An Efficient Implementation of Radix 3 Multiplier Less Stream Processor Using Verilog HDL

Manoj M.K
Department of Electronics & Communication Engineering,
SDM College of Engineering and Technology,
Dharwad, India

Dr. Bairu K. Saptalakar
Assistant Professor, Department of E&CE,
SDM College of Engineering and Technology
Dharwad, India

***Abstract* - A new Two Dimensional convolution-based filter is presented specifically designed to improve Visual Search applications. It exploits a new radix-3 partitioning method of integer numbers, derived from the weight partition theory, which allows substituting multipliers with simplified floating-point adders, working on 32 bits floating point filter coefficients. The memory organization allows elaborating the incoming data in raster scan order, as those directly provided by an acquisition source, without frame buffers and additional aligning circuitry. This Proposed System Implemented using Verilog HDL and Simulated by Modelsim 6.4 c and Synthesized by Xilinx tool. The proposed system implemented in FPGA Spartan 3 XC3S 200 TQ-144.**

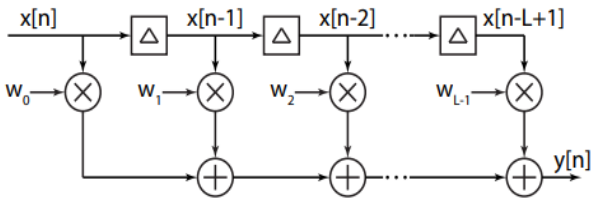
I. INTRODUCTION

Recent advancements in the elaboration of high-quality media contents have promoted an intense research activity for the improvement of filtering operators, whose hardware (HW) complexity is a major concern in applications aimed to pure speed, such as image and video elaboration. Such complexity, indeed, usually relapses in the allocation of a large number of arithmetic operators and a consequent slackening of the overall circuit. The recent literature shows that the aforementioned issue is usually managed either by recurring to the full/partial serialization of the filters and folding techniques or by intervening on the intrinsic complexity of fused multiply adders and multiply accumulators (MAC). Since the former way usually causes a significant reduction of the filter performances , the latter approach remains the most accurate way to achieve a good power, performance, and area (PPA) tradeoff. In this case, the complete removal of the multiplier circuitry is by far the preferred choice of several authors , who recur to fast adders and shifters in place of multipliers, according to the coding of the operands, canonical signed digit (CSD), and modified booth (MB), primarily. The simplification of filtering circuits becomes particularly effective when one of the operands can be reduced to a bounded set of precalculated values, as in the case of predefined filter kernels. In such cases, the distributed arithmetic (DA) method can be successfully applied in order to partition multiplications in simpler shifts and additions. By using memories to store precalculated partial sums, whose number can be reduced by the help of multiple-constant multiplication (MCM) techniques , DA can be, in principle, advantageously used in place of MB and CSD . However, actual performances of DA result from a careful tradeoff between its “natural” bitserial operation and the parallelism by which the partial sums are calculated, which can lead to the excessive increment of mapped physical resources.

Digital signal processing (DSP) is widely used in many applications ranging from audio and speech processing to image and video processing to radar and sonar processing. DSP algorithms are implemented in hardware using computers, specialized processors called digital signal processors (DSPs), field-programmable gate arrays (FPGAs), or custom built hardware called application-specific integrated circuits (ASICs). The choice of hardware platform depends on the requirements imposed by the application. ASICs are the traditional solution to high performance applications, but the high development costs and time-tomarket factors prohibit the deployment of such solutions for certain cases. DSP processors offer high programmability, but the sequential execution nature of their architecture can adversely affect their throughput performance. FPGAs are somewhere in between ASICs and DSPs, offering programmability and improved performance through parallelization. At the core of digital signal processing applications is the digital filter. Digital filters are generally used for: • Separation of signals that have been combined • Restoration of signals that have been distorted • Transform operations Signal separation is used when a signal has been corrupted with noise or other forms of interference or combined with other signals. Echo cancellation in a telecommunications network is an example of separation. Signal restoration is used when the signal has been distorted. An example of signal restoration is channel equalization, where the losses of the channel are identified and compensated for.

Transform operations involve the conversion or mapping of the signal from one domain to another. The Fourier transform is such an operation. The Fourier transform converts the signal from a time-domain representation to a frequency-domain representation. Other transforms include the discrete cosine transform (DCT), which is used in image compression, and the modified DCT (MDCT), which is used in audio compression. Digital filtering is implemented in two ways, using either finite impulse response (FIR) filters or infinite impulse response (IIR) filters. The primary

difference between FIR and IIR is that for FIR filters, the output is dependent only on the inputs, while for IIR filters the output is dependent on the inputs and the previous outputs. FIR filters also do not suffer from stability issues that affect IIR filters. In this thesis, an architecture for FIR filtering based on distributed arithmetic is presented. The proposed architecture has the ability to implement large FIR filters using minimal hardware and at the same time is able to complete the FIR filtering operation in minimal amount of time and delay when compared to typical FIR filter implementations. The proposed architecture is then used to implement the fast affine projection adaptive algorithm, an algorithm that is typically used with large filter sizes. The implementation of the fast affine projection adaptive algorithm using distributed arithmetic is unique to this thesis. The constructed adaptive filter shares all the benefits of the proposed FIR filter: low hardware requirements, high speed, and minimal delay. The proposed FIR filter and adaptive filter are then analyzed in the context of digital hearing aids. Example hearing aid types are presented along with comparison results. Figure 1.1 shows the basic diagram of a FIR filter. For an adaptive filter, the coefficients wk are not fixed, after every output sample is computed the coefficients are updated by an adaptive algorithm. It is evident that as the filter length (L) increases, the number of components, and therefore the hardware resources, increases. For large filter sizes, the number of components may become too large to be practical. Many applications require large FIR filters. Pulsed radar uses matched filters which can have filter lengths of up to 1000 points or more. Acoustic echo cancellation requires long filter lengths (> 10,000)to capture the echo which can take up to a few seconds to return. MDCT and the inverse, IMDCT, used in the encoding and decoding of audio streams such as MP3 and AAC, use filter lengths ranging from 128 points 1024 points. The examples listed not only require large filter lengths, but also require the filtering operation to be realtime, resulting in the need for high speed and low latency filtering methodologies.



The simplest way to achieve the speed and latency requirements of demanding applications is to simply use as much hardware resource as necessary. While this approach may be applicable to certain applications such as ground-based radar systems, power consumption demands for other applications such as portable music devices prevent such a direct approach.

Distributed arithmetic (DA) is an efficient multiplication-free technique for calculating inner products The multiplication operation is replaced by a mechanism that generates partial products and then sums the products together. The key difference between distributed arithmetic and standard multiplication is in the way the partial products are generated and added together. Since its introduction, distributed arithmetic has been widely adopted in many digital signal processing applications, including but not limited to digital filtering , discrete cosine transform , discrete Fourier transform

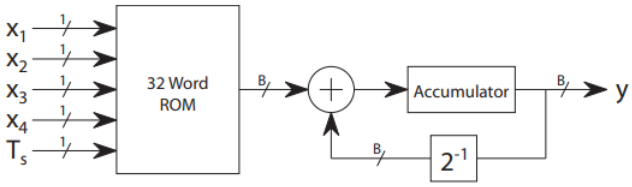


Fig.Block diagram of a 4-tap DA with full table

FIR Filtering Traditional implementations of an L-tap FIR filter

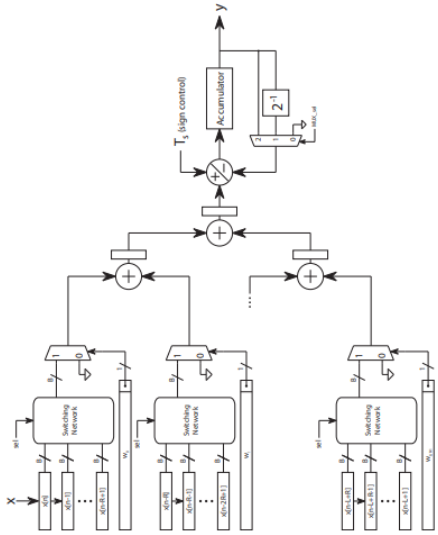
$$y[n] = \sum_{k=0}^{L-1} h[k]x[n-k]$$

where h[k] are the filter coefficients and x[n − k] are the input samples, require the use of L multiply-accumulate (MAC) units. Modern implementations employ M MAC units that are reused, where typically M L. A multiplier-based architecture can be easily configured for a given set of speed, area, and power specifications. High throughput can be obtained by adding more multipliers. Smaller area can be obtained by using fewer multipliers. If high throughput and small area are desired, simply use a small enough number of multipliers and clock the system at a fast enough rate to meet throughput requirements, all the while acknowledging the trade-offs among area, speed, and power. For distributed arithmetic-based FIR filters, similar configuration scenarios exist. When implementing FIR filters using DA, the area resource is dictated by the size of the lookup table, which in turn is dictated by the length of the filter and the bit precision. Throughput is proportional to bit precision, as is power consumption. One advantage that distributed arithmetic architectures possesses over multiplier architectures is a smaller hardware footprint for the same filtering operation. This advantage comes from implementing a multiply operation using smaller, simpler components. Another advantage is that a filtering operation with DA requires only B cycles to complete, regardless of filter length.

FIR FILTERING WITH MULTIPLIERS

Typical implementations of the FIR filtering equation utilize one or multiple multipliers, depending on the throughput requirements of the application. A multiplier-based FIR filter is constructed for comparison with the RDA approach to FIR (RDA-FIR) filtering. The

multiple multiplier design is termed MM-FIR. The multiple multiplier design consists of memories containing the coefficients and input samples. Instead of having multiple MAC units, multipliers are used, with a single accumulator as compared to multiple accumulators.



Having a single or multiple accumulators does not change the operation of the MM-FIR filter, however by separating the accumulator from the MAC and pushing it to the end of the data path reduces the area of the design. The multipliers compute in parallel, each multiplier requiring a memory for coefficients and a memory for input samples. The figure for the multiplier memories combination block is shown in Fig. 2.6. In contrast to the RDA-FIR system where the throughput is proportional to the bit precision, the throughput of an MM-FIR system is proportional to the number of times the multipliers are used. By having a large number of multipliers, the MM-FIR design can achieve high sampling rates. The same location in all the coefficient memories is accessed simultaneously with a circular pointer system for incrementing the address during the filtering operation. After a full iteration, the pointer returns to the starting address. A similar pointer system is used for the memories holding the input samples. As with the coefficient memories, the same location in all input sample memories is accessed simultaneously. The pointer loops through the L M addresses during a sample period. The stopping address of the pointer becomes the start address for the next sample period. This is important since the oldest input samples in each memory block are shifted to the next memory block.

II. LITERATURE SURVEY

We observed that the optimized HW solutions have been essentially addressed to the improvement of memory architectures and the search for the optimal strategy for exchanging data with the data path circuitries. The buffer less solution preserves a high degree of accuracy by using a custom coding and a partial serialization of the filtering, in order to reduce the number of mapped physical resources. The design, however, is very tailored for DoG, since it exploits the separability of Gaussian kernels, and it can be hardly generalized to generic VS methods.

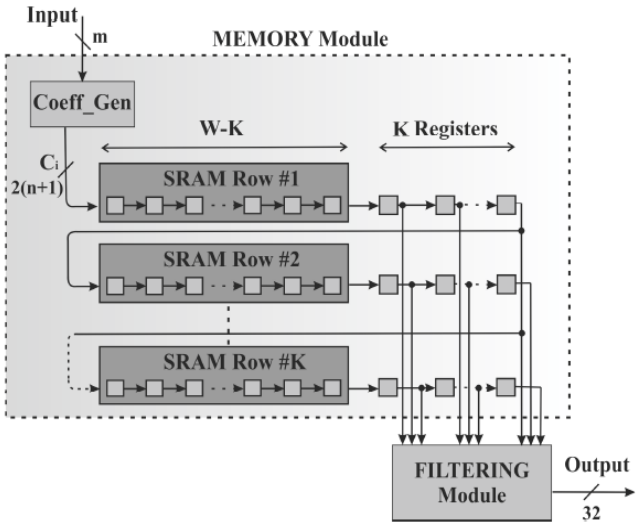
The design does not exploit the separability of Gaussian kernels but proposes a complex arrangement of SRAMs to implement sliding window and row-shuffling operation by means of a switching network whose complexity increases with the filter dimensions. On the other hand, very few optimizations have been addressed to the arithmetic units, which are usually simplified by recurring to fixed-point (FI) codes in place of 32 bits floating-point (FP32), with impact on the accuracy of the overall VS systems.

We developed the Single Octave Scale Invariant Feature Transform (Single Octave SIFT). This solution drastically reduces the computational load and memory bandwidth requirements while providing an accurate image-based terrain referenced navigation system for micro- and small-sized Unmanned Aerial Vehicles (UAVs). The Gaussian filtering and Key point extraction stages are the most computationally intensive parts of the Single Octave SIFT.

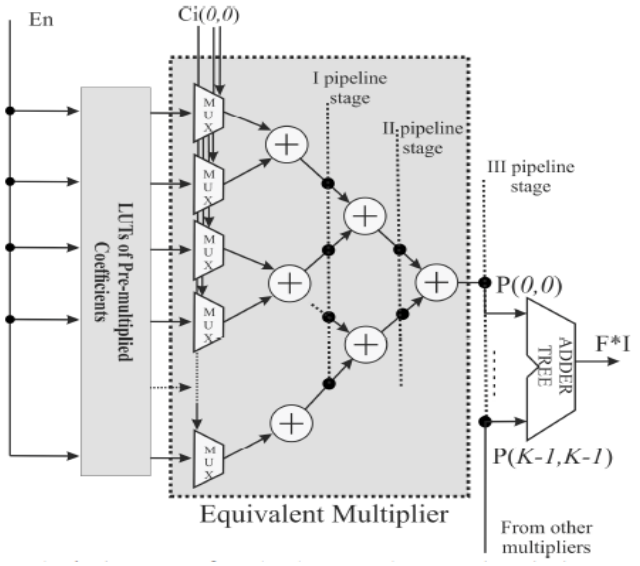
III. METHODOLOGY

In this work a new design is proposed, which is capable to outperform existent FP32 implementation of 2D filters, by exploiting a new partitioning method of the operands, in the case that one of that assumes multiple constant values. The design works on a continuous stream of data, directly provided by the input source, and avoids the use of frame buffers by means of a careful organization of small intermediate buffers. The accuracy of the elaborated results is ensured by the use of FP32 coding, while its compactness is given by the complete absence of multiplier circuits.

BLOCK DIAGRAM:



Block diagram of proposed design

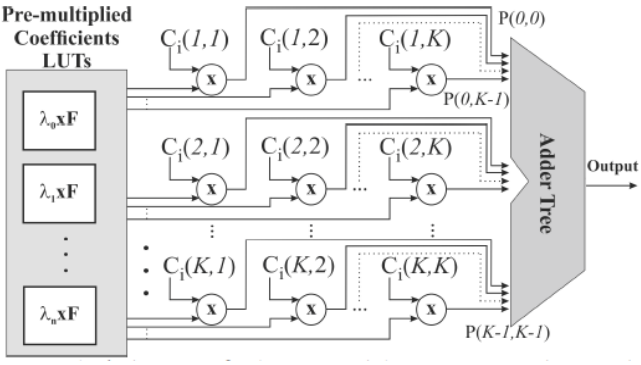


Block diagram of proposed design for Multiplier

In mathematical numeral systems, the radix or base is the number of unique digits, including the digit zero, used to represent numbers in a positional numeral system. For example, for the decimal system (the most common system in use today) the radix is ten, because it uses the ten digits from 0 through 9. In any standard positional numeral system, a number is conventionally written as (x)y with x as the string of digits and y as its base, although for base ten the subscript is usually assumed (and omitted, together with the pair of parentheses), as it is the most common way to express value. For example, (100)dec = 100 (in the decimal system) represents the number one hundred, while $(100)_2$ (in the binary system with base 2) represents the number four. $(5)_{10}$ = 1-1-1 (in the Radix 3 system) represents the number one hundred, while $(1\text{-}1\text{-}1)_3$ (in the binary system with base 3) represents the number four.. Finite impulse response (FIR) filters are extensively used in various digital signal processing applications such as digital audio, image processing, data transmission, biomedical and etc. In some applications, the FIR filter circuit must be capable to operate at high sample rates, while in other applications, the FIR filter circuit must be a low power circuit operating at moderate sample rates. FIR filters design implementation consist a large number of multiplications, which leads to excessive area and power consumption. So, the topology of the multiplier circuit also affects the resultant power consumption. Many works have focused on development to optimizing multiplications by decomposing them into simple operations such as addition, subtraction and shift or sharing common sub-expressions using different methods. Various multiplier structures and algorithms i.e. combinational, serial, add and shift and modified booth are used for FIR filters which define the consumption of resource, and its performance. In this work, booth radix-8 multiplier is used in FIR filter and its effects on power and speed is analyzed.

IV. IMPLEMENTATION

Filtering Module Fig. 2 shows the block diagram of the Filtering Module, representing the organization and interconnections of the equivalent K×K multipliers. The K small LUTs store the (n+1) parts pre-multiplied by the K coefficients of the filter, (,) i F h j l in (2), coded with length lS, calculated in the following. In order to simplify the adder structure, the K LUTs are used to store also the 2's complement of the pre-multiplied coefficients, which are selected when Ci=−1, without additional overhead. Therefore, each LUT has dimensions 2(1) S n+ ×l. It has been substituted by n adders distributed along depth tree, which calculates equation by using the pre-multiplied coefficients, selected by a multiplexer bank, and the Ci coefficients provided by the stripe buffer.



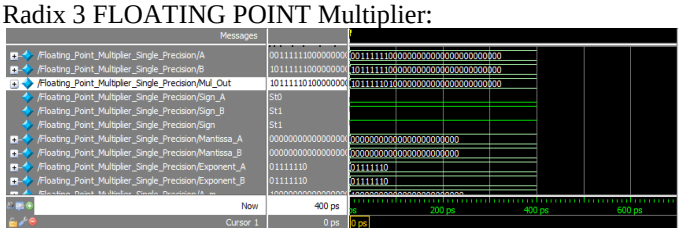
Filter Design

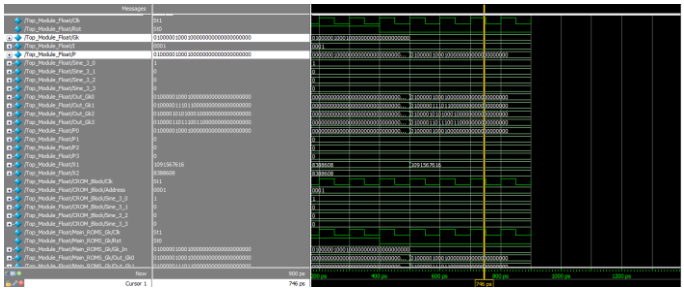
The input is used to access the ROMs that, in principle, are sharable I by all the multipliers. Outputs from the C-ROMs provide the signals to select the inputs to the first adder's row of multipliers. This ROM Contains the Radix 3 Values for Required Numbers.

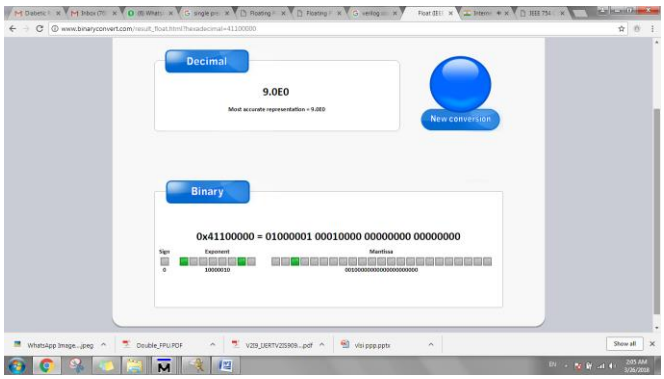
Input	*Partition*						
	3^0	3^1	3^2	3^3	3^4	...	λ_n
0	0	0	0	0	0	...	0
1	+1	0	0	0	0	...	0
2	-1	+1	0	0	0	...	0
3	0	+1	0	0	0	...	0
4	+1	+1	0	0	0	...	0
5	-1	-1	+1	0	0	...	0
...	...	...	...	...	...	...	...
q	C_0	C_1	C_2	C_3	C_4	...	C_n
...	...	...	...	...	...	...	...
r	+1	+1	+1	+1	+1	...	+1

V. SIMULATION AND IMPLEMENTATION RESULTS

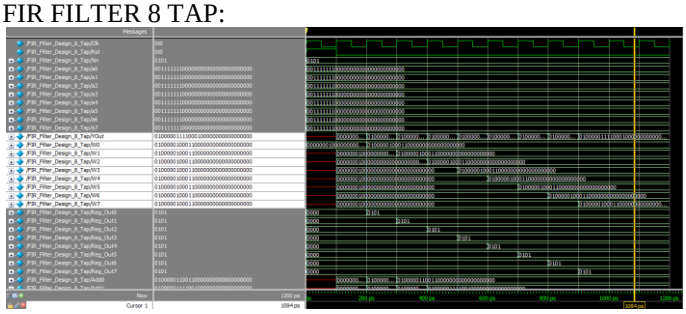
Radix 3 FLOATING POINT Multiplier:



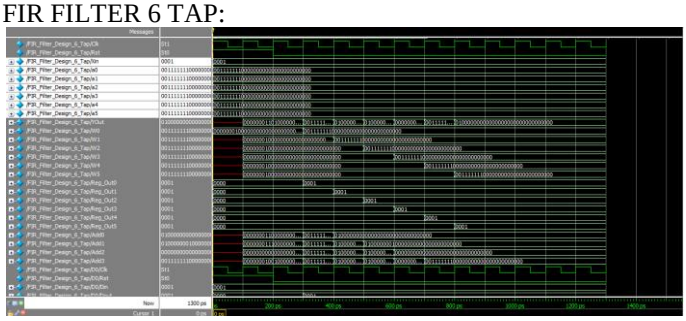




FIR FILTER 8 TAP:



FIR FILTER 6 TAP:



DEVICE UTILIZATION :

RADIX 2 FLOATING POINT MULTIPLIER:

Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	121	55,296	1%	
Number of 4 input LUTs	1,031	55,296	1%	
Logic Distribution				
Number of occupied Slices	565	27,648	2%	
Number of Slices containing only related logic	565	565	100%	
Number of Slices containing unrelated logic	0	565	0%	
Total Number of 4 input LUTs	1,051	55,296	1%	
Number used as logic	1,031			
Number used as a route-thru	20			
Number of bonded IOBs	70	489	14%	
IOB Flip Flops	9			
Number of MULT18X18s	8	96	8%	
Number of GCLKs	1	8	12%	
Total equivalent gate count for design	40,925			
Additional JTAG gate count for IOBs	3,360			

RADIX 3 FLOATING POINT MULTIPLIER:

Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	106	3,840	2%	
Number of 4 input LUTs	1,204	3,840	31%	
Logic Distribution				
Number of occupied Slices	646	1,920	33%	
Number of Slices containing only related logic	646	646	100%	
Number of Slices containing unrelated logic	0	646	0%	
Total Number of 4 input LUTs	1,223	3,840	31%	
Number used as logic	1,204			
Number used as a route-thru	19			
Number of bonded IOBs	70	97	72%	
IOB Flip Flops	32			
Number of MULT18X18s	6	12	50%	
Number of GCLKs	1	8	12%	
Total equivalent gate count for design	34,605			
Additional JTAG gate count for IOBs	3,360			

VI. CONCLUSION

New HW architecture has been presented for 2D convolution-based filtering of images and video-frames. It is particularly useful for VS applications, where performances strongly contrast with the number of arithmetic operators and required memory. Both the memory and the arithmetic apparatus have been design in order to improve the throughput and the amount of mapped resources.

REFERENCES

[1] J. Luo and G. Oubong, “A comparison of SIFT, PCA-SIFT and SURF”, International Journal of Image Processing, Vol. 3, No. 4, pp. 143–152, Aug. 2009.

[2] S. L. Chen, “VLSI implementation of an adaptive edge – enhanced image scalar for real - time multimedia applications”, in IEEE Trans. on Circuits and Systems for Video Technology, Vol. 23, No. 9, pp.1510–1522, Sep. 2013.

[3] M. Basu, “Gaussian-Based Edge-Detection Methods—A Survey”, in IEEE Trans. on Systems, Man and Cybernetics - Part C: Applications and Reviews, Vol. 32, No. 3, pp.234–240, Aug. 2002.

[4] D. G. Lowe, “Object Recognition from Local Scale-Invariant Features”, in Computer Vision, 1999. The Proc. of the Seventh IEEE International Conf. on, Kerkyra, Vol. 2, pp.1150–1157, Sep. 1999.

[5] D. G. Lowe, “Distinctive image features from scale-invariant key points”, in International Journal of Computer Vision, vol. 60, no. 2, pp. 91–110, Jan. 2004.

[6] K. Mizuno et al., “A low power real-time SIFT descriptor generation engine for full hdtv video recognition”, in IEICE Trans. Electron, Vol. E94-C, No. 4 Apr. 2011.