VLSI Architecture Design Approaches for Real-Time Video Processing

A. AHMAD[1], K. K. LOO[2] & J. COSMAS[3]
[1, 2, 3]Electronics and Computer Engineering Department
School of Engineering and Design
Brunel University
UB8 3PH, Uxbridge
UNITED KINGDOM

[1]Computer Engineering Department
Faculty of Electrical and Electronic Engineering
University Tun Hussein Onn Malaysia
P. O. Box 101, 86400 Batu Pahat, Johor
MALAYSIA

Afandi.Ahmad@brunel.ac.uk; Jonathan.Loo@brunel.ac.uk; John.Cosmas@brunel.ac.uk

Abstract: - This paper discusses the programmable and dedicated approaches for real-time video processing applications. Various VLSI architecture including the design examples of both approaches are reviewed. Finally, discussions of several practical designs in real-time video processing applications are then considered in VLSI architectures to provide significant guidelines to VLSI designers for any further real-time video processing design works.

Key-Words: - Dedicated, programmable, Very Large Scale Integration (VLSI) architecture, video processing

1 Introduction

In the digital world, compression is used to exploit limited storage and transmission capacity as efficiently as possible. Since we are in the middle of a digital revolution in which the fields of broadcasting and information technology are merging together, video compression is at the heart of this process. Video compression techniques have played an important role and these are expected to continue to reduce the size of data for transmission and storage purposes, especially since the bandwidth is still a valuable commodity.

The growth of digital video applications and technology in the past few years has been explosive and to date, a wide variety of digital video applications exist, ranging from simple low-resolution and low-bandwidth applications (multimedia, picture phone) to very high-resolution and high-bandwidth (HDTV) demands. With all these examples, there clearly exists a challenge for VLSI designers to design an effective VLSI architecture that meets the requirements of real-time applications. The availability of low-cost and low-power hardware with sufficiently high performance is essential for video processing applications [32]. In addition, real-time processing involves operating continuous data streams of huge volumes. Thus, efficient hardware solutions are of vital importance to meet real-time constraints with desired low-cost and low-power.

A well-known survey on hardware architecture for image and video coding was proposed by Pirsch *et al.* in [1] has was followed with [2,3]. However, the major focuses of these related works are of particular standards such as JPEG, MPEG-1, MPEG-2, H.261 and H.263. However, in [2] the survey only concentrates on the hardware architectures for MPEG-4 video coding and JPEG2000 image coding.

Another survey paper by Dang in [4] provides a brief idea about the VLSI architecture for real-time image and video processing systems. This paper provides a survey of VLSI approaches for video processing applications, mainly the programmable and dedicated approaches. The information contained in this paper is significant and may serve as a reference to VLSI designers about the key consideration in VLSI implementation for video processing applications.

This paper is organized as follows. An overview of fundamental video processing issues and fundamental of hardware architectures are discussed in Section 2 and Section 3, respectively. Section 4 and 5 present several design examples and discussion, respectively for programmable and dedicated approaches. Finally, Section 6 concludes the paper.

2 Video Processing Applications

The landscape of evolution that utilizes the advancement in video processing has undergone rapid changes. From the ideas of satellite imaging, magnetic resonant imaging (MRI) and high-definition televisions (HDTV), apparently a significant improvement to satisfy the commercial and technical requirements are required. Additionally, there is a wide variety of applications for the government purposes, medical industries, for entertainment and also household needs. There is no doubt that this growth has happened because of advances in research and development in algorithms and theory as well as in the areas of systems architectures and VLSI circuit designs [4].

Real-time processing is a vital constraint especially in some applications such as video processing. In general, VLSI designs promise to make this requirement possible, for example efficient circuit design, pipeline architecture and parallel processing. However, to meet with real-time processing, the solution would vary with respect to standard features, like the throughput, the VLSI area and the power dissipation.

A comprehensive example about an application of a surveillance camera is given in [5]. It requires the H.264 encoder performing at real-time with a low-bit rate and also with minimal cost of implementation. In order to optimize for both the bit-rate (compression ratio) and the VLSI area, the following challenges have to be considered: i) A profile selection which has to meet an acceptable bit-rate by providing a performance close with the reference H.264 encoder, ii) The design of a data flow graph that works to minimize the internal memory and register count, iii) An efficient mapping of computations onto the hardware resources and an enabling of the elimination of those resources showing small utilization rates, and iv) The architecture implementation with low clock rates and minimized power consumption.

To meet all these challenges, the VLSI designer has to take significant action in their designs; for example the designer has to choose a profile to give the required performance with respect to the compression ratio and also a low area of architecture. Furthermore, the use of asynchronous-circuits promises further improvement in VLSI area by reducing the number of registers. In regard to memory, the local memory organization is of significant importance since it can provide a temporal storage and also efficient data communications among the encoder modules. In brief, this example acts as clear proof that VLSI designers need to be considering all the commercial and technical requirements especially for any real-time applications designs.

All in all, further investigation and consideration are required for real-time applications for huge market demand, and the advances in VLSI technology promises a great possibility of exploration to accomplish the commercial and technical requirements in any application designs.

3 Fundamentals of Hardware Architectures

Section 2 gives an overview on the ideas, challenges and possibilities in VLSI architecture for video processing applications. Generally, it appears that efficient VLSI design for video processing involves several subtasks with different levels of computational complexity to be manipulated. The subtasks as discussed in [2,3,6,7] and summarized in Table 1 can be clustered into low-level, medium level, and high level types.

In regard to computational characteristics in the algorithms, it is necessity to conduct a detailed algorithm analysis as proposed in [3], as follows: i) Analysis by task profiling to capture the computational share of each task, and ii) Analysis for each task. As video coding algorithms are typically dominated by several computation-intensive tasks, the computational-characteristics of computation-intensive tasks are mostly low-level, for example in JPEG, MPEG-1, MPEG-2, H.261 and H.263 [3].

In the case of H.261 [7], motion estimation, discrete cosine transform (DCT), inverse DCT, and loop filtering account for about 90% of the total number of operations in the low-level tasks. But, it is noticed that, for the sake of overall system performance, the other tasks (medium- and high-level tasks) may not be ignored [2,3].

Moreover, to efficiently process multiple tasks, there are two categories of architectural strategies: programmable and dedicated [1,2,3,4]. Programmable architectures propose the solution that allows various tasks to be implemented on the same hardware.

Programmable architectures execute tasks under software control and it allows future modifications without any hardware redesign. As a consequence of its flexibility, programmable architectures are a suitable choice for larger application fields and are also capable to execute highly irregular tasks with unpredictable operation flow [9]. However, the penalties of flexibility are an inefficiency of hardware resources utilization and an additional hardware cost for control units and storage elements, and higher power consumption. Moreover, time and cost factors also increase for the software development of programmable processors.

Conversely, dedicated architectures are derived by full adaptation to specific algorithms or classes of algorithms. The adoption of dedicated architectures is suitable only for specific-defined application with fixed functionality and as a results, high efficiency of VLSI

implementation can be obtained by manipulate of the special computational features. Concerning hardware overhead and power consumption, dedicated architectures give minimum hardware overhead of control and lower power consumption.

On the other hand, the drawbacks of dedicated are the impossibility of further extensions without hardware modification, and its restricted flexibility architectures which complicate the execution of tasks with largely varying computational demands. Table 2 depicts the comparison of programmable and dedicated architectures.

Instead of programmable and dedicated architectures, a hybrid architecture design which is a combination of programmable and dedicated architectures [2], promises better strategy, and many existing architectures actually involve some combination of these two.

Automated partitioning of applications into hardware (dedicated modules) and software (programmable modules) is the subject of current research in the field of hardware/software co-design [8,9]. The following section will discusses about the architectural strategies in programmable approaches.

3.1 Programmable Approaches

The discussion so far has been concentrated on the capabilities of each approach for VLSI implementations for video processing. Obviously, for larger applications fields with varying computational demands, programmable processors promise a relevant choice by performing computational under software control [1,34]. With the capability to execute several different algorithms on the same hardware, a programmable approach also employs features such as upgradeability by only software changes. Nevertheless, problems with poor utilization of available hardware resources and spending too many clock cycles, require a special architectural approached targeting instruction set, arithmetic units, and a memory system in order to enhance their multimedia processing capabilities [2,34].

There are two architectural techniques [2] that are possible to utilize: parallelization and adaptation strategies. These two strategies aim to increase the multimedia performance and at the same time could be considered as architectural approaches for programmable processors. Fig. 1 clearly shows the approaches for parallelization and adaptation strategies and in the following, an explanation of and examples for each approach are presented.

Parallelization strategies are examined separately on: i) Data level (Single Instruction stream, Multiple Data streams (SIMD), split Arithmetic Logic Unit (ALU), ii) Instruction level - Very Long Instruction Word (VLIW), and iii) Task level (Multiple Instruction Streams, Multiple Data Streams (MIMD), associative controlling). Besides, adaptation strategies are divided into set design modifications (specialized instructions) and the incorporation of dedicated hardware modules (coprocessor). In the following section, several parallelization strategies are presented.

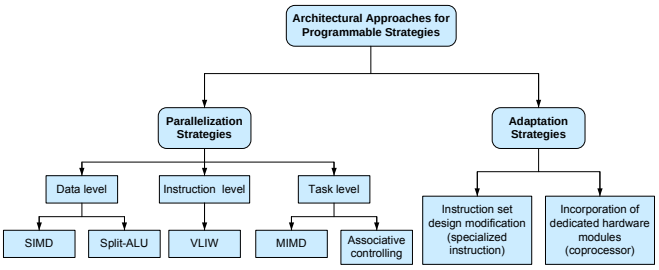

Fig. 1. Architectural strategies for programmable approaches.

3.1.1 SIMD

SIMD as shown in Fig. 2, is a concept with several processors executing the same instructions on different data and it aims to exploit the data parallelism in video processing schemes. In a typical SIMD architecture, a number of parallel data paths are centrally controlled by a main control unit [10,11,17] and all the data paths execute the same stream of instruction with particular data items. Moreover, a single control unit and multiple data paths lead to a high degree of parallelism.

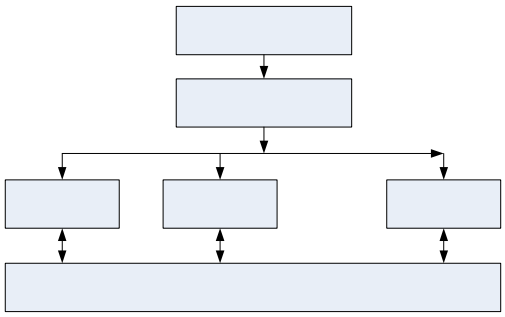

Fig. 2. SIMD multiprocessor architecture [17].

However, to deal with conditional operations, binary masking [2] can be utilized to exclude individual data paths from the execution of single instruction. Hence, for more diverse computation requirements, an SIMD suffers for lack of flexibility. SIMD processors are suitable for low-level tasks [11] where simple operations are applied to multiple data, such as high regular computation patterns algorithms. But, for more irregular algorithms that involve a higher partition of data-dependent processing, there is a rapid decrease in the utilization. Consequently, SIMD architectures [2] are not well-suited to the implementation of complex processing schemes of heterogeneous nature as frequently encountered in multimedia applications.

3.1.2 Split-ALU

In general, split-ALU proposes similar principle to SIMD and targeted to data level parallelism. The split-ALU [12] incorporates subword parallelism, which involves parallel processing of several low-precision data items on a single ALU of higher word length. In terms of implementation, it requires minor hardware extensions. The carry signal arising in arithmetic operations has to be prevented from being propagated across the boundary of separate data items [2]. The small increase in hardware cost makes this concept well suited for the extension of existing general purpose processors and typical operations that may include to be executed in split-ALU including parallel addition/subtraction, multiplication, or comparison [12,13]. However, split-ALU instructions are generally not supported by current compilers, as a result of the lack of adequate high-level language constructs to express desired operations.

3.1.3 VLIW

The aim of VLIW is to exploit the instruction level parallelism. It is particularly important to exploit the instruction level parallelism due to the fact that multiple operations are specified within a single long instruction word for concurrent execution. In order to deal with concurrent, it is necessary to apply multiple parallel function units. As shown in Fig. 3 [14], a static mapping of operation slots defined in the VLIW assigns the individual operation to the function units. Besides, a read/write crossbar connects to a common register file which has to be multiported to enable simultaneous accesses to multiple brands [2].

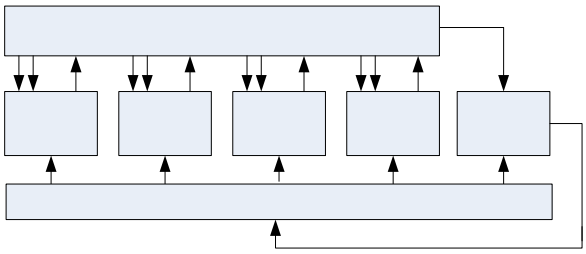

Fig. 3. VLIW architecture [14].

In terms of performance gain, the degree of exploitable instruction-level parallelism in the target algorithms needs to be considered. In addition, sophisticated compiler techniques [15,16] for instance loop unrolling, software pipelining, trace scheduling, or guarded execution, may be applied as a mechanism to increase the pool of operations, which can be scheduled into a single instruction word. Moreover, an even higher degree of parallelism may be exploited in VLIW architectures [33] than in superscalar processors as larger units of code can be inspected at a time.

Comparing VLIW to super scalar processors, it relies on static operation scheduling at compilation time to assemble the long instruction words [15]. The resulting additional hardware units for dynamic reordering are obsolete, and the saved chip area can be used for more functional units [2]. On the other hand, the burden of operation scheduling is shifted from hardware to the compiler, and the resulting utilization and the obtained parallelism fundamentally depend on the available compiler technology.

3.1.4 MIMD

The basic principle of MIMD is about several processors execute different instructions on different data [11]. Each processor can access either its own or a shared memory and they can run the same or different instructions and process different data streams asynchronously. Besides, MIMD could also be consider as an approach to exploit parallelism on both data and tasks level. In contrast with SIMD, an MIMD structure offers private control unit for each single data paths as shown in Fig. 4.

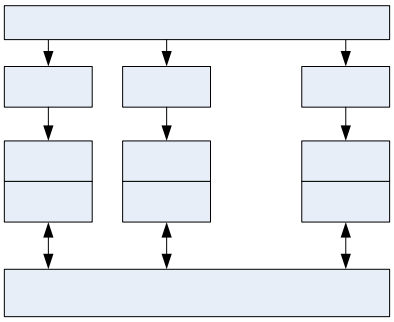

Fig. 4. MIMD multiprocessor architecture [2].

Since each data path is provided with its own individual stream, it is possible to enable concurrent execution of different programs or tasks. Similarly, the data parallelism can be targeted as well by supplying data paths with identical instruction streams. Obviously, this strategy promises high flexibility since each data can be controlled individually and this makes MIMD better suited than SIMD in order to execute compound multimedia processing schemes (low-, medium-, and high-level tasks) [2,11].

Besides, duplication of the control unit and the high bandwidth requirement for continuous supply of several instruction streams dramatically increase the hardware cost by limiting the number of data paths that can be economically implemented on a single chip [2]. Instead of duplication issues, poor programmability is further disadvantage of MIMD, which the programmer has to develop and configure scalar programs manually for individual data paths and also to synchronize the processing elements [11]. The amount of time required

for application development, the limited data paths and the highly complex nature of MIMD are likely to be the main reasons for limited the commercial success of MIMD processors.

3.1.5 Associative Controlling

In order to overcome SIMD and MIMD demerits, associative controlling [17] proposed a new concept with lower number of control units than parallel data paths. Fig. 5 illustrates a structural overview of an associative controlling that consists of control units, multiple instructions and data memory.

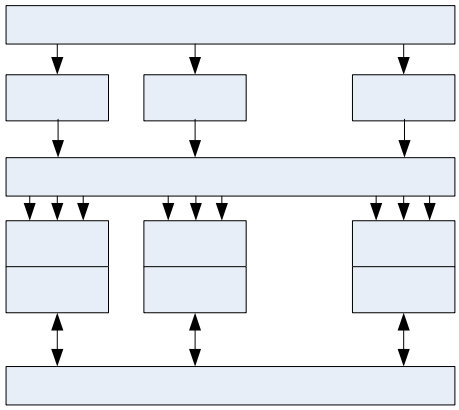

Fig. 5. Associative controlling architecture [17].

In the associative controlled processor, the control units utilize the concurrent strategy to issue their individual instruction streams via an instruction broadcast network to all parallel data paths. Additionally, the principle of associative controlling comes with each data path autonomously selecting an appropriate instruction stream for execution among the multiple instruction streams offered [2,17].

In terms of data path capability, individual data paths can dynamically switch between different instruction streams offered by the control units and this strategy provides an efficient execution of data-dependent control flow. Simultaneous operation accommodates the instructions' alternative branches following a decision to all data paths and letting each data path select the proper branch according to its local status [2,17].

Furthermore, an associative controlled processor also has the ability to adapt with varying parallelization degrees during computation by supporting a dynamic clustering of data paths. Even this approach promises solution to overcomes the drawbacks of SIMD and MIMD, issues such as area, power overhead, selection of concurrent instruction streams and compiler support are further to be explored for better performance and capability of associative controlling approach.

3.1.6 Specialized Instruction

In order to adapt the programmable processor to specific algorithm, specialized instructions can be introduced to deal with frequently occurring operation or higher complexity. Specialized instruction including extended shift, minimum/maximum average, and add-sign operations are the examples of it [18,19]. In this case, the functions of specialized instructions are to reduce the instruction count and accelerate program execution.

However, the incorporation of specialized instructions leads for incremental hardware cost especially for additional functional units such as multiply adder, but the design complexity of additional units can usually moderated as a result of high specialization and optimization toward the targeted operations. In terms of decision, specialized instruction involves a trade-off between additionally required software effort and probability of their use. Unfortunately, the specialized instruction is not universal since only subsets of algorithms will experience of acceleration.

3.1.7 Coprocessor/Heterogeneous Multiprocessor

In this strategy, there are combinations of parallelization and adaptation principles. Concerning architectures, generally it comprises of a flexible general-purpose processing module such as a standard RISC core, coupled with a less flexible hardware module adapted toward a specific type of processing. For the implementation, RISC core performs high-level tasks of lower computational requirements such as control and I/O functions, while the adapted modules executes the computation-intensive but regular low-level tasks [20].

To deal with lower computational requirements, specialized modules may also be considered, but it requires special algorithm characteristics and obviously difficult to implement on standard processors [18]. In addition, the concept of heterogeneous multiprocessor architectures appears as a result of combination several programmable and dedicated modules and it can be seen as a generalization of the coprocessor concept. In contrast with homogenous multiprocessors, heterogeneous multiprocessor allow us to reach higher performance levels by emphasizing parallel execution of several task, but they are still more tied to a specific processing scheme due to the adaptation of individual modules.

The coprocessor/heterogeneous multiprocessor architectures are efficient particularly in use of hardware resources due to the specialization and also reduced control overhead of the adapted modules. However, the adaptation leads to restriction in flexibility even it is noticed that most algorithms can still be executed in software. Significant changes in the targeted application may result in a highly unbalanced resource utilization

and serious performance degradation. As a result, coprocessor architectures are best suited for well-defined and fixed application fields [2,20].

3.2 Dedicated Approaches

In Section 3.1 the architectural approaches for programmable strategies has been discussed, while this subsection will gives an idea about the dedicated approach. With the advancement of video processing applications and at the same time huge requirement for consumer electronic market, the roles of dedicated approach is significant. Dedicated approach offers for high volume production and relatively the optimization of silicon area leads to lower production cost.

The study in [1] shows that the demand for high production volumes though the narrow application field, the number of candidates for dedicated implementations is rather limited. Moreover, with varies design complexity and since for each dedicated design targeted for specific function, there is no general assessment can be made and it is agreed in [1,2]. Consequently, only a general comparison available in order to give a landscape of ideas about the dedicated approaches works. Reviews of the dedicated approaches design examples for video processing applications specifically are available in Section 6.

3.3 Memory and Interconnection Strategy

Despite of programmable and dedicated architecture approaches, memory and interconnection architecture also requires significant strategy for better overall system architecture. Regarding data communication, video processing involves memory and interconnects architecture to deal with large data of computation during processing continuous data streams. The date access through the frame buffer dominated by the memory architecture, while the data access between different module architectures and the memory architecture is primarily dominated by the interconnect architecture.

Concerning memory architecture, the data access through the frame buffer is a slow and power-consuming process. In order to solve these problems, two architectural approaches [2] can be implemented as follows: i) Implementation of special local memory buffers, and ii) Integration of the off-chip frame buffer as on-chip memory.

For the first approach, since most of the tasks in video processing are predictable and contain many repetitive data access patterns, it is possible to adopt special local memory buffers. With this approach, it significantly reduces redundant data access by using data-reuse property. Besides, the second approach propose the advances of VLSI technology with various designs of integrated embedded DRAM or SRAM as on-chip frame buffer to overcome the data access problem. Despite it can improve the data access performance, it also help to reduce the I/O power consumption.

Another key factor issue in data communication for VLSI architectures is interconnection architectures. It is vital to have depth understanding of the intermodule communication pattern to design efficient interconnect architecture, with higher bandwidth and consume less power [3]. However, flexibility and efficiency in interconnect architecture is inter related factors [3], for instance the global bus provides higher flexibility but lower efficiency, whereas the dedicated data link with full adaptation to specific algorithm provides the highest efficiency but without flexibility.

In the following sections, programmable approaches including the design and examples are provided.

4 Programmable Approaches: Design and Examples

In the Section 3.1, several strategies to enhance the multimedia processing performance have been discussed. In the following, several design examples of various video processing applications are presented. The designs employ a combination of different strategies in order to achieve optimum performance and high efficiency.

4.1 Design Examples

With the huge demand on market particularly for low-power systems, Liu *et al.* in [22] proposed a solution of a low-power quadtree video encoder with the following modules: i) A low-complexity motion estimation algorithm, ii) An application-specific ME-MC processor, iii) A dedicated quadtree encoder engine, and iv) A processor control-based clock-gating technique. The proposed solution could be considered as a hybrid approaches called application specific instruction-set processors (ASIPs). Their proposed solution contributes with capability to process VGA at 30 fps and the power consumption only 40.87mW at 80MHz and 1.97mW at 3.3MHz for QCIF at 15 fps.

Dedicated quadtree encoder engine and the processor control-based clock gating techniques has been applied in [22], and a quadtree is defined as a tree data structure in which each internal node has up to four children. Quadtrees are most often used to partition a two-dimensional space by recursively subdividing it into four quadrants or regions. The QT algorithm has been chosen in this case as a replacement of DCT. The DCT works to transform the data into the frequency domain to perform compression easily. But, the DCT

transformation requires a multiply operation, which has higher calculation complexity. Conversely, the QT algorithm can transform the data into a simple compression form without using multiply operation and results in low complexity computation.

In concern to power saving, clock gating techniques has been utilized in this design. Clock gating [22] refers as an activity to activate the clocks in a logic module only if there is work to be done and also the processor will shut down some modules when these modules will not be utilized again. However, in ME or MC the number of active clock cycles are varies with the video contents, but the clock-gating technique can be used to switch off the clock for idle modules. Thus, power saving of these modules can be achieved.

On the other hand, Iwasaki *et al.* in [23] proposes a new architecture for VASA, a single-chip MPEG-2 422P@HL CODEC LSI with multichip configuration. The VASA architecture is derived from the remodelling of parallel encoding and the propose architecture is targeted for large scale processing beyond the HDTV level. There are two major problems involved in implementing single-chip HDTV codec LSI, as follows: i) Complexity and memory bandwidth, and ii) Multichip configuration, but both major problems have been solved using a single-chip MPEG-2 422P@HL CODEC LSI with multichip configuration. An LSI was successfully fabricated using the 0.13μm eight-metal CMOS process and the multichip system configuration depicted in Fig. 7.

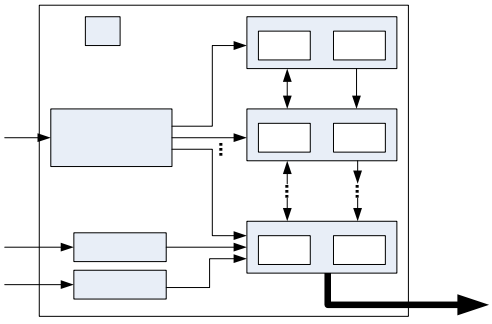

Fig. 7. Multichip system configuration [23].

Sohn *et al*. in [24] proposes fixed-point co-processor architecture with dual operations uses advanced 3-D graphics algorithms and various streaming multimedia functions in a single hardware to achieve low-power consumption. It is targeted to balance 3D graphics pipeline within the limited system resources and provide a single hardware solution for various multimedia applications. The co-processor architecture composed of fixed-point SIMD datapath has been used in [24] for maximum throughput and easy programmability. The dual operations from conventional co-processor architecture allow parallel operations in streaming multimedia processing. A fixed-point multimedia co-processor is designed and tested into an ARM-10 based mobile graphics processor for portable 2-D and 3-D multimedia applications. Fig. 8 shows the block diagram of the multimedia coprocessor with two parts – control and datapath.

In the control part, there is a 2kB code memory that stores vertex program codes of graphics instructions, while in the datapath part, there is a fixed-point vector unit that is responsible for all SIMD arithmetic operations such as addition and multiplication. In order to optimize the power consumption, the clock gating is performed on an instruction-by-instruction basis.

Suggestion in [24] for future research lies in efficient memory streaming system for the co-processor requires further investigation to boost overall performance with high-sustained throughput. Also, scalable data stream engine including direct memory access (DMA) and multi-layer bus architecture require further consideration.

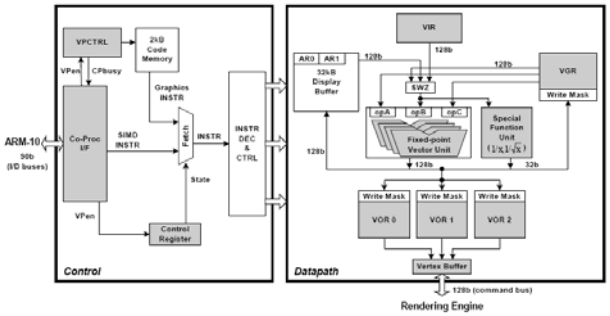

Fig. 8. Block diagram of the multimedia co-processor [24].

Another design example is the latest Philips media-processor based on the Trimedia architecture called TM3270 [21]. It addresses the requirement of multi-standard video processing at standard resolution and the associated audio processing requirements. In the architectural perspectives, the TM3270 is a VLIW-based media-processor and inherit the C-code written in the previous TriMedia processors to lead for compatibility with other processors in the TriMedia family. Typically, the TM3270 architecture specification is used as an embedded processor in a System-on-a-Chip (SoC).

The TM3270 proposed a new enhancement such as the (instruction set architectural) ISA enhancements, collapsed load operations with interpolation, new CABAC operations and also prefetching. With new CABAC operations introduced, the performance of complete decoding for a standard resolution 4.5 Mbits/sec bitstream results in a speed up in the range of [1.5, 1.7]. By increasing the operating frequency when compared to its predecessor, TM3270 is able to either

encode or decode H.264 video material at standard definition resolution and it influenced by the CABAC and collapsed load operations with interpolation.

The ISA extensions with roughly 40 new operations improve the performance on video processing kernels. Additionally, the data cache policies and prefetching techniques allow for efficient access to multimedia data. Since it is established that TM3270 is the first processor that supports two-slot operations [25], it open up the possibility to directly support functions that have up to four inputs and produce to two outputs. Moreover, they allow for the combining of multiple two input operations and it reduces overall function latency and register pressure.

Context-based Adaptive Variable Length Code (CAVLC) proposed a technique to handle the demand of high quality video and high data compression. Many previously related studies as reviewed in [25] adapt the bit-serial method and bit-parallel method. Though the bit serial-method is simple, it is not suitable for real-time processing applications. Based on the bit-parallel method, Chang et al. solution [25] exploit five different techniques as follows: i) PCCF (Partial combinational component Freezing), ii) HLLT (Hierarchical logic for Look-up tables), iii) ZTEBA (Zero-left table elimination by arithmetic), iv) IDS (Interleaved Double Stacks), and v) ZCS (Zero Codeword Skip). Implementing these techniques shows the achievement of low hardware cost, low-power consumption, and high data throughput rate. Based on the experimental data, results shows that the design can decode every syntax element per cycle and achieves the real-time processing requirement for H.264 video decoding on HD1080i format video. Interestingly, the proposed design presents 23% and 40 % respectively, in terms of hardware cost and speed improvements as compared to the existing design as reported in [28].

A compact DSP core in [26] is a compact DSP core for dual-core multimedia SoC and its complete software development tools. In regard to architectures, the DSP cores incorporate a pure dataflow engine which consists of off-the shelf memory modules with limited ports. For the software solution, it inherits the principle of VLIW processors that promises extensive reduction of hardware complexity. Moreover, the DSP is also equipped with novel static floating-point units that work to emulate expensive floating-point DSP operations at low cost.

Besides, static floating-point arithmetic contributes to emulate expensive floating-point DSP operations and consequently helps to further shrink the DSP core. For the VLSI implementation [26], the DSP core module has been prototyped in the 0.35μm CMOS technology and operates at 100MHz with 122mW power consumptions, and physically the core size is 2.8mm^2 with embedded DMA controller and the AMBA AHB interface.

Empirically, the DSP cores has about thrice the performance of ADSP-218x in the execution cycles, and the 16-bit static Floating Point (FP) arithmetic has 31.1165 dB signal to round-off noise ratio over IEEE single precision FP units.

Finally, work of Woo *et al* in [27] is designed and targeted for mobile multimedia applications. With utilization of pure DRAM technology, the proposed design is envisioned to reduce the fabrication cost while keeping the huge memory bandwidth. Moreover, the DRAM process also significantly results in further reduction of power consumption since the off-chip loading to render memory is completely eliminated. Experimental data demonstrates that the embedded DRAM drastically reduces the power consumption since the external I/Os for 3D rendering are completely eliminated, and an additional 22% reduction is obtained by low-energy texturing unit and pipeline clock gating.

The design [27] also optimizes the 3D pipeline that results in 210 mW at a drawing speed of 66Mpixels/s and 264Mtexels/s bilinear MIPMAP texturing and antialiasing. The design meets the performance requirements of mobile applications with little leakage current through and eventually cause large gate delay and routing area compared with pure logic gates. In brief, the proposed design has a 121 mm^2 chip area and has been implemented with 0.16 μm pure DRAM process to reduce the fabrication cost. The real-time 3D graphics applications are successfully demonstrated by the fabricated chip on two PDA systems boards.

4.2 Discussion

All the examples presented in Section 5.1 appear to clarify a wide range of architectural approaches, particularly for programmable approaches in VLSI architectures for real-time video processing schemes.

Obviously, all the applied strategies are influenced by commercial and technical requirements. Commercial requirements are related to general consumer requirements such as the factors of size, cost and reliability. Moreover, technical requirements could be considered as the methodology to achieve commercial requirements including low-power design strategy, low computational complexity, simple data path, efficient algorithms and high throughout.

There is no doubt that most of the presented architectures inherit coprocessor concept with flexible programmable modules as a strategy to match with real-time applications. An overview of programmable architectures targeted for various real-time video processing applications is given in Table 3.

Based on the comparison that has been made, the following ideas are useful for VLSI designers. First and

the foremost for the low-power design, the clock gating technique has been implemented in designs in [22] and [24] which significantly contributes to power saving schemes.

On the other hand, to allow compatibility with other processor, the C-code and VLIW-media processor have been incorporated into the TM3270, which prove that by inheriting the same programmability scheme, it can enhance the processor compatibility. Lin *et al*. in [26] agrees that this strategy promises extensive reduction of hardware complexity.

Moreover, the introduction of collapsed load operations and CABAC operations in TM3270 gives significant results concerning speed and processor capabilities to either encoding or decoding H.264 video material at standard definition resolution. Furthermore, for the HDTV application, the bit-parallel method with PCCF, HLLT, ZTEBA, IDS, and ZCS could be considered for further implementation since it is proven that it offers advantages in terms of hardware cost, power-consumption, and data throughput rate. In addition, utilization of DRAM technology promises the reduction of the fabrication costs and at the same time retains the huge memory bandwidth. As a result, the DRAM process also lead to a significant reduction in power consumption since the off-chip loading to render memory is completely eliminated.

Based on this comparison, it can be concluded that for each application, there is still room for VLSI designers to contribute and it is clear that each application requires low hardware cost, low power-consumption, and high data throughput rate.

5 Dedicated Approaches: Design and Examples

In this section, several works that related with dedicated approach for real-time video processing application will be discussed. They are the works of Song *et al.* [29], Wei and Gang [30] and Kim *et al.* [31]. For the sake of simplicity and general comparison, we only discuss the proposed solution and their contribution towards the video processing applications.

5.1 Design Examples

A novel partial distortion sorting (PDS) algorithm and a simplified local PDS (LPDS) algorithm are presented in [29] as a mechanism to solve demerits with fast ME algorithms, including the instable image quality, irregular work flow and variable computation time. The basic idea of the proposed PDS algorithm is to disable the search points which have larger partial distortions during the ME process, and only keep those search points with smaller ones.

To further reduce the computation overhead, a simplified LPDS algorithm is also presented. The proposed two algorithms can be integrated into different FSBMA architectures to save power consumption. All the contributed two algorithms has been realized with 1-D systolic array in Verilog HDL and synthesized by Synopsys Design Compiler with TSMC 0.18 um 1P6M technology. Moreover, all the designs are placed and routed by Synopsys Astro and the proposed system architecture shown in Fig. 9.

In PDS and LPDS algorithms, since all the search points cannot be directly skipped before calculation, it is hard to reduce PE number in order to save the hardware cost. However, the PE can be disabled during ME process to reduce power consumption. Therefore, the PDS and LPDS algorithms are more suitable for low-power applications and in practice it can be integrated into various FSBMA architectures. Experimental result shows that the PDS and LPDS have almost the same quality as FSBMA. Moreover, it also has been integrated into the 1-D systolic array to prove its hardware-friendliness. In summary, the PDS and LPDS are envisioned for different requirements. For applications whose image quality is critical, PDS algorithm is suitable, otherwise, LPDS more desirable in regard to the factor of hardware cost and power consumption.

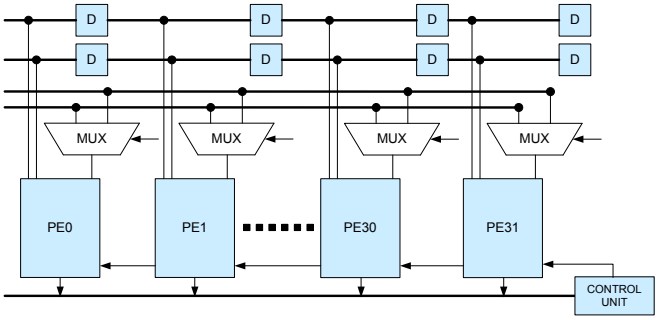

Fig. 9. 1-D systolic array architecture with proposed PDS and LPDS algorithms (block size 16x16, search range 32x32) [29].

Wei and Gang in [30] suggest a new design of enhanced motion estimation (ME) with variable block size as a result to fulfil the feature requirements in the MPEG-4 AVC/H.264 standard. Basically, there are two kinds of architecture for VBSME in the MPEG-4 AVC/H.264, which is a 1-D array and a 2-D array. In this proposed design, a 2-D array architectures has been used as it envisioned to support real-time processing. In brief, this architecture can support seven block patterns and also can reuse the smaller blocks' SADs to calculate 41 motion vectors (MVs) of a 16x16 block in parallel. For the reason of high data-reuse in the search area, the architecture can be reconfigured to support a

"meander"-like scan format of the search area. The proposed architecture consist of 16 computing units (CU), a 4-staged adder tree, 41 comparators and many programmable switches.

The designs has been implemented with Verilog-HDL description and synthesized by Synopsys Design Compiler with TSMC 0.25µm CMOS standard cell library. It is proven that the architecture allows the real-time processing of 720x576 at 30fps in a search range [-16, +15].

The aim of implementation work by Kim et al. in [31] is to find all 41 motion vectors and their minimum SADs for the seven blocks (16x16, 16x8, 8x16, 8x8, 8x4, 4x8, and 4x4). A 2-D array architecture has been chosen even it is noted that a 1-D array architecture is simple, easy to control, and requires smaller area than a 2-D array architecture. But, a 1-D array architectures search only based on one row or column of the block at a time, so it slower than a 2-D array architecture which computes the sum of absolute differences (SAD) of a search point at every cycle. Therefore, a 2-D array architecture is more suitable for high-end real-time video processing. In the full-search 2-D array architecture, a processing element (PE) array computes the SADs between a pixel of the current frame and a pixel of one of its reference at every cycle, so the pixel data should be changed every cycle.

The data flow of a 2-D array architectures can be listed into three classes as follows: Class 1: The current frame pixel data are moving while the reference frame pixel data are fixed, Class 2: While the current frame pixel data are fixed, the reference frame pixel data are changed, Class 3: Both current and reference frame pixel are changed. Based on the list of data flow classes, the 2-D array architectures, the proposed class 3 has been adopted since it offers high PE utilization and simple data path.

The proposed architecture [31] as illustrated in Fig. 10 comprises of four basic blocks: i) The processing element array that computes 16 4x4 SADs of a 16x16 marcoblock, ii) The adder and comparator block to sum up the 4x4 SADs to form the SADs for seven different block sizes and finds the minimum distortions and corresponding motion vectors, iii) The search area SRAM contains the reference frame pixel data within a given search range to reduce I/O memory bandwidth, and iv) The ME control block generates the addresses of the search area SRAMs and the control signals to other blocks.

The proposed architecture has been utilized a VHDL description and synthesized by Synopsys Design Compiler with TSMC 0.18 µm standard cell library, and in overall on-chip memory requirements for this proposed solution are 60Kb with total 154k gates.

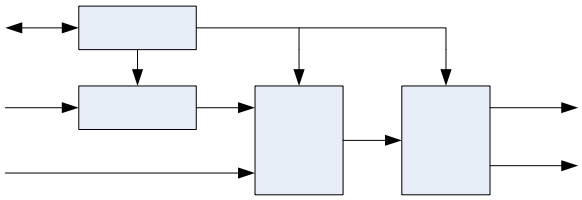

Fig. 10. Block diagram of the VBSME architecture [31].

5.2 Discussion

Section 6.1 provides the examples of video processing applications that optimized the dedicated architecture capability. Since the dedicated approach is literally for specific functions, Table 4 proposes a general comparison of several works.

In general, this comparison gives a perspective of novelty for each design to manage the advancement in video processing applications. For applications that require high image quality, the PDS design proposed by [29] is suitable, otherwise, LPDS more desirable due to the factor of hardware cost and power consumption. However, since the proposed PDS and LPDS are unable to make all the search points to be directly skipped before calculation, it suffers to minimize the PE number in order to save the hardware cost. Yet, the PE can be disabled during the ME process to reduce power consumption.

The proposed architecture by Wei and Gang in [30] contributes to real-time processing with the support provided to seven block patterns and also can reuse the smaller blocks' SADs to calculate 41 motion vectors (MVs) of a 16x16 block in parallel. Additionally, the reconfigurable tools to support a "meander"-like scan format of the search area allows for high data-reuse of the search area. Experiments show that the architecture allows the real-time processing of 720x576 at 30fps in a search range 32x32. The design by Kim *et al*. in [31] is more suitable for high-end real-time video processing by utilizing a 2-D array architecture, even though the 1-D array architecture is simple, easy to control, and requires smaller area.

All in all, recent developments in video processing applications have led to an increase in the complexity of both module' and specific functions to achieve real-time applications. It is clear that the dedicated approaches are an option to consider for reducing power consumption and hardware cost.

6 Summary

According to the discussion and comparison that has been made for both approaches and also for each designs provided in Section 5 and Section 6, video processing applications are used in various fields. Commercial and technical requirements are inter-related and play a significant role in designing a better VLSI architecture for video processing.

Commercial requirements in hardware architectures have to be meet by all video processing applications. Moreover, the architectures also must be competitive with huge low-cost consumer market demands. Architectural approaches for video processing applications can be divided into programmable and dedicated approaches each which has advantages and disadvantages.

Both will continue contributing to future implementations, however, an increasingly irregular and unpredictable computation flow in video processing applications demands a higher degree of flexibility. Thus, in future it is predicted that programmable and hybrid approaches will become the domain of VLSI architectures' implementation in video processing applications.

Finally, this paper provides useful information and ideas to the VLSI designer for further design consideration in order to fulfil the commercial and technical requirements with maximum advantages.

References:

[1] P. Pirsch, N. Demassieux and W. Gehrke, "VLSI architectures for video compression-A survey," in *Proc. of the IEEE,* 1995. vol. 83, pp. 220 – 246.

[2] P. Pirsch and H.-J Stolberg, "VLSI implementations of image and video multimedia processing systems," *IEEE Trans. Circuits and Systems for Video Technology,* vol. 8, no. 7, pp. 878 – 891, Nov. 1998.

[3] P. Tseng, Y. –C. Chang, Y. –W. Huang, H. –C. Fang, C. –T. Huang and L. –G. Chen, "Advances in hardware architectures for image and video coding - A survey," in *Proc. of the IEEE,* 2005, vol. 93, pp. 184 –197.

[4] P. Dang, "VLSI architecture for real-time image and video processing systems," *Journal of Real-time Image Processing,* vol. 1, pp. 57 – 62, Aug. 2006.

[5] K. Babionitakis, G. Doumenis , G. Georgakarakos, G. Lentaris, K. Nakos, D. Reisis, I. Sifnaios, N. Vlassopoulos, "A real-time H.264/AVC VLSI encoder architecture," *Journal of Real-time Image Processing,* vol. 3, pp. 43 – 59, Nov. 2007.

[6] A. Choudhary and S. Ranka, "Parallel processing for computer vision and image understanding," *IEEE Computer*, vol. 25, pp. 7 – 10, Feb. 1992.

[7] M. Maresca, "Parallel architectures for image processing," *Proc. IEEE* (*Special Issue*), vol. 84, July 1996. pp. 915 – 916.

[8] G. De Micheli, "Hardware-software codesign," *Proc. IEEE* (*Special Issue*), vol. 85, Mar. 1997. pp. 349 – 365.

[9] K. A. Olukotun, R. Helaihel, J. Levitt and R. Ramirez, "A software-hardware co-synthesis approach to digital system simulation," *IEEE Micro*, vol. 14, no. 4, pp. 45 – 58, Aug. 1994.

[10] M. J. Flynn, "Very high speed computing systems," *Proc. IEEE*, vol. 54, Dec. 1966, pp. 1901–1909.

[11] L. C. Sim, H. Schroder and G. Leedham, "MIMD–SIMD hybrid system-towards a new low cost parallel system," *Parallel Computing,* vol. 29, pp. 21– 36, 2003.

[12] Y. –K. Chen, S. Y. Kung, "Multimedia signal processors: an architectural platform with algorithmic compilation," *Journal of VLSI Signal Processing*, vol. 20, pp. 181– 204, Oct. 1998.

[13] V. Bhaskaran, K. Konstantinides, R. B. Lee, and J. P. Beck, "Algorithmical and architectural enhancements for real-time MPEG-1 decoding on a general purpose RISC workstation," *IEEE Trans. Circuits and Systems for Video Technology,* vol. 5, pp. 380 – 386, Oct. 1995.

[14] S. Dutta, A. Wolfe, W. Wolf, and K. J. O'Connor, "Design issues for very-long-instruction-word VLSI video signal processors," in *VLSI Signal Processing IX, IEEE*, Oct. 1996, pp. 95 – 104.

[15] R. R. Hoare, A. K. Jones, D. Kusic, J. Fazekas, J. Foster, S. Tung, and M. McCloud, "Rapid VLIW processor customization for signal processing applications using combinational hardware functions," *EURASIP Journal on Applied Signal Processing,* vol. 2006, pp. 1 – 23, 2006.

[16] W.-M. W. Hwu, R. E. Hank, D. M. Gallagher, S. A. Mahlke, D. M. Lavery, G. E. Haab, J. C. Gyllenhaal, and D. I. Aug., "Compiler technology for future microprocessors," *Proc. IEEE*, Dec. 1995. vol. 83, pp. 1625 – 1676.

[17] W. Gehrke and K. Gaedke, "Associative controlling of monolithic parallel processor architectures," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 5, pp. 453 – 464, Oct. 1995.

[18] K. Nadehara, I. Kuroda, M. Daito, and T. Nakayama, "Low-power multimedia RISC," *IEEE Micro*, vol. 15, pp. 20 – 29, Dec. 1995.

[19] E. Holmann, A. Yamada, T. Yoshida, and S. Uramoto, "Real-time MPEG-2 software decoding with a dual-issue RISC processor," in *VLSI Signal Processing IX*, *IEEE,* Oct. 1996, pp. 105 – 114.

[20] S. –Y. Kung and Y. –K. Chen, "On architectural styles for multimedia signal processors," in *Proc. IEEE Workshop Multimedia Signal Processing,* June 1997, Princeton, NJ, pp. 427 – 432.

[21] J. –W. Waerdt, S. Vassiliadis, S. Das, S. Mirolo, C. Yen, B. Zhong, C. Basto, J. –P. Itegem, D. Amirtharaj, "The TM3270 media-processor," in *Microarchitecture, 2005. MICRO-38. Proc. 38th Ann. IEEE/ACM Int. Symp.,* pp. 12 – 23.

[22] Q. Liu, S. Hiratsuka, K. Shimizu, S. Ushiki, S. Goto and T. Ikenaga, "A 41 mW VGA@30 fps quadtree video encoder for video surveillance systems," *IEICE Trans. Electron,* vol. E91-C, pp. 449 – 456, April 1. 2008.

[23] H. Iwasaki, J. Naganuma, K. Nitta, K. Nakamura, T. Yoshitome, M. Ogura, Y. Nakajima, Y. Tashiro, T. Onishi, M. Ikeda, M. Endo, "Single-chip MPEG-2 422P@HL CODEC LSI with multichip configuration for Large Scale Processing beyond HDTV level," *IEEE Trans. Very Large Scale Integration (VLSI) Systems,* vol. 15, pp. 1055 –1059, 2007.

[24] J. –H. Sohn, J. –H. Woo, J. Yoo and H. –J. Yoo "Design and test of fixed-point multimedia co-processor for mobile applications," in *Design, Automation and Test in Europe, 2006. DATE '06. Pro,* vol. 2, pp. 1 – 5.

[25] H. –C. Chang, C. –C. Lin, and J. –I. Guo, "A novel low-cost high-performance VLSI architecture for MPEG-4 AVC/H.264 CAVLC decoding," in *Circuits and Systems, 2005. ISCAS 2005. IEEE Int. Symp.* vol. 6, pp. 6110 – 6113.

[26] T. –Y. Lin, H. –Y. Lin, C. –M. Chao, C. –W. Liu and C. –Y. Jen, "A compact DSP core with static floating-point unit and its microcode generation," in *Proc. of the 14th ACM Great Lakes Sym. on VLSI,* Boston, MA, USA, 2004, pp. 57 – 60.

[27] R. Woo, S. Choi, J. –H. Sohn, S. –J. Song, Y. –D. Bae and H. –J. Yoo, "A low-power graphics LSI integrating 29Mb embedded DRAM for mobile multimedia applications," in *Proc. of the Asia and South Pacific Design Automation Conf.* 2004, pp. 533 – 534.

[28] W. Di, G. Wen, H. Mingzeng and J. Zhenzhou, "A VLSI architecture design of CAVLC decoder," in *ASIC, 2003. Proc. 5th Int. Conf.* vol. 2, pp. 962– 965.

[29] Y. Song, Z. Liu, T. Ikenaga, and S. Goto, "Low-power partial distortion sorting fast motion estimation algorithms and VLSI implementations," *IEICE - Trans. Inf. Syst.* E90-D, pp. 108 – 1171, Jan. 2007.

[30] C. Wei and M. –Z. Gang, "A novel VLSI architecture for VBSME in MPEG-4 AVC/H.264," in *IEEE Int. Symp. on Circuits and Systems,* 2005, pp. 1794 – 1797.

[31] M. Kim, I. Hwang and S. –I. Chae "A fast VLSI architecture for full-search variable block size motion estimation in MPEG-4 AVC/H.264," in *Proc. of the Asia and South Pacific Design Automation Conf.* 2005, vol. 1, pp. 631– 634.

[32] S. –Y. Chien, Y. –W. Huang, C. –Y. Chen, H. H. Chen and L. –G. Chen "Hardware architecture design of video compression for multimedia communication systems," *IEEE Communications Magazine,* vol. 43, pp. 122 – 131, 2005.

[33] K. Gaedke, M. Borsum, M. Georgi, A. Kluger, J.-P. Le Glanic and P. Bernard, "Architecture and VLSI Implementation of a programmable HD real-time motion estimator," in *Circuits and Systems, 2007. ISCAS 2007. IEEE Int. Symp.* pp. 1609 – 1612.

[34] P. Ranganathan, S. Adve and N.P Jouppi, "Performance of image and video processing with general-purpose processors and media ISA extensions," in *Proc. of the 26th Int. Symp. on Computer Architecture,* 1999, pp. 124 – 135.

Table 1
Subtasks with Different Computational Characteristics

Parameter	Low-level	Medium-level	High-level
Data structures	Characterized by predefined sequences of operations and data access, both input and output consist of simple data structures (such as pixels)	Operates on simple data structures (pixels, symbolic information), both input and output consist of symbolic data	Deal with smaller number of symbols and objects, complex and of variable size
Computational	Involves identical operation on large number of samples	Frequently irregular and involves data dependent decision	Irregular and cannot be predicted in advance
Parallelism	Highly regular and offer large potential of data parallelism	Degree of data parallelism varies dynamically	Opportunity to apply may exist mainly on instruction and task level
Performance	Usually represent the most computation-intensive parts in processing schemes	Lower than low-level tasks	Less demanding for high-level tasks

Table 2
Comparison for VLSI Architecture Approaches

Parameters	VLSI architectures approaches	
	Programmable	Dedicated
Application	Several algorithms can run on the same hardware, suitable for larger application fields	Derived by full adaptation to specific algorithms, suitable for well-defined application with fixed functionality
Flexibility	Enable execution of highly irregular tasks with unpredictable operation flow	Complicated to execute task with high computational demand
Further extension	Possible for future algorithm modification by software changes without hardware redesign	Impossible for future extension without HW modification
Hardware cost	Additional for control units and storage module	Minimum
Power consumption	High power	Kept low

Table 3
Overview of Dedicated Architectures for Video Processing Applications

References	Parameter				
	Process	Chip area (mm^2)	Power consumption	Operating frequency	Applications
[22]	0.18 µm	17.5	1.97 mW	3.3 MHz	Video surveillance
[23]	0.13 µm	196	5.0 W	200 MHz	Beyond HDTV
[24]	0.18 µm	10.2	75.4 mW	200 MHz	Mobile applications
[21]	90 nm	8.08	3.32 mW at 0.8 V	350 MHz	MP3 decoder
[25]	0.18 µm	4270 gates	No data	175 MHz	HD1080i
[26]	0.35 µm	2.8	122 mW	100 MHz	Dual core multimedia SoC
[27]	0.16 µm	121	210 mW	132 MHz	2D MPEG-4 video decoding

Table 4
Overview of Dedicated Architectures for Video Processing Applications

References	Parameter					
	Process	Search range	Number of PE	Block Size	Frequency	Gate Count
[29]	0.18μm	32 x 32	32	16 x 16	166MHz	27.36k
[30]	0.25μm	32 x 32	256	16 x 16 to 4 x 4	100MHz	154k
[33]	0.18μm	64 x 64	256	16 x 16 to 4 x 4	100MHz	154k