Multi-spectral Satellite Image Processing on a Platform FPGA Engine

M. Fleury, R. P. Self, and A. C. Downton
University of Essex
Electronic Systems Engineering Dept.
Multimedia Architectures Laboratory
Colchester, Essex, CO4 3SQ, United Kingdom
Tel.: +44 1206 872817
Fax.: +44 1026 872900
{fleum;rpself;acd}@essex.ac.uk

Abstract

Multi-spectral satellite image sets present a storage and transmission problem. These image sets can also contain line features that typical detection methods do not resolve sharply enough for some purposes. The Karhunen-Loève transform (KLT) presents a solution to both these problems. Firstly, the KLT can be used as a form of lossy data compression, only retaining the higher-order images in the transformed set, as these contain the significant features. The KLT can also substitute as a coder of wavelet transform coefficients, equally for the purposes of compression. Secondly, when an image contains buildings then one technique is to use a KLT in order to highlight these features. The KLT can also provide a characteristic signature of differing regions in multi-spectral imagery and unlike 2D methods prior image fusion is not required.

However, the KLT's kernel is data-dependent unlike related orthogonal transforms. In other words, it has to be re-calculated for each image set to which it is applied. This causes a computational bottleneck when batches of satellite images are processed. Neither conventional uniprocessors or medium-grained parallel machines are well-matched to KLT processing, either calculation of the kernel or subsequent application of the transform to the image set data. This is because the KLT exhibits fine-grained parallelism in two of its three processing stages. An intermediate third stage of the KLT can be mapped to a RISC processor or an ASIC. In contrast to other architectures, a platform FPGA may be viewed as a *fine-grained* parallel-processing architecture. Pipelining of the image sets is then required to overlap data streaming with computation.

Based on a prototype KLT engine, this paper demonstrates that the prototype design can be incrementally scaled to provide real-time processing of satellite image sets, either for the purposes of compression or feature extraction. Further replication of FPGAs to form a KLT engine allows single-pass processing for medium-sized image sets without the need for repeated data streaming. Each FPGA-based KLT could form a 'System-on-Chip'. The result is incrementally scalable to a desired performance. The design has been targeted at Xilinx Virtex series FPGAs, and the prototype produced through Celoxica's DK environment and Handel-C hardware compiler.

1 Introduction

In multi-spectral satellite imagery, large data-banks of images are usually created. For example, the Landsat satellite was capable of producing hundreds of image ensembles per day, each ensemble consisting of seven thematic spectral channels at 512×512 resolution. In [8], there is a set of such

images for the Washington D.C. area. The Karhunen-Loève transform (KLT)[1] can act as lossy data compressor [13], only retaining the higher-order images in the transformed set, as these contain the significant features. In an example from [8], only two of a set of six images from an airborne scanner were retained. Alternatively, when compressing images through a wavelet transform, a coder such as SPIHT [16] suppresses high-frequency coefficients to gain its effect. Unfortunately, satellite images of interest contain line features which this form of compression will blur. In [22], the KLT is proposed as a simple alternative to vector quantization of wavelet coefficients. In addition, conventional edge-detection methods do not resolve sharply enough for some purposes. Besides 2D methods are unsuited to multi-spectral image sets, whereas the KLT is normally applied [5] to an image set or ensemble. When an image contains buildings then one technique is to use a KLT in order to highlight these features. Alternatively, a Gaussian filter or similar low-pass filter precedes an edge detector, for example Sobel or Canny, in order to reduce ringing [8]. The low-pass filter blurs lines, which is not a problem for natural scenes (ones forming a Markov order one field) but becomes a problem when the precise location or configuration of the lines is vital. In contrast, the KLT not only preserves such features but can also provide a characteristic signature of differing regions in multi-spectral imagery [15].

The KLT's kernel is data-dependent unlike related orthogonal transforms such as the discrete cosine transform. In other words, it has to be re-calculated for each image set to which it is applied. This causes a computational bottleneck when batches of satellite images are processed. A platform Field-Programmable Gate Array (FPGA) may be viewed as a fine-grained parallel-processing architecture. The KLT has a matching fine-grained structure in two of its three phases. An intermediate phase of the KLT can be mapped to a RISC or a custom Application Specific Integrated Circuit (ASIC) to exploit pipelined parallelism, which is also required for data streaming of the image set. Based on a prototype KLT engine, this paper demonstrates that the prototype design can be incrementally scaled to provide real-time pre-processing of satellite image sets. A single Virtex-I Field-programmable Gate Array (FPGA) can outperform a high-performance microprocessor by a factor of five [7], when processing image sets of size $6 \times 512 \times 512$ gray-scale pixels. Based on a prototype KLT engine, this paper demonstrates that the prototype design can be incrementally scaled up to provide real-time identification of significant features in an image set.

2 Background

2.1 KLT pipeline

Figure 1 shows the KLT equations represented as a pipeline. The KLT pipeline comprises three distinct processing stages: covariance matrix formation, covariance eigenvector calculation, and eigenspace transform. Since the covariance matrix is generally too small to justify parallel decomposition in general or hardware implementation on an FPGA in particular, this intermediate stage is mapped to a coprocessor. The pipeline can be fully synchronous, corresponding to the systolic model of processing whereby data are 'pumped' across a processing array.

Parallelism arises in various ways. First and obviously, the stages of the KLT pipeline can be partially overlapped in time. For example, the second stage could be elided with the third and final stage as it consumes relatively little time, making for a reasonably balanced pipeline. Secondly there is internal parallelism in the first stage between the covariance and mean-vector sub-stages. Thirdly and most importantly, there is data parallelism across the images' width.

[1]The KLT is known as the Hotelling Transform and the transformed output as the principal components in some texts.

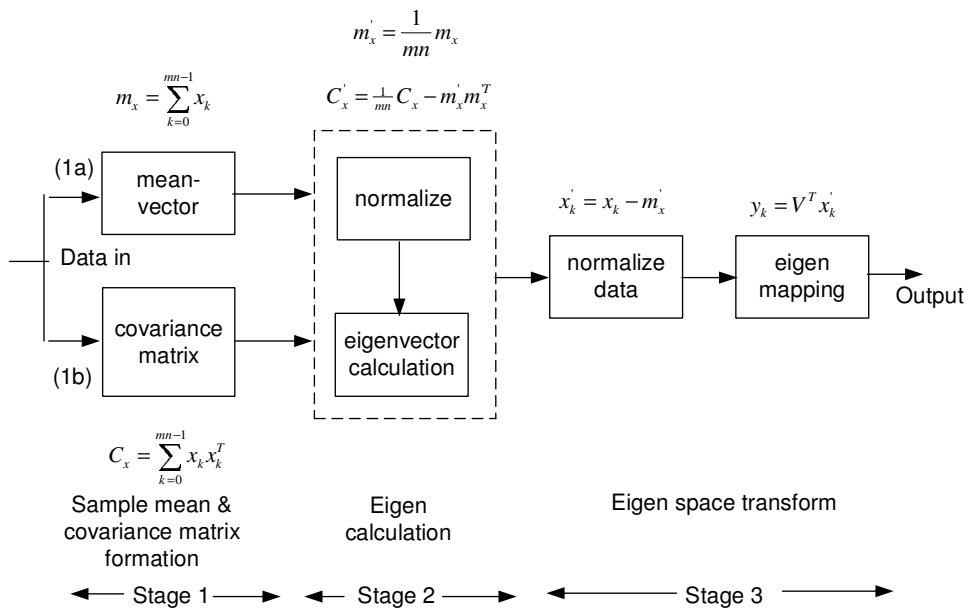

Figure 1: Mapping the KLT algorithm to a pipeline

The KLT algorithm easily maps to a pipeline but to see this requires analysis of the algorithm. Consider a sample set of real-valued images taken from a set of multi-spectral images:

- Create vectors with the equivalent pixel taken from each of the images. If there are D images each of size $N \times M$, then form the column vectors $x_k = (x_{ij}^0, x_{ij}^1, \ldots, x_{ij}^{D-1})^T$ for $k = 0, 1, \ldots, MN-1, i = 0, 1, \ldots, M-1$ and $j = 0, 1, 2, \ldots, N-1$.

- Calculate the sample mean vector:

$$m_x' = \frac{1}{MN} \sum_{k=0}^{MN-1} x_k$$

- Use a computational formula to create the sample covariance matrix by:

$$C_x = \sum_{k=0}^{MN-1} x_k x_k^T,$$

$$C_x' = \frac{1}{MN} C_x - m_x' m_x'^T.$$

  It is important to note that C_x' has rank D, which being the number of images in an ensemble is, in practice, usually below ten.

- Form the eigenvector set:

$$C_x' u_k = \lambda_k u_k, \quad k = 0, 1, \ldots, D-1,$$

  where u_k are the eigenvectors with associated eigenvalue set λ_k.

- Finally, the KLT kernel is a unitary matrix, V, whose columns, vectors u_k (arranged in descending order of eigenvalue amplitude), are used to transform each zero-meaned vector:

$$y_k = V^T(x_k - m_x)$$

2.2 FPGAs and development environment

The present attraction of platform FPGAs is their scale of integration which has risen from about 1 M equivalent system gates [24] in the Virtex-I from Xilinx to about 10 M gates in the Virtex-II, with potential for further integration in line with Moore's law for at least the present decade. The Virtex series are reconfigurable via an (encrypted) bitstream which sets switches on interconnect and look-up tables on the Configurable Logic Blocks (CLBs) arranged as slices (two CLBs per slice in the Virtex family) in a mesh geometry. The feature size is similar to an Application Specific Integrated Circuits (ASIC), now approaching 0.11 μm. On these devices, the I/O bandwidth is also an important feature, making them suitable for data intensive applications. In the Virtex-I , there are 512 I/O pads, and in the later Virtex-II there are 1,108 buffered I/O pins on the largest member of the family of ICs. Additionally, on the Virtex-II Pro between zero and twenty-four Rocket serial transceivers with individual data rates of 3.125 Gbps can be added as embedded cores. Selectable block RAM acts as buffering. While add/subtract is chained, embedded multipliers have eliminated one weakness of earlier members of the family and more specialized, application-oriented families are now being introduced such as the Virtex-4 series, which, for instance, allows a specialization to digital signal processing. Though smaller FPGAs have been placed in satellites as reconfigurable elements, for example in [11], even the lower-powered Virtex 'e' members, are in practice better co-located with the base station.

FPGAs have started to be used as SIMD array elements, taking advantage of their reconfigurability, which allows optimal parallel configurations to meet varying computational needs. The Garp project [9] employs an FPGA as a coprocessor, selecting and extracting suitable computationally intensive parts of an algorithm, for example inner loops, to be targeted onto the FPGA. Alternatively, a two-level multicomputer model has been proposed [19] that will consist of an invariant network-level structure and a variant node-level structure. In one embodiment of this architecture, the first level consists of a Sparc host with a Myrinet NIC (Network Interface Card) [2], itself controlled by a LANai communication coprocessor. The host is responsible for initialization of its computation coprocessor and subsequent message handling. The computation coprocessor was actually composed of four interconnected FPGAs with on-board NIC.

In some cases, the FPGA is dominant, and it may be possible to regard the conventional host as a co-processor. Certainly for the pipeline that we have designed two of the stages are performed by FPGAs and are massively parallel, with the brief intervening stage performed on a conventional RISC. The intervening stage is iterative and (because of the size of the matrix involved) does not warrant parallel decomposition. A pipelined design allows data transfer to be overlapped with computation. Such a design assumes a continuous flow of image sets to be processed. In our prototype design, the microprocessor is a conventional general-purpose microprocessor, but the Virtex-II supports an on-FPGA MicroBlaze 32-bit soft-core RISC processor (running at up to 150 MHz) [25], and the more recent Virtex-II Pro System-on-Chip supports up to four on-FPGA 32-bit PowerPC 405x3 hard cores (at 400 MHz). Both of these RISCs operate in fixed-point format, as is normal for embedded microprocessors, with the PowerPC having fused multiply-accumulate (MAC) hardware, which is particularly appropriate to the design considered in this paper. Clock speeds are limited to those

supported by the FPGA design. The advantage of an embedded core is the anticipated considerable reduction in data transfer latency.

Platform FPGAs, because of the number of equivalent system gates available can be regarded as a new form of computer architecture [6], especially if they are programmed using a *hardware* compiler such as Handel-C [3]. The hardware compiler runs within Celoxica's DK environment [20], which has the 'look-and-feel' of MS Visual Studio. Compared to *silicon* compilers [4], which arise from hardware description languages and thus provide a high-level way of modelling circuits, a hardware compiler maps a software program onto an FPGA, providing a high-level way of modelling program constructs directly in hardware.

The constructs of the program, such as loops, if . . . then statements, expressions and assignments are represented by parameterizable logic templates. Handel-C employs (nested) barrier parallelism, which when replicated in hardware achieves the reported gains in performance. Handel-C's semantics are based on the 'simplicity principle' embodied in Tony Hoare's Communicating Sequential Processes (CSP) [10]. Automatic synthesis takes place through the agency of the Xilinx Foundation Series tools. In fact, the output of Handel-C is either a netlist, which can be in the industry standard EDIF (Electronic Data Interchange Format), or in RTL (Register Transfer Level) VHDL. The Virtex-II MicroBlaze core, for which data widths are parameterizable, is currently programmed and interfaced to the reconfigurable logic through a customized tool chain. However, the Virtex-II Pro PowerPC hosts the Unix-like VxWorks real-time operating system from Wind River Inc. , and, hence, is a flexible development environment.

3 Design prototype

The KLT algorithm is deterministic in nature except than for the eigenvector calculation, if performed iteratively by Jacobi rotation. This implies that a small scale design can be extrapolated to a larger implementation.

3.1 Development board

An economical way to prototype the pipeline is to use a PC-based development board. The RC1000-PP board, Figure 2, has four banks of 2 MB SRAM (Static RAM) memories, which are accessible four bytes individually (or one word) at a time. 'Fast' switch isolation avoids access contention to the memory banks.

The critical part of the KLT algorithm is the eigenvector extraction pipeline stage, as it forms a bottleneck in a pipeline because of the need to perform I/O between stages. In the prototype, the eigenvector calculations were performed on a Pentium host connected through a PCI bus to the RC1000-PP system development board from Celoxica Inc. [21]. This creates a potential I/O bottleneck. In [23], the bandwidth bottleneck on this and similar boards is essentially recognized, as a task-parallel processing model is implemented for the RC1000-PP.

A realistic possibility is to include a specialist eigenvector engine, and indeed systolic designs have long existed for symmetric matrices, for example [18]. Naturally, this latter proposal would result in a customized board, and not a development board. A more likely solution is an upgrade to either a Virtex-II or a Virtex-II Pro, as on-chip RISC cores avoid the need for intermediate data to pass over a PCI bus. The issue of streaming data on and off the custom board still remains and this issue is returned to in Section 4.2.

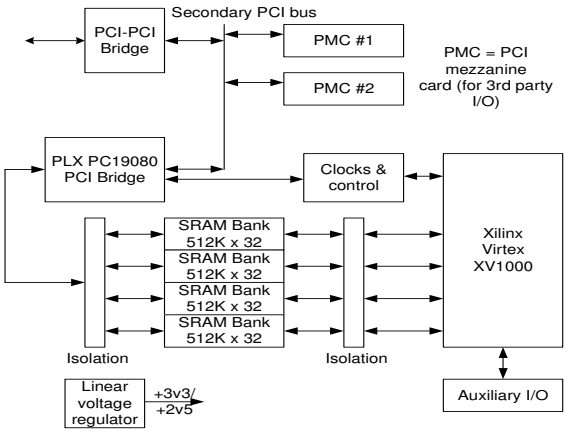

Figure 2: RC1000-PP board

3.2 Data modelling

A pilot implementation of the design was made before performing scaling experiments. In fact, six images with just 20 data items in each were employed. One of the reasons for the small size of the pilot implementation was the I/O bottleneck on the RC1000-PP development board already referred to. Another reason was that performing place-and-route of FPGA components on a full-scale design, even with a high-performance host can be a time-consuming activity.

Apart from feasible clock speed, assuming for the moment a single clock domain design, the other main purpose of the pilot implementation is to establish data widths, which in turn indicates area usage. Fixed-point modelling of floating point numbers was applied. Other possibilities include logarithmic arithmetic [17] and distributed arithmetic [1]. The dynamic fixed-point range for a given accuracy was established through modelling with FrontierDesign's C++ class library, available as part of the SystemC release [12]. The input test data were integers in the range $0 - 100$.

A fixed-point implementation of the stage one covariance calculation was found to require a 22-bit data width, whereas integer (direct) calculation required only 17-bits. This impacts on multiplier size. Specifically, a 17-bit Virtex-I multiplier occupies 17 slices and runs at 39 MHz, whereas a 22-bit multiplier consumes 113 slices, and runs at 28 MHz. More significantly for other designs, the Virtex-II family of FPGAs is equipped, for high-speed operation, with 18-bit twos complement hardware multipliers, maximally 192 multipliers in all. Therefore, direct implementation was chosen for stage one of the KLT pipeline.

Figure 3 shows the data widths found for stage three of the KLT pipeline to avoid overflow errors, and to avoid quantization errors exceeding 0.5% for the test data set. Therefore, the results in Figure 3 are data-dependent. Refer forward to Section 4.1 for scaled data widths replicated to a realistic sized image set. The tests in Section 4.1 are also general for all 7-bit data (dynamic range 0-127).

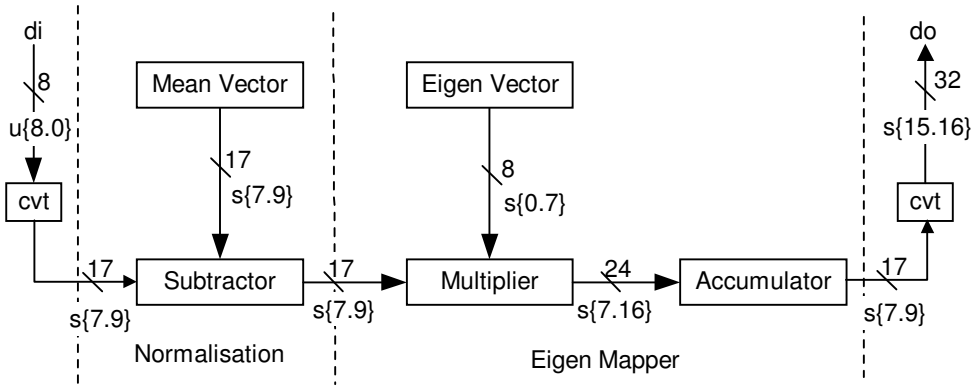

Figure 3: Fixed-point data widths

The data width notation appearing in Figure 3 takes the form: $t(i, u)$, where t is the data type, either `unsigned` (u), or `signed` (s), and i and u correspond to the width specifications of the integer and decimal elements of the data types concerned. The notation `cvt` refers to the task of converting dissimilar data types, which is necessary in order to align decimal points prior to add and subtract calculations.

3.3 Prototype results

Stages one and three, consisting of 953 slices occupied approximately 10% of the Virtex-I capacity for the small implementation. The feasible clock speed was 35.418 MHz, based on the very reliable estimates originating from Xilinx's place-and-route tool. The sub-stages of the pipeline were fully decoupled to increase the clock speed. In addition, the CoreGen soft core multiplier was used to optimize the design.

4 Dimensioning the design

The prototype design was now scaled to take account of parallelism.

4.1 Scaling to full size: Virtex-I Solution

The pilot KLT pipeline, consisting of first and third stages, was replicated n times. When the mean vector and covariance totals are accumulated on the FPGA, then the data width will be larger, respectively increasing from 11 to 26 bits and 17 to 34 bits for a $6 \times 512 \times 512$ dataset. These data widths are for all 7-bit data consisting of integers in the range 0–127.[2]

Table 1 records the results of accounting for the increase in dynamic range, establishing the number of replications that can be supported. In these measurements, no account is taken of coordination of data I/O, as only the first pipeline in every replication set is properly connected to memory banks. For the same reason, the clock frequency should only be taken to indicate an approximate speed.

What the results show is that the number of replications is limited to twelve. This represents an image of just 240 bytes, whereas a representative image is obviously either 128×128 or 512×512 pixels. To accommodate images of these dimensions, many times what can be accommodated in one

[2]As an indication, the data widths for un-accumulated 8-bit data for the mean vector and covariance totals respectively increase from 11 to 13 bits and from 17 to 21 bits.

Replic-ations (n)	Gate count	Number of slices	Capacity (%)	Clock speed (MHz)
1	39,705	1,077	9	25.295
8	270,126	8,065	66	22.444
10	334,306	9,920	80	22.077
12	400,334	11,921	97	20.115

Table 1: FPGA usage with scaled data widths of 26 bits for mean vector and 34 bits for covariance totals

pass on the Virtex, it is necessary to pass repeated streams of data many times through the KLT engine, accumulating results either on the host processor or on the FPGA, before the eigenvector calculation is performed.

Timings for a realistically-sized image ensemble are shown in Table 2. Taking absolute timings for the first stage for 12 iterations of the pilot design resulted in a time of 54 μs and for the third stage 38 μs. The first stage is now dominant, and scaling this result to the full image ensemble, requiring 1,093 passes of data ($(512 \times 512)/240$), gives the result in Table 2. The result in Table 2 assumes that the stages are pipelined and neglects the cost of stage two on a high-end microprocessor, (less than 50 μs on a Pentium IV at 1.7 GHz). To make the timings, the MS Windows High-Resolution Timer API on the microprocessor used Intel's high-resolution 64-bit timers and the privileged `rtdsc` instruction, which in turn examines a hardware register on the Pentium. The accuracy of the timings was further established by averaging over many thousands of tests.

Description	Timing (ms)
Pentium IV (1.7 GHz)	301.0
Virtex-I(20 MHz)	59.0

Table 2: KLT execution times for a $6 \times 512 \times 512$ pixel image ensemble

4.2 A Virtex-II Solution

The scaled timing for the FPGA is five times faster than the Pentium IV, but this pre-supposes that there is a streaming I/O sub-system to sustain the Virtex data requirements.

In the case of the KLT, the Virtex-I design operated at 20 MHz allowing six updates for reasonable DRAM access speeds of 120 MHz. In fact, there is a greater latency because the KLT pipeline takes about sixty 20 MHz clock cycles per stage, allowing about 360 DRAM accesses before a new set of data is required. The data width for one stage of the KLT is 6×12 bytes. Each of the twelve data streams requires a 48-bit width, typically formed from three 16-bit wide DRAM memory modules. A write of the original images from an external source and two reads for each of the KLT stages is required for each of the twelve data streams, making 36 accesses in all, certainly allowing a further set of writes if the processed image ensemble were to be written to sufficiently large memory modules. In fact, a streaming sub-system has been constructed for a different purpose [14] with four 8 Mb $\times 16$ bits access width for Virtex-I modules.

On the RC1000-PP board, access to memory is limited by the clock speed of the FPGA, whereas a stream sub-system can independently control memory access. In general, dual-ported selectRAM

blocks can act as a buffer, so that data can be accumulated before being read-out for consumption by the FPGA engine. Though this requires two clock domains, Virtex-II family FPGAs already have up to 12 parameterizable digital clock managers, each with a maximum frequency of 400 MHz. Multiple clock domains are necessary, as otherwise there is insufficient time for a streaming I/O system to update all KLT pipeline stage data buffers before the next row has to be processed, some 50 clock cycles later. This 50 cycle time frame approximately equates to a four cycle update window per KLT pipeline. Increasing the DRAM controller clock speed to (say) 120 MHz extends the update window to 24 clock cycles, which is sufficient to allow one write to DRAM memory from an external source inputting the next image set, and one read to each pipeline stage. This update procedure is repeated for each of the six row data elements giving a total update time of 18 cycles, well within the 24 clock cycle budget allowed.

Figure 4 shows a system architecture for such a design on a Virtex-II Pro. Notice that stage two is now performed on-FPGA by means of a RISC core, as mentioned in Section 2. The system consists of the streaming I/O and the KLT pipeline sub-system, respectively running at 120 MHz and 20 MHz. The I/O controller coordinates DRAM read/write activity and row data transfers from DRAM to KLT pipeline buffers. The summation unit accumulates the sample mean and covariance partial results generated by each of the twelve KLT pipeline units, and interrupts the microprocessor to start the eigen calculator on the host. Additional system controller logic and further DRAM memory would be needed to store the transformed data output. In practice, careful choice of FPGA device is necessary to ensure that sufficient selectRAM is available. Another issue to bear in mind is that it is not usual to design a board with a higher clock speed than the processing device, and the board design is, therefore, more critical than it otherwise would be.

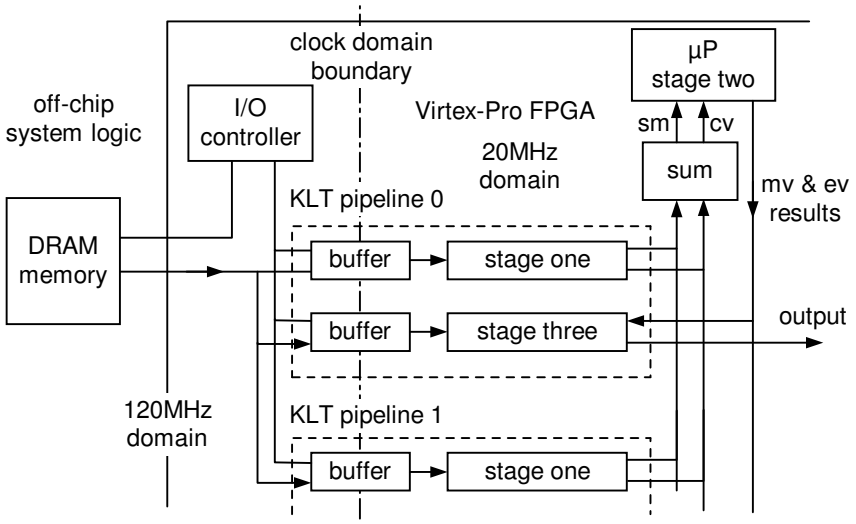

Figure 4: Virtex-II Pro design

Further speed-up of the pipeline in Figure 4 is accomplished by replicating further 'stage one & three' units on Virtex-IIs. One Virtex-II Pro hosting a single stage two unit is required, with an external bus to communicate intermediate results, Figure 5. Table 3 compares the features of the Virtex-I family XCV1000 used on the RC-1000 board, two alternative Virtex-II Pro devices each with just one PowerPC (PPC) 450 core on chip, the XC2VP4 and XCVP7 (as found on the ML300 Evaluation board from Xilinx), and several Virtex-II devices. Apart from reconfigurable logic capacity, the Virtex-II Pro devices differ in the number of high-bandwidth (3.25 Gbps) Rocket I/O serial bus transceiver blocks

present, 4 and 8 respectively for the XC2VP4 and XCVP7. Assuming use of conventional I/O blocks (IOBs), this argues for use of the smaller of the two devices. Of the two Virtex-II devices, the smaller is approximately equivalent to the RC-1000 board device, while the second is the largest of the family and for economies of scale and I/O efficiency is the preferred choice for the design in Figure 5.

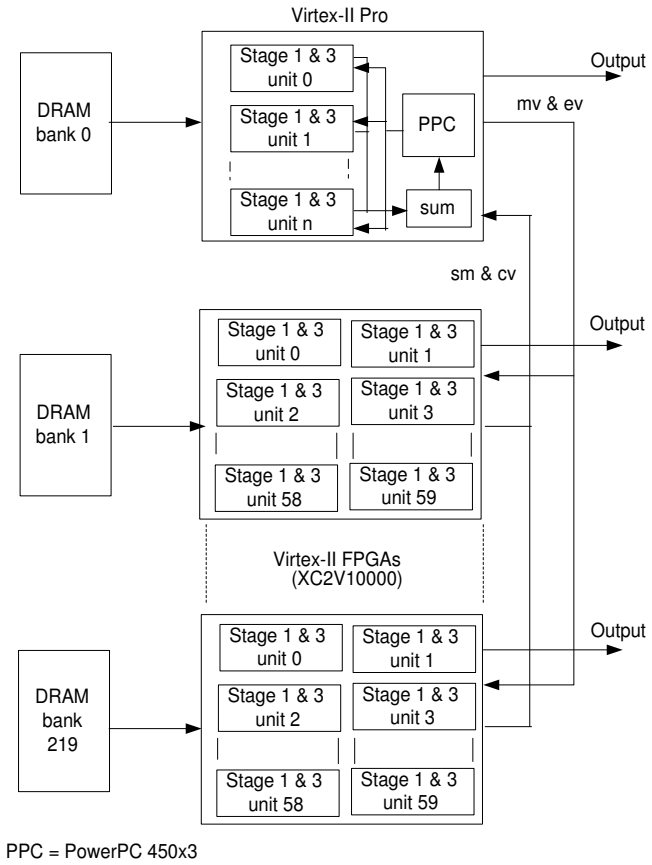

Figure 5: Virtex-II-based single-pass KLT Engine

In Figure 5, the number of pipelines possible on the largest Virtex-II device, the XC2V10000, has been conservatively estimated by extrapolation of slice usage, based on the figures in Table 3. There is a scaling of approximately five times. This implies that about $1093/5 = 219$ data passes would be needed, and that equivalently, to achieve a single pass computation of a $6 \times 512 \times 512$ image ensemble, would require 219 Virtex-II units. The throughput would then be approximately one image ensemble per 54 μs or no less than 18,519 ensemble/s, with a latency of approximately $3 \times 54 = 162$ μs. These figures assume that the clock speed is unaltered in the Virtex-II pipeline over the Virtex-I pipeline, whereas the reduction in feature size from 5-layer 0.22 μm to 8-layer 0.13 μm CMOS will lead to reduced timings.

5 Conclusions

The Karhunen-Loève transform (KLT) is applicable to real-time processing of multi-spectral satellite image sets for compression or to isolate target features. The architecture of general-purpose

Family	Device	CLBs	Slices	IOBs	Multipliers	PPCs	Max. Clock (MHz)
Virtex-I	XCV1000	6,144	12,288	512	0	0	200
Virtex-II	XC2V1000	1,280	5,120	432	40	0	450
	XC2V10000	15,360	61,440	1,108	192	0	450
Virtex-II Pro	XC2VP4	752	3,008	348	28	1	420
	XC2VP7	1,232	4,928	396	44	1	420

Table 3: Virtex-I, Virtex-II, and Virtex-II Pro with embedded PowerPC device statistics

uni-processors or medium-grained parallel processors is not ideally suited to this task. Instead, a fine-grained parallel architecture, as currently exemplified by platform FPGAs, matches the algorithm's structure. Virtex-I platform FPGAs, running at far lower clock speeds, compete favorably with a high-end general-purpose microprocessor, if sufficient data parallelism can be exploited. However, as a single Virtex-I device is insufficient to allow single-pass processing, in practice, it is necessary to repeatedly stream data across the internal replicated pipelines formed by the FPGA. In a prototype KLT pipeline, estimates suggest a maximum speed-up of five over a Pentium IV processor with the largest Virtex-I device for a multi-spectral image set of $6 \times 512 \times 512$ pixels. This prototype design allows an estimate of the speed-up from transfer of the task to Virtex-II devices. The current largest Virtex-II device is approximately five times as big. In part due to a reduction in feature size (0.13 μm), a Virtex-II has approximately doubled theachievable clock speed. Moreover, it is found that the pipelines in the case study scale linearly in slice usage, suggesting a further increase in performance of about ten times. Finally, a KLT engine is possible exploiting the embedded RISC processor on the Virtex-II Pro device to perform second stage processing. In this design, devices are replicated to allow single-pass processing without repeated streaming of data. In fact, the design is incrementally scalable to give a range of performances depending on the number of FPGA deployed.

References

[1] F. Bensaali, A. Amira, and A. Bouridane. An efficient architecture for color space conversion using distributed arithmetic. In *14$^{\underline{th}}$ Conference on Field Programmable Logic and Applications (FPL 2004)*, pages 991–995, 2004. LNCS # 3203.

[2] N. J. Boden, D. Cohen, R. E. Felderman, A. E. Kulawik, C. L. Seitz, J. N. Seizovic, and W-K. Su. Myrinet: A gigabit-per-second local area network. *IEEE Micro*, pages 29–36, 1995.

[3] M. Bowen. *Handel-C Language Reference Manual, 2.1*. Celoxica Ltd, Abingdon, UK, 1998.

[4] S. D. Brown, R. J. Francis, J. Rose, and Z. G. Vransic. *Field-Programmable Gate Arrays*. Kluwer, Boston, 1992.

[5] P. A. Devijver and J. Kittler. *Pattern Recognition: A Statistical Approach*. Prentice-Hall, London, 1982.

[6] M. Fleury, R. P. Self, and A. C. Downton. Hardware compilation for software engineers: an ATM example. *IEE Proceedings Software*, 148(1):31–42, 2001.

[7] M. Fleury, R. P. Self, and A. C. Downton. Development of a fine-grained karhunen-loève transform. *Journal of Parallel and Distributed Computing*, 64(4):520–535, 2004.

[8] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Prentice Hall, Upper Saddle River, NJ, $2^{\underline{nd}}$ edition edition, 2002.

[9] J. R. Hauser and J. Wawrzynek. Garp: A MIPS processor with a reconfigurable coprocessor. In K. L. Pocek and J. Arnold, editors, *IEEE Symposium on FPGAs for Custom Computing Machines*, pages 12–21, 1997.

[10] C. A. R. Hoare. Communicating sequential processes. *Communications of the ACM*, 21(8):666–677, 1978.

[11] A. Koch. A comprehensive prototyping platform for hardware-software codesign. In *Workshop on Rapid System Prototyping*, 2000.

[12] S. Y. Liao. Towards a new standard for system level design. In $8^{\underline{th}}$ *International Workshop on Hardware/Software Codesign, CODES 2000*, pages 2–7, 2000.

[13] D. J. Percival. Compressed representation of a backscatter ionogram database using karhunen-loève techniques. In *Image Processing and its Applications*, 1995. IEE conference publications # 410.

[14] D. Protnove, A. Sulejmanpasic, and C. Ebeling. An emulator for exploring RaPiD configurable computing architectures. In *Field-Programmable Logic and Applications, FPL2001*, pages 17–26. Springer, Berlin, 2001. LNCS 2147.

[15] P. J. Ready and P. A. Wintz. Information extraction, SNR improvement, and data compression in multispectral imagery. *IEEE Transactions on Communications*, 31(10):1123–1130, 1973.

[16] A. Said and W. A. Pearlman. A new fast and efficienet image codec based on set partitioning in hierarchical trees. *IEEE Transactions on Circuits and Systems for Video Technology*, 6:243–250, 1996.

[17] J. Schier and A. Hermanelc. Using logarithmic arithmetic to implement the recursive least squares (QR) algorithm in FPGA. In $14^{\underline{th}}$ *Conference on Field Programmable Logic and Applications (FPL 2004)*, pages 1149–1151, 2004. LNCS # 3203.

[18] R. Schreiber. Solving eigenvalue and singular value problems on an undersized systolic array. *SIAM Journal of Scientific and Statistical Computing*, 7:441–451, 1986.

[19] C. Seitz. A prototype two-level multicomputer, 1996. Project information on DARPA/ITO project available from `http://www.myri.com/research/darpa/96summary.html`.

[20] C. Sullivan. Codesign comes to Virtex-II Pro and MicroBlaze systems. *Xcell Journal*, pages 36–39, Winter 2002.

[21] C. Sweeney and B. Blyth. *RC1000-PP Hardware Reference Manual*. Celoxica Ltd., Didcot, UK, 2001.

[22] J. Vaisey, M. Barlaud, and M. Antonini. Multispectral image coding using lattice VQ and the Wavelet transform. In *IEEE International Conference on Image Processing*, 1998. Paper 712.

[23] M. Weinhardt and W. Luk. Task-parallel programming of reconfigurable systems. In *Field-Programmable Logic and Applications, FPL2001*, pages 172–181. Springer, Berlin, 2001. LNCS 2147.

[24] Xilinx Inc., San Jose, CA. *Gate Count Capacity Metrics for FPGA's Application note*, 1997.

[25] Xilinx Inc., San Jose, CA. *MicroBlaze^{tm} Processor Reference Guide: Embeedded Development Kit*, 2003.