.

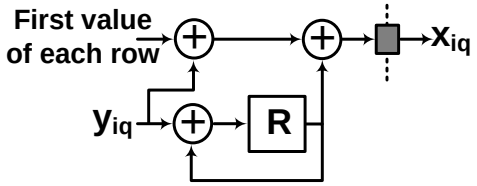
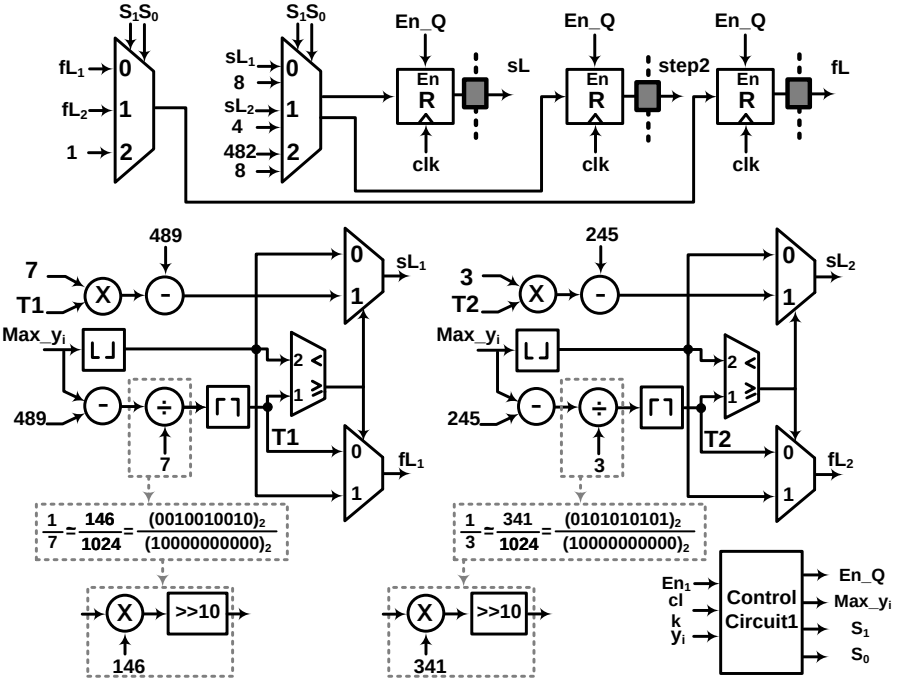


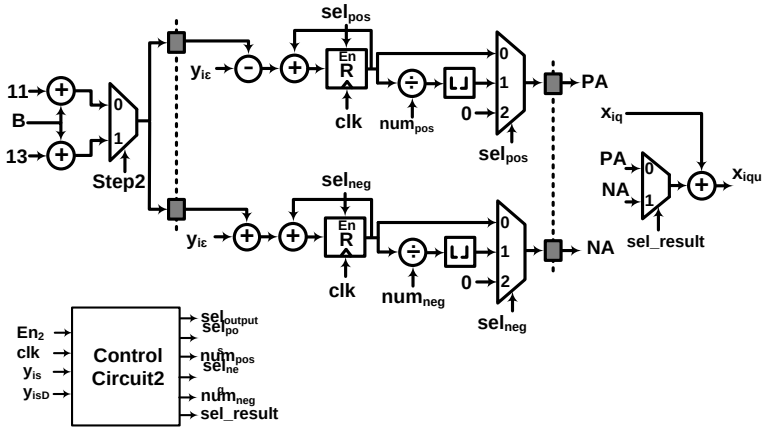
Figure 3-17. I.Differential Predictor hardware implementation.

Quantizer tuner consists of quantizer parameters update and average update blocks. Figure 3-18(a) shows the hardware implementation of quantizer parameters update block, realizing Equations 3-2 and 3-3. Control Circuit1 calculates the $Max(y_i)$ of each row. If $Max(y_i)$ is smaller than or equal to 242, $Step2$ is set to 4 and quantizer parameters are accordingly updated as Equation 3-2 through multiplexers in ure 3-18(a) controlled by $S1$ and $S0$. Similarly, if $Max(y_i)$ is larger than 242 and smaller than or equal to 482, $Step2$ is set to 8 and quantizer parameters are accordingly updated as Equation 3-3. In the case that $Max(y_i)$ is larger than 482, $Step2$ is set 8 and fL and sL are selected as 1 and 482. To avoid implementing two dividers in Equations 3-2

and 3-3, each one is converted to a multiplier and a right shift as shown in dotted box in Figure 3-18(a).



(a)



(b)

Figure 3-18. Quantizer Tuner consists of two blocks which are (a) Quantizer Parameters Update block. (b) Average Update block.

Average update block is implemented in Figure 3-18(b). Each input which is quantized into 63 or -64 is sent to it. Then the difference between input ($y_{i\varepsilon}$) and corresponding y_{iq} is calculated and averaged for both positive and negative values separately. In this block, there are two dividers in which both numerator and denominator are variable; however, they are enabled once per W cycle right after filling the FIFO. As a result, PA and NA are obtained as the average of difference values for positive and negative parts of the quantizer and applied to retrieved data to compensate the error. It is noted that the new FIFO architecture can function in two modes which are named as adaptive and non-adaptive modes. In non-adaptive mode which is mainly used for testing, Quantizer tuner is disabled and therefore the quantizer parameters are constant. While in adaptive mode all the parameters are updated as explained so far.

3.7.2 Pipelined Implementation of LP

Since all the mentioned filters are linear phase, symmetrical coefficients can be used to reduce the number of multipliers. For example, for a 5^{th}-order filter, the number of multipliers is reduced from five to three as shown in Figure 3-19. Similarly, remaining filters can be fit into an equivalent direct form implementation with two pipeline stages. We did not put more effort on optimizing the filters hardware implementation because the post-layout power simulation showed that all filter blocks contribute only less than 5% of the total power. Similarly the whole design excluding FIFOs contributes less than 7% of the area. However, when filter blocks significantly affect the power, speed, and area, efficient single/multiple constant multiplication (SCM/MCM) techniques such as canonical signed digit (CSD) or RADIX-2r recoding [72, 73] must be

applied. In addition, the fixed-point two's complement arithmetic is used to calculate the output. So, filter coefficients (t_{ij}) are quantized into M-bit integer and input data is assumed to be K-bit wide. The M-bit right shift in Figure 3-19 is to compensate the M-bit integer representation of the filter coefficients. Finally the result is rounded with respect to discarded M bits.

Having implemented FIFO blocks and new extension method, we are now able to replace them with equivalent blocks in Figure 3-5 and draw the pipelined implementation of LP as illustrated in Figure 3-20. It consists of eight pipeline stages indicated as S1 to S8. It also incorporates other pipeline stages within the filter and FIFO blocks. Figure 3-20 shows that there are two inputs required to start the LP process. The first two blocks indicated as D1 and D2 perform decimation on their inputs. They provide two subsequent data at the input of the filter blocks located in the next stage. So, data arrival is controlled by a clock ($Clk1$) that is twice faster than second clock ($Clk2$). $Clk2$ is also connected to remaining blocks of decimation part. Decimation operation in "h_y" block is implicitly applied by providing two data from neighbouring rows through FIFOs. In addition, interpolation in "g_x" block is performed by a multiplexer. It sends inputs to output one after another using the faster clock ($Clk1$). Hereafter the following blocks are all controlled by $Clk1$. As a result, LFC and two high frequency outputs are streamed out with $Clk2$ and $Clk1$, respectively. The "Control Unit" is responsible to reset the blocks at the beginning, disable/enable the New FIFO adaptive part and most importantly control the whole processing flow of the image.

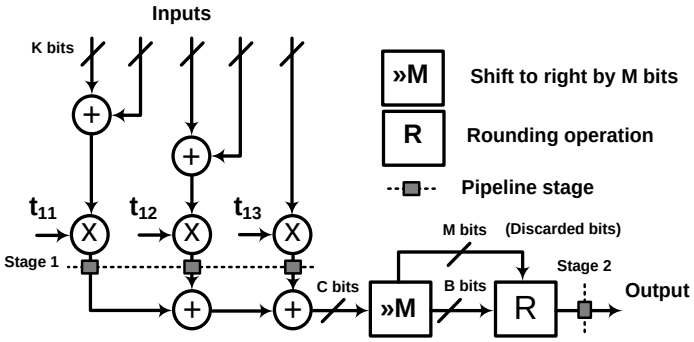


Figure 3-19. Direct form implementation of a general 5th-order linear phase filter with reduced numbers of multipliers.

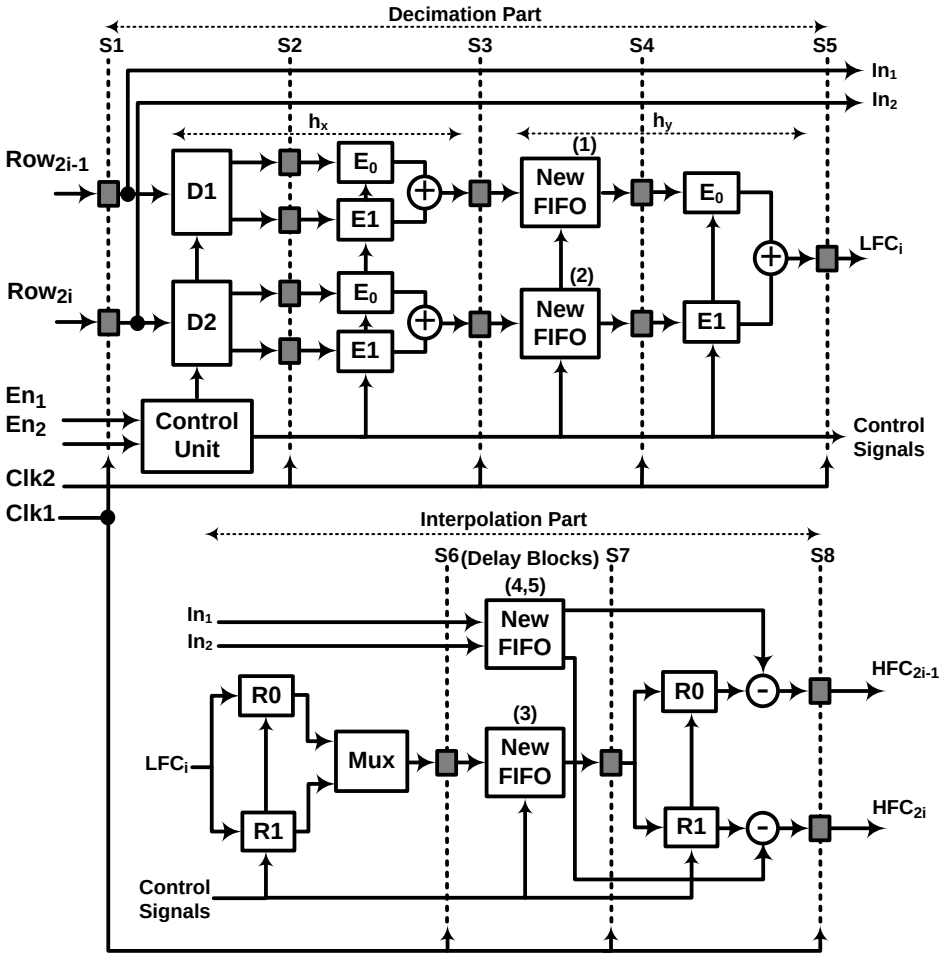


Figure 3-20. Pipeline implementation of LP. There are only eight pipeline stages shown here.

3.7.3 Near-threshold Operation Technique

Besides reducing the power consumption on architecture level by adopting proposed FAERDC technique, near-threshold operation [45] is also adopted to further reduce the power as explained in details in Section 2.5. In addition, ultra-low voltage level shifters [74] are also adopted in the design to interface between the near-threshold core and I/Os as shown in Figure 3-21.

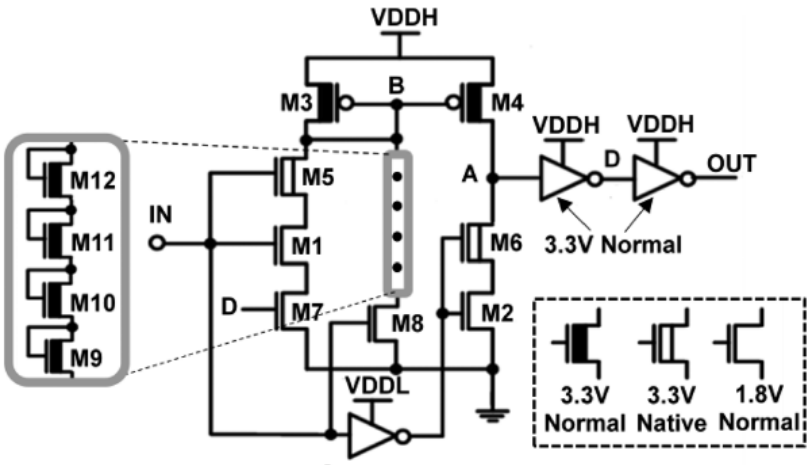


Figure 3-21. Level shifter for fast and energy-efficient wide-range voltage conversion from near/sub-threshold up to I/O voltage [74].

3.8 Experimental Results

We have fabricated the proposed LPPE at a 0.18 µm CMOS process technology. The proposed LPPE is described in VHDL and then synthesized using the Cadence RTL Compiler. The placement and routing (PnR) was done by the Cadence SOC Encounter applying the CPF (common power format) flow for the low power design. The full-customized level-shifter is exploited during synthesis and PnR to connect the core outputs to the inputs of the I/O pads operating at 1.8V. Since the main core is operating at 0.5 V, level-shifter voltage conversion is from 0.5 V to 1.8 V. Figure 3-22 shows the chip micrograph of the fabricated LP with the size of 4 x 4 mm^2 in 0.18 µm CMOS technology. Detailed specifications of chip are summarized in Table 3-2. A Xilinx ZYNQ-7000 SOC video and imaging kit with Xilinx FMC XM105 debug cards is used to test the chip. To do so, the input image is stored in embedded memory of ZYNQ device and then later two pixels are simultaneously sent to the chip and the results are read to logic analyzer, as shown in Figure 3-23. Finally the results collected by the logic analyzer are verified in MATLAB to check the functionality of the chip.

Table 3-2 Chip specifications

Technology	0.18µm
Pad/Core Voltage	3.3-1.8 V/ 0.50 V
Frames per second (fps)	112
Power (µW)	452
Frequency (MHz)	3.68
Energy/cycle (pJ/cycle)	122.83
Image size	256x256

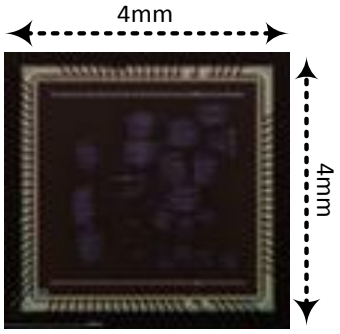


Figure 3-22. Chip micrograph.

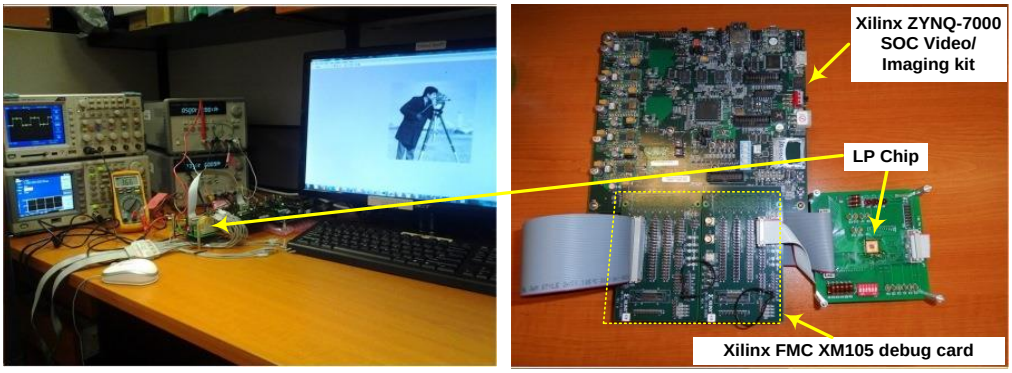


Figure 3-23. Testing setup using Xilinx ZYNQ-7000 SOC video and imaging kit with Xilinx FMC XM105 debug cards.

3.8.1 MSE Performance

All the "New FIFO" blocks in Figure 3-20 are implemented with 7-bit wide FIFO. The extensive simulation results over 100 standard test images [47], which are popular in image processing and computer vision applications evaluation, show that it outperforms the architecture in by more than 90% for LFC and more than 86% for HFC as shown in Figure 3-24. The standard deviation of MSE in Figure 3-24 reveals that the MSE just varies 0.85 and 0.43 for LFC and HFC, respectively, which is 98% better than those of others. In addition, if more area and power reduction are desired, the FIFO cell

width should be reduced more. In this case the MSE of other methods will be very large. Suppose that FIFO cells width is reduced to 6; therefore, the proposed design is performing 92% better than [24] as illustrated in Figure 3-25. The standard deviation in Figure 3-25 also exposes that the MSE variation is 96% better.

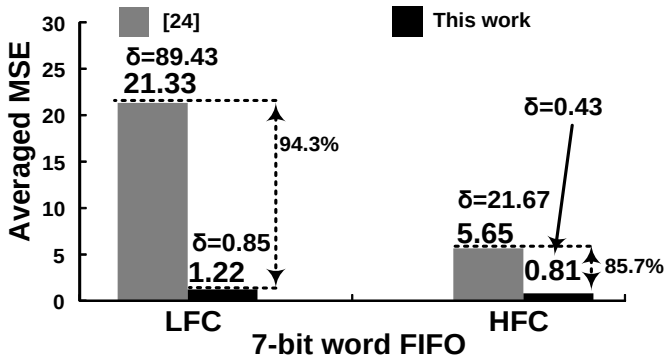


Figure 3-24. MSE performance and standard deviation of MSE of LPPE outputs applying FIFO architectures in [24] and FAERDC with 7-bit wide FIFOs over 100 standard test images.

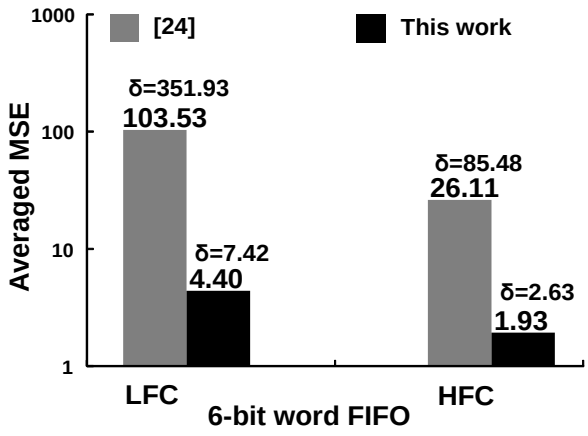


Figure 3-25. MSE performance and standard deviation of new LP outputs applying FIFO architectures in [24] and FAERDC with 6-bit wide FIFOs over 100 standard test images.

3.8.2 Power and Area

Since the proposed FIFO architecture has adaptive and non-adaptive operating modes, the chip can function in two different modes. The chip operates at 0.50V and 3.68 MHz for processing 112 images (256 x 256) per second. The resultant power for adaptive and non-adaptive modes is 452 µW and 445 µW, respectively shown in Figure 3-26(a) for "Cameraman" input image. It turns out that the Quantizer Parameters Update block responsible for adaptive operation just contributes 1.55% of the total power. However, the MSE of output in adaptive operating mode is more than 95% better than that of non-adaptive one as given in Figure 3-26(b). Power simulation result of post-layout netlist using real input images shows that the LPPE consumes 58 mW at nominal voltage of 1.8V and operating frequency of 10 MHz.

Figure 3-27(a) shows the low and high frequency outputs of LP for "Cameraman" input image. Figure 3-27(b) illustrates the difference between adaptive and non-adaptive outputs in which adaptive mode outperforms nonadaptive one on the edges which conveys vital information of the image. Since interpolation part includes more FIFO blocks compared to decimation part, 70.85% of the core area is occupied by interpolation part and the rest by decimation part. Table 3-3 summarizes the MSE and its variation for LFC and HFC over 100 standard test images as well as the sample input image (Cameraman).

Table 3-4 compares the proposed LP engine with other hardware implementations. Since the proposed design is operating at near threshold voltage, the frequency is degraded. Other designs in Table 3-3 are all operating at nominal voltage and therefore

the operating frequency is higher and suitable for high performance applications. However, all are almost suffering from redundant and zero-operand arithmetic operations. The proposed LPPE is designed for biomedical applications such as CE which requires 256×256 input image as well as IoT applications. The achieved operating frequency which is 3.68 MHz is far enough for example for real-time observation of internal digestive tracts. The frame per second (*fps*) can be calculated as below:

$$fps = \frac{2 \times f}{N \times N} \quad \text{3-6}$$

Where f is the operating frequency and N × N is the input image size. The Proposed design can support VGA input standard with the operating frequency of 3.84 MHz for $fps = 25$. The required operating frequency for HD input standards is 12 MHz (1280×720) and 26 MHz (1920×1080), respectively, for fps = 25. If the required fps is doubled, the abovementioned frequencies are also doubled. For such a high performance applications, the LPPE can be also operated at a higher voltage in order to achieve the required frequency. As a result the LPPE can support VGA and HD standards with lower operating frequency and smaller area and power consumption as compared to the existing ones.

The latency of LPPE to generate the first output can be calculated as tion 3-7 as a function of input image size and operating frequency. The constant value 4 is introduced due to the proposed extension method.

$$Latency = \frac{4 \times N}{f} \quad second \quad \text{3-7}$$

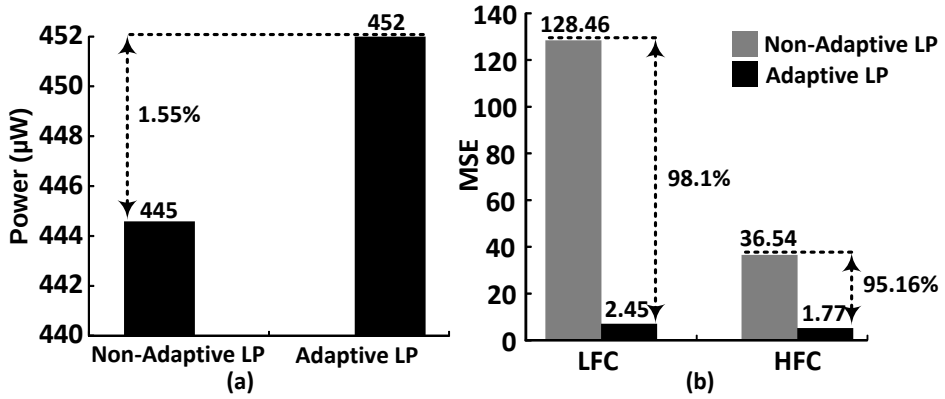


Figure 3-26. (a) Power measurement result for LP operating at 0.5 V with both non-adaptive and adaptive modes. (b) MSE comparison of the "Cameraman" input image in non-adaptive and adaptive modes.

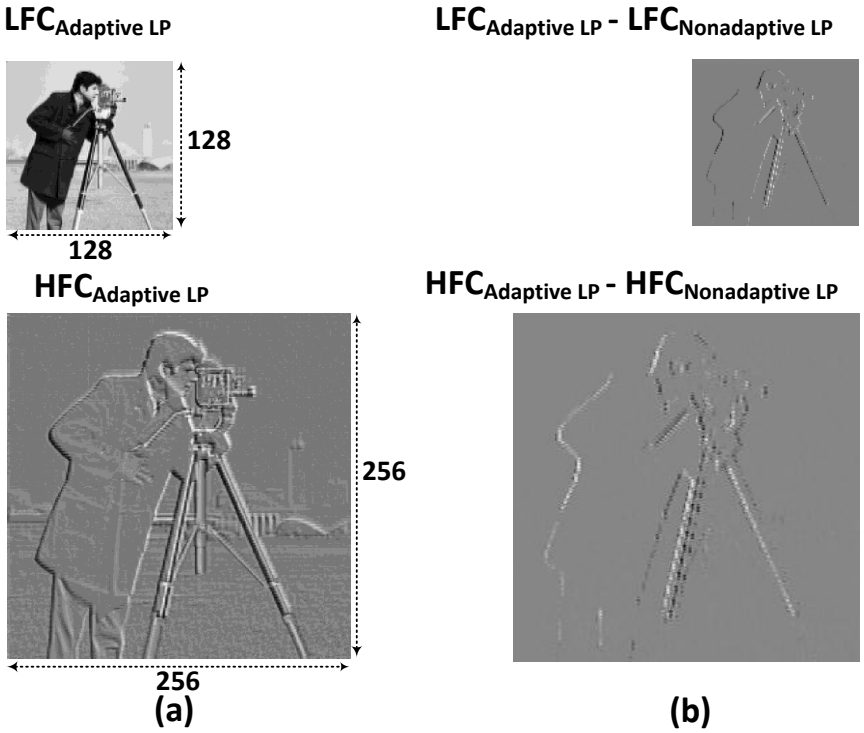


Figure 3-27. (a) The LPPE output for Cameraman Image. (b) The difference between the outputs of adaptive and nonadaptive modes.

Table 3-3 MSE for LFC and HFC over 100 test images

		[20]		[24]		This work	
		LFC	HFC	LFC	HFC	LFC	HFC
7-bit	Averaged MSE	NA	NA	21.33	5.65	1.22	0.81
	Variation	NA	NA	89.43	0.85	21.67	0.43
6-bit	Averaged MSE	NA	NA	103.53	26.11	4.40	1.93
	Variation	NA	NA	351.93	85.48	7.42	2.63
MSE (Cameraman Image)		5960	2339	128.46	36.54	2.45	1.77

NA: Not applicable

Table 3-4 Comparison with other works

	[20]	[57]	[62]	[63]	[64]	[68]	[69]	**This work**
Freq. (MHz)	200	420	20	25	31	108	83	**3.68**
fps	1525	94	55	35	101	400	270	**112**
Image Size	512× 512	HD	512× 512	-	VGA	VGA	VGA	**256× 256**
App.	A1	A6	A1	A2	A3	A4	A5	**A1**
ASIC (A) FPGA (F)	F	F	A	F	F	A	F	**A**
Latency (ms)	~0	-	-	-	~50	~1	-	**0.28**
Energy(pJ) /cycle/pixel	-	-	-	-	-	-	-	**0.002**

A1: GP and LP, A2: Target tracking and Image stabilization, A3: Image fusion
A4: Vision processor, A5: Motion and depth estimation, A6:Image blending

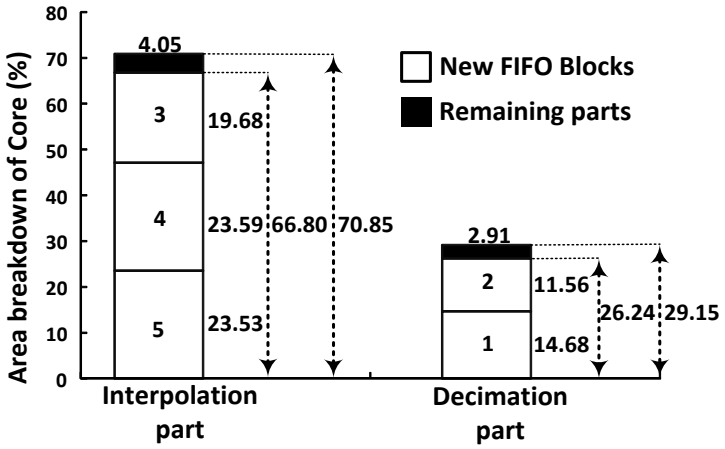
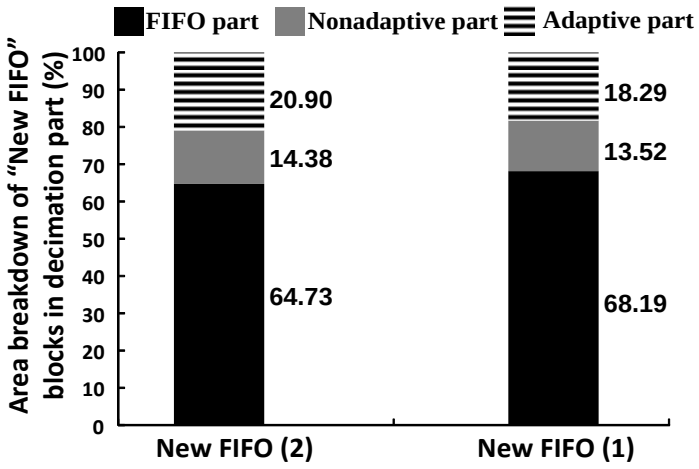
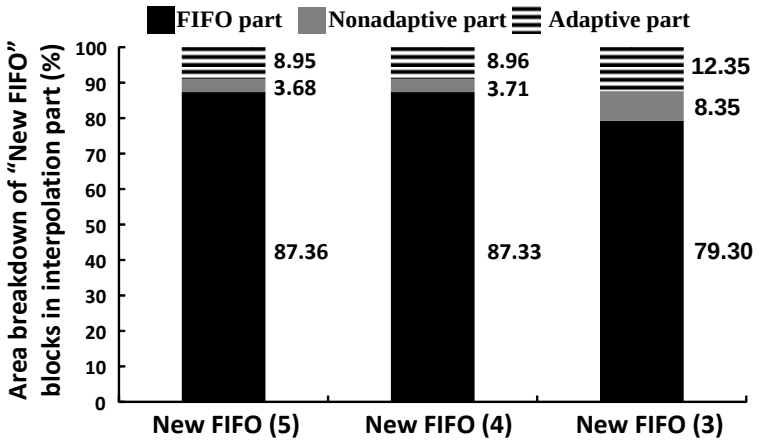


Figure 3-28. Area breakdown of LP core. The number written in the boxes are referred to the number assigned to "New FIFO" blocks in Figure 3-20.

Figure 3-28 shows that New FIFO blocks in interpolation and decimation parts (Figure 3-5) occupies 66.80% and 26.24% of the core area, respectively, which is 93.31% of the whole area. The numbers shown in Figure 3-28 refer to the New FIFO blocks indexed in Figure 3-20. More than 66% of the area of New FIFO blocks in decimation part is occupied by the FIFO parts as shown in Figure 3-29(a) and the rest is used by nonadaptive and adaptive parts. Figure 3-29(b) also shows that 85% of the area of New FIFO blocks in interpolation part is occupied by FIFO parts and remaining area is used by nonadaptive and adaptive parts. The area breakdown of non-adaptive and adaptive parts in New FIFO block named 3 is shown in Figure 3-30. In non-adaptive part I.Q. dominates the area because each FIFO core requires a separate one. In adaptive part, the area is almost divided between average update and quantizer parameters update blocks; however, further reduction in average update block can be achieved by replacing two dividers with the simpler design. As a result, the total area improvement estimated for 512×512 and 1024×1024 input image sizes are 28.79% and 36.06% respectively as shown in Figure 3-31.



(a)



(b)

Figure 3-29. Area breakdown of New FIFO blocks in (a) decimation, (b) interpolation parts. The numbers used to name the New FIFO blocks referred to the numbers in Figure 3-20.

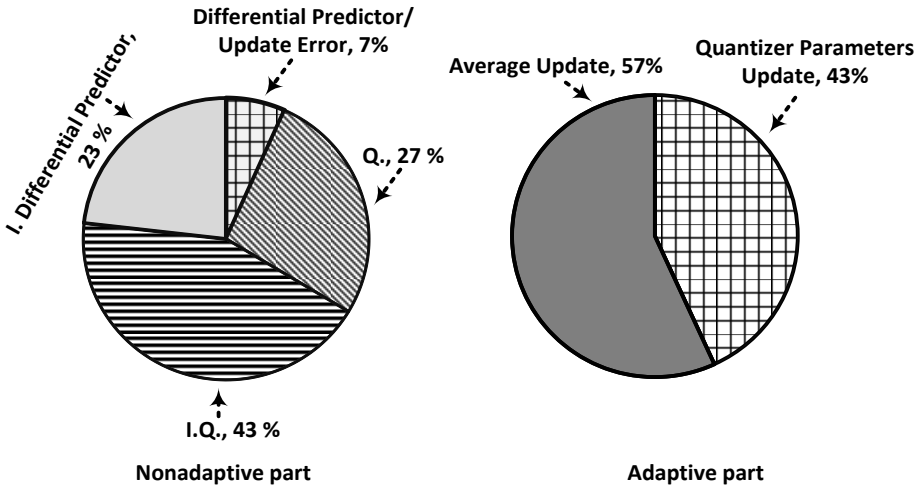


Figure 3-30. Area breakdown of non-adaptive and adaptive parts of New FIFO block numbered as 3 in Figure 3-29.

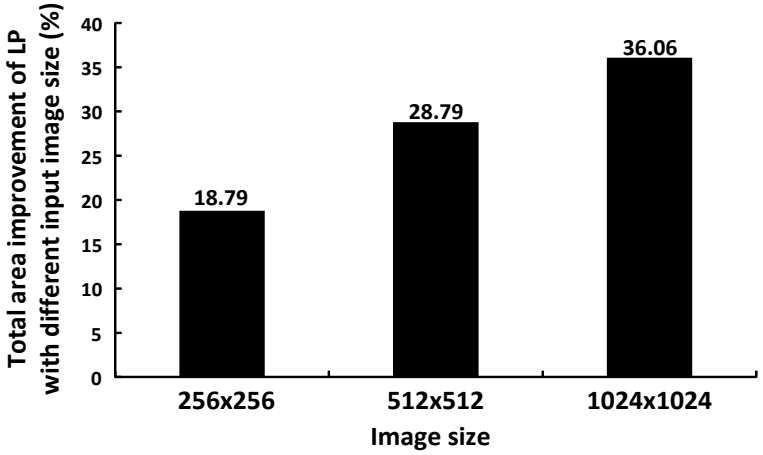


Figure 3-31. Total area improvement vs. different input image size.

3.9 Computational Complexity Analysis

The required number of arithmetic operations throughout the whole process can be generally calculated as given in Table 3-5 for an $W \times W$ input image, $1 \times K$ decimation filter and $1 \times P$ interpolation filter. It shows that the proposed LP architecture reduces the arithmetic operations by half in decimation part. Similarly, the zero operand arithmetic operations is reduced by $P \times 100\%/(2P-1)$ in the proposed LP architecture and therefore there is 54% reduction for the 7^{th}-order filter ($P = 7$). Moreover, since the proposed method is processing the two rows of the image simultaneously, the whole process requires some $W^2/2$ clock pulses to finish the analysis of an $W \times W$ image.

Table 3-5 Comparison of Number of addition and Multiplication in decimation and interpolation parts (Excluding the operation in FIFO)

	Operation	Conventional Methods	Proposed Method
Decimation	Addition	$1.5 \times W^2(K-1)$	$0.75 \times W^2(K-1)$
	Multiplication	$1.5 \times W^2K$	$0.75 \times W^2K$
Interpolation	Addition	$1.5 \times W^2(P-1)$	$0.75 \times W^2(P-2)$
	Multiplication	$1.5 \times W^2P$	$0.75 \times W^2P$
Processing time (Number of clock cycle)		$\cong W^2$	$\cong \frac{W^2}{2}$

3.10 Conclusion

In this chapter, we have presented a power and area efficient Laplacian Pyramid processing engine (LPPE) for image/video processing applications. LPPE applies architecture-level and circuit-level techniques to reduce the power and area and improve the MSE. On architecture-level, FEARDC and new extension method are proposed to compress the FIFO size, reduce the power and area and improve the MSE. On circuit level, near-threshold design is adopted to further reduce the power consumption. As a result, the area is reduced by 18.98%~36.06%. Measurement results show that the proposed LPPE achieves the operating frequency of 3.68 MHz at 0.50 V for processing 112 *fps* while consuming only 452 μW.

Chapter 4. Structuring of Contourlet Transform for Pipeline-based Implementation

4.1 Introduction

Digital signal processing has been developed with the aids of various transform algorithms such as discrete cosine transform (DCT) [75], discrete Fourier transform (DFT) [76], Wavelet transform [77], etc. Basic transforms such as DCT and DFT provide coefficients in the frequency domain regardless of the space or time occurred. The Wavelet transform has been introduced to obtain signal properties in the specified time or space coordinated. 2D wavelet transform detects edge points efficiently, but cannot detect the smoothness of contours. To tackle the above limitations, various new schemes such as curvelet [78] and contourlet [17, 79] have been proposed, trying to overcome the limitations of the standard wavelets.

Contourlet transform (CT) is a powerful and contemporary tool in extracting vital information from an image [17, 79]. It is mainly composed of two parts named LP and directional filter bank (DFB). CT is widely used in many image/video processing applications such as image quality assessment, watermarking, denoising, compression, etc [80-85]. Hardware implementation of transform algorithms is highly required due to numerous real time applications for signal and image processing. Even though CT has been increasingly used for image and video processing applications, hardware architecture of CT has not completely been reported. However, various LP algorithms have been

proposed as a part of systems [56, 62, 86, 87] and fully reviewed in Chapter 2. In [88], a video denoising algorithm applying CT was implemented on FPGA. It uses the conventional LP architecture and only performs QPDEC decomposition in DFB part; however, the details of the design are not included.

In this chapter, a structure for CT, composed of the LP and DFB that is suitable for efficient hardware implementation is proposed. A new architecture of the LP proposed in Chapter 2 uses the poly-phase representation and the noble identities to reduce the number of mathematical operations. We also present hardware architecture of the directional filter bank utilizing 2-D filters and several arithmetic structures with the reduced number of multipliers. Moreover, for the CT to extract data from different directions of an input image, a memory-based structure is designed to perform the rotation operation. Quantization analysis is utilized to determine the width of memory for the registers and FIFO cells in different pipeline stages. The final CT architecture is obtained by concatenating the proposed LP and DFB and then is verified by FPGA implementation.

The organization of this chapter is as follows. In Section 4.2, a brief and intuitive principle of CT is presented. Section 4.3 explains the proposed hardware-oriented structure of CT, followed by its hardware architecture given in Section 4.4. Performance evaluation, including quantization and complexity analyses of CT and FPGA implementation is presented in Section 4.5. Finally, the conclusions are drawn in Section. 4.6.

4.2 Review of Contourlet Transform Principle

CT consists of two parts, LP and DFB (Figure 4-1(a)). LP comprises a decimation part and an interpolation part. These two parts produce low frequency (LF) and high frequency coefficients from the original input image [53]. DFB analyzes the high frequency coefficients, generated by LP, to show directionality in the frequency domain. CT can be constructed hierarchically by applying low frequency coefficients from one LP (e.g. n_1 in Figure 4-1(b)) to the input of another LP (e.g. n_2 in Figure 4-1(b)) as illustrated in Figure 4-1(b). The directional filter bank decomposition at each pyramidal level is defined by a vector, $nLevs = [n_t \dots n_2\, n_1]$. Each scale divides its high frequency coefficients (*HF*) into 2^{n_i} wedge-shaped sub-bands where $i = 1, 2, \dots, t$. For an $N \times N$ input image, the size of HF_i sub-bands and LF is $N \times N/(4^{(i-1)}2^{n_i})$ and $N \times N/4^i$, respectively. Figure 4-1(c) illustrates the 2D frequency plane decomposition of $nLevs = [2, 3]$. It shows two sets of directional sub-bands (A and B) that are 8 and 4 wedge-shaped directional sub-bands. Figure 4-1(d) shows the CT transform of Barbara image for $nLevs = [2, 3]$. Note that the number of the output coefficients of CT transform is larger than that of input image pixels by about 25%.

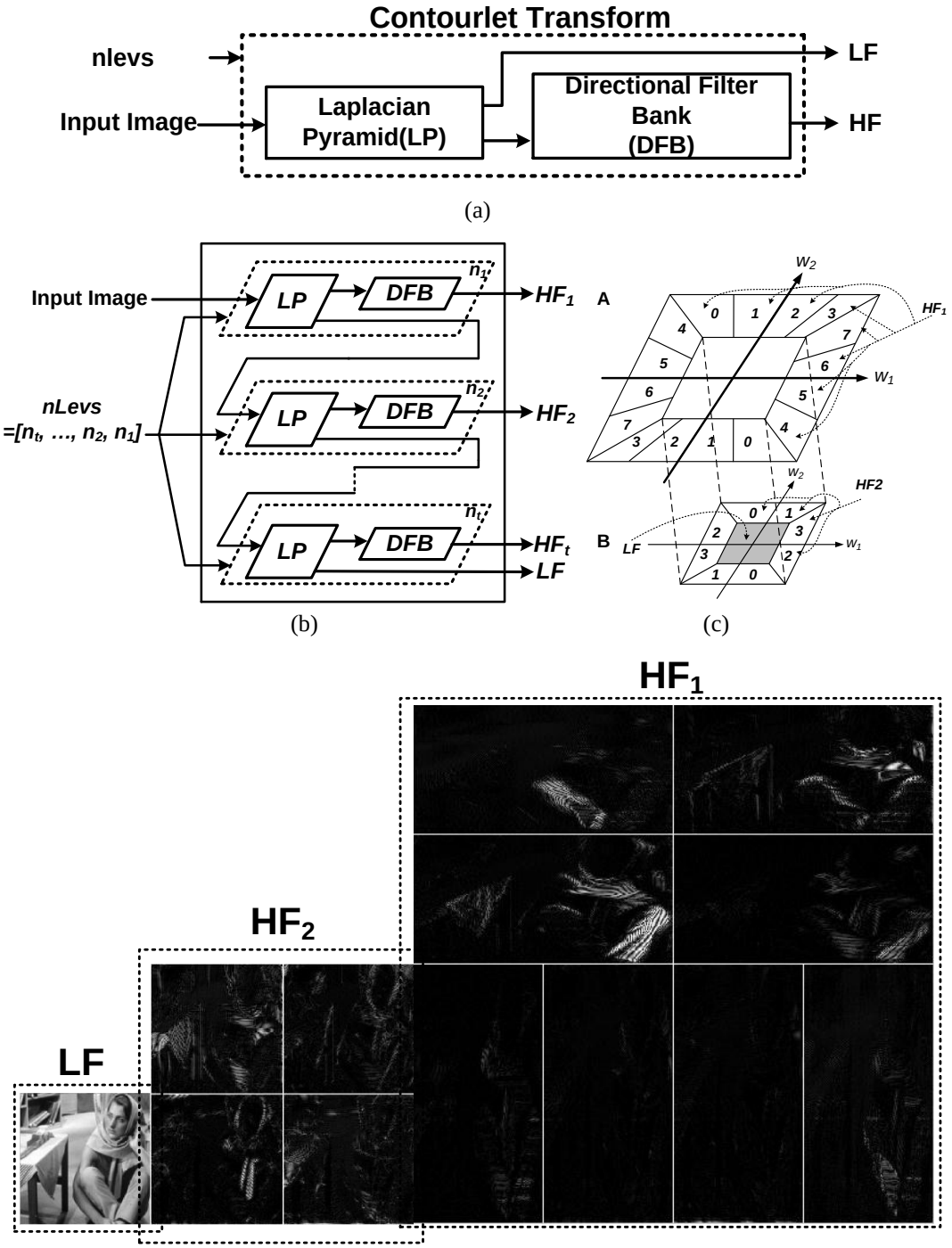


Figure 4-1. (a) High level block diagram of CT, (b) hierarchical block diagram of CT, (c) 2D frequency plane decomposition for $nLevs = [2, 3]$, and (d) "Barbara" image for $nLevs = [2, 3]$.

The general structure of the LP as explained in Chapter 2 is illustrated in ure 4-2 [53, 79]. C_0, C_1 and D_1 represent input image, low and high frequency coefficients, respectively. H and G are filters used in the decimation and interpolation parts. LP is a hierarchical algorithm where the input image, represented by C_0, determines the zero level of LP. The first pyramidal output (C_1) is a low-pass filtered version of C_0. It then goes to interpolation part to produce D_1.

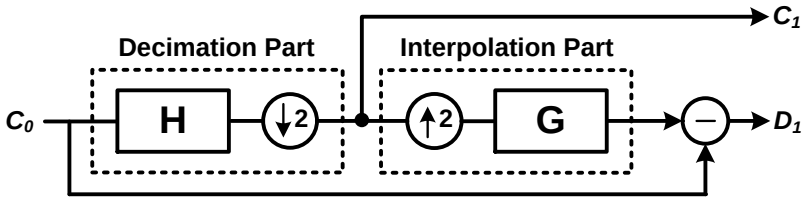


Figure 4-2. General structure of Laplacian Pyramid (LP).

Filter banks (FB) are powerful tools for decomposing and analyzing discrete signals in digital signal processing [70, 89]. Advancements in new 2D multi-scale representation have revealed that directionality is a crucial feature for an efficient image representation [79]. In [90], non-separable FB is explored for constructing a 2D directional filter bank. This 2D DFB involves input signal modulation using diamond-shaped filters. (See [91] for details).

To simplify the analysis of the iterated DFB, a new formulation has been proposed based only on Quincunx Filter Bank (QFB) with fan filters in [17, 79]. This eliminates the modulation of the input image and has a simpler rule for expansion. In summary, DFB partitions frequency planes into wedge-shaped parts by combining QFBs

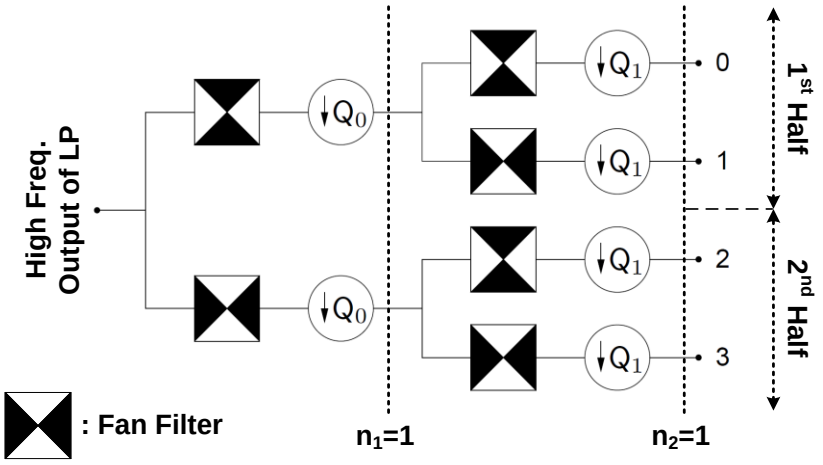
with “rotation” operations through resampling. To avoid higher mathematical complexity of DFB, a descriptive explanation of DFB decomposition using QFB and Resampled Quincunx Filter Bank (RQFB) is given (Figure 4-3). In this decomposition, QFB is employed for $n_i = 1, 2$ while QFB and RQFB are both utilized for $n_i > 2$. Figure 4-4(a) shows the DFB decomposition of the first two levels using QFB.

Figure 4-3. Block diagram of DFB

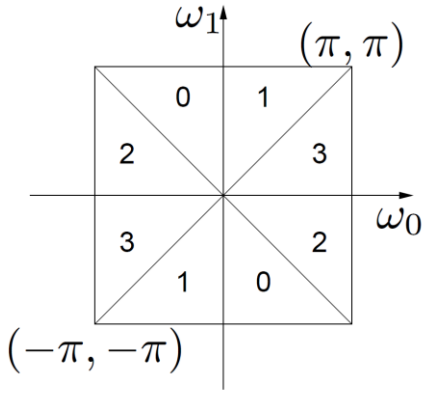
Matrices Q_0 and Q_1 followed by fan filters represent the two-dimensional quincunx sub-lattice [92]. Using this decomposition method, we could divide the frequency plane into four parts as depicted in Figure 4-4(b). The four outputs of Figure 4-4(a) (i.e. 0, 1, 2, and 3) correspond to the four distinct regions in Figure 4-4(b). For $n_i > 2$, frequency planes are partitioned into finer regions by RQFBs. Four types of RQFB are established by resampling matrices (R_0, R_1, R_2 and R_3), which are used for each RQFB type [17]. Figure 4-4(c)-(d) illustrates Type0 and Type1 of RQFBs including R_0, R_1, and fan filters (for Type3 and Type4, refer to [79]).

Finally, the above RQFBs are combined with Figure 4-4(a) to obtain the finer wedge-shaped parts in Figure 4-4(a). RQFBs of Type0 and Type1 are used in the first half whose outputs are indexed as ‘0’ and ‘1’ while RQFBs of Type2 and Type3 are

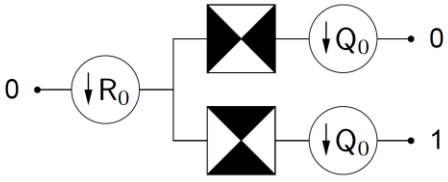
adopted in the second half whose outputs are indexed as ‘2’ and ‘3’. This procedure continues iteratively for the outputs of Figure 4-4(c)-(d), that is, RQFBs of type0 and type1 are applied again based on the index of outputs mentioned. This trend is called expansion rule and applied until the proper number of fine partitions is acquired.

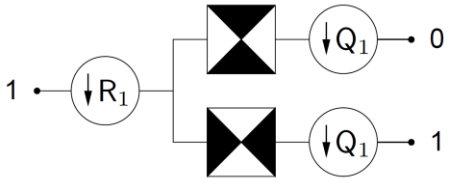


(a)



(b)





(c) (d)

Figure 4-4. (a) The first two levels of DFB, each level is a QFB, (b) directional quadruple division of the frequency plane, (c) RQFB Type0, and (d) RQFB Type1 [79].

4.3 Proposed Decomposition of CT Structure for Pipeline Suitability

Although CT has been gaining more and more popularity in digital image/video processing, comprehensive research works on hardware architecture have rarely been conducted. In this section, the structures of LP and DFB that are suitable for the intended pipeline implementation of CT are proposed. Since LP structure has already been discussed in detail in Chapter 4, we just recap it here very briefly to make the whole structure understandable. Later we will explain DFB in more details.

4.3.1 Proposed LP Structure

The proposed LP structure is illustrated in Figure 4-5. It uses one dimensional (1D) filters (h_x and h_y , g_x and g_y). By applying poly-phase representation $H(z)$ and $G(z)$ (z-transform of g and h in Figure 4-2) can be represented by $E_l(z)$ and $R_l(z)$ named as polyphase components. Then by using nobble identities, $E_l(z)$ and $R_l(z)$ can be reversed by resampling [70]. M and L are the decimation and interpolation values [70]. So the proposed LP structure processes two rows of an image simultaneously. Therefore, two high frequency coefficients (HFC_{2i-1} and HFC_{2i}) are calculated concurrently in addition to the low frequency coefficient (LFC_i). h_x calculates the horizontal convolution while h_y calculates the vertical convolution through which even-indexed and odd-indexed results coming from h_x are processed by E_0 and $E_{1,}$ respectively. The outputs of E_0 and E_1 are added to produce the LFC_i. LFC_i is processed by R_0 and R_1, and their results are interleaved to produce the results of a horizontal convolution. The output

of the horizontal convolution is applied to R_0 and R_1, which generates the high frequency coefficients. For synchronization, delayed input signals, which are implemented by FIFOs, are used in the high frequency coefficient calculation.

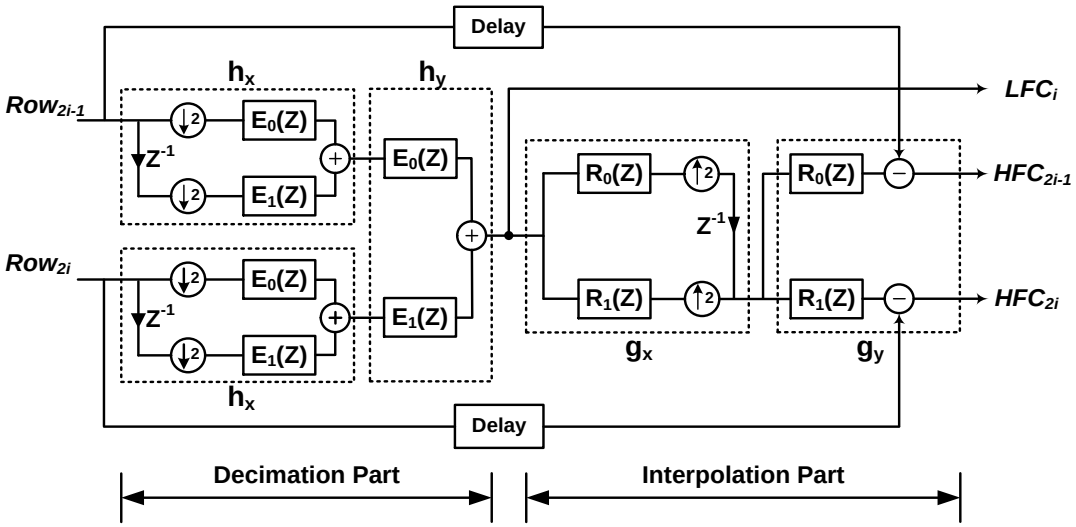


Figure 4-5. Proposed LP architecture.

4.3.2 Proposed Directional Filter Bank (DFB) Structure

Since DFB has high mathematical complexity, as was explained in Section 4.2, it is necessary to figure out a structure which facilitates achieving the hardware architecture. To do so, we employ DFB decomposition using QFB and RQFB whose structures are illustrated in Figure 4-6 and elaborated in [17, 79]. RQFB consists of Parallelogram Poly-phase Decomposition (PPDEC) and ladder structure. Similarly, QFB comprises Quincunx Poly-phase Decomposition (QPDEC) and ladder structure. In order to reach the hardware-oriented structure, new rotation symbols are introduced in Figure 4-7 to explain QPDEC and PPDEC block diagrams (Figure 4-6). Each rotation symbol indicates the rotation direction of an input image. Figure 4-8 shows sample images after rotation. For example, by applying the rotation of Figure 4-7(d) to an input image (Figure 4-8(a)), firstly all of the rows of the image are shifted to right by $i-1$ (Figure 4-8(b)) where i is the row index starting from '1'. However, the vertically extended image has more columns than the input image. Secondly, the empty columns are filled with the pixels in the additional columns, generating the image shown in Figure 4-8(c). Compared to Figure 4-7(d), Figure 4-7(h) rotates all the rows in an image by i where i is the row index. Therefore, the difference between Figure 4-8(c) and Figure 4-8(d) cannot be visually recognized. Similarly, Figure 4-7(b) rotates the columns of the image downward by $j-1$, where j is the column index. Figure 4-8(e) illustrates the resultant image.

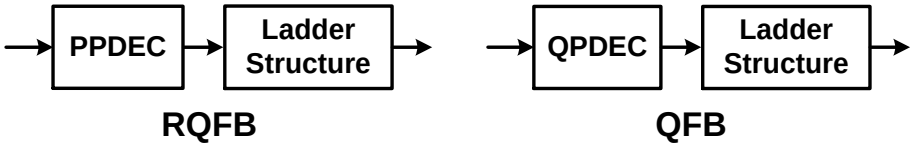


Figure 4-6. Block diagrams of RQFB and QFB.

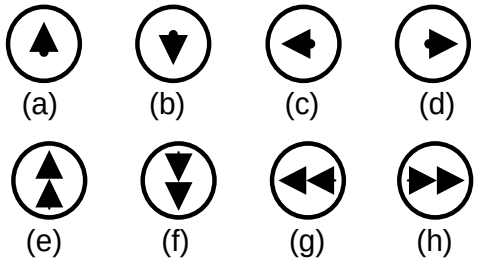


Figure 4-7. Eight distinct rotation symbols.

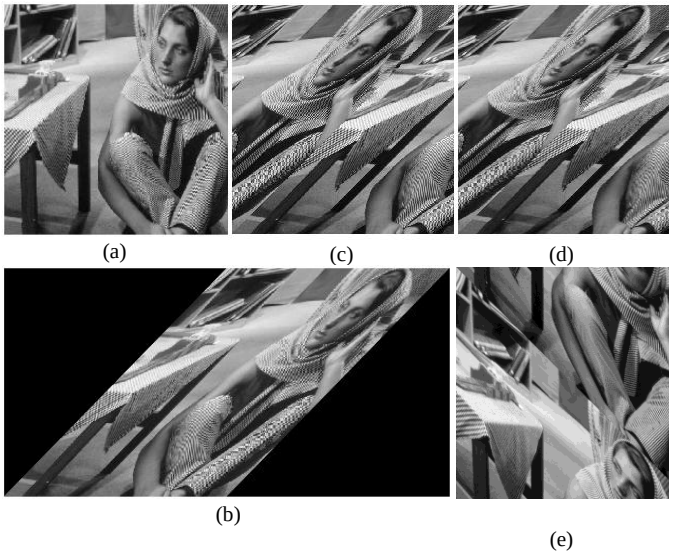


Figure 4-8. (a) Input image, (b) Image after shifting rows to right by $i - 1$, (c) Output image after applying Figure 4-7(d), (d) Output image after applying Figure 4-7(h), and (e) Output image after applying Figure 4-7(b).

By using the above symbols, the equivalent block diagrams for QPDEC and PPDEC can be obtained as illustrated in Figure 4-9 and Figure 4-10. The term , z_R^{-1}, denotes down-sampling applied on the input rows; in other words, odd and even rows are

sent to the upper branch and the lower branch, respectively. Similarly, z_C^{-1} represents down-sampling for the input columns.

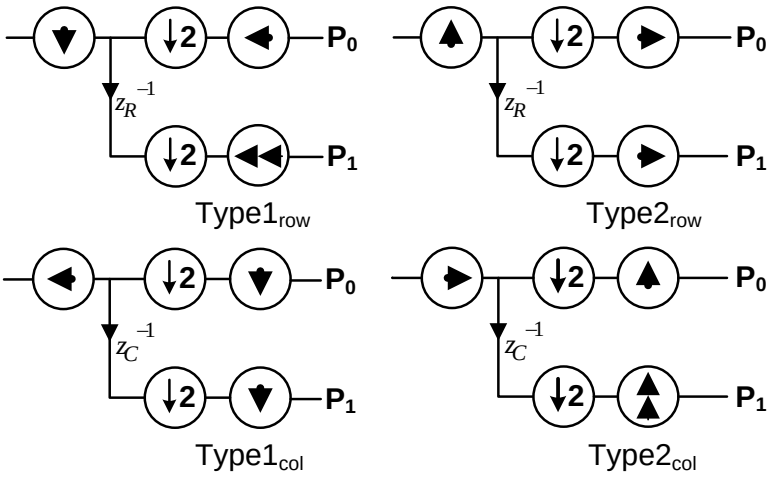


Figure 4-9. Equivalent block diagrams for QPDECs named $Type1_{row}$, $Type2_{row}$, $Type1_{col}$ and $Type2_{col}$.

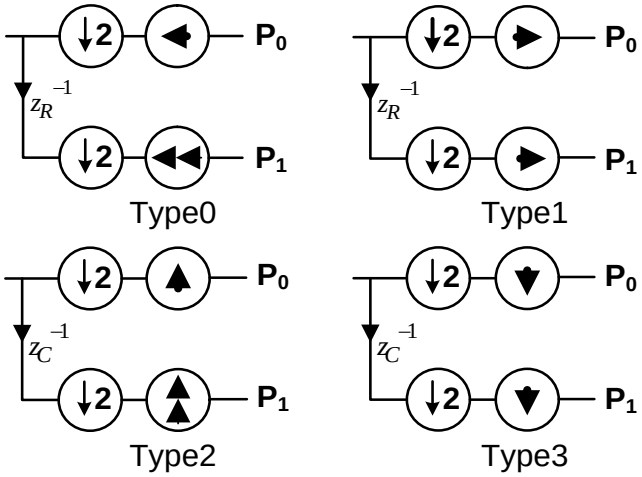


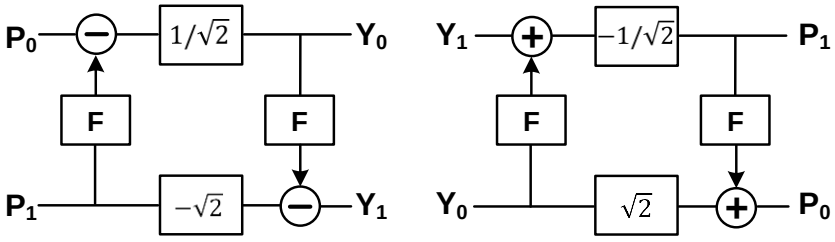
Figure 4-10. Equivalent block diagrams for PPDECs named Type0, Type1, Type2 and Type3.

In Figure 4-9, '$Type1_{row}$' and '$Type2_{row}$' represent the quincunx matrices for rows while '$Type1_{col}$' and '$Type2_{col}$' indicate the quincunx matrices for columns. In Fig-

ure 4-10, ‘Type0’, ‘Type1’, ‘Type2’ and ‘Type3’ correspond to the matrices, R_0 to R_3, respectively. For example, if the input image in Figure 4-8(a) is applied to ‘Type1_{row}’ (QPDEC), firstly the input image will be rotated downward by the symbol of ure 4-7(b) generating the image in Figure 4-8(e). After that, the odd row part is rotated by the symbol of Figure 4-7(c) and the even row part is rotated by Figure 4-7(g). The outputs of ‘Type1_{row}’ (P_0 and P_1) are shown in Figure 4-11(a)-(b). The operations of PPDECs described in Figure 4-10 are similar since they are a part of QPDECs in Figure 4-9. The final step for realizing the QFB and RQFB block diagrams is filter implementation. A ladder structure based on 2D separable filters named F [17, 79] is employed to estimate the directions (Figure 4-12(a)). Figure 4-12(b) is a reconstructing ladder structure that will be used in the CT reconstruction flow.

(a) (b)

Figure 4-11. Outputs of QPDEC Type1$_{row}$ (a) P_0 and (b) P_1



(a) (b)

Figure 4-12. Ladder structures: (a) Ladder structure in CT construction and (b) Ladder structure in CT reconstruction.

4.3.3 Proposed CT Block Diagram

By combining the proposed hardware-oriented structures of LP and DFB, the proposed CT block diagram with $nLevs = [1]$ is depicted in Figure 4-13, where $nLevs$ represents the number of pyramidal levels as well as the number of wedge-shaped directional sub-bands. As explained in Section 4.2, QFB is utilized for DFB at $n_i = 1$ or 2. An $N \times N$ image is analyzed by LP generating low frequency and high frequency outputs.

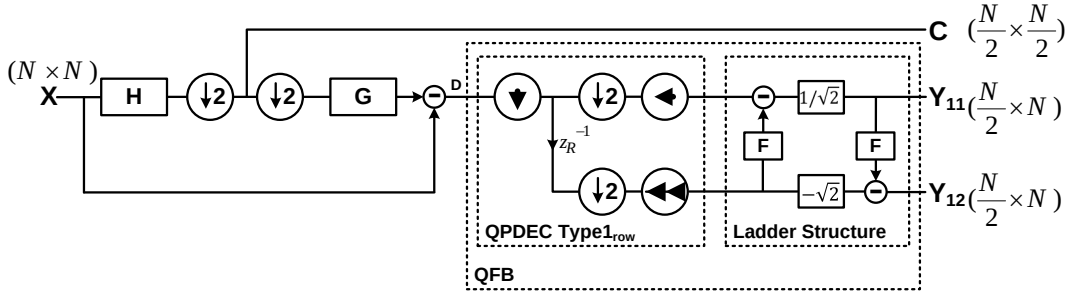


Figure 4-13. CT block diagram with $nLevs = [1]$. (The numbers written in parentheses are the size of input and outputs)

The high frequency output is further processed by QFB implemented by $\text{Type1}_{\text{row}}$ (QPDEC) and the ladder structure as explained in the previous section. In Figure 4-13, LP produces low frequency output (C) with the size of $(N/2) \times (N/2)$ and two high frequency outputs (Y_{11} for odd rows and Y_{12} for even rows) with the size of $(N/2) \times N$. Similarly, the CT Block diagram for $nLevs = [3]$ is presented in Figure 4-14 where QPDEC 'Type2$_{\text{col}}$' accompanied by a ladder structure constitutes the second level QFB. The high frequency outputs for $nLevs = [2]$ divides the frequency plane into four parts (Y_{21}, Y_{22}, Y_{23}, and Y_{24}), directionally, which is explained in Figure 4-4(b). In the third level RQFB is implemented by PPDECs and ladder structures. The four outputs of the second level QFB are further divided into eight parts (i.e. Y_{3j} where $j = 0, 1, \dots, 8$).

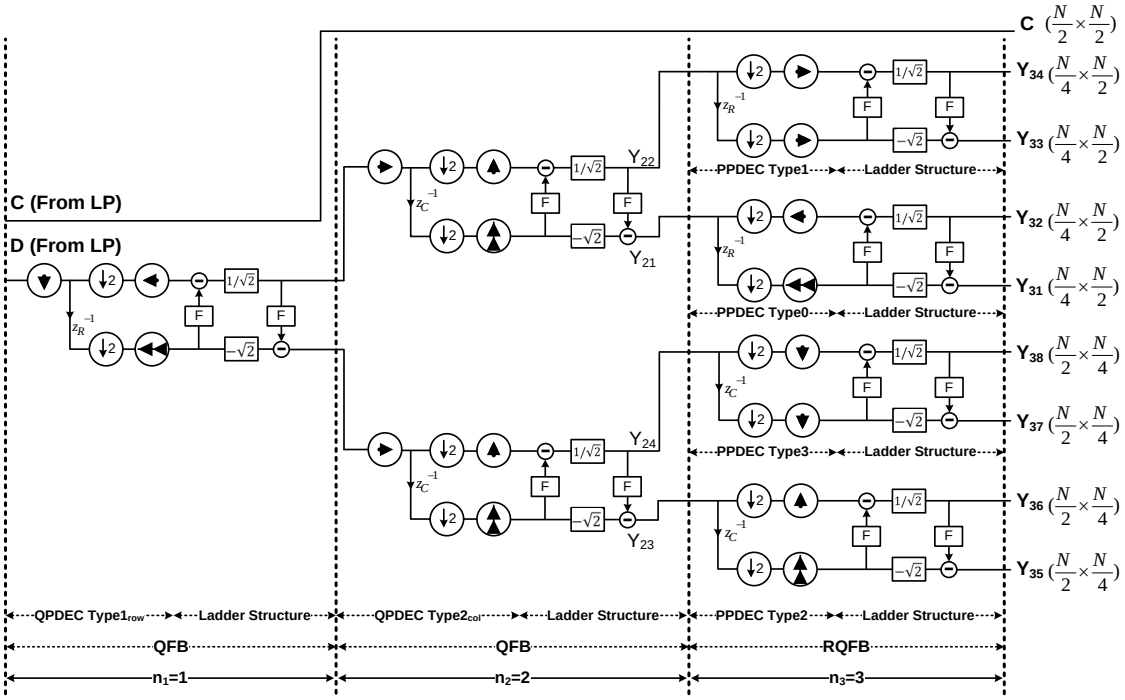


Figure 4-14. CT block diagram with $nLevs = [3]$. (The numbers written in parentheses are the size of outputs).

To realize efficient hardware architecture of CT, its reconstruction block is also critical. The CT reconstruction structure is achieved by reversing the rotation directions and using the corresponding ladder structure for filtering (Figure 4-12(b)). The CT reconstruction block diagram for $nLevs = [1]$ is shown in Figure 4-15. The reconstruction will be lossless if the output of CT is not modified [79].

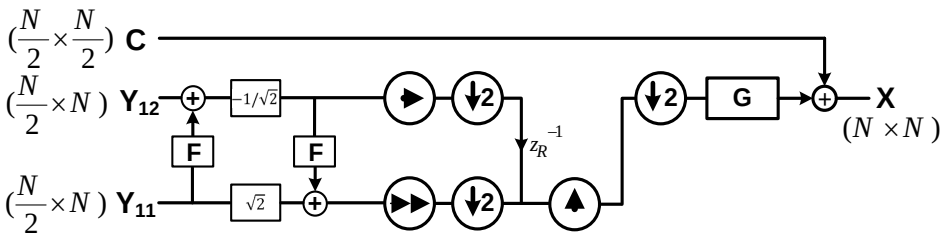


Figure 4-15. CT reconstruction block diagram with $nLevs = [1]$. (The numbers written in parentheses are the size of inputs and outputs)

4.4 Proposed Hardware Architecture for Contourlet Transform

4.4.1 LP Architecture

For achieving high performance and cost-effective design, the LP architectures proposed in Chapter 3 can be realized as a pipeline-based hardware redrawn in Figure 4-16. In the first stage, two neighbouring pixels of Row_{2i-1} are saved in block 1 and then sent simultaneously to the second stage of the pipeline. Same thing happens for adjacent row (Row_{2i}). This approach requires two separate clocks where the pixels arrival is controlled by a $Clk1$ and sending them out to the second stage is done by $Clk2$ with half the frequency of $Clk1$. Horizontal convolution is applied in the second stage of pipeline and essential number of data ($N \times 5/2$ and $N \times 4/2$ pixels for two FIFOs in stage 3) is stored in FIFO located in stage 3 of pipeline. Vertical convolution is performed in stage 4 and low frequency coefficients are calculated. All these stages are making a decimation part of LP. In stage 5, interpolation part starts horizontal convolution followed by interleaving operation. Then sufficient number of data is stored in FIFO in stage 6. Finally in stage 7 the vertical convolution is performed and high frequency

coefficients of two neighboring rows are obtained [20]. The FIFO-related design issues have been discussed in Chapter 3

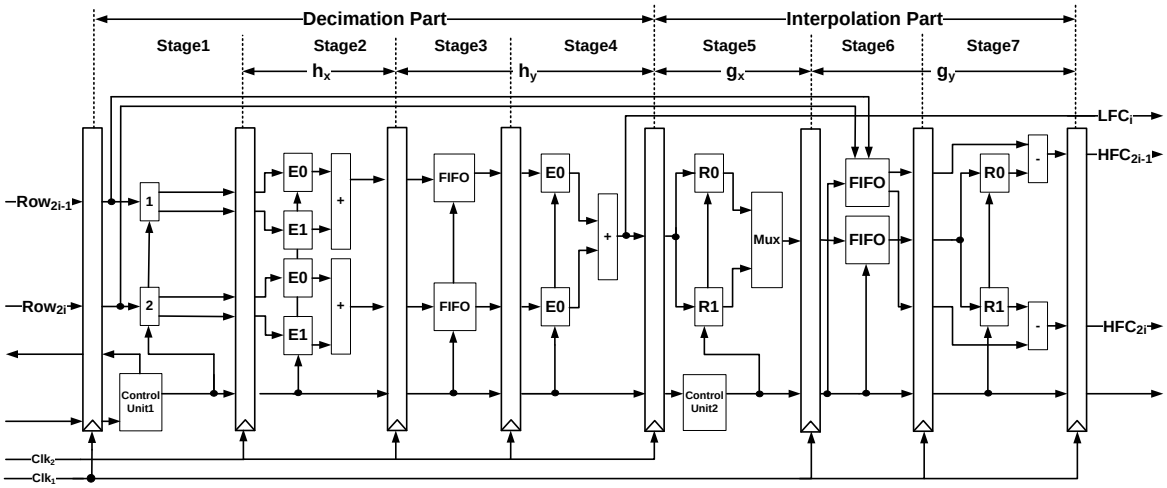


Figure 4-16. Hardware Implementation of the LP proposed architecture.

4.4.2 QFB and RQFB Architecture

In this section, we explain the hardware architecture of QFB and RQFB utilizing QPDEC and PPDEC (as shown in Figure 4-9 and Figure 4-10). For instance, Figure 4-18 shows the data processing of QPDEC $\text{Type1}_{\text{row}}$ (Figure 4-17) when an 8×8 matrix (Matrix A) is applied. Matrix A is assumed as LP's high frequency output shown by $a_i, b_i, c_i, d_i, e_i, f_i, g_i$ and h_i where $i = 1, 2, \dots, 7$. Figure 4-18(b)-(d) are the output matrixes at three rotations ($B, C,$ and D). For example, the gray pixels in the first row in Matrix A are rotated downward (Figure 4-7(b)), generating Matrix B. After that, the decomposed odd and even rows are rotated leftward (Figure 4-7(c) and (g)), which produces Matrix C and Matrix D (Figure 4-18). Because the data arrangement in Matrix C and D is non-causal, a memory has to be employed to save the number of pixels for imple-

menting the required filter operation with the different memory access patterns and a ladder structure.

Figure 4-19(a) shows the memory content corresponding to QPDEC Type1$_{row}$. Figure 4-19(b) reveals the data flow in the memory where the locations with black squares indicate the firstly arrived pixels of each column. The locations with black circles indicate the final pixels of each column. The zigzag lines show the memory scanning direction. This method is also used for other types of QPDEC and PPDEC in realizing their memory address generators. It is worthy of mentioning that QPDEC Type1$_{row}$ address generator can be used as QPDEC Type2$_{col}$ (Figure 4-9) address generator if the coordinate *x* and coordinate *y* are reversed and the input is horizontally swept. Therefore, the implementation of QPDEC and PPDEC needs a memory as well as memory address generator producing the suitable addresses required. Since QPDEC and PPDEC have two outputs, a dual-input and dual-output memory is used in this work. Note that the proposed LP architecture also produces two outputs as shown in Figure 4-5.

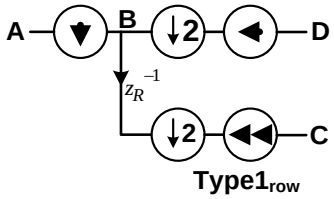


Figure 4-17. QPDEC Type1$_{row}$.

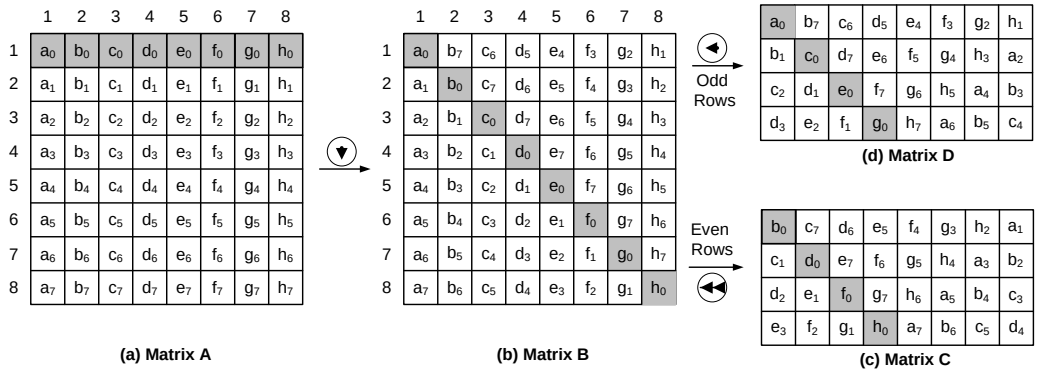
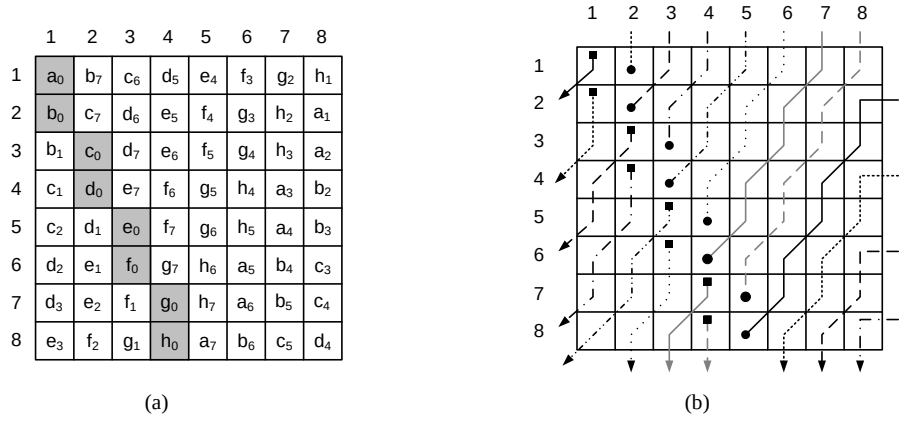


Figure 4-18. QPDEC Type1$_{row}$ applied on an 8×8 input.



(a) (b)

Figure 4-19. (a) Memory content related to QPDEC Type1$_{row}$. (b) Data scanning flow of the memory for QPDEC Type1$_{row}$.

Even though the data flow described in Figure 4-19(b) needs memory with 2D address line (($Xaddr_a, Yaddr_a$) and ($Xaddr_b, Yaddr_b$)), its hardware is impractical due to many required resources. Therefore, a dual-input and dual-output random access memory with two single address lines accompanied by an address converter, changing 2D address to 1D address, are utilized (Figure 4-20). Following are the expressions for the above address conversion for a $Z \times R$ memory:

$$Addr\_a = (Yaddr_a - 1) + [(Xaddr_a - 1) \times Z]$$
$$1 \le Yaddr_a \le R, 1 \le Xaddr_a \le Z$$
$$\Rightarrow 0 \le Addr\_a \le (Z * R) - 1$$

4-1

$$Addr\_b = (Yaddr_b - 1) + [(Xaddr_b - 1) \times Z]$$
$$1 \le Yaddr_b \le R, 1 \le Xaddr_b \le Z$$
$$\Rightarrow 0 \le Addr\_b \le (Z * R) - 1$$

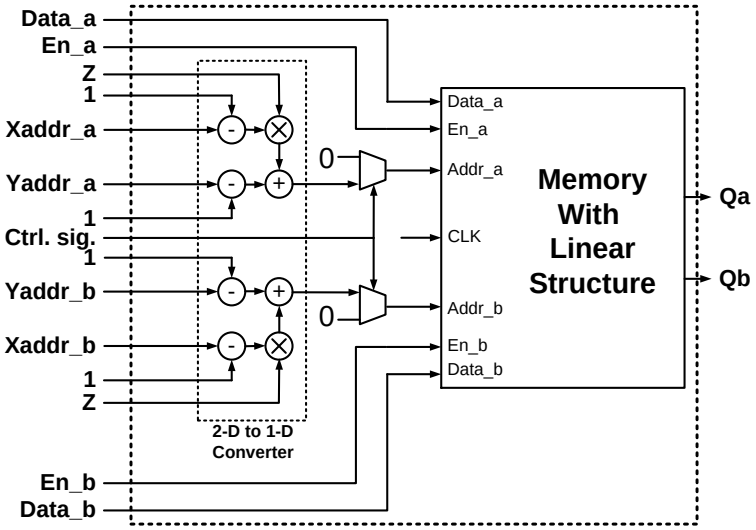


Figure 4-20. Dual-input and dual-output Memory with the address converter.

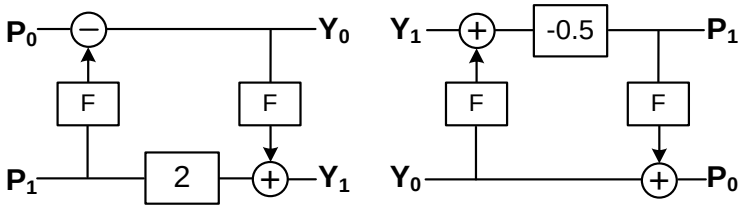
4.4.3 Ladder Architecture

In this section, our hardware-efficient architecture of ladder structure is proposed for DFB implementation. The original ladder structure (Figure 4-12) used in the DFB design has two coefficients (i.e. $1/\sqrt{2}$ and $-\sqrt{2}$), which increases the hardware complexity. It also degrades the accuracy if implemented in a fixed-point mode. These coefficients can be removed by merging them in both construction and reconstruction blocks by manipulating the equations describing them.

Figure 4-21(a)-(b) illustrate the proposed equivalent ladder structures of those in Figure 4-12(a)-(b). The proposed structures only have 2 and 0.5 as coefficients, which

can be realized by simple shift operation. Using a 2D filter with the size $L \times L$ of leads to $L^2 - 1$ adders and L^2 multipliers. A 2D filter based upon the PKVA filter [93] with the filter size of 6×6 is described in Equation 4-2.

$$\mathrm{F} = \begin{bmatrix} +a & +b & +c & -c & -b & -a \\ +b & +d & +e & -e & -d & -b \\ +c & +e & +k & -k & -e & -c \\ -c & -e & -k & +k & +e & +c \\ -b & -d & -e & +e & +d & +b \\ -a & -b & -c & +c & +b & +a \end{bmatrix} \qquad 4\text{-}2$$

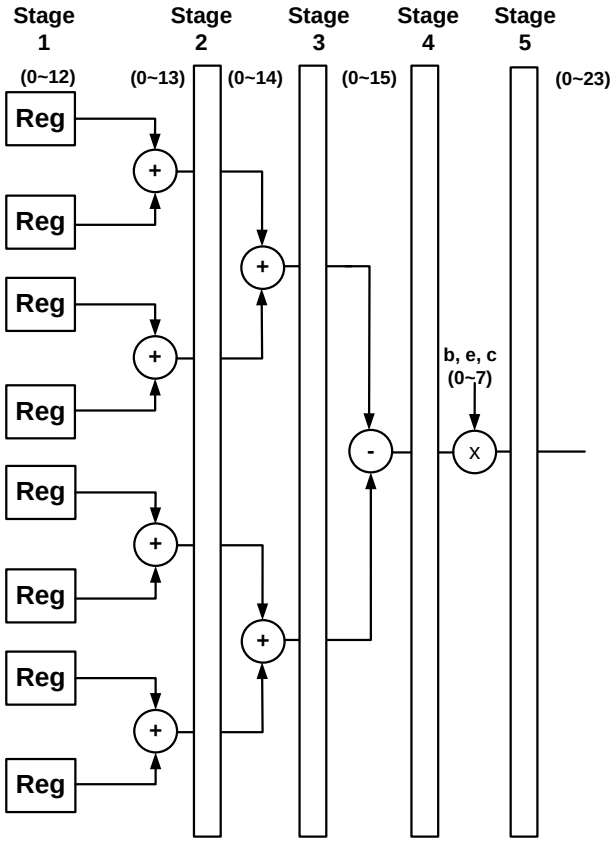


(a) (b)

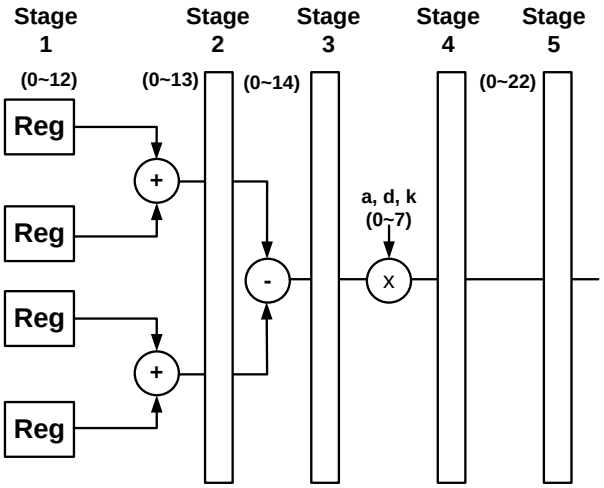
Figure 4-21. Ladder structure simplification: (a) Equivalent ladder structure for Figure 4-12(a), and (b) Equivalent ladder structure for Figure 4-12(b).

The 2D filter (F) consists of six distinct coefficients (i. e. a, b, c, d, e, k). Three coefficients ($a, d,$ and k) are repeated four times and the rest of the coefficients ($b, e,$ and c) are repeated eight times. In order to get the output, pixels surrounded by filter F are multiplied in parallel by the corresponding filter coefficients and added. Since the multipliers use a lot of resources, three pipeline tree architectures are designed to decrease the number of multipliers to L. The corresponding tree architectures depicted in Figure 4-22(a)-(b) are used to facilitate an output generation. Figure 4-22(c) is a tree structure for adding the results of the above two tree structures.

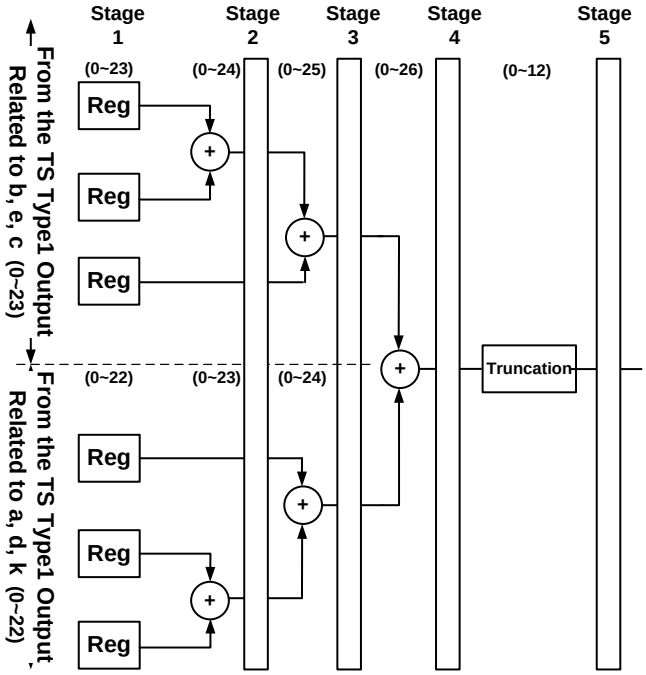
Figure 4-23 illustrates the pipeline architecture for implementing the ladder structures. Two 6×6 Register matrixes R_{ij} (where $i, j = 0, 1, 2, 3, 4, 5$) store 36 pixels within the windows to calculate the outputs (Y_0 and Y_1). The aforementioned tree structures are doing the filter function by taking four or six pixels from R_{ij} as indicated in Figure 4-23. Two series of five $N \times 1$ FIFOs in parallel with the window are to buffer the recently arrived data which should be processed through 6×6 windows. As a result, DFB is a combination of memories, various memory address generators and ladder structures.



(a)



(b)



(c)

Figure 4-22. Three tree pipeline architectures: (a) Tree architecture related to coefficients b, e, c (TS Type 1), (b) Tree pipeline architecture related to coefficients a, d, k (TS Type 2), (c) Tree pipeline architecture used to add the results of other tree pipeline architectures (TS Type 3). (The numbers written in parentheses are the width of stage's output. They are obtained based on the quantization analysis given in Section 4.5)

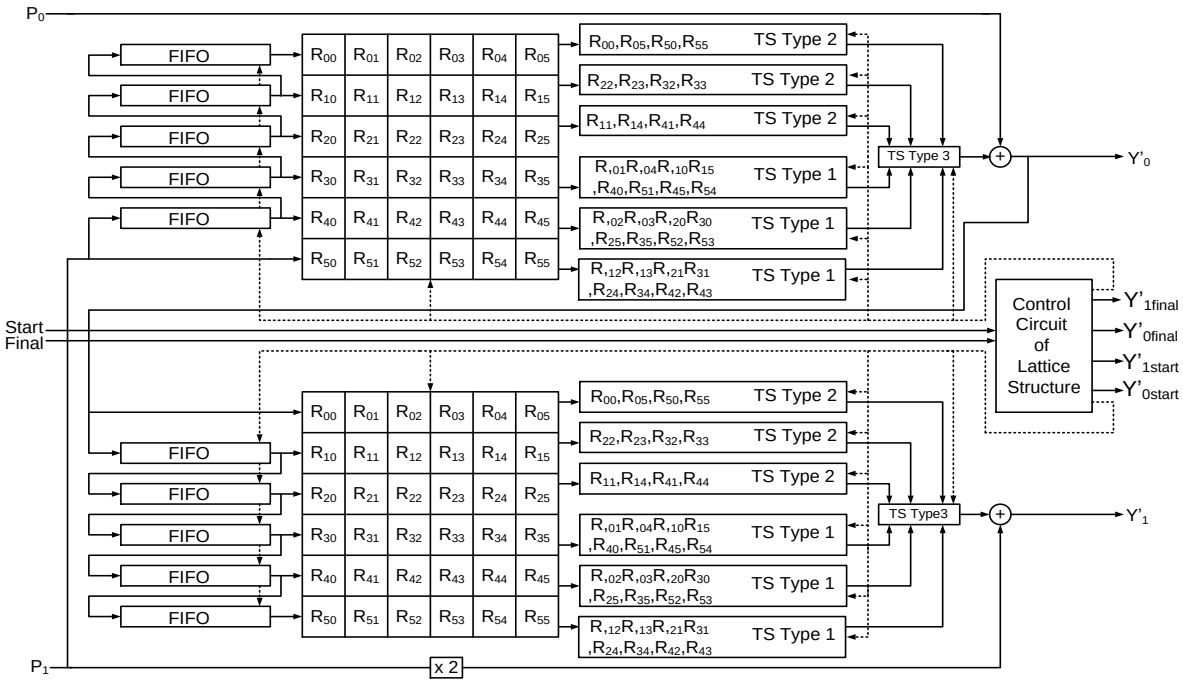


Figure 4-23. Proposed pipeline architecture for ladder structure.

4.4.4 Proposed Hardware Architecture of Contourlet Transform

Having proposed hardware architectures of LP and DFB in the previous sections, we can thus achieve the proposed hardware architecture of CT by concatenating LP and DFB as shown in Figure 4-24. In the proposed architecture, input image is first analyzed by LP to generate the low and high frequency outputs. High frequency output is then stored in the memory through multiplexers (M_1 and M_2), sharing the memory between LP and ladder structure. After that memory is devoted to DFB and control circuits check the whole flow of directional division of high frequency part with respect to the $nLevs$.

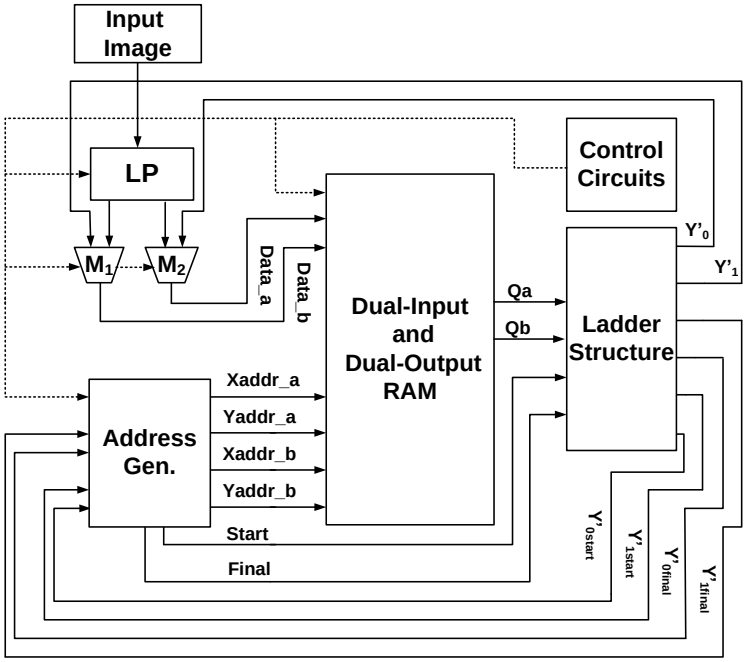


Figure 4-24. CT Proposed Architecture.

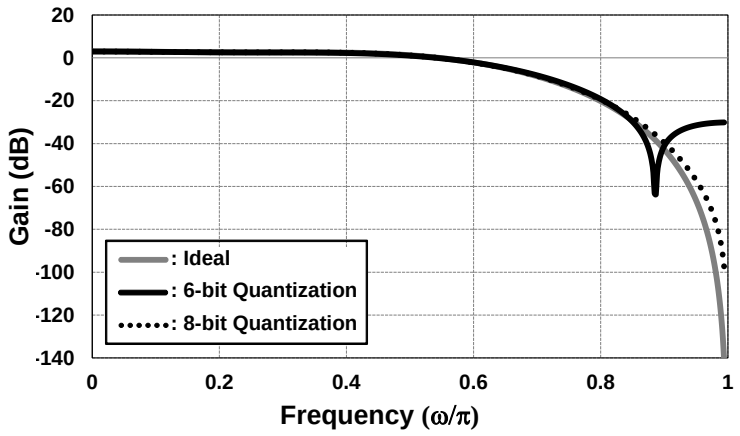
4.5 Performance Evaluation

In this section, we first conduct extensive quantization analysis to evaluate the error occurred due to the fixed-point arithmetic for representing filters coefficients and executing mathematical operations in the proposed CT architecture both in Laplacian pyramid and directional filter bank. Then the time complexity of LP as well as the whole flow of the CT architecture is provided. Finally the proposed architecture is verified by FPGA implementation and the results are given in details.

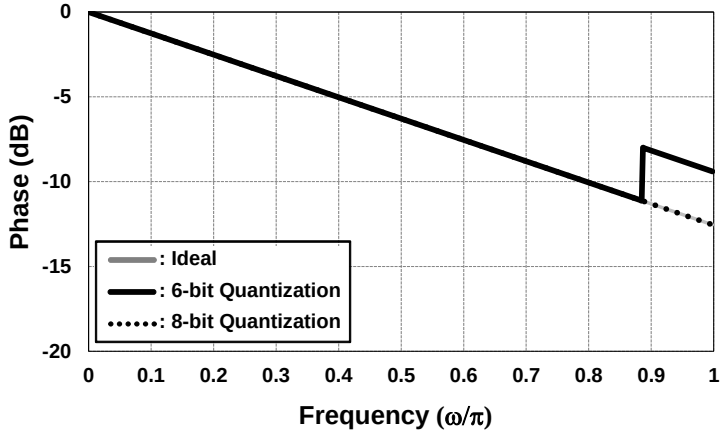
4.5.1 Quantization Analysis of Laplacian Pyramid

The LP algorithm is implemented by the '9/7' filters that were introduced in [94]. To reduce the amount of errors, the frequency response of the quantized filters need to

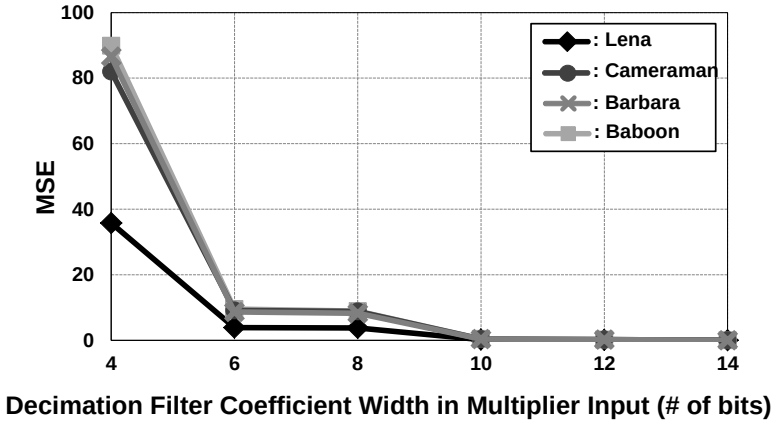
be preserved close to the ideal case [95]. Figure 4-25(a)-(b) explain that the 8-bit quantization for the decimation filter in Figure 4-5 provides a similar frequency response compared to the ideal one. In addition, we need to estimate the effect of the number of coefficient bits on the LP outputs. The number of required coefficient bits can be determined by examining various images in terms of MSE and PSNR. Figure 4-25(c) presents the MSE values of the low frequency coefficients (LFC in Figure 4-5) of several images over different numbers of coefficient bits for decimation filter. It reveals that MSE is less than 10 if 6 or more bits are employed in the decimation filter (h_x and h_y). Similarly, Figure 4-25(d) illustrates the MSEs in the interpolation part (g_x and g_y). By adopting 8 bits for the interpolation filters (g_x and g_y), the MSE of the high frequency coefficients (HFC in Figure 4-5) becomes less than 5. In this works, we employ 8-bit coefficients, which results in PSNR greater than 40 dB.



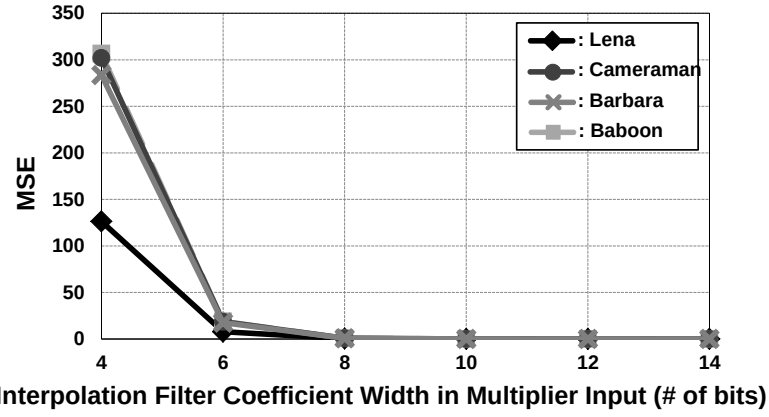
(a)



(b)



(c)



(d)

Figure 4-25. Quantization analysis: (a), (b) '9' filter frequency response, (c) Decimation part, and (d) Interpolation part.

Table 4-1 summarizes the simulated MSE and PSNR results of some standard images for LP. Good values can be observed since the PSNR values are more than 30 dB and MSE values are all less than 7 [96]. Since 8 bits are used to implement the LP filters, the high frequency output can be truncated to 13 bits without losing output accuracy for the standard test images. Therefore, the memory width should be equal to or greater than 13 bits.

Table 4-1. MSE of LFC, HFC and PSNR of the Reconstructed Images for LP.

Image	MSE(LFC)	MSE(HFC)	PSNR(R)
Cameraman	6.4354	3.2741	40.5050
Barbara	6.4063	2.7167	40.9748
Lena	6.2835	2.1329	44.8239
Baboon	6.4563	2.4270	40.7610

Note: Filters are quantized to 8 bits.

4.5.2 Quantization Analysis of Directional Filter Bank

Similarly, the effect of DFB quantization on the filter F (Figure 4-21) is illustrated in Figure 4-26. Note that the frequency response of 8-bit representation is almost equal to floating-point one (focusing on band-pass regions). It turns out that MSE of both 8-bit and 6-bit representations are small. By selecting 13-bit, the maximum and the minimum possible values are 4095 and -4096, respectively. Figure 4-27 shows the maximum absolute value of Y_0 and Y_1 (Figure 4-21) versus n_i. According to Figure 4-27, for $n_i = 4, 5$ there are some outputs that are greater than 4095 or less than -4096. Their numbers are tabulated in Table 4-2. If 4095 and -4096 replace all these values when n_i equals four, PSNR between ideal and quantized reconstructed images is calculated as $\text{PSNR}_{n_i=4}$ in Table 4-2. The results in Table 4-2 validate that the proposed CT architec-

ture shows good performance. Similar replacement with n_i = 5 leads to acceptable results as summarized in Table 4-2. For better PSNR, it is required to increase the memory width by one bit.

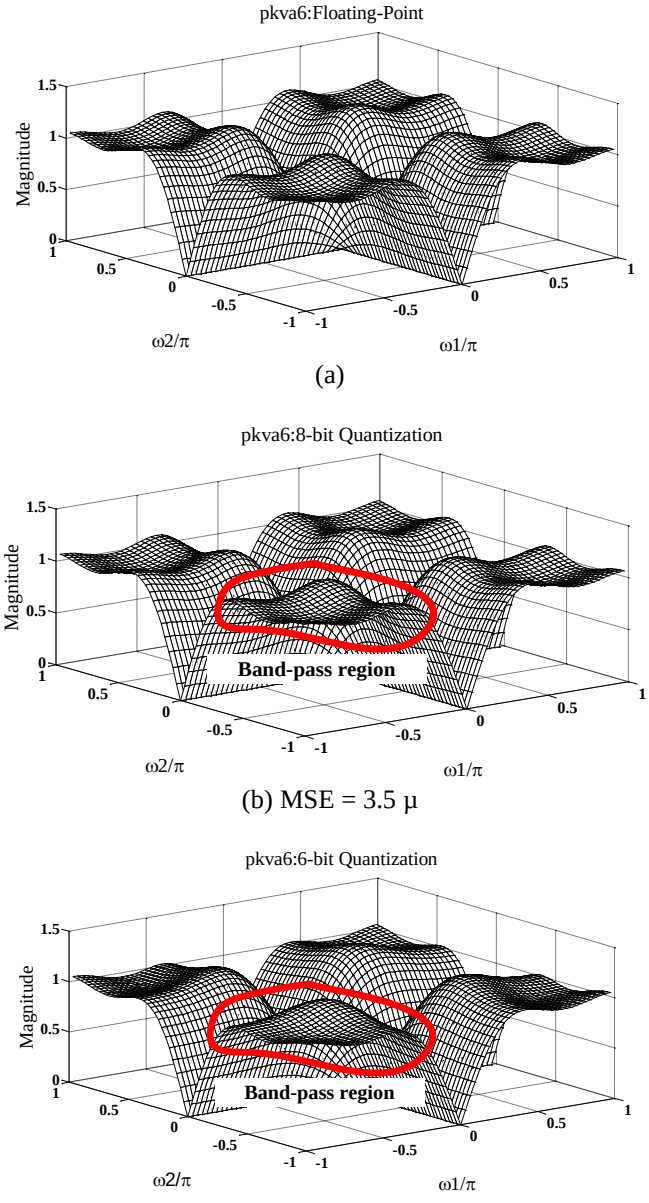


Figure 4-26. Quantization Analysis of 2D PKVA6 filter: (a) floating point, (b) 8-bit quantization, and (c) 6-bit quantization. (The frequency response in Figure 4-26(b) is quite closer to the frequency response in Figure 4-26(a), focusing on band-pass region.)

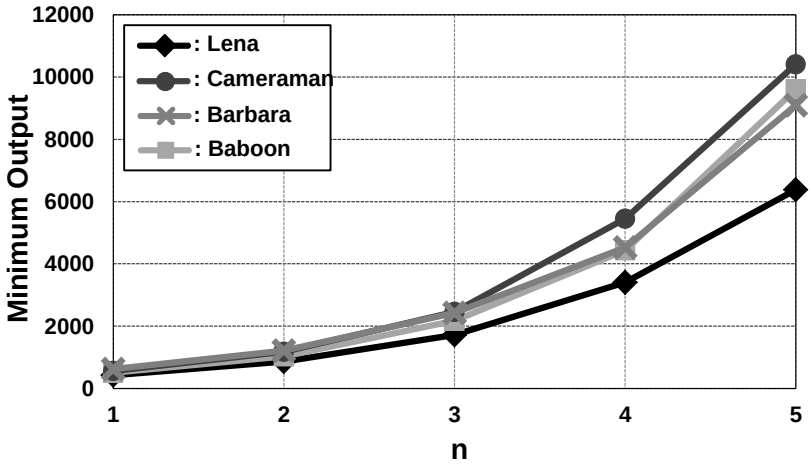


Figure 4-27. Maximum numbers exceeding by 4095 for different n.

Table 4-2. The number of coefficients exceeds by 4095 or -4096.

Image	$n_i = 4$	$n_i = 5$	PSNRn_i=4	PSNRn_i=5
Cameraman	12	761	51.3	29.33
Barbara	6	573	70.61	31.86
Lena	1	235	74.43	39.11
Baboon	5	642	62.98	31.04

4.5.3 Computational Complexity of the Proposed LP and DFB

In order to compare the proposed LP architecture with the existing ones [56, 62, 86, 87], the computational complexity is analyzed in this section. Suppose that for an $N \times N$ input image, the $K \times 1$ decimation filter and the $P \times 1$ interpolation filter, the number of required additions and multiplications throughout the whole process can be calculated. In the proposed LP architecture, The numbers of additions and multiplications in the decimation part are respectively $3N^2(K-1)/4$ and $3KN^2/4$. In addition, the numbers of additions and multiplications in the interpolation part are $3N^2(P-2)/4$ and $3PN^2/4$. Whereas in the conventional architectures, the numbers of additions and multiplications in decimation part are $3N^2(K-1)/2$ and $3KN^2/2$ and the numbers of

additions and multiplications in the interpolation part are $3N^2(P-2)/2$ and $3PN^2/2$, respectively. We can thus conclude that half of the arithmetic operations in decimation part are redundant and the percentage of the redundant additions and multiplications in the interpolation part are $P \times 100\% \,/(2P-2)$ and 50 %, respectively.

In order to analyze the computational complexity of filtering in DFB, the filter size and input image size are generalized as $G \times G$ and $\frac{\mathrm{N}}{2} \times \mathrm{N}$ (for example refer to Figure 4-13). The total numbers of multiplications and additions are N^2G and $N^2(G^2-1)$, respectively. Ac compare to conventional DFB implementation, the number of multiplications is reduced by $\frac{G-1}{G} \times 100$, which is 83.33% for $G = 6$.

4.5.4 Time Complexity of the Proposed Hardware Architecture of Contourlet

For an $N \times N$ image, during the first $N^2/2$ cycles, low and high frequency coefficients concerning LP are calculated and the high frequency coefficients are stored in the memory whose addresses are generated by the address generator for QPDEC $\mathrm{Type1_{row}}$. During the next N^2 cycles, Y_{11} and Y_{12} corresponding to $nLevs = [1]$ (Figure 4-13) are obtained and saved in the memory. After that, Y_{11} is fetched from the memory in order of QPDEC $\mathrm{Type2_{col}}$ and is analyzed with the ladder structure. Then the results, Y_{21} and Y_{22} are saved in the memory in $N^2/2$ cycles. After that, the ladder structure is devoted to the Y_{12} in the next $N^2/2$ cycles calculating Y_{23} and Y_{24}. The timing diagram given in Table 4-3 exemplifies these processes for $nLevs = [3]$ (n_i is equal to 3), which is acquired by total cycle (TC):

$$TC = \frac{N \times N}{2} + n(N \times N), \quad n = 1, 2, \ldots \tag{4-3}$$

Finally by assuming that CT is implemented at the most by $nLevs = [i, j, k]$, the *TC* is:

$$TC = \frac{231N^2}{32} \quad for\ i, j, k = 5 \tag{4-4}$$

For N which is equal to 512, TC is 1892352. So for real-time application, the frame per second equal to 40 (fps) requires:

$$fps = 40 \Rightarrow \text{f} \geq 75.7 \text{ MHz} \tag{4-5}$$

Table 4-3. Timing Diagram

nLevs=[3]								
Cycle	LP	*n* = 1	*n* = 2		*n* = 3			
$N^2/2$	LFC HFC	-	-	-	-	-	-	-
$3N^2/2$	-	Y_{11} Y_{12}	-	-	-	-	-	-
$2N^2$	-	-	Y_{22} Y_{21}	-	-	-	-	-
$5N^2/2$	-	-	-	Y_{24} Y_{23}	-	-	-	-
$11N^2/4$	-	-	-	-	Y_{32} Y_{31}	-	-	-
$12N^2/4$	-	-	-	-	-	Y_{34} Y_{33}	-	-
$13N^2/4$	-	-	-	-	-	-	Y_{36} Y_{35}	-
$14N^2/4$	-	-	-	-	-	-	-	Y_{38} Y_{37}

4.5.5 FPGA Implementation and Comparison

To verify the proposed architecture, the whole design is described in VHDL and then implemented on Altera Stratix II EP2S180F1020CS FPGA device [97] operating at 100MHz. This FPGA has three internal RAM blocks named as M512, M4K and M-RAM that can be configured as dual-input dual-output RAM. In the design, 36% of total RAM blocks are utilized to realize the 512×512 memory with word-length of 13 bits. The basic building block of logic in this chip is named the adaptive logic module (ALM). In EP2S180F1020CS FPGA chip, there are 143520 ALM blocks out of which 62518 are deployed to implement the design. FIFOs utilized in ladder structure (Figure 4-23) use 23353 ALM blocks of 24980 ALM blocks (Table 4-4). In addition, 28571 of 33911 ALM blocks utilized in LP are used to implement the FIFOs (Figure 4-16). Therefore it is seen that FIFOs dominates around 83% of the ALM blocks in the whole design.

Table 4-4. Usage of Hardware Resources

	LP	Ladder struc-ture	Dual-input dual -output Memory	Other parts[1]
Number of ALM	33911	24980	3538	89
ALM (%)	23.63	17.41	2.47	.06

[1]Other parts include "Address Gen." and "Control Circuits"

The only hardware implementation of Contourlet transform is presented in [88] for an image denoising application. It performs the Contourlet with $nLevs = [2]$ on the 8x8 block of the input image. Since the input image size is highly limited to an 8x8 block, the hardware implementation is very much simplified. In addition, the conventional LP architecture is implemented, leading to more than 50% redundant and zero-

operand arithmetic operations. There is not enough detail on DFB in order to do comparison. The design in [88] can operate at maximum frequency of 150 MHz and uses 22k logic cells and 75k memory cells.

4.6 Conclusion

This chapter has presented proposed hardware architecture for Contourlet Transform. The objective is to firstly bring an intuitive explanation of the process leading to directionally dividing of 2D frequency plane. It is done by proposing a new set of symbols and block diagrams making CT more comprehensible. Secondly, a CT proposed architecture is developed by separating into main parts named Laplacian Pyramid and directional filter banks. In LP, more than half of the arithmetic operations in decimation and interpolation parts have been reduced compared to conventional designs. In DFB a memory-based architecture along with various address generators are employed to get the wedge-shaped sub-bands. Tree structures and 2D filtering are utilized to reach the real time performance. In addition, quantization analysis extensively applies to estimates the FIFOs and memory cells width and the required number of bits for fixed-point arithmetic. Finally, it is verified through FPGA implementation. it should be pointed out that we are going to design integrated circuit of CT with further architecture and circuit improvements to reduce the power for low power applications.

Chapter 5. A 128-Channel Spike Sorting Processor

5.1 Introduction

Multi-electrode intracranial recording technology is highly required for various applications such as brain studies, brain machine interface (BMI) and treatment of disorders like epilepsy, memory loss and paralysis [98, 99]. The intracranial recording signals are generally recorded by intracellular or extracellular electrodes. Intracellular electrode is attached to a cell and records the activity of the cell whereas the extracellular electrode records the activity from outside of the cell through the cell medium. Implanting large number of electrodes to intracellularly record the brain signals is very difficult. Therefore, the extracellular recording has become dominant technique in many brain studies and it has provided the opportunity of exploring complex brain activities such as perception and movement [8]. In BMI [100], the neuronal signals from patients can be real-time translated to command signals to use in assistive devices. In these applications, the chronic electrode arrays are needed to be implanted for long-life use and moreover the signal interpretation should be performed in real-time. As shown in Figure 5-1, a severe spinal chord injury prevents transmitting movement command to the hand muscles. As an alternative approach, the command signal can be recorded from motor cortex and after being processed by computer is converted to a steering command [101]. In this way patient regain some sort of movement.

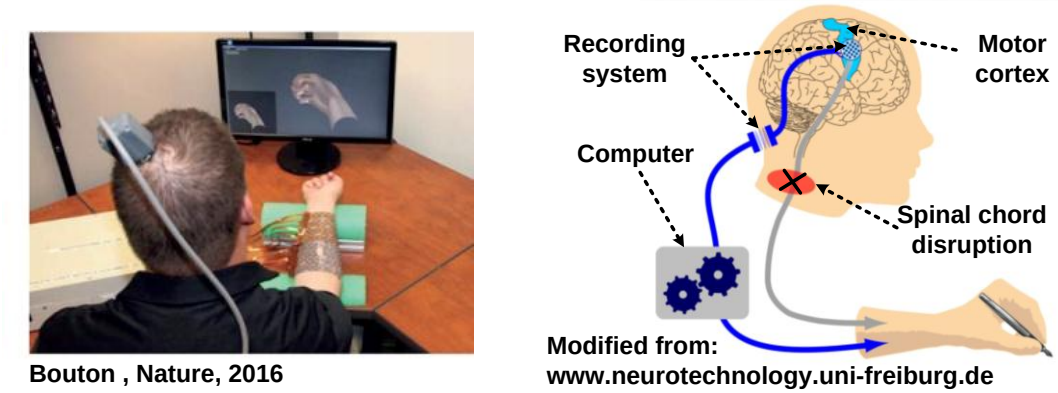


Figure 5-1. Conceptual BMI application to help patient with severe neurological disorder.

The recorded signal by extracellular electrodes as shown in Figure 5-2 is composed of high-frequency content (300Hz ~ 6KHz) named unit activity or spike, and low-frequency content (< 600Hz), named local field potential (LFP) [102]. LFP is significantly originated from large populations of cells and is an indication of the "cooperative actions" of neurons. The high-frequency signal, which is our band of interest, is the activity of neurons close enough to the electrode tip (single-unit and multiunit activity) plus the background noise generated by distant neurons. Therefore to obtain the unit activity signal, the raw data is first band-pass filtered and then digitized by sampling frequency around 25 KHz. The crucial step to extract the information from high frequency content of extracellular recording is called spike sorting [8] and shown in Figure 5-3. It basically assigns the detected spikes to their source neurons located near recording electrodes. Spike sorting typically consists of spike detector (SD), feature extractor (FE) and dimensionality reduction (DR), and clustering including classifier and training engine. Detector first identifies the spikes from the neural signal and aligns them to a common

point to offset the sampling jitter. Then a set of features is extracted and sent to the clustering assigning detected spikes to a specific neuron near by the electrode.

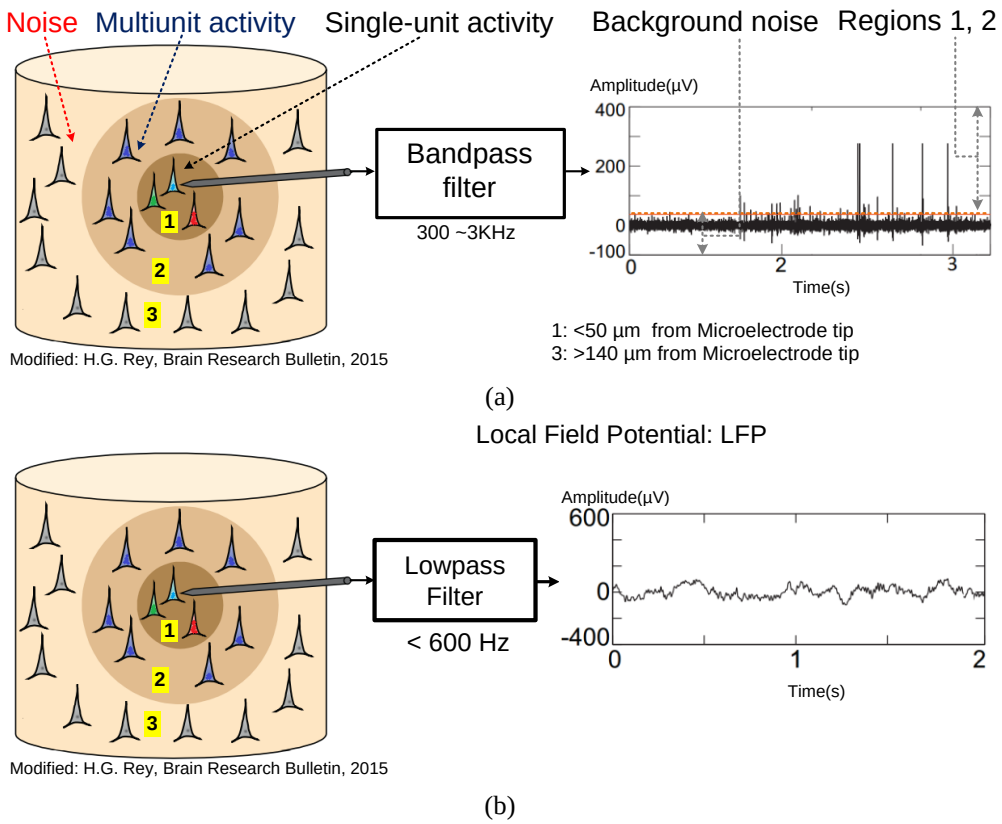


Figure 5-2. Extracellular recording signal. (a) High-frequency content, (b) Low-frequency content.

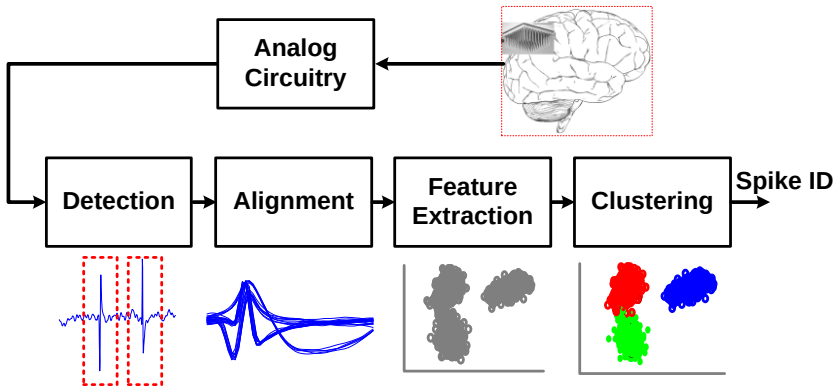
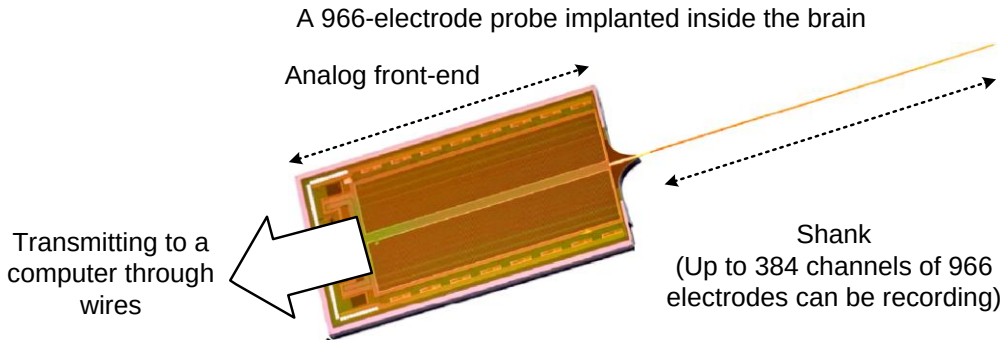


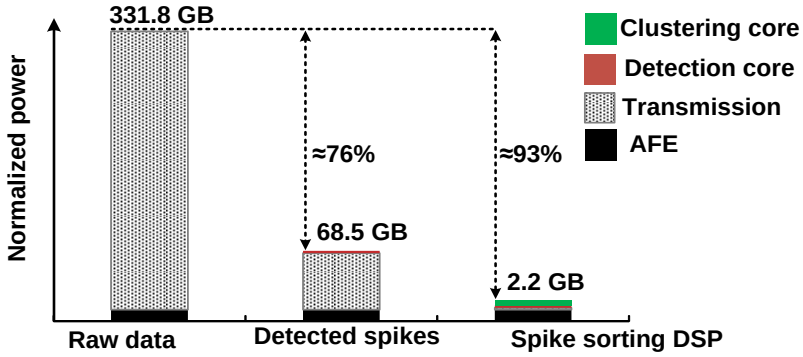
Figure 5-3. A typical brain signal recording system consists of an analog front-end and spike sorting back-end.

In general, neural recording system has exponentially grown since the 1950s and now there are multi-electrode arrays which allow recording hundreds of neurons [21, 22, 102]. Neural probe [22] shown in Figure 5-4(a) has achieved the highest electrode count reported so far. It can record up to 384 channels at the same time. In traditional off-line spike sorting, raw signals from different neurons are transmitted to a nearby computer through wires for further analysis (Figure 5-4(a)). However, this approach faces practical limitations such as very high data rate, real-time closed-loop experiment, freedom of movement and infection on subjects [8, 102]. For instance, the data rate for 128 channels, 25 kS/s recording system using 8 bit ADCs is 25.6 Mb/s. Therefore, practically transmitting this amount of data violates the power density requirement on implantable devices which is expected to be less than 277 $\mu W/mm^2$ [103]. As estimated in Figure 5-4 (b), on-chip spike sorting can reduce required storage memory for one day recording by 99% from 332 GB to 2.2 GB. More importantly, the transmitted data rate and thus power consumption is reduced by 93%, making implantable chip much more feasible.

As a result, an IC implementation of spike sorting has been proved to be essential for multi-channel neural signal processing research [7-9, 102, 104]. Integrating whole or a part of spike sorting algorithm on-chip leads to a significant reduction of the data to be transmitted to the subsequent blocks [98-100]. Moreover, such a realization of spike sorting is much faster and more power-efficient than that in the software domain. As a result, it makes multi-channel real-time processing applications much more feasible.



(a)



(b)

Figure 5-4. (a) a 966-electrode probe for neural signal recording. Up to 384 channels can be simultouneously recorded. (b) Estimated power and the required memory size for one day recording for various scenarios in 128-channel recording system. The assumptios for power calculations are Analog front-end: 10 μW/ch, communication: 1 nJ/bit, Spike detection: 2 μW/ch, Clustering: 4.65 μW/ch.

To our best knowledge, none of the state-of-the-art spike sorting chips [6, 99, 100, 103, 104] has implemented the whole spike sorting flow. [99] performs detection and feature extraction on 128-channel design. [7, 104] applies detection, alignment and feature extraction for 64-channel design. In [7, 9, 99], clustering is performed offline. [9] is the first 16-channel spike sorting chip performing online unsupervised clustering algo-rithm. However it does not include the feature extraction since the clustering algorithm

requires operating on the original spike data. Therefore it requires a large memory and thus the number of channels is more or less restricted.

In this chapter, we present a new spike sorting DSP performing detection, alignment, feature extraction and clustering for 128 channels. I propose a new detection and feature extraction operators which outperform the existing designs in terms of detection accuracy. Most importantly, its power consumption is only linearly dependent on the number of channels and thus they are more suitable for the hardware implementation. To improve the real-time clustering accuracy, an improved *k*-means algorithm that addresses the existing issues in *k*-means algorithm is proposed. Besides, novel circuit level techniques are also exploited to achieve energy-efficient multi-channel design. The rest of this chapter is as follows. Section 5.2 explains about dataset and design approach. Working principles of the proposed detector, feature extractor and improved *k*-means for online clustering is explained in Sections 5.3 and 5.4. Hardware implementation of 128 channels chip is described in Section 5.5. The measurement results are presented in Section 5.6. Section 5.7 concludes our chapter.

5.2 Neural Dataset and Design Approach

Evaluating the spike sorting algorithm requires the extensive realistic spike data with the defined true spikes and their corresponding clusters for classification accuracy verification. The most practical approach is to use the synthetic data [7, 9, 105]. This approach also makes it easier to compare our design with existing state-of-the-art. In this work, we use the popular database available in [105]. This database has 21 datasets, each contains 1440000 data points representing a digitized version of a continuous neural recording. Since the typical neural signal sampling rate is 20~30 kS/s [8], each dataset is equivalent to a waveform of 60 seconds recording time. True spike locations and cluster IDs are also available for each waveform. Each waveform has different number of spikes (1636-1787), spike locations, noise levels as well as number of spike clusters. This wide range of spike signal scenarios allow us to extensively test our sorting algorithm. From now on, the dataset refers to one complete waveform given in the database.

By using this database, our spike sorting algorithm was simulated using MATLAB to verify its clustering accuracy. Several existing algorithms were also simulated for comparison. The proposed spike sorting algorithm was then optimized for digital core RTL implementation. The core is subsequently synthesized, placed and routed (PnR) before being connected to the full-customized SRAM.

5.3 Proposed Detector and Feature Extractor

5.3.1 Detection and alignment

Spike detection is an essential step in the spike sorting flow separating the spikes from the neural signal [8]. All spike detection methods involve two stages of pre-emphasizing and thresholding. Having applied the pre-emphasis part, a threshold value checks if a spike is available or not. The main pre-emphasis methods are absolute thresholding and nonlinear energy operator (NEO) [106] and hybrid algorithm [107]. Once a spike is detected, a 3-ms window of the input signal is captured for further processing [8]. This is because brain spikes usually last less than 3 ms [8]. Finally, the captured spikes are aligned in specific ways to mitigate the sampling jitter and noise effects which deteriorate the spike clustering. There are some popular alignment methods by which all spikes are aligned to a specific points such as maximum or minimum value, or maximum slope. We will discuss about this later in this section.

In hardware realization, there are always trade-offs between the detection accuracy (i.e. probability of detection), probability of false alarm, the power consumption and the circuit complexity [108]. Both absolute thresholding and NEO schemes are popular thanks to their simple circuit structures and good detection accuracy [109, 110]. In general, NEO outperforms absolute thresholding at the cost of adding complexity and power consumption.

Absolute thresholding, Equation 5-1, compares the absolute of input data ($s(n)$: spike sample) and a predefined threshold value [105]. If $abs(s(n))$ is larger than Thr

(threshold value), a spike is detected. Only one comparator is required to implement this detector.

$$Thr = 4\sigma_N \quad \sigma_N = median\left(\frac{|s(n)|}{0.6745}\right) \tag{5-1}$$

NEO ψ takes into account the differences between $s(n)$, $s(n-1)$ and $s(n+1)$, as shown in Equation 5-2. Intuitively, if we approximate $s(n-1) = s(n) - \varepsilon$ and $s(n+1) = s(n) + \varepsilon$, $\Psi(s(n)) = \varepsilon^2$. Thus $\psi[s(n)]$ will be very large if a spike is present. This operator amplifies the spike signal and suppresses noise and other low-frequency components. Therefore, it provides better detection accuracy. Similar to absolute thresholding, $\psi[s(n)]$ is compared with Thr_NEO in Equation 5-3 to make decision.

$$\psi[x(n)] = s^2(n) - s(n-1) \times s(n+1) \tag{5-2}$$

$$Thr\_NEO = 4\frac{1}{N}\sum_{1}^{N}\psi[s(n)] \tag{5-3}$$

Instead of using NEO or absolute thresholding, an integer coefficient filter-based pre-emphasis method, expressed in Equation 5-4, is proposed. The proposed filter y_D acts as a short-window convolution to capture spike-like waveforms. As a result, it provides better accuracy performance compared to both NEO and absolute thresholding, especially for noisy signals. Furthermore, because the filter coefficients are integers, its

hardware complexity is reduced to shift and addition. Its hardware implementation will be discussed in more details in Section 5.5.2.

$$y_D[n] = 128s(n) - 48s(n-1) - 156s(n-2) - 36s(n-3) + 56s(n-4) + 32s(n-5) \quad \text{5-4}$$

Figure 5-5 shows the Receiver Operating Characteristic (ROC) comparing various detection methods using two different datasets with different noise levels. The ROC is characterized using probability of detection and probability of false alarm (P_{FA}). To draw ROC curve, any of pre-emphasis methods are applied to the input signals provided in [105] and then output is subjected to a threshold. The threshold is swept from very low to very high value. Simultaneously P_D and P_{FA} are recorded for each threshold value. When the threshold is too low, everything is considered spike and thus the probability of detection approaches 100% but it also has a lot of false alarm. On the other hand, when the threshold is too high, only a few true spikes are detected when no false alarm happens. Apparently, there is a trade-off between the two. In Figure 5-5, only data points with $P_D > 90\%$ are shown since we are more interested in high spike detection accuracy regions. It can be seen that, when we lower the threshold value, more spikes are captured but it also results in higher false alarm rate. This is true for all three designs. However, the proposed design offers a better trade-off when compared to the other two designs. It improves both P_D and P_{FA}, especially for noisy signals. Note that reducing P_{FA} facilitates the clustering operation by allowing the cluster means to converge to the accurate final values smoothly without being interfered by outliers' effect on the cluster means.

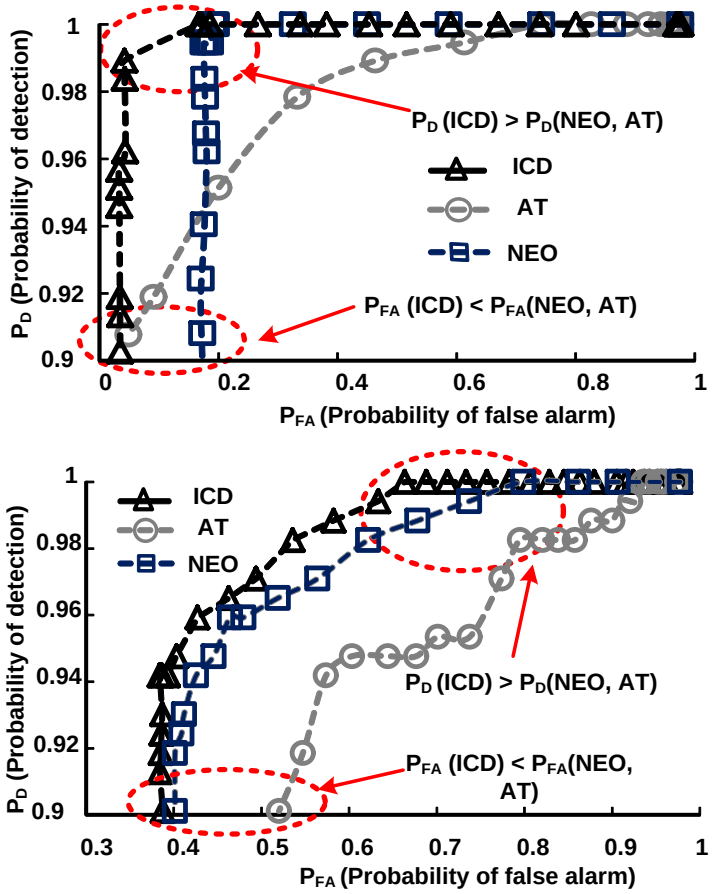
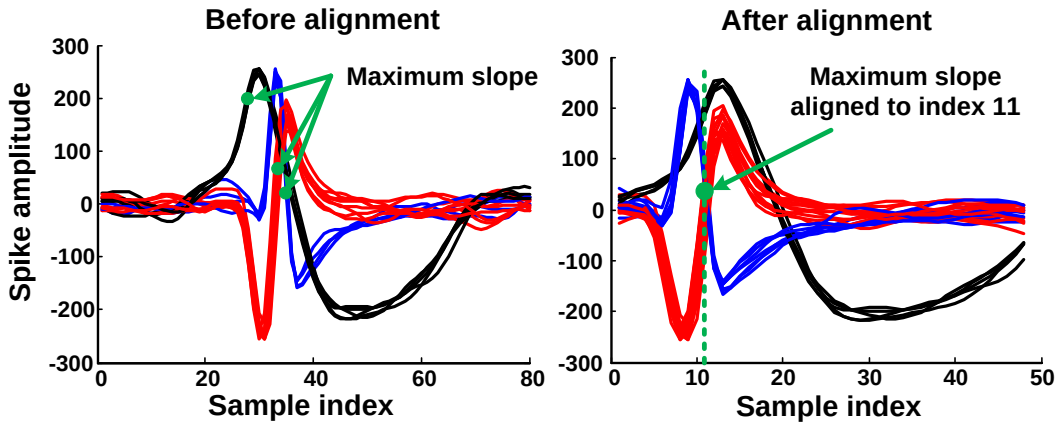


Figure 5-5. Various detection methods comparison over two input signals with different noise levels.

For spike alignment, we use the maximum slope since it is biologically significant and results in better clustering accuracy as compared to other most common method [8]. Correspondingly, every captured spike waveform is shifted so that index 11 has the maximum slope. Therefore all spikes are aligned with respect to index 11 as shown in Figure 5-6. Next, 48 samples of each detected spike are stored and undergone the feature extraction process explained in the next section.



(a) (b)

Figure 5-6. Detected spikes are aligned to their maiximum slope whose index is 11. (a) Before alignment; (b) After alignment.

5.3.2 Feature extraction (FE) and dimensionality reduction (DR)

FE along with DR is the second step of spike sorting flow. It basically extracts the major components of detected spikes and condenses them into several most important features to provide faster and more accurate clustering. The popular feature extraction methods [8] are Principal-Component Analysis (PCA), Discrete Wavelet Transform (DWT), Discrete Derivatives (DD) and Integral Transform (IT). As comprehensively studied in [8, 109], PCA is the most common approach in off-line spike sorting because of its high accuracy. PCA and DWT compute the same number of coefficients as the detected spike then reduces them to several largest coefficients before sending them to the clustering engine. However, they are not suitable for the hardware realization due to its complexity and high memory requirement.

DD approach is inspired by DWT but it is much simpler and thus more suitable for hardware implementation. It is described in Equation 5-5 where $s(n)$ is the spike sample, δ is the time step and *n* is the current sample index. In [7], δ is set to 1, 3, and 7, thereby generating a $3\times$ coefficients compared to the total number of spike samples. Then, seven coefficients are uniformly selected for each level, which result in 21 coefficients per spikes. The way through which the coefficients are selected is very important and affects the clustering accuracy.

$$dd_{\delta}(n) = s(n) - s(n-\delta) \qquad 5\text{-}5$$

IT method computes the area under the spike curve both in positive and negative phases. It generates two features per spike and can be easily implemented in hardware. However, its accuracy is not satisfactory and some parameters are required to be calculated through training phase. From above observation we conclude that DD feature extraction method is the most suitable choice for the hardware implementation, thanks to its simple hardware requirement and reasonable clustering accuracy. In fact, DD reaches higher accuracy compared to DWT for various DR methods though it requires more features.

We realize the FE using another integer-coefficient filter, y_{FE} in Equation 5-6, to process the detected spike. Subsequently, some samples are selected to reduce the dimensions. Since the sample selection greatly impacts on clustering accuracy, sampling method must be carefully chosen. Our observations on MATLAB simulations of various datasets reveals that samples whose indexes are 8, 11, 18 and 25 give the best clustering

results. They also outperform other hardware-oriented feature extraction methods. Figure 5-7 shows how the FE filter amplifies spike components and the significant of these indices. It can be seen that these indices coincides or very close to the local peaks. Most importantly, since index 11 has the maximum slope, it coincides with the global peak of the filtered waveform and thus is a crucial feature for classification.

$$y_{FE}[n] = 8x(n) - 2x(n-1) - 6x(n-2) - 4x(n-3) \qquad 5\text{-}6$$

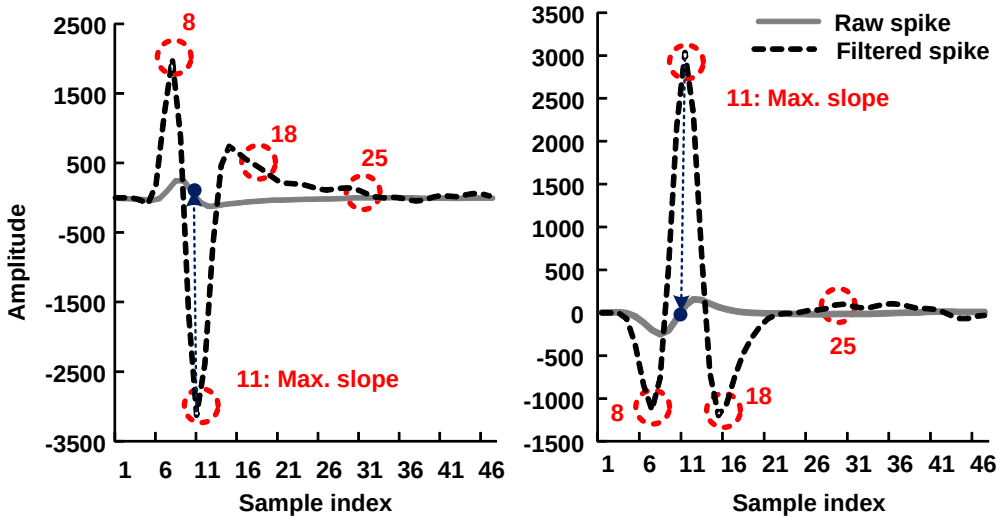


Figure 5-7. Four selected features whose indexes are 8, 11, 18 and 25 on two sample spikes.

Figure 5-8 shows how y_{FE} can improve the clustering accuracy by making very good separation in the feature spaces, which is a necessity for the good clustering. In order to visualize it, samples 11 and 18 are selected from both the spike samples and y_{FE} in Figure 5-8(a)-(d) for two difficult (i.e. spike classes are similar) datasets. In Figure 5-8(b) data points from different classes are hardly discriminated from each other. After going through our filter, they have much better features separation. Figure 5-9 shows that four selected features leads to the better clustering accuracy for various datasets compared to other scenarios. In Figure 5-9, MATLAB-based k-means clustering was used on top of the extracted features to compare the choice of features.

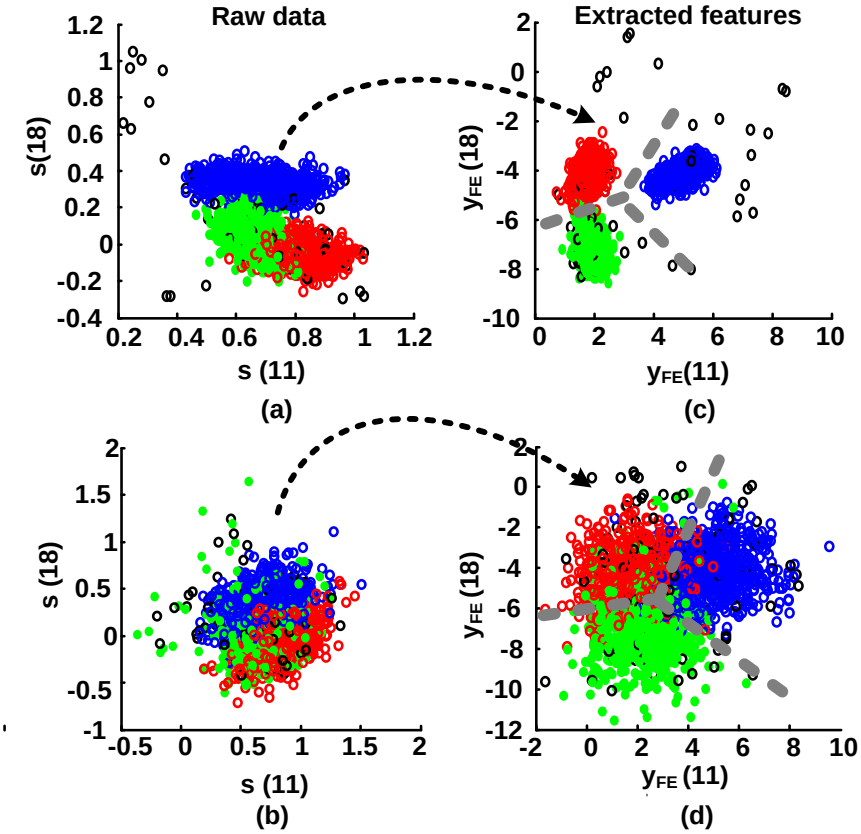


Figure 5-8. (a)-(b) Original spike samples whose indexes are 11 and 18. (c)-(d) Outputs of y_{FE} whose indexes are 11 and 18. Green, red and blue circles are representing the actual neurons and the black ones are outliers.

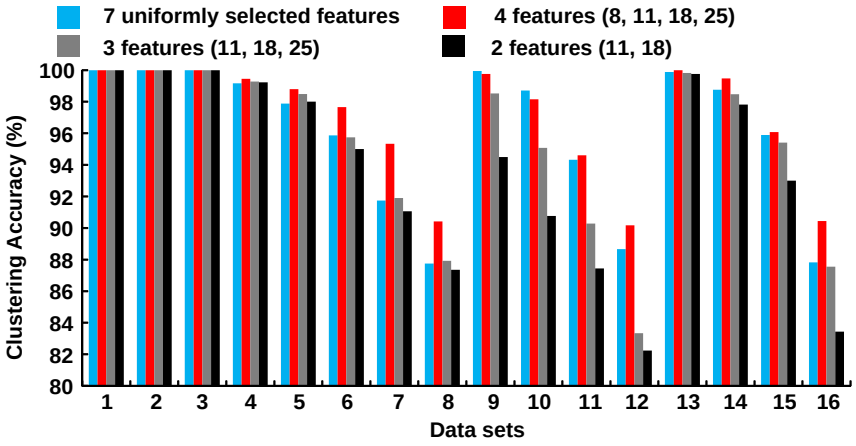


Figure 5-9. Performance comparison of different choice of features across several datasets. MATLAB $k-$mean was used for clustering in conjunction of our feature extraction to compare the effectiveness of the choice of features.

Figure 5-10 compares the proposed method with various numbers of features with DD whose δ is set to 1, 3, and 7, using standard MATLAB k-means function. It shows that the proposed feature extraction method with four, three and two features outperforms DD with 24 uniformly selected features and DD with the same four features. Furthermore, applying Equation 5-6 to detected spikes and then selecting 4 features leads to much smaller required memory compared to DD with 24 features. For instance, for a 3-neuron input signal, only 120 bits are required for four features, each with 10 bits. However it is 720 bits for the DD counterpart with 24 features. In addition, the number of features highly affects the clustering complexity in terms of computations and memory access power. For example, in [9] where no feature extraction is implemented, clustering algorithm requires the whole spike samples for time-domain comparison. It results in a large memory size which contributes to 88% of the DSP power and around 60% ~70% of active area. The proposed method therefore significantly reduces the complexity of the clustering engine when compared to existing state-of-the-arts.

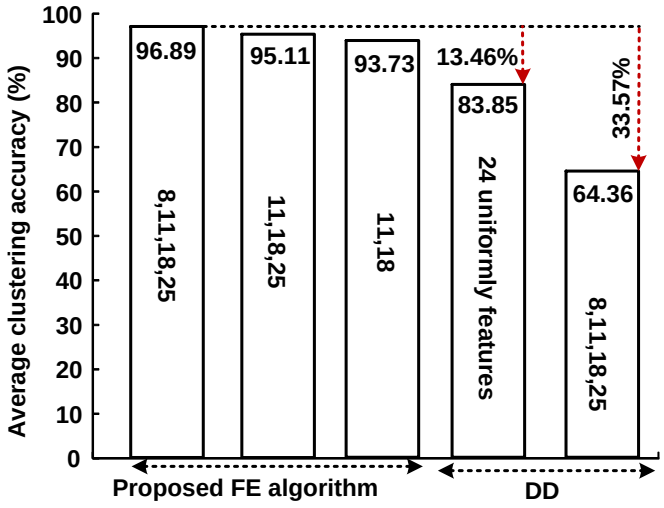


Figure 5-10. Averaged clustering accuracy for different feature extraction.

5.4 Proposed Improved *k*-means for Clustering

Clustering algorithms can be widely divided into two classes: supervised and unsupervised. In supervised learning, neuron ID of each spike in the training set is provided so that the model can predict the correct ID of the new spikes in the test set. However, in spike sorting applications, this supervised learning is both undesirable and infeasible. This is because both ground truth is hard to obtain (requires a lot of expert hours) and expert themselves can make mistakes when the number of spikes are large. Therefore, unsupervised algorithm is much more desirable. Another approach is to use semi-supervised algorithm when experts can interferes to enhance the clustering accuracy.

Most of currently available clustering algorithms are run on software with large memory capacity and powerful processors. (WaveClus [105], KlustaKwik [111] and Osort [112]). Osort is the reasonable clustering implemented on hardware. However, it requires large memory to implement. In this work, an improved *k*-means to improve both clustering accuracy and hardware efficiency is proposed.

k-means is a popular unsupervised/semisupervised clustering algorithm which classifies the input data based on their distance to the existing trained cluster means. During training, cluster means are initiated using randomly chosen some data points and then adjusted by looping through the whole training set. One prohibiting feature of *k*-means is that it requires the whole dataset to be stored for training. This is not feasible for hardware real-time training because of the limited memory capacity. Furthermore, the initial cluster means can be assigned wrongly (i.e. two data points belong to the same

clusters can be assigned to different means). In this case, k-means will have difficulty converging to the correct mean.

For example if we have two classes A and B but somehow the initial means are initiated both from data points in class A, $M_1 = A_i$ and $M_2 = A_j$. Therefore, with any other data points, they will be assigned to either M_1 and M_2, regardless of their actual class. Even those in class B can also be assigned to M_1 and M_2 with similar probability. As a result, cluster means will never converge. Therefore, k-means is usually repeated several times to find the best mean convergence. Thus, conventional k-means cannot be used in real-time hardware implementation. Because data only passes through the system once and large data storage are not available.

An improved k-means algorithm which functions in unsupervised fashion or semi-supervised is proposed. It also addresses the convergence issue of k-means. In unsupervised mode, the number of clusters is not specified but we assume that it has less than six clusters. In semi-supervised mode, number of clusters is provided to guide the cluster means convergence.

We tackle the issue of wrongly initiated means by allows the new data to form a new cluster, instead of forcing it to one of the existing means. Similar to k-means, distances from the new data to existing means are computed. In addition, distances between the existing means are also calculated. If the distances between the new data to existing means are much larger than distance between the means themselves, it indicates that the new data is quite far away from the means and thus deserve to form a new cluster. In this case, two of the existing means should be merged, to maintain the same number of clus-

ter. Extensive simulations have shown that, this allows us to converge even though initial means are purposely assigned wrongly.

Detailed implementation of the algorithm is as follows. Our cluster means' features are denoted as C_{ik} where i changes from 0 to 5 (represent 6 clusters) and k changes from 0 to 3 indicating four features obtained in feature extraction. The number of clusters is selected as six because there are usually less than six to eight neurons [9, 113]. This is the upper bound and is not the actual cluster means. If the actual number of clusters is less than 6 (e.g. 3), only the first 3 clusters are meaningful. Initially, C_{ik} is filled by the first 24 features from the first 6 spikes in the training set. Then the next four features of the 7th spike are stored in the temporary cluster (7th cluster) whose mean is named as C_{6k}. After that two set of l_1-distances represented by $d(C_{ik}, C_{jk})$ and $d(C_{ik}, C_{6k})$ are calculated as Equation 5-7. l_1-distance is used due to its good clustering accuracy and simple circuit implementation compare to l^2-distance.

$$d\left(C_{ik}, C_{jk}\right)_{i,j=0:5;i\neq j} = \sum_{k=1}^{4} \left|C_{ik} - C_{jk}\right|$$

$$d(C_{ik}, C_{6k})_{i=0:5} = \sum_{k=1}^{4} |C_{ik} - C_{6k}| \qquad \text{5-7}$$

$$d_u\left(C_{ik}, C_{jk}\right)_{i,j=0:5,i\neq j} = 1.5 * d(C_{ik}, C_{jk}) \qquad \text{5-8}$$

$$C_{ik,new} = \frac{15 * C_{ik} + C_{jk}}{16} \qquad \text{5-9}$$

$d(C_{ik}, C_{jk})_{i,j=0:5}$ represents the distances between existing clusters while $d(C_{ik}, C_{6k})_{i=0:5}$ represents the distance between the new data point (i.e. the temporary cluster) to the existing clusters. After calculating all $d(C_{ik}, C_{jk})$, they are weighted as given in Equation 5-8, so that the two main clusters (i changes from 0 to 5) are less subjected to merge together as compared to the case of main and temporary clusters. As a result, the minimum of all values among $d_u(C_{ik}, C_{jk})$ and $d(C_{ik}, C_{6k})$ are selected and two corresponding clusters are merged together. Once two clusters i and j are merged in cluster i, the cluster i mean is updated as Equation 5-9. Finally, after the training period is finished, the clusters means converge to their final values.

In supervised mode, the user should determine the number of clusters and the proposed clustering algorithm is adapted to it and operates similar to what is explained in unsupervised mode. Figure 5-11 shows how cluster means converge to their final values for user-predefined four clusters. Each cluster mean converges to its final value after training phase and it is shown by a dark circle. Since input signals incorporate three spikes classes, Figure 5-11 indicates three main clusters and one cluster for outliers.

Figure 5-12 compares the proposed clustering method with other clustering methods when applied to the whole dataset. As shown in Figure 5-12, the proposed one outperforms [9] in supervised and unsupervised modes. Note that in semi-supervised mode, the proposed algorithm has better accuracy when compared to unsupervised mode. The original k-means running in MATLAB using option "Replicates" which is set to 100, gives the better result. It is because original k-means is set to run 100 times with the new initial values to get the better clustering accuracy. As it is seen in Fig-

ure 5-12, the averaged clustering accuracy is achieved 72% for the proposed unsupervised clustering and it goes up to 86% if the number of clusters is known in advance.

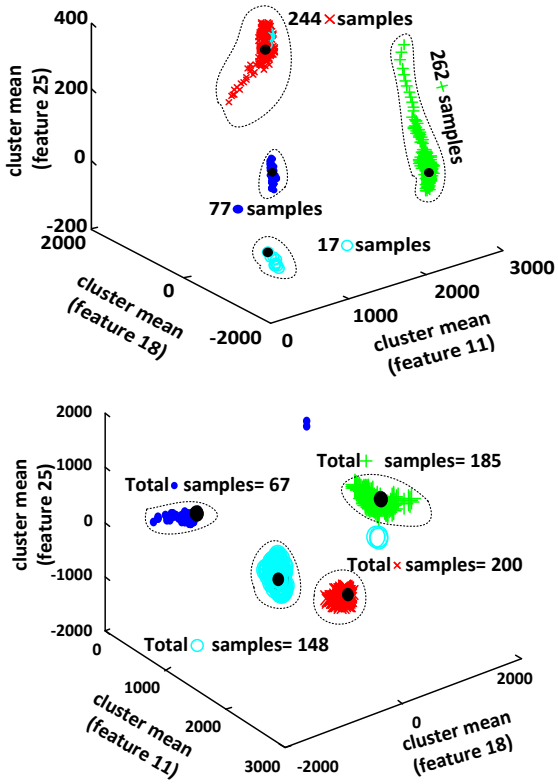


Figure 5-11. Cluster means convergence during the training phase for three input signals. Four clusters are initially selected and after training phase each cluster mean is converged to its final value shown in black circle. Three out of four clusters correspond to three various neurons and the forth one is outliers. The number given for each cluster is indicating the total samples assigned to the cluster during the training phase.

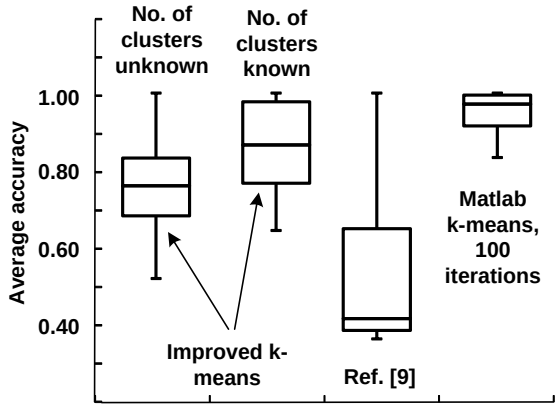


Figure 5-12. Clustering accuracy of Improved *k*-means compared to [9] and original *k*-means using 'Replicates' option set to 100.

5.5 Spike Sorting Chip

5.5.1 Architecture and Operation

Block diagram of the proposed design is illustrated in Figure 5-13. It closely follows the sorting flow. Input data of 128 channels are serially transmitted to the chip with the frequency of 3.2 MHz, thus the actual data rate for each channel is 25 kS/s. The chip works in two main modes: training modes and classifying mode. It has two clock domains, CLK and $CLKS$. CLK frequency is 3.2 MHz, corresponding to the input data rate and $CLKS$ frequency is 25 KHz, corresponding to the ADC sampling rate/channel. In general $fCLK = fCLKS \times N$ where N is the number of channels and $fCLKS$ is 25 KHz.

Training is only required to run once in a while to update the cluster means. Each channel is trained sequentially. The training period for a specified channel is controlled by the control circuit and it can vary from one channel to another one. During the training mode, after a spike is detected, features are calculated and passed through the training engine. After the training, the final clusters means are stored in the SRAM memory for classifying.

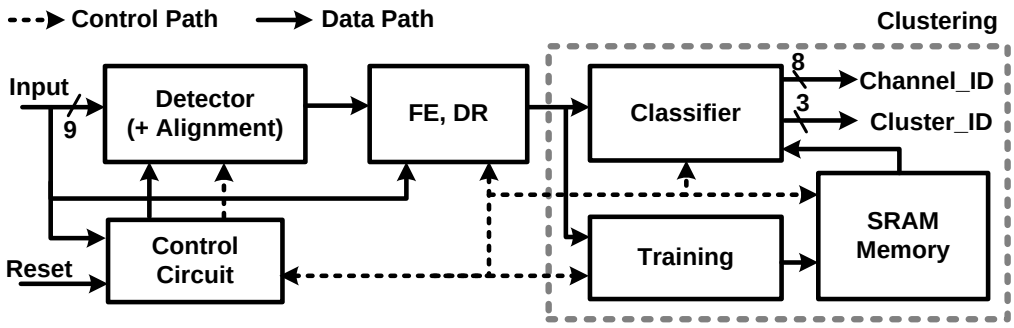


Figure 5-13. Functional block diagram of spike sorting.

During the classification phase, in parallel with performing the feature extraction, the previously trained clusters means of the corresponding channel are read out from the SRAM to a local memory inside the Classifier. This helps reducing the whole processing time. Furthermore, it allows the SRAM to operate using slow clock, which in turns allows us to use HVT device to suppress SRAM leakage because its speed performance is now relaxed. The Classifier runs and looks for the nearest cluster using l_1-distance to assign to it in 8 clock cycles. Finally the Classifier is clock-gated and reset for the next similar operation. Note that, in training mode, the Classifier is clock-gated to save power and vice-versa. Furthermore, Detector, FE and DR are shared between these two phases to save hardware resources.

In particular, Figure 5-14 shows the timing and scheduling of "Detector", "FE, DR" and clustering after a spike is detected. The detector is fully clock-gated by the control circuit. Since the detector filter is fifth order, there are six samples available for the corresponding channel in detector. "FE, DR" (shown in Figure 5-13) continues reading next 46 samples from that channel as shown in Figure 5-14. Simultaneously, it calculates

the maximum slope and features which will be stored in the local memory. Because input data of 128 channels is sequentially streamed in, the data for the detected channel is available every 128 cycles. So “FE, DR” block is only clocked once in a 128 clock cycles to reduce unnecessary switching activity. Right after reading 46 samples, “FE, DR” block is running with the fast clock to finish maximum slope and features calculation and selecting 4 corresponding features aligned to the maximum slope. As a result, the whole “FE, DR” block takes $5898(46 * 128 + 10)$ fast clock cycles to extract the features and then “FE, DR” is clock-gated and reset. 5898 clock cycles is almost equal to a single spike window width. Next depending on the operating phase, the features are sent either to training or to classifier as shown in Figure 5-13.

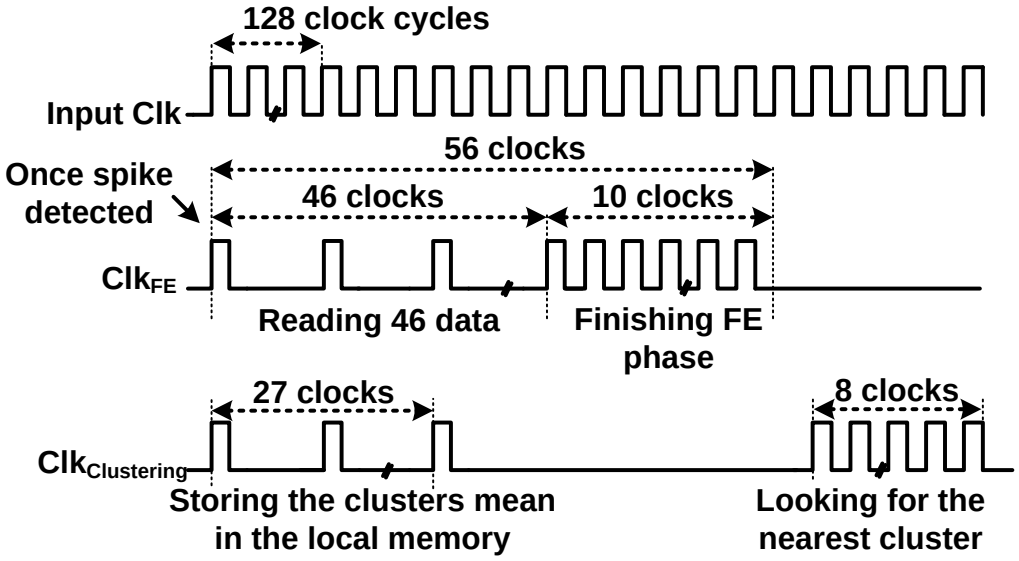


Figure 5-14. Timing and scheduling diagram of “Detector”, “FE, DR” and clustering after spike is detected.

5.5.2 Circuit Design Consideration

From the hardware point of view, each channel needs five 9-bit (i.e 45 bits) registers in detector. Therefore, 5.625 kb of memory is required in total to simultaneously perform detection for 128 channels. Because detector is almost operating all the time, it is the main source of dynamic power consumption. Furthermore, both its clock frequency (i.e. $fCLK = fCLKS \times N$) and loading (i.e. number of flip-flops) linearly grow with the number of channels, its dynamic power increases linearly with square of number of channels. As a result, the detector's dynamic power becomes prohibitive when N is high. Other blocks such as "FE, DR", "Training" and "Classifier" are shared between all channels and only activated once required to perform their operations through gated clock. We discuss a novel architecture to reduce dynamic power of the detector.

Interleaved architecture in [7] leads to a huge dynamic power loss because all the registers are connected in sequence and therefore their content is changing each clock cycle as depicted in Figure 5-15 for 128 channels. To avoid data transition of all registers at each clock cycle, a time-multiplexing architecture as shown in Figure 5-16 is designed to perform the clock gating technique. In Figure 5-16, RB_i where i varies from 1 to 128 shows the register bank for each channel. Mux_i where i is from 1 to 5 is selecting 5 data of the same channel and stores them in the output register. Clk_i where i is from 0 to 128 is the gated clock attributed to channel i.

The power comparison between two architectures shown in Figure 5-15 and Figure 5-16 for various numbers of channels is illustrated in Figure 5-17. The power is improved by 74% for 128 channels using time-multiplexing architecture applying extensive

clock gating technique. For 8 channels and less the interleaved implementation outperforms parallel one since the clock gating power overhead is significant.

To further reduce the computational complexity of arithmetic unit in detector, all multiplications introduced by the filter coefficients are converted to shift and addition operations. Therefore, power-of-two coefficients such as 128 and 32 are easily realized by the shift operations. Other coefficients such as -48, -156, -36 and 56 are first converted to the summation of power-of-two values and then implemented by the shift operations as shown in Figure 5-18. Four pipeline stages are incorporated to improve performance, especially in near-threshold voltage where latency of the long adder chain becomes critical and affects the whole design performance.

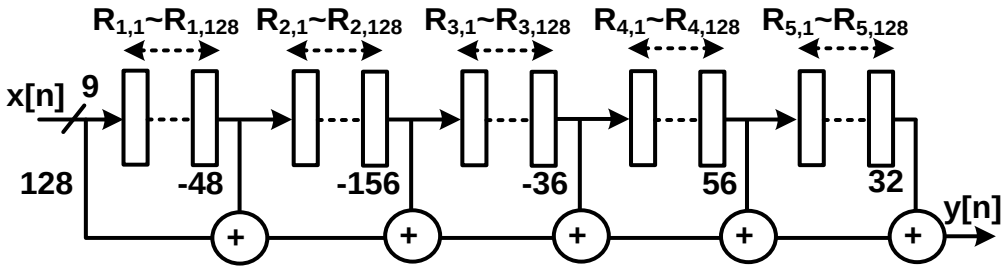


Figure 5-15. Simplified interleaved architecture for 128-channel detection.

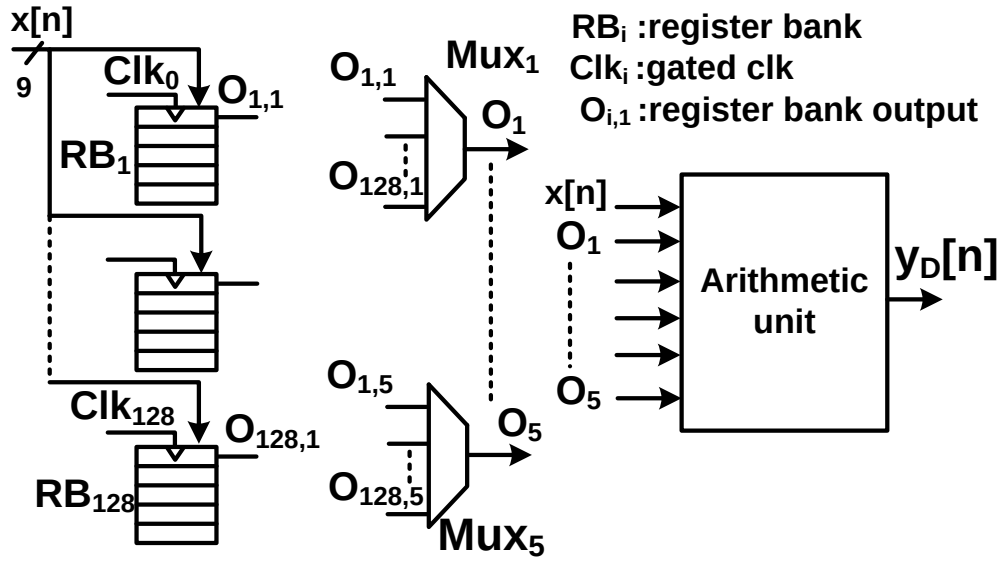


Figure 5-16. Parallel implementation of the required memory in 128-channel detection. RB_i blocks indicate the register banks for each channel. Each channel is controlled by its specified clock signal named as Clk_i.

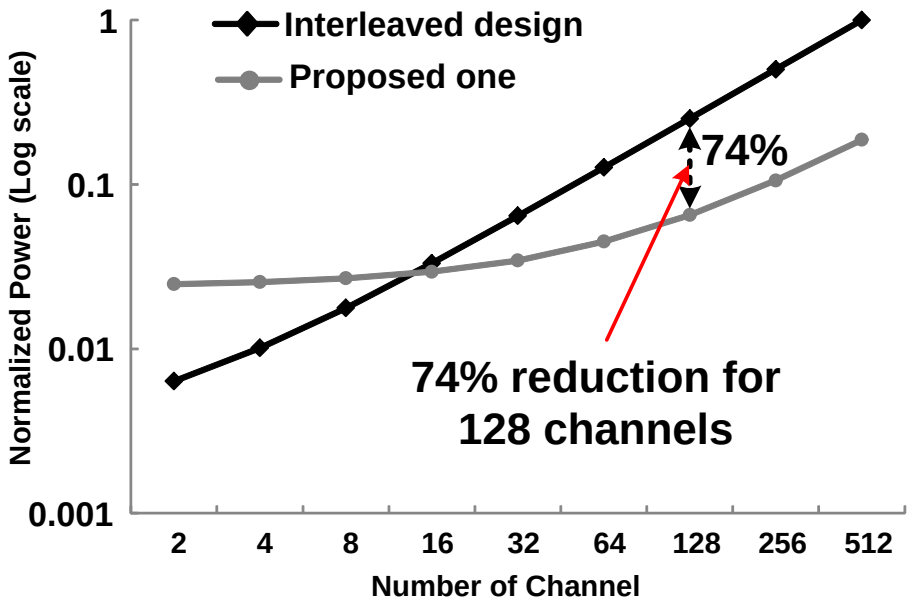


Figure 5-17. Power comparison between interleaved and parallel architecture which is normalized to maximum value.

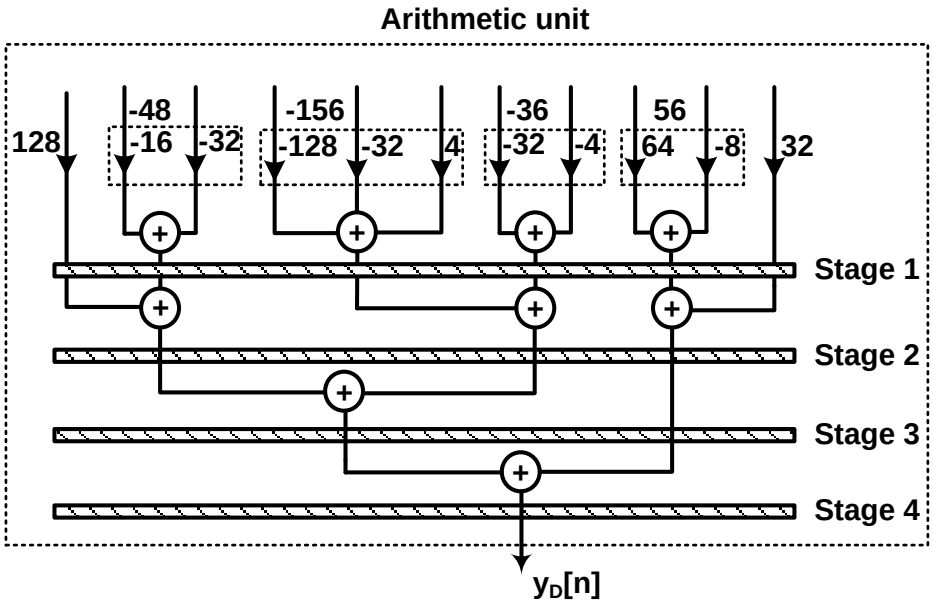


Figure 5-18. Arithmetic unit implemented by converting filter coefficients to power-of-two values and using left shift as a multiplier.

5.5.3 SRAM Memory

A fully-customized 8T SRAM were designed and implemented to support the 128-channel spike-sorting core. Since each channel may contain up to 6 spike clusters, each cluster mean requires 40 bits storage (4 features, each 13 bits), a total capacity of 39 kbits is required. Figure 5-19 shows the block diagram of the implemented SRAM. 8T cell was chosen due to its suitability for low-voltage operation. Note that the memory is written once after training and read only when a spike is detected. Therefore, in this implementation memory active power is insignificant. Consequently, special care was taken to minimize leakage power of the memory.

Apparently HVT (high threshold voltage) device is the first choice to reduce leakage power but access speed will be greatly reduced around the threshold voltage. To overcome this, HVT devices are used for the 6T structure while SVT (standard threshold voltage) devices are used at the read port and peripheral circuits. Furthermore, as mentioned before, SRAM operates under $CLKS$ therefore its speed constraint is only of secondary concern. Our simulation (and later measurement) shows that its maximum operating frequency at 0.5 V and 0.4 V are 4 MHz and 500 KHz, respectively. This gives a safe margin as our intended $fCLKS$ is only 25 KHz.

Another concern of the SRAM at sub- and near-threshold design is the cell stability margin under half-select condition. This happens even with 8T cell when some cells on a row are written while the rest are half-selected, i.e. their world-line is turned on but their data are meant to stay unchanged. Figure 5-19(b) shows the Monte-Carlo simulation of our 8T cell under half-select condition at 0.2 V and 0.3 V supply condition. It is

noticeable that when $V_{DD} = 0.2\,V$, some cells flipped and are deemed to be unstable. However, when V_{DD} is raised to 0.3 V, none of them fails. As the intended operating voltage of the whole chip is around 0.5 V, this confirms that the cell is stable enough to overcome the half-select disturb issues.

A similar memory is separately implemented on the same die to check the performance and reliability of the design. Measurement results as depicted in Figure 5-19(c) confirms that the proposed SRAM operates correctly down to 0.36 V and a maximum operating frequency of 250 KHz. Figure 5-19(d) shows memory maximum operating frequencies at different supply voltages.

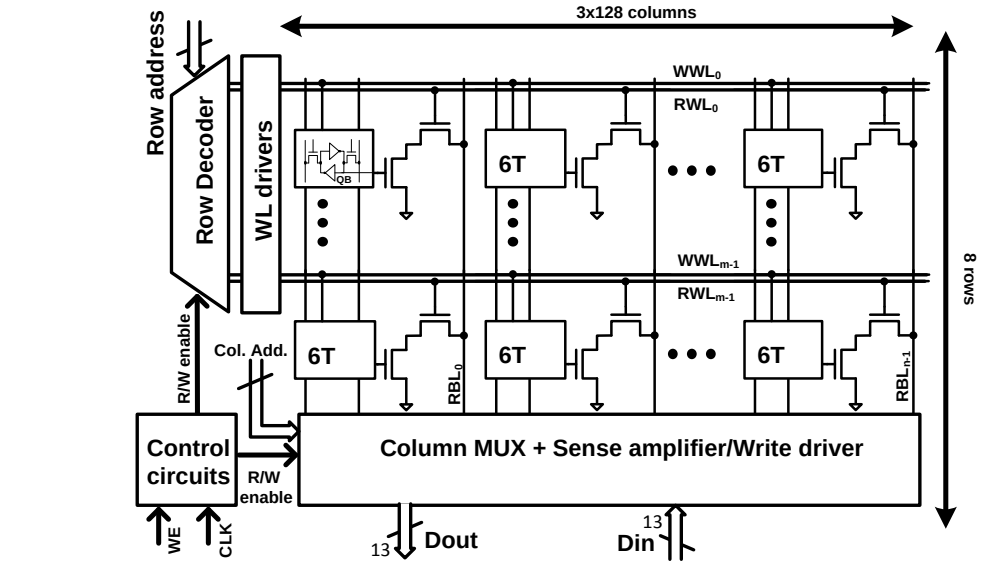


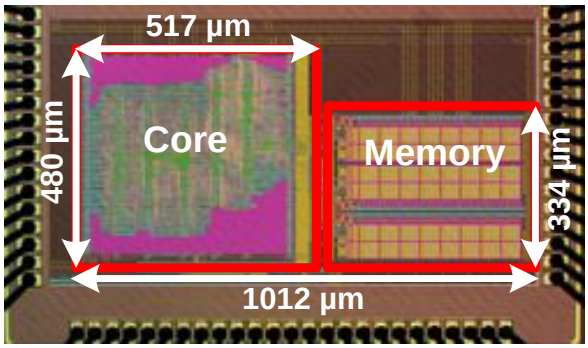
Figure 5-19. (a) Block diagram of the customized 8T SRAM (b) 1000-iteration Monte-Carlo simulation of the memory cell to verify its stability under half-select condition. (c) Measured consecutive write and read waveforms of the proposed SRAM at 0.36 V, room temperature. (d) Maximum operating frequency of the SRAM at different supply voltages (Room temperature).

5.5.4 Power/Speed Optimization

During synthesis, standard library choice greatly impacts the overall performance of the chip. For example, LVT (low threshold voltage) device can easily satisfy speed requirement but may incur more leakage. As a result, we may need to push the operating voltage to deep sub-threshold to maintain the same leakage level as SVT and HVT While it is feasible to do so, operating at these voltages incurs unprecedented amount of process variations and thus timing errors in digital circuits. On the other hand, HVT devices can reduce leakage but requires higher supply voltage to operate. Note that in our design, CLK and $CLKS$ frequency is predetermined by the analog front end and hence are unlikely to be scalable. As a result, using HVT will significantly increase the dynamic power consumed by the detection circuit. Finally, SVT devices are chosen as they provide a good balance between speed performance dynamic and leakage power.

5.6 Measurement Result

The 128-channel spike sorting chip has been fabricated in 65-nm CMOS process technology. The chip microphotograph is shown in Figure 5-20, alongside with chip specification. It has two distinct parts which are digital processing core including the "Detector", "FE, DR", Classifier and Training engine and the fully customized 8T SRAM. The typical ADC sampling frequency for spike signal recording is 25 KHz [7, 9, 104] and therefore the operating frequency of 128 channels is set to 3.2 MHz. A Xilinx ZYNQ-7000 SOC video and imaging kit with Xilinx FMC XM105 debug cards is used to test the chip as shown in Figure 5-21. The logic analyzer is used to capture output data. Digitized neural signals [105] is streamed out through ZYNQ board to the chip and finally results are recorded by the logic analyzer for later verification in MATLAB.



CHIP SPECIFICATION

Technology	65-nm
Total Area ($mm^{2)}$	0.414
Core voltage	0.54
Number of Ch.	128
Total Power (µW)	22.4
Input Frequency(MHz)	3.2
Spike Data Reduction	257

Figure 5-20. 128-channel spike sorting chip micrograph.

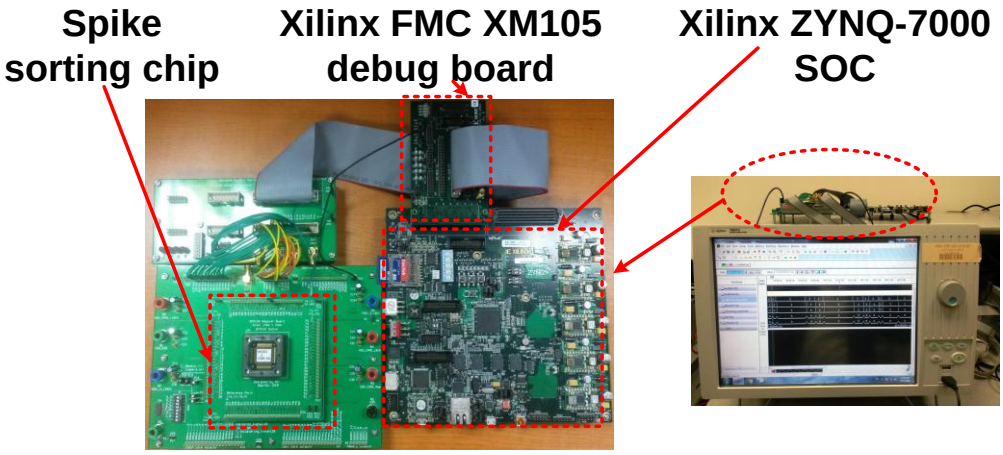


Figure 5-21. Testing setup using ZYNQ-7000 SOC video and imaging kit with the Xilinx FMC XM105 debug card.

During classification, the designed chip can be configured to operate in three different modes: Detection only, (2) Detection and FE; and (3) Detection, FE and Classification. Figure 5-22 shows the measurement waveforms during different modes. As shown in Figure 5-22, in Detection only mode, a spike is detected and the detection flag is enabled accordingly. In Detection/FE mode, in addition to a detection flag, a FE flag is also enabled showing the features are calculated and streamed out. Finally in the third mode, neuron IDs are also available corresponding to each detected spike. These three

modes of operations provide user flexibility to control over data rate and power consumption. Figure 5-23 shows a short window capture of the measurement result from the logic analyzer.

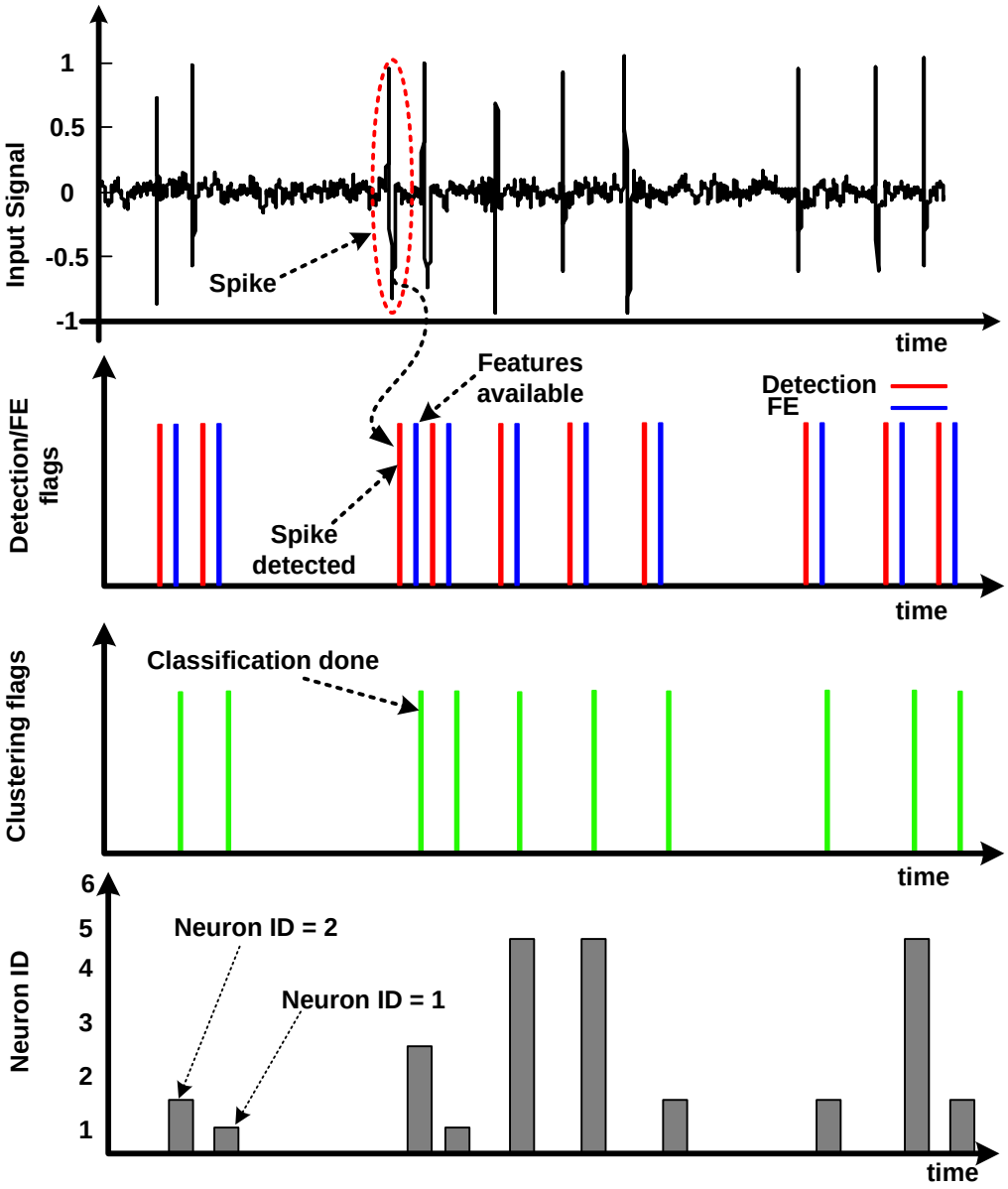


Figure 5-22. The main outputs of the 12-channel spike sorting chip for a sampled input.

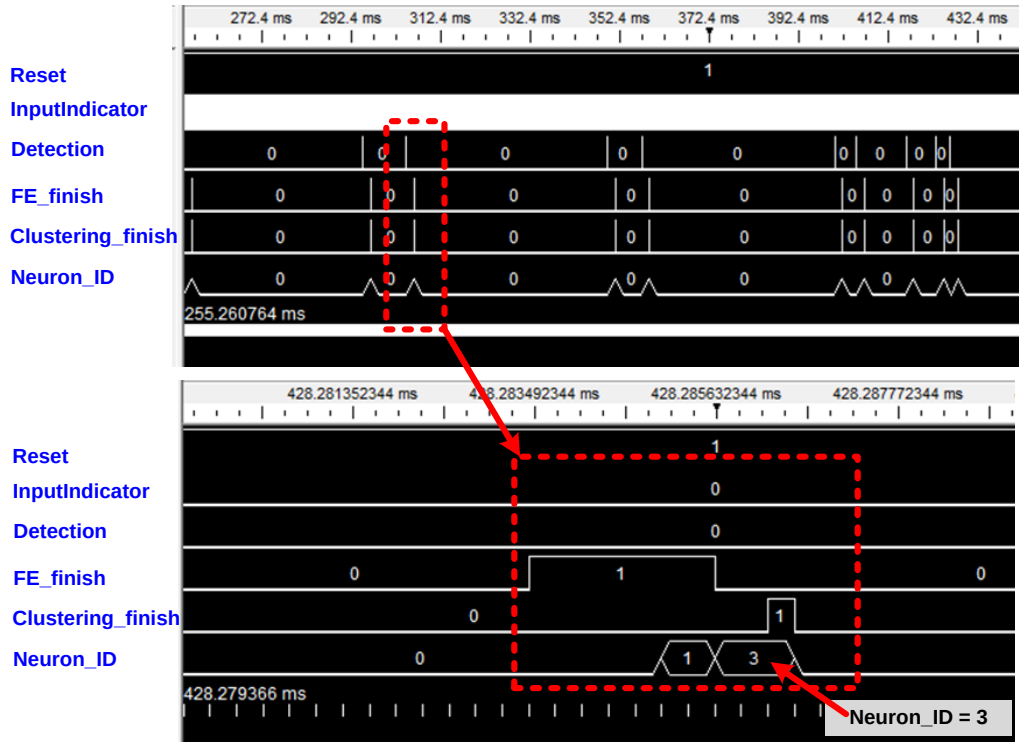


Figure 5-23. The measurement results shown by logic analyzer for a sampled input.

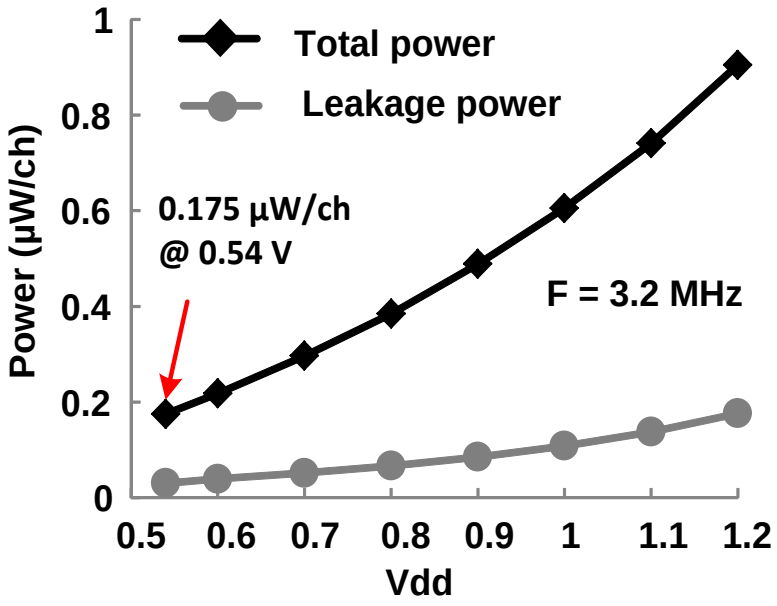


Figure 5-24. Power consumption per channel.

After verifying the chips functionality, its power consumption at different voltages is recoded, keeping the same clock frequency. Its minimum operating voltage that can support 3.2 MHz is 0.54V. Figure 5-24 shows the power consumption per channel when supply voltage changes from 1.2 V to 0.54 V. Unlike existing designs in which leakage dominates the total power consumption at low voltages, our designs' leakage is successfully suppressed thanks to the customized low-leakage SRAM. As illustrated in Figure 5-25, 82% of the total power is dynamic power and the rest is leakage power. Detector contributes 67% of the total power because it is active almost all the time.

Table I compares our chip with existing state-of-the-arts. [9] is the first design performing the spike sorting flow for 16 channels. In comparison to [9], power consumption and area per channel is reduced by 26× and 23×, respectively.

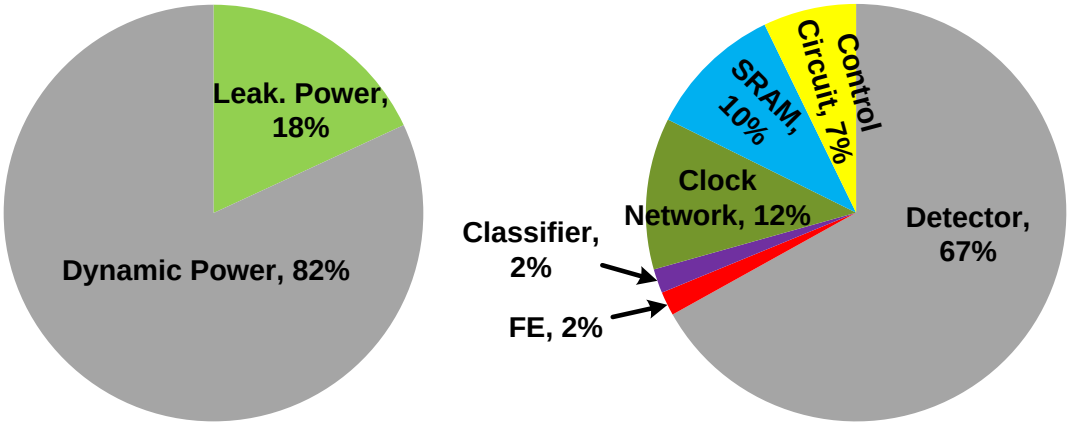


Figure 5-25. The power breakdown of 128-channel spike sorting chip at 0.6 V and 3.2 MHz.

Table 5-1. Performance Comparison

Reference	[7]	[9]	[99]	[104]	**This work**
No. of Chs.	64	16	128	64	**128**
Detection	Y	Y	Y	Y	**Y**
Feature Extraction	Y	N	Y	Y	**Y**
Clustering	N	Y	N	N	**Y**
Core Voltage (V)	0.55	0.27	3	0.25	**0.54**
Power (μW/Ch)	2.03	4.68	75	0.46	**0.175**
Area (mm^2/Ch)	0.06	0.07	0.11	0.03	**0.003**
Power Density (μW/mm^2)	33.8	66.8	682	15.33	**58.33**
Process (nm)	90	65	500	65	**65**

5.7 Conclusion

This chapter has presented a 128-channel spike sorting chip to analyze the brain recorded activities. This design proposes a new on-line spike sorting algorithm to improve the clustering accuracy and to lower the power and area per channel. It includes three main parts which are new detection, feature extraction and improved k-means clustering algorithms. In hardware implementation, a time-multiplexing detector and SRAM is proposed to reduce the power and area. Measurement results show that the proposed design operates down to 0.54 V and reduces the power and area by 26× and 23×. The averaged clustering accuracy is between 72% ~ 86%.

Chapter 6. Conclusions and Future Works

In this thesis, several architecture- and circuit-level techniques have been developed to address the main issues of power and area in designing the ultra-low power processor. These processors are frequently applied in biomedical and IoT applications where both power and area are very critical. The general architecture-level techniques named as FERDC and FAERDC are introduced to reduce the power and area in FIFO which is extensively used in signal and image filtering. After that, FAERDC is exploited in hardware implementation of LP and the measurement and testing results verify the functionality as well as power and area reduction. Then, CT is comprehensively studied from hardware point of view and verified through FPGA implementation. CT in fact consists of LP and DFB and it can be considered as a hardware accelerator to perform complex image processing applications. Finally, a complete spike sorting chip has been designed and proposed using the various architectures and circuit level techniques.

In Chapter 2 and Chapter 3, FERDC and FEARDC techniques are proposed to reduce the area and power consumption of the FIFO by utilizing the spatial correlation between neighbouring pixels and performing error-reduced data compression together with nonadaptive and adaptive quantization to minimize the MSE. For future work, a reconfigurable FIFO can be designed in which the cell bit-width can be voluntarily adjusted with respect to applications required precision. Moreover, it gives more flexibility to error-resilient applications to reduce the power. In addition to that, FIFO core can be designed in more customized approach to reduce the area and power more significantly.

In chapter 3, a power and area efficient Laplacian Pyramid processing engine is presented for image/video processing applications. LPPE applies architecture-level and circuit-level techniques to reduce the power and area as well as improving the MSE. The proposed architecture optimizes the number of arithmetic operations and removes all redundant operations. Besides, it uses FEARDC to significantly reduce the power and area which are mainly dominated by FIFOs. For future work, the increase of *fps* can be investigated by using more parallel architecture. Currently, the proposed architecture halves the input image processing time as compared to the conventional approach. The more parallel architecture further reduces the operating frequency by the power of two in the price of larger area.

In chapter 4, new hardware architecture for CT is proposed. CT consists of LP and DFB. LP architecture was studied in Chapter 3. For DFB, a memory-based architecture along with various address generators are employed to get the wedge-shaped subbands. For future work, CT can be developed to do the reconfigurable computing on input image for various applications, requiring the block-based image processing. To do so, the LP and DFB should be adaptively adjusted to the block size in order to reduce the power.

Chapter 5 has presented a 128-channel spike sorting chip to analyze the brain recorded activities. The proposed design operates down to 0.54 V and reduces the power and area by 26× and 23×. The averaged spikes classification accuracy is between 72% ~

86%. For future work, some vital contributions are possible to improve the spike sorting chip. For instance, in the current chip, once a spike is detected, the chip is dedicated to spike classification of the corresponding channel and other channels are skipped within this period. This one can be addressed by employing more processing unit to stop losing any concurrent spike. To do so, the extensive study on multi-electrode probe should be made to figure out the statistics of concurrent occurrence of spikes on different channels.

Author's Publications

Journal Papers

- A. T. Do, **S. M. A. Zeinolabedin**, D. Jeon, D. Sylvester, and T. T. Kim, "A 128-Channel Spike Sorting Processor Featuring 0.175 µW and 0.0033 mm^2 per Channel in 65-nm CMOS," IEEE J. Solid-State Circuits, in Submission, 2016.
- **S. M. A. Zeinolabedin**, J. Zhou, and T. T. Kim, "A Power and Area Efficient Ultra-low Voltage Laplacian Pyramid Processing Engine with Adaptive Data Compression," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 63, no. 10, pp. 1690-1700, 2016.
- **S. M. A. Zeinolabedin**, J. Zhou, X. Liu, and T. T. H. Kim, "An Area- and Energy-Efficient FIFO Design Using Error-Reduced Data Compression and Near-Threshold Operation for Image/Video Applications," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 23, no. 11, pp. 2408-2416, Nov. 2015.
- **S. M. A. Zeinolabedin**, N. Karimi, S. Samavi, and T. T.-H. Kim, "Structuring of Contourlet Transform for Pipeline-Based Implementation," Circuits, Systems, and Signal Processing, vol. 35, no. 3, pp. 953-976, 2015.

Conference Papers

- A. T. Do, **S. M. A. Zeinolabedin**, and T. T. Kim, "A 0.3 pW/Access 8T Data-Aware SRAM Utilizing Column-based Data Encoding for Ultra-Low Power Applications," in IEEE Asian Solid-State Circuits Conference, Toyama, Japan, Nov. 2016., pp. 173-176.
- **S. M. A. Zeinolabedin**, A. T. Do, D. Jeon, D. Sylvester, and T. T. Kim, "A 128-Channel Spike Sorting Processor Featuring 0.175 µW and 0.0033 mm2 per Channel in 65-nm CMOS," in IEEE Symp. VLSI Circuits, Hawaii, USA, Jun. 2016.
- **S. M. A. Zeinolabedin**, J. Zhou, X. liu, and T. T. Kim, "A 0.5V Power and Area Efficient Laplacian Pyramid Processing Engine using FIFO with Adaptive Data Compression," in IEEE European Solid-State Circuits Conference (ESSCIRC), Graz, Austria, Sep. 2015, pp. 104-107.
- **S. M. A. Zeinolabedin**, J. Zhou, X. Liu, and T. T. Kim, "An Area- and Power-Efficient FIFO with Error-Reduced Data Compression for Image/Viedo Processing," in Proc. IEEE Int. Symp. Circuits Syst. (ISCAS), , Melbourne, Australia, Jun. 2014, pp. 2277-2280.
- **S. M. A. Zeinolabedin**, A. T. Do, K. S. Yeo, and T. T.-H. Kim, "Design of a hybrid neural spike detection algorithm for implantable integrated brain circuits," in Proc. IEEE Int. Symp. Circuits Syst. (ISCAS), Lisbon, Portugal, May. 2015, pp. 794-797.

Student Design Contest

- **S. M. A. Zeinolabedin,** A. T. Do, D. Jeon, D. Sylvester, and T. Kim, "A 128-Channel Spike Sorting Processor Featuring 0.175 µW and 0.0033 mm_2 per Channel in 65-nm CMOS," IEEE International Symposium on Low Power Electronics and Design (**ISLPED**), **Student Design Contest Winner**, August 2016.

References

[1] W. Al-Atabany, B. McGovern, K. Mehran, R. Berlinguer-Palmini, and P. Degenaar, "A Processing Platform for Optoelectronic/Optogenetic Retinal Prosthesis," *IEEE Trans. Biomed. Eng.*, vol. 60, no. 3, pp. 781-791, Mar. 2013.

[2] R. Rithe, P. Raina, N. Ickes, S. V. Tenneti, and A. P. Chandrakasan, "Reconfigurable processor for energy-scalable computational photography," in *IEEE ISSCC Dig. Tech. Papers*, Feb. 2013, pp. 164-165.

[3] H. C. Tsung, M. Tikekar, C. Juvekar, V. Sze, and A. Chandrakasan, "A 249Mpixel/s HEVC video-decoder chip for Quad Full HD applications," in *IEEE ISSCC Dig. Tech. Papers*, Feb. 2013, pp. 162-163.

[4] Y. J. Chen, S. Y. Chuang, C. Y. Hung, C. H. Hsu, C. M. Chang, and S. Y. Chien, and L. G. Chen, "A 130.3mW 16-core mobile GPU with power-aware approximation techniques," in *Proc. IEEE Asian Solid State Conf. (A-SSCC)*, Nov. 2013, pp. 297-300.

[5] T. H. Khan and K. A. Wahid, "Low Power and Low Complexity Compressor for Video Capsule Endoscopy," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 21, no. 10, pp. 1534-1546, Oct. 2011.

[6] A. M. Sodagar, G. E. Perlin, Y. Yao, K. Najafi, and K. D. Wise, "An Implantable 64-Channel Wireless Microsystem for Single-Unit Neural Recording," *IEEE J. Solid-State Circuits*, vol. 44, no. 9, pp. 2591-2604, Sept. 2009.

[7] V. Karkare, S. Gibson, and D. Markovic, "A 130-uW, 64-Channel Neural Spike-Sorting DSP Chip," *IEEE J. Solid-State Circuits*, vol. 46, no. 5, pp. 1214-1222, May. 2011.

[8] S. Gibson, J. W. Judy, and D. Markovic, "Spike Sorting: The First Step in Decoding the Brain: The first step in decoding the brain," *IEEE Signal Process. Mag.*, vol. 29, no. 1, pp. 124-143, Jan. 2012.

[9] V. Karkare, S. Gibson, and D. Markovic, "A 75-uW, 16-Channel Neural Spike-Sorting Processor With Unsupervised Clustering," *IEEE J. Solid-State Circuits*, vol. 48, no. 9, pp. 2230-2238, Sept. 2013.

[10] V. Venkatachalam and M. Franz, "Power reduction techniques for microprocessor systems," *ACM Comput. Surv.*, vol. 37, no. 3, pp. 195-237, Sep. 2005.

[11] D. Jeon, Y. Kim, I. Lee, Z. Zhang, D. Blaauw, and D. Sylvester, "A 470mV 2.7mW feature extraction-accelerator for micro-autonomous vehicle navigation in 28nm CMOS," in *IEEE ISSCC Dig. Tech. Papers*, Feb. 2013, pp. 166-167.

[12] Z. Kuchar and J. Doubrava. (2004). *Human Eye and Artificial Retina Implants*. Available: http://remort.wz.cz/retina/

[13] I. Second Sight Medical Products. (2014). *Argus II*. Available: http://2-sight.eu/en/home-en

[14] X. Zou, G. Shi, Y. Jin, and Y. Zheng, "Extraocular image processing for retinal prosthesis based on DSP," in *IEEE International Conference on Nano/Micro Engineered and Molecular Systems*, 2009, pp. 563-566.

[15] R. C. Gonzalez and R. E. Woods, *Digital image processing / Rafael C. Gonzalez, Richard E. Woods*: Harlow : Prentice Hall, 3rd ed., 2008.

[16] W. K. Pratt, *Digital image processing [electronic resource] : PIKS Scientific inside / William K. Pratt*: Hoboken, N.J. : Wiley-Interscience, 2007.

[17] M. N. Do and M. Vetterli, "The contourlet transform: an efficient directional multiresolution image representation," *IEEE Trans. Image Process.*, vol. 14, no. 12, pp. 2091-2106, Dec. 2005.

[18] D. Jeon, N. Ickes, P. Raina, H.-C. Wang, D. Rus, and A. Chandrakasan, "24.1 A 0.6V 8mW 3D vision processor for a navigation device for the visually impaired," in *IEEE ISSCC Dig. Tech. Papers*, Feb. 2016, pp. 416-417.

[19] P. Ou, J. Zhang, H. Quan, Y. Li, M. He, Z. Yu, *et al.*, "A 65nm 39GOPS/W 24-core processor with 11Tb/s/W packet-controlled circuit-switched double-layer network-on-chip and heterogeneous execution array," in *IEEE ISSCC Dig. Tech Papers*, Feb. 2013, pp. 56-57.

[20] S. M. A. Zeinolabedin, N. Karimi, and S. Samavi, "Low computational complexity hardware implementation of Laplacian Pyramid," in *Proc. IEEE 18th Iranian Conf. Electr. Eng.*, May. 2010, pp. 465-470.

[21] C. M. Lopez, A. Andrei, S. Mitra, M. Welkenhuysen, W. Eberle, C. Bartic, *et al.*, "An implantable 455-active-electrode 52-channel CMOS neural probe," in *IEEE ISSCC Dig. Tech. Papers*, Feb. 2013, pp. 288-289.

[22] C. M. Lopez, S. Mitra, J. Putzeys, B. Raducanu, M. Ballini, A. Andrei, *et al.*, "22.7 A 966-electrode neural probe with 384 configurable channels in 0.13um SOI CMOS," in *IEEE ISSCC Dig. Tech. Papers*, Feb. 2016, pp. 392-393.

[23] S. M. A. Zeinolabedin, J. Zhou, X. Liu, and T. T. Kim, "An Area- and Power-Efficient FIFO with Error-Reduced Data Compression for Image/Viedo Processing," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS),* , Melbourne, Australia, Jun. 2014, pp. 2277-2280.

[24] S. M. A. Zeinolabedin, J. Zhou, X. Liu, and T. T. H. Kim, "An Area- and Energy-Efficient FIFO Design Using Error-Reduced Data Compression and Near-Threshold Operation for Image/Video Applications," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 11, pp. 2408-2416, Nov. 2015.

[25] S. M. A. Zeinolabedin, J. Zhou, X. liu, and T. T. Kim, "A 0.5V Power and Area Efficient Laplacian Pyramid Processing Engine using FIFO with Adaptive Data Compression," in *IEEE European Solid-State Circuits Conference (ESSCIRC)* Graz, Austria, Sep. 2015, pp. 104-107.

[26] S. M. A. Zeinolabedin, J. Zhou, and T. T. Kim, "A Power and Area Efficient Ultra-low Voltage Laplacian Pyramid Processing Engine with Adaptive Data Compression," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 63, no. 10, pp. 1690-1700, 2016.

[27] S. M. A. Zeinolabedin, N. Karimi, S. Samavi, and T. T.-H. Kim, "Structuring of Contourlet Transform for Pipeline-Based Implementation," *Circuits, Systems, and Signal Processing*, vol. 35, no. 3, pp. 953-976, 2015.

[28] S. M. A. Zeinolabedin, A. T. Do, K. S. Yeo, and T. T.-H. Kim, "Design of a hybrid neural spike detection algorithm for implantable integrated brain circuits," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Lisbon, Portugal, May. 2015, pp. 794-797.

[29] S. M. A. Zeinolabedin, A. T. Do, D. Jeon, D. Sylvester, and T. T. Kim, "A 128-Channel Spike Sorting Processor Featuring 0.175 µW and 0.0033 mm2 per Channel in 65-nm CMOS," in *IEEE Symp. VLSI Circuits, ,* Hawaii, USA, Jun. 2016.

[30] A. T. Do, S. M. A. Zeinolabedin, D. Jeon, D. Sylvester, and T. T. Kim, "A 128-Channel Spike Sorting Processor Featuring 0.175 µW and 0.0033 mm2 per Channel in 65-nm CMOS," *IEEE J. Solid-State Circuits*, in Submission, 2016.

[31] C. Zhang, C. Wang, and M. O. Ahmad, "A Pipeline VLSI Architecture for Fast Computation of the 2-D Discrete Wavelet Transform," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 59, no. 8, pp. 1775-1785, Aug. 2012.

[32] R. Rithe, C. Chih-Chi, and A. P. Chandrakasan, "Quad Full-HD Transform Engine for Dual-Standard Low-Power Video Coding," *Solid-State Circuits, IEEE Journal of*, vol. 47, pp. 2724-2736, 2012.

[33] T. T. Kim, J. Liu, J. Keane, and C. H. Kim, "Circuit techniques for ultra-low power subthreshold SRAMs," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May. 2008, pp. 2574-2577.

[34] D. Salomon, *Data Compression [electronic resource] : The Complete Reference / by David Salomon*, Fourth Edition. ed.: London : Springer-Verlag London Limited, 2007.

[35] D. G. Bailey, *Design for embedded image processing on FPGAs [electronic resource]* Singapore : John Wiley & Sons (Asia), 2011.

[36] C.-H. Tsai, H.-T. Wang, C.-L. Liu, Y. Li, and C.-Y. Lee, "A 446.6K-gates 0.55-1.2V H.265/HEVC decoder for next generation video applications," in *Solid-State Circuits Conference (A-SSCC), 2013 IEEE Asian*, 2013, pp. 305-308.

[37] K. J.-W., S. J., L. S., and P. I.-C., "Tiled Interleaving for Multi-Level 2-D Discrete Wavelet Transform," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS), New Orleans, LA* May. 2007, pp. 3984-3987.

[38] S. J. and P. I.-C., "Novel pipelined DWT architecture for dual-line scan," in *IEEE Int. Sym. Circuits Syst. (ISCAS), Taipei*, May. 2009, pp. 373-376.

[39] S. L. Chen, "VLSI Implementation of a Low-Cost High-Quality Image Scaling Processor," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 60, no. 1, pp. 31-35, Jan. 2013.

[40] B. K. Mohanty, P. K. Meher, S. Al-Maadeed, and A. Amira, "Memory Footprint Reduction for Power-Efficient Realization of 2-D Finite Impulse Response Filters," *IEEE Trans. Circuit Syst. I, Reg. Papers*, vol. 61, no. 1, pp. 120-133, Jan. 2014.

[41] L. Chang, D. J. Frank, R. K. Montoye, S. J. Koester, B. L. Ji, P. W. Coteus, *et al.*, "Practical Strategies for Power-Efficient Computing Technologies," *Proceedings of the IEEE*, vol. 98, no. 2, pp. 215-236, Feb. 2010.

[42] A. Wang and A. Chandrakasan, "A 180-mV subthreshold FFT processor using a minimum energy design methodology," *IEEE J. Solid-State Circuits*, vol. 40, no. 1, pp. 310-319, Jan. 2005.

[43] L. Chang and W. Haensch, "Near-threshold operation for power-efficient computing? It depends..." in *Proc. IEEE Design Autom. Conf. (DAC)*, , 2012, pp. 1155-1159.

[44] D. Jeon, M. Seok, Z. Zhang, D. Blaauw, and D. Sylvester, "Design Methodology for Voltage-Overscaled Ultra-Low-Power Systems," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 59, no. 12, pp. 952-956, Dec. 2012.

[45] X. Liu, J. Zhou, X. Liao, C. Wang, J. Luo, M. Madihian, *et al.*, "Ultra-low-energy near-threshold biomedical signal processor for versatile wireless health monitoring," in *Proc. IEEE Asian Solid State Circuits Conf. (A-SSCC)*, Nov. 2012, pp. 381-384.

[46] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image coding using wavelet transform," *IEEE Trans. Image Process.*, vol. 1, no. 2, pp. 205-220, Feb. 1992.

[47] U. o. Granada. (Last Updated: 2014). *Test Images*. Available: http://decsai.ugr.es/cvg/dbimagenes/

[48] Z. Chengjun, C. Wang, and M. O. Ahmad, "A Pipeline VLSI Architecture for Fast Computation of the 2-D Discrete Wavelet Transform," *IEEE Trans. Circuits and Syst. I, Reg. Papers*, vol. 59, no. 8, pp. 1775-1785, Aug. 2012.

[49] J. Diaz, E. Ros, F. Pelayo, E. M. Ortigosa, and S. Mota, "FPGA-based real-time optical-flow system," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 16, no. 2, pp. 274-279, Feb. 2006.

[50] M. Genovese and E. Napoli, "ASIC and FPGA Implementation of the Gaussian Mixture Model Algorithm for Real-Time Segmentation of High Definition Video," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 22, no. 3, pp. 537-547, Mar. 2014.

[51] J. Kwon, I. J. Chang, I. Lee, H. Park, and J. Park, "Heterogeneous SRAM Cell Sizing for Low-Power H.264 Applications," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 59, no. 10, pp. 2275-2284, Oct. 2012.

[52] T. H. Tsai and C. N. Liu, "Low-Power System Design for MPEG-2/4 AAC Audio Decoder Using Pure ASIC Approach," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 56, no. 1, pp. 144-155, Jan. 2009.

[53] P. Burt and E. Adelson, "The Laplacian Pyramid as a Compact Image Code," *IEEE Trans. Commun.*, vol. 31, no. 4, pp. 532-540, Apr. 1983.

[54] S. Chen, Q. Guo, H. Leung, and E. Bosse, "A Maximum Likelihood Approach to Joint Image Registration and Fusion," *IEEE Trans. Image Process.*, vol. 20, no. 5, pp. 1363-1372, May. 2011.

[55] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91-110, Nov. 2004.

[56] Y. Song, K. Gao, G. Ni, and R. Lu, "Implementation of real-time Laplacian pyramid image fusion processing based on FPGA," 2007, pp. 683316-683316-8.

[57] V. Popovic, K. Seyid, A. Schmid, and Y. Leblebici, "Real-time hardware implementation of multi-resolution image blending," in *IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, May. 2013, pp. 2741-2745.

[58] M. N. Do and M. Vetterli, "Framing pyramids," *IEEE Trans. Signal Process.*, vol. 51, no. 9, pp. 2329-2342, Sep. 2003.

[59] C. Zhao, Q. Ding, and J. Li, "The Performance Analysis of Image Fusion Algorithm," in *IEEE Int. Symp. on Computational Intelligence and Design*, Oct. 2008, pp. 83-86.

[60] M. A. Mahraz, J. Riffi, and H. Tairi, "Motion estimation using the fast and adaptive bidimensional empirical mode decomposition," *Journal of Real-Time Image Processing*, vol. 9, no. 3, pp. 491-501, Jun. 2012.

[61] P. J. Burt, "Multiresolution Pyramid Architectutres for Real-Time Motion Analysis," in *IAPR Workshop on Machine Vision Applications*, Tokyo, Japan, 1990.

[62] G. S. Van der Wal and P. J. Burt, "A VLSI pyramid chip for multiresolution image analysis," *International Journal of Computer Vision*, vol. 8, no. 3, pp. 177-189, Sep. 1992.

[63] M. A. Nuno-Maganda, M. O. Arias-Estrada, and C. Feregrino-Uribe, "Three video applications using an FPGA based pyramid implementation: Tracking, Mosaics and Stabilization," in *Proc. IEEE Int. Conf. on FPL*, Dec. 2003, pp. 336-339.

[64] O. Sims and J. Irvine, "An FPGA Implementation of Pattern-Selective Pyramidal Image Fusion," in *IEEE Int. Conf. Field Programmable Logic and Applications*, Madrid , 2006, pp. 1-4.

[65] Y. Song, K. Gao, G. Ni, and R. Lu, "Implementation of real-time Laplacian pyramid image fusion processing based on FPGA," *Proceedings of the SPIE*, pp. 683316-683316, 2007.

[66] J. Zhang, Y. Han, B. Chang, Y. Yuan, Y. Qian, and Y. Qiu, "Real-time color image fusion for infrared and low-light-level cameras," in *Proc. SPIE*, Jul. 2009, pp. 73833B-73833B-7.

[67] G. Van der Wal, M. Hansen, and M. Piacentino, "The Acadia vision processor," in *Proc. IEEE Int. Workshop on Computer Architectures for Machine Perception*, 2000, pp. 31-40.

[68] G. S. Van der Wal, "Technical overview of the Sarnoff Acadia II vision processor," in *Proc. SPIE*, May. 2010, pp. 77100O-77100O-12.

[69] F. Barranco, M. Tomasi, M. Vanegas, J. Diaz, S. Granados, and E. Ros, "Hierarchical architecture for motion and depth estimations based on color cues," *Journal of Real-Time Image Processing*, pp. 1-18, 2012/11/20 2012.

[70] P. P. Vaidyanathan, *Multirate systems and filter banks* Englewood Cliffs, N.J. : Prentice Hall, 1993.

[71] K. Danckaert, K. Masselos, F. Cathoor, H. J. De Man, and C. Goutis, "Strategy for power-efficient design of parallel systems," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 7, no. 2, pp. 258-265, Jun. 1999.

[72] A. K. Oudjida and N. Chaillet, "Radix-2^r Arithmetic for Multiplication by a Constant," *IEEE Trans. Circuits Syst. II, Exp. Briefs,* vol. 61, no. 5, pp. 349-353, May. 2014.

[73] A. K. Oudjida, A. Liacha, M. Bakiri, and N. Chaillet, "Multiple Constant Multiplication Algorithm for High-Speed and Low-Power Design," *IEEE Trans. on Circuits Syst. II, Exp. Briefs,* vol. 63, no 2, pp. 176-180, Feb. 2016.

[74] J. Zhou, C. Wang, X. Liu, X. Zhang, and M. Je, "An Ultra-Low Voltage Level Shifter Using Revised Wilson Current Mirror for Fast and Energy-Efficient Wide-Range Voltage Conversion from Sub-Threshold to I/O Voltage," *IEEE Trans. Circuits Syst. I: Reg. Papers,* vol. 62, no. 3, pp. 697-706, Mar. 2015.

[75] A. Madisetti and A. N. Willson, "A 100 MHz 2-D 8x8 DCT/IDCT processor for HDTV applications," *IEEE Trans. Circuits Syst. Video Technol.,* vol. 5, no.2, pp. 158-165, 1995.

[76] C. L. Yu, K. Irick, C. Chakrabarti, and V. Narayanan, "Multidimensional DFT IP Generator for FPGA Platforms," *IEEE Trans. Circuits Syst. I: Reg. Papers,,* vol. 58, no. 4, pp. 755-764, 2011.

[77] C. C. Cheng, C. T. Huang, C. Y. Chen, C. J. Lian, and L. G. Chen, "On-Chip Memory Optimization Scheme for VLSI Implementation of Line-Based Two-Dimentional Discrete Wavelet Transform," *IEEE Trans. Circuits Syst. Video Technol.,* vol. 17, no. 7, pp. 814-822, 2007.

[78] J. L. Starck, E. J. Candes, and D. L. Donoho, "The curvelet transform for image denoising," *IEEE Trans. Image Process.,* vol. 11, no. 6, pp. 670-684, 2002.

[79] M. N. Do, "Directional Multiresolution Image Representations," Ph.D. , Department of Communication Systems, Swiss Federal Institute of Technology Lausanne, 2001.

[80] M. A. Akhaee, S. M. E. Sahraeian, and F. Marvasti, "Contourlet-Based Image Watermarking Using Optimum Detector in a Noisy Environment," *IEEE Trans. Image Process.,* vol. 19, no. 4, pp. 967-980, 2010.

[81] D. Tao, X. Li, W. Lu, and X. Gao, "Reduced-Reference IQA in Contourlet Domain," *IEEE Trans. Syst., Man, Cybern.,* vol. 39, no. 6, pp. 1623-1627, 2009.

[82] A. L. Da Cunha, J. Zhou, and M. N. Do, "The Nonsubsampled Contourlet Transform: Theory, Design, and Applications," *IEEE Trans. Image Process.,* vol. 15, no. 10, pp. 3089-3101, 2006.

[83] R. Eslami and H. Radha, "Translation-Invariant Contourlet Transform and Its Application to Image Denoising," *IEEE Trans. Image Process.,* vol. 15, no. 11, pp. 3362-3374, 2006.

[84] H. Sadreazami, M. O. Ahmad, and M. N. S. Swamy, "A Study of Multiplicative Watermark Detection in the Contourlet Domain Using Alpha-Stable Distributions," *IEEE Trans. Image Process.,* vol. 23, no. 10, pp. 4348-4360, 2014.

[85] V. P. Shah, N. H. Younan, and R. L. King, "An Efficient Pan-Sharpening Method via a Combined Adaptive PCA Approach and Contourlets," *IEEE Trans. Geosci. Remote Sens.,* vol. 46, no. 5, pp. 1323-1335, 2008.

[86] A. L. Abbott, P. M. Athanas, L. Chen, and R. L. Elliott, "Finding lines and building pyramids with SPLASH 2," in *Proc. IEEE FPGAs for Custom Computing Machines,* 1994, pp. 155-163.

[87] A. Darabiha, W. J. MacLean, and J. Rose, "Reconfigurable hardware implementation of a phase-correlation stereoalgorithm," *Machine Vision and Applications*, vol. 17, no. 2, pp. 116-132, 2006.

[88] T. Q. Vinh, L. Q. Bao Tri, and N. N. Tai, "A real-time video denoising implementation on FPGA using contourlet transform," in *International Conference on Computing, Management and Telecommunications (ComManTel)*, 2013, pp. 203-207.

[89] T. T. Nguyen and S. Oraintara, "A Class of Multiresolution Directional Filter Banks," *IEEE Trans. Signal Process.*, vol. 55, no. 3, pp. 949-961, 2007.

[90] R. H. Bamberger and M. J. T. Smith, "A filter bank for the directional decomposition of images: theory and design," *IEEE Trans. Signal Process.*, vol. 40, no. 4, pp. 882-893, 1992.

[91] Y. M. Lu and M. N. Do, "Multidimensional Directional Filter Banks and Surfacelets," *IEEE Trans. Image Process.*, vol. 16, no. 4, pp. 918-931, 2007.

[92] M. Vetterli, "Multi-dimensional sub-band coding: Some theory and algorithms," *Signal Processing*, vol. 6, no. 2, pp. 97-112, 1984.

[93] S.-M. Phoong, C. W. Kim, P. P. Vaidyanathan, and R. Ansari, "A new class of two-channel biorthogonal filter banks and wavelet bases," *IEEE Trans. Signal Process.*, vol. 43, no. 3, pp. 649-665, 1995.

[94] M. Martina and G. Masera, "Multiplierless, Folded 9/7-5/3 Wavelet VLSI Architecture," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 54, no. 9, pp. 770-774, 2007.

[95] S. M. A. Zeinolabedin and N. Karimi, "A new quantization algorithm for fir filters coefficients," in *IEEE Conference on Electrical Engineering (ICEE)*, 2012, pp. 1120-1124.

[96] Q. Huynh-Thu and M. Ghanbari, "Scope of validity of PSNR in image/video quality assessment," *Electronics Letters*, vol. 44, no. 13, pp. 800-801, 2008.

[97] *Stratix II Device Handbook*: Altera Co., San Jose, CA, 2007.

[98] A. M. Kamboh and A. J. Mason, "Computationally Efficient Neural Feature Extraction for Spike Sorting in Implantable High-Density Recording Systems," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 21, no. 1, pp. 1-9, Jan. 2013.

[99] M. S. Chae, Z. Yang, M. R. Yuce, L. Hoang, and W. Liu, "A 128-Channel 6 mW Wireless Neural Recording IC With Spike Feature Extraction and UWB Transmitter," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 17, no. 4, pp. 312-321, May. 2009.

[100] G. Santhanam, S. I. Ryu, B. M. Yu, A. Afshar, and K. V. Shenoy, "A high-performance brain–computer interface," *Nature*, vol. 442, no. 7099, pp. 195-198, Jul. 2006.

[101] C. E. Bouton, A. Shaikhouni, N. V. Annetta, M. A. Bockbrader, D. A. Friedenberg, D. M. Nielson, *et al.*, "Restoring cortical control of functional movement in a human with quadriplegia," *Nature*, vol. 533, no. 7602, pp. 247-250, 2016.

[102] H. G. Rey, C. Pedreira, and R. Quian Quiroga, "Past, present and future of spike sorting techniques," *Brain Research Bulletin*, vol. 119, Part B, pp. 106-117, Oct. 2015.

[103] S. Kim, P. Tathireddy, R. A. Normann, and F. Solzbacher, "Thermal Impact of an Active 3-D Microelectrode Array Implanted in the Brain," *IEEE Trans. Neural Syst. Rehabil. Eng.,* vol. 15, no. 4, pp. 493-501, Dec. 2007.

[104] T. T. Liu and J. M. Rabaey, "A 0.25 V 460 nW Asynchronous Neural Signal Processor With Inherent Leakage Suppression," *IEEE J. Solid-State Circuits,* vol. 48, no. 4, pp. 897-906, Apr. 2013.

[105] R. Quian Quiroga, Z. Nadasdy, and Y. Ben-Shaul, "Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering," *Neural Comput.,* vol. 16, no. 8, pp. 1661-1687, Aug. 2004.

[106] I. Obeid and P. D. Wolf, "Evaluation of spike-detection algorithms for a brain-machine interface application," *IEEE Trans. Biomed. Eng.,* vol. 51, pp. 905-911, Jun. 2004.

[107] A. T. Do and K. S. Yeo, "A hybrid NEO-based spike detection algorithm for implantable brain-IC interface applications," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS), Melbourne VIC*, Jun. 2014, pp. 2393-2396.

[108] S. Mukhopadhyay and G. C. Ray, "A new interpretation of nonlinear energy operator and its efficacy in spike detection," *IEEE Trans. Biomed. Eng.,* vol. 45, no. 2, pp. 180-187, Feb. 1998.

[109] S. Gibson, J. W. Judy, and D. Markovic, "Technology-Aware Algorithm Design for Neural Spike Detection, Feature Extraction, and Dimensionality Reduction," *IEEE Trans. Neural Syst. Rehabil. Eng.,* vol. 18, no. 5, pp. 469-478, Jun. 2010.

[110] S. Gibson, J. W. Judy, and D. Markovic, "Comparison of spike-sorting algorithms for future hardware implementation," in *Proc. IEEE Engineering in Medicine and Biology Conf., Vancouver, BC,* Aug. 2008, pp. 5015-5020.

[111] K. D. Harris, D. A. Henze, J. Csicsvari, H. Hirase, and G. Buzsaki, "Accuracy of tetrode spike separation as determined by simultaneous intracellular and extracellular measurements," *J. Neurophysiol.,* vol. 84, pp. 401-414, 2000.

[112] U. Rutishauser, E. M. Schuman, and A. N. Mamelak, "Online detection and sorting of extracellularly recorded action potentials in human medial temporal lobe recordings, in vivo," *Journal of Neuroscience Methods,* vol. 154, no. 1, pp. 204-224, 2006.

[113] C. Pedreira, J. Martinez, M. J. Ison, and R. Quian Quiroga, "How many neurons can we see with current spike sorting algorithms?," *Journal of Neuroscience Methods,* vol. 211, no. 1, pp. 58-65, 2012.