EURASIP Journal on Applied Signal Processing 2005:13, 2018–2025

Scalable IC Platform for Smart Cameras

Richard P. Kleihorst
Philips Research Laboratories, Professor Holstlaan 4, 5656 AA Eindhoven, The Netherlands
Email: richard.kleihorst@philips.com

Anteneh A. Abbo
Philips Research Laboratories, Professor Holstlaan 4, 5656 AA Eindhoven, The Netherlands
Email: anteneh.a.abbo@philips.com

Vishal Choudhary
Philips Research Laboratories, Professor Holstlaan 4, 5656 AA Eindhoven, The Netherlands
Email: vishal.choudhary@philips.com

Harry Broers
Philips Industrial Vision, P.O.Box 218, 5600 MD Eindhoven, The Netherlands
Email: harry.broers@philips.com

Received 19 December 2003; Revised 24 January 2005

Smart cameras are among the emerging new fields of electronics. The points of interest are in the application areas, software and IC development. In order to reduce cost, it is worthwhile to invest in a single architecture that can be scaled for the various application areas in performance (and resulting power consumption). In this paper, we show that the combination of an SIMD (single-instruction multiple-data) processor and a general-purpose DSP is very advantageous for the image processing tasks encountered in smart cameras. While the SIMD processor gives the very high performance necessary by exploiting the inherent data parallelism found in the pixel crunching part of the algorithms, the DSP offers a friendly approach to the more complex tasks. The paper continues to motivate that SIMD processors have very convenient scaling properties in silicon, making the complete, SIMD-DSP architecture suitable for different application areas without changing the software suite. Analysis of the changes in power consumption due to scaling shows that for typical image processing tasks, it is beneficial to scale the SIMD processor to use the maximum level of parallelism available in the algorithm if the IC supply voltage can be lowered. If silicon cost is of importance, the parallelism of the processor should be scaled to just reach the desired performance given the speed of the silicon.

Keywords and phrases: smart cameras, IC architectures, image processing, SIMD, parallel processing, architecture scaling.

1. INTRODUCTION

Real-time video processing on (low-cost and low-power) programmable platforms is now becoming possible thanks to advances in integration techniques [1, 2, 3, 4]. This is relevant to a number of applications such as mobile communications, home robotics, and even industrial image processing [5, 6]. It is important that these platforms are programmable since new applications for smart cameras emerge every month. The complexity (and possible error-proneness) of the algorithms and the fickleness of real-life scenes are also strongly motivating complete programmability. Repeatedly building application-specific ICs or weakly programmable ICs is simply too costly and by far not sufficient for this changing market.

It seems like a daunting task to create programmable hardware that is able to process multimillion pixels per second for complex decision tasks. However, we will show in this paper that this is possible by exploiting the inherent parallelism present in the various levels of image processing.

A desirable "silicon" property of the resulting parallel architectures is that they are easily scaled up or down in performance and/or power consumption whenever the application changes. This not only lowers the need to develop new architectures from scratch for new vision application areas, but it also allows the design team to use the same software suite with only some settings changed in include files. This significantly reduces the overall cost of vision solutions as they can be reused among the portfolio of the producer.

The two types of processors that we propose to be definitely included in smart camera architectures are the SIMD (single instruction multiple-data) massively parallel processor and (one or more) general purpose DSPs [7]. Enough has

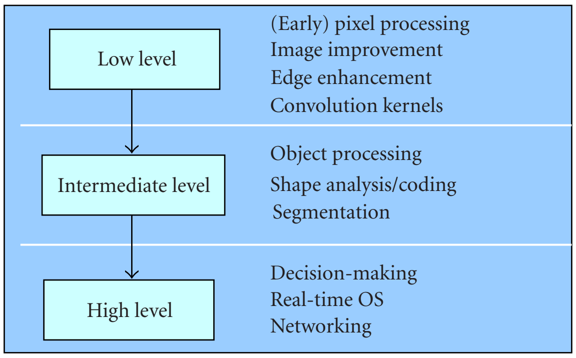

Figure 1: Algorithm classification with respect to the type of operations.

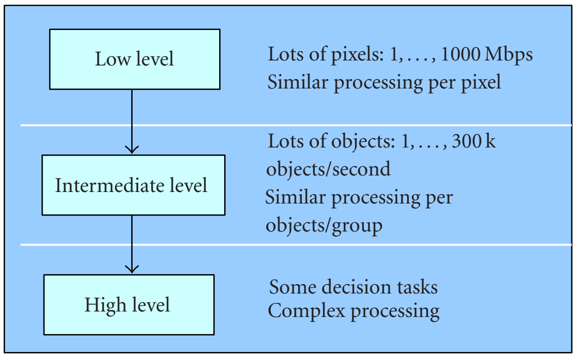
Low level
Lots of pixels: 1, . . . , 1000 Mbps
Similar processing per pixel
Intermediate level
Lots of objects: 1, . . . , 300 k objects/second
Similar processing per objects/group
High level
Some decision tasks
Complex processing

Figure 2: Data entities with processing characteristics and possible ways to increase performance by exploiting parallelism.

been written about general purpose DSPs, so we will mainly focus in this paper on the merits of SIMD processors for the computationally demanding image processing tasks. After reading the paper, it will be clear that they have unique and very clear merits from a silicon and algorithmic point of view for realistic IC implementations of programmable smart cameras.

This paper deals with hardware processor architectures and their properties for scalable platforms. However, designing an easy-to-use software environment for these scalable platforms with multiple processor cores is of course an immense task. Even though all processors might be programmable in a similar language, still (automatic) decisions have to be made to decide on which processor to run which task, when, and how to communicate data. A recent result of the smart camera project is a programming method where algorithmic kernels in the shape of skeletons are used to describe the applications. Based on the available processor cores and memory in the final (scaled) architecture, different skeletons are chosen, linked, and scheduled. The virtue of this design suite is that several skeletons are available for a certain task, optimized for different processor cores. An implementation of a complete application can now survive scaling of the hardware. For more information on this software environment, the interested reader is kindly referred to [8].

The remainder of the paper is organized as follows. In Section 2, we show how well-known processor architectures map to the image processing algorithms. Section 3 deals with some application areas and their performance demands. This is followed by the proposed SIMD-VLIW-based vision platform for smart cameras in Section 4. The scaling properties of SIMD regarding silicon are discussed in Section 5. Finally, conclusions are drawn in Section 6.

2. ALGORITHM CLASSIFICATION

Applications on smart cameras have typically as input images or life video from the observed scene and produce low-rate data output in the form of decisions or identification results. Among the examples that are worked on now are person and object identification, gesture control, event recognition, and data measurement. More challenging applications are expected to come in the near future.

The algorithms in the application areas of smart cameras can be grouped into three levels: *low-level*, *intermediate-level*, and *high-level* tasks. Figures 1 and 2 show the task classification and the corresponding data entities, respectively.

The *low- or early-image processing level* is associated with typical kernel operations like convolutions and data-dependent operations using a limited neighbourhood of the current pixels. In this part, often a classification or the initial steps towards pixel classification are performed. Because every pixel could be classified in the end as "interesting," the algorithms per pixel are essentially the same. So, if more performance is needed in this level of image processing, with (now!) up to a billion pixels per second, it is very fruitful to use this inherent data parallelism by operating on more pixels per clock cycle. The processors enabling this have an SIMD architecture [9, 10]. In SIMD architectures, the same instruction is issued on all data items in parallel. This lowers the overhead of instruction fetch, decoding, and data access leading to economical solutions to meet the high performance and throughput found in this task level.

Also from a power consumption point of view, SIMD processors prove to be very good [11]. The parallel architecture namely reduces the number of memory accesses, clock speed, and instruction decoding, thereby enabling higher performance at lower power consumption [3, 4], see Section 5.1.

An important silicon property of SIMD processors is the regularity of the design. This enables cost effective scaling of the platform for different performance regions by simply increasing or reducing the number of processors in the parallel array. The hardware design can be reused and what is even more important is that inherently the software suite remains the same. So, using SIMD for pixel processing lowers the design cost (and time-to-market) of rapidly changing applications.

In the *intermediate level*, measurements are performed on the objects found to analyze the quality or properties of objects in order to make decisions on the image contents. It appears that SIMD type of architectures can do these tasks, but they are not very efficient because only part of the image (or line) contains objects and the SIMD processors are always processing the entire image (or line). However, they can be

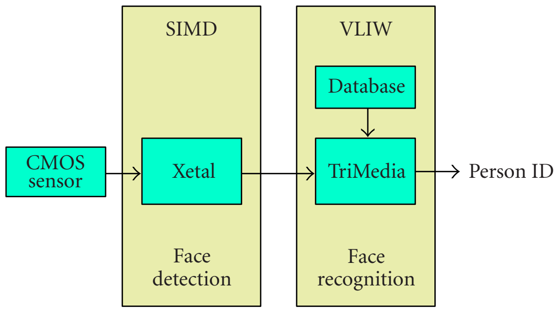

Figure 3: Face recognition actually has two parts, the detection and the recognition. The complete detection part is perfectly mapped to the SIMD processor doing the low-level operations. The recognition part is dealt with by the VLIW processor.

easily performed by a general purpose DSP in the system, if the performance demands are met. If the performance needs to be increased, a viable way is to use the property that similar algorithms are performed on the various objects, which leads to task-parallel object processing on different processors.

Finally, in the *high-level part* of image processing, decisions are made and forwarded to the user. General-purpose processors are ideal for these tasks because they offer the flexibility to implement complex software tasks and are often capable of running an operating system and doing networking applications.

An illustrative application where we can easily indicate the three processing levels is face recognition and detection (see Figure 3). Here the low-level part of the algorithm classifies *each pixel* as belonging to a face or not (face detection). This is mapped on "Xetal," an SIMD processor [4]. The intermediate level determines the features and identifies *each* found face *object*. Finally, in the high-level part of the algorithm, the *decisions* to open a door, sound an alarm, or start a new guest program are taken. The latter two tasks are performed by "TriMedia," a VLIW (very large instruction word) processor [12]. This example was mentioned purely because it shows the three levels quite clearly, more information on this application for the interested reader can be found in [11, 13].

3. APPLICATION PROFILING

The nature of the application determines the scaling that needs to be applied to the SIMD processor and the accompanying DSP in order to meet the performance. Although the basic operations handled by the SIMD processor remain identical, the application dictates the way the computations are performed and the intensity. Consequently, the SIMD architecture needs to be scaled up or down. To a varying degree, all application segments look for better performance at lower-cost and lower-power consumption.

Mobile multimedia processing

This class of applications is characterized by low cost, moderate computational complexity, and low power. The application can usually live with reduced quality of services, for example, lower-frame rate and compressed video streams. The objective from the point of view of product manufacturers is cost reduction which translates to reduction in silicon area and power-efficient computation. The latter objective can often be compromised for the former since mobile devices are active for a short period of time compared to standby duration and the battery energy is wasted mainly in the standby phase.

Thus, to cut costs, the SIMD has to be scaled down from the power-optimal massively parallel architecture. However, as technology shrinks, cost-per-unit area decreases and scaling down becomes less interesting since the tendency in mobile video applications are increased frame sized and rates, and more complex applications.

The computational complexity for this class of applications varies from 300 MOPs for basic camera preprocessing (VGA at 30 fps : $640 * 480 * 30$ pix/s $* 30$ OPs/pix $=$ 277 MOPs) to 1.5 GOPs for more complex preprocessing including auto white balance, exposure time control (about 150 operations per pixel).

Intelligent home interfaces and home robotics

This class of applications corresponds to emerging house robots with vision features. Typical examples include intelligent devices with gesture and face recognition [13], autonomous video guidance for robots, and smart home surveillance cameras. These devices cover the medium-cost range. From a user point of view, the response times and accuracies of the intelligent devices are of high importance and imply faster burst performance. They need to operate in uncontrolled environments (lighting conditions, etc.), and smarter (complex) algorithms are needed to achieve the desired performance. The power aspect remains an issue especially in standalone modules such as battery powered surveillance robots.

Because of the extra intelligence needed in this class of applications, the number of operations per pixel is in the order of 300 or more. This translates to more than 3 GOPs for a 30-frame-per-second VGA size video stream. Even though the devices can exhibit long idle times until they are excited by an event, for example, an intruder in a scene, in their active duration, the same degree of performance is required to guarantee real-time behaviour.

Industrial vision

Unlike the previous two cases, applications in this segment are cost-tolerant and more emphasis is given in achieving top performance sometimes at a given power budget. The emergence of smart cameras has made it possible for various industrial applications to replace large expensive PC-based vision systems with compact and light modules having different vision functionalities. The massively parallel SIMD is very suited for this class of applications from cost, performance, and power consumption points of view.

The scaling here is mainly in accordance with the incoming video format, one can expect a corresponding increase in the SIMD array with increase in the resolution of image sensors.

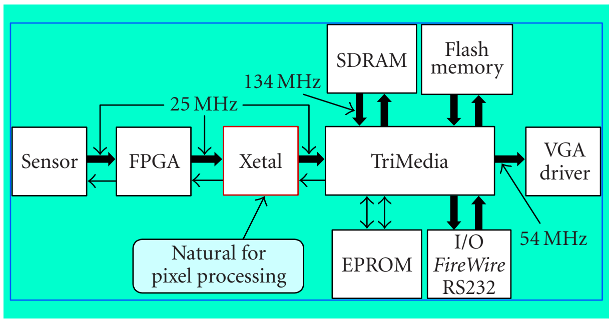

FIGURE 4: Proposed smart camera architecture.

In this class of applications, a number of basic pixel-level operations such as edge detection, enhancement, morphology, and so forth, need to be performed. Because of the high video rates often in excess of 500 fps, the computational complexity is easily more than 4 GOPs.

4. THE SIMD-VLIW VISION PLATFORM

The platform discussion in this section is based on the application profiling and algorithm classification discussed in the previous sections. The different aspects of the algorithmic levels have made us choose for a dual processor approach where the low-level image processing and part of the intermediate level are (as in Figure 3) mapped on a massively parallel SIMD processor "Xetal" [3]. The high-level image processing part and the remaining intermediate-level parts are mapped on a high-performance DSP core "TriMedia" [12]. This DSP has a VLIW architecture where instruction fetch, data fetch, and processing are performed in a pipelined fashion. For most applications, the two processors can be simply connected in series as shown in Figure 4.

The first part of the smart camera architecture is a CMOS image sensor, it can take up to 100 frames per second with a resolution of 1280×960 pixels. The Xetal processor contains 320 pixel-level processors organized as a linear processor array. The array processes an entire image line in 1 to 4 instructions depending on the line width, when each pixel processor is responsible for 1 to 4 columns. Around 1600 instructions can be handled by each processor per line time, depending on the clock setting. It has 16 line memories to save information and a control processor with the program memory to host the programs [3]. Figure 5 shows the architecture of the Xetal processor in more detail.

The TriMedia functions as the high-level DSP in the architecture. It exploits limited instruction-level parallelism and can handle up to 5 operations in parallel.

5. PERFORMANCE-DRIVEN SIMD SCALING

The discussion on application profiling indicates that the performance requirements vary by orders of magnitude from 300 MOPs to more than 4 GOPs. In this section, we address the scaling of a massively parallel SIMD architecture to match the computational complexity of a given application.

The impact of scaling is studied with respect to the different SIMD building blocks and quantified in terms of silicon area and power dissipation. The silicon area directly relates to the cost while the power dissipation dictates the applicability of the device in a system with maximum power constraint.

5.1. *SIMD performance and power scaling*

In mobile applications, both battery life and packaging are important issues. For a given performance requirement, scaling the number of processors in the SIMD machine has direct impact on the power consumption. Power dissipation determines the complexity and cost of packaging and cooling of the devices.

In this section, we look at the performance-power tradeoff when scaling SIMD processors. The analysis is based on the assumption that dynamic power dissipation is the dominant component and uses the well-known CMOS dynamic power dissipation formula: Power $\propto CV^2 f$, where C is the switched capacitance, V is the supply voltage, and f the switching frequency.

Energy consumption of an SIMD machine is decomposed into the following components: computation modules (E_{comp}), communication network (E_{comm}), memory blocks (E_{mem}), and control and address generation units (E_{caddr}). This decomposition helps to identify where most of the power is spent. Equations (1)–(5) give the intrinsic energy model of the components as a function of the convolution filter width (W), number of processing elements (P), number of pixels per image line (N), and the size of the working line memory ($N(W-1)$). The model parameters have been derived based on a high-level power estimation Petrol [14, 15] and were later calibrated with measurement results of the Xetal chip. The basis for choosing a convolution algorithm in our investigation is the fact that convolution involves all the four components (computation, memory, communication, and control) that contribute to energy consumption. In the formulae, it is assumed that the different SIMD configurations operate at the same supply voltage:

$$E_{\text{comp}} = \alpha_1 W^2, \tag{1}$$

$$\begin{aligned} E_{\text{comm}} &= \alpha_2 W\left(\frac{N}{P}+2\right) + \alpha_2 W\left(\frac{N}{P}+2\right) \\ &\quad \times \left[2\frac{P}{N}\sum_{k=1}^{\lfloor W/2 \rfloor}\left(\frac{N}{P}\left\lfloor\frac{k-1}{N/P}\right\rfloor + k \bmod \frac{N}{P} + \frac{N}{P}\right)\right], \end{aligned} \tag{2}$$

$$\begin{aligned} E_{\text{mem}} &= \left(\frac{W^2}{P}+W\right) \\ &\quad \times \left(\beta_1 + \beta_2 N + \beta_3(W-1) + \beta_4 N(W-1)\right) \\ &\quad + \left(\frac{2W^2}{P}\right)\left(\beta_1 + \beta_2 P + \beta_3\frac{N}{P} + \beta_4 N\right), \end{aligned} \tag{3}$$

$$E_{\text{caddr}} = \left(\frac{W^2}{P}+W\right)\left(\beta_5 + \beta_6(W-1)\right) + \frac{2W^2}{P}\left(\beta_5 + \beta_6\frac{N}{P}\right), \tag{4}$$

$$E_{\text{tot}} = E_{\text{comp}} + E_{\text{comm}} + E_{\text{mem}} + E_{\text{caddr}}. \tag{5}$$

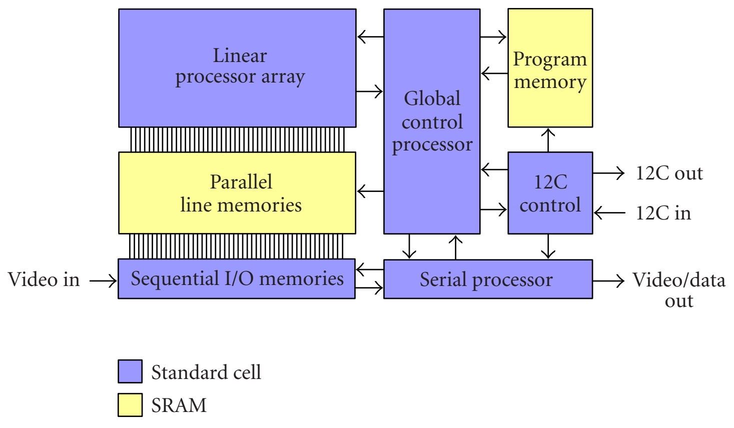

Figure 5: Xetal architecture.

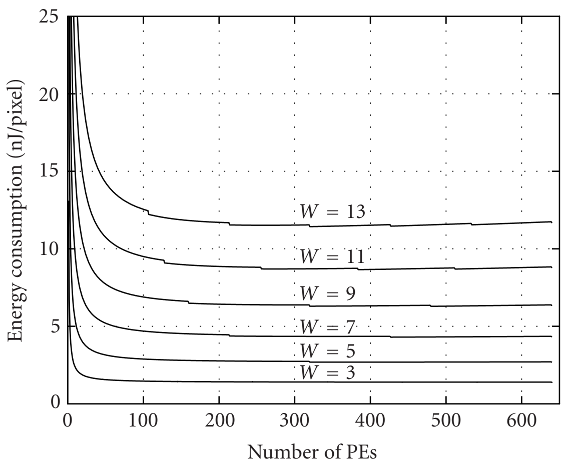

Figure 6: SIMD energy consumption for different filter kernels.

The computation energy (E_{comp}) is a quadratic function of the filter width (W) and does not depend on the number of processing elements as the same number of arithmetic operations needs to be done for all configurations. On the other hand, the other energy components depend on all three dimensions (W, P, N) and have been modelled by a first-order approximation. In essence, scaling the SIMD architecture affects the number of accesses to the working line memories. With each access, a certain amount of energy is consumed by the communication channel, the control, address, and generator, and the memory block. As the number of processors increases, the number of accesses to memory decreases thereby reducing the total energy dissipation. In general, the following relationship holds between the energy components: $E_{\text{mem}} > E_{\text{comp}} > E_{\text{caddr}} > E_{\text{comm}}$.

Figure 6 shows curves of the total energy for $N = 640$ (the number of pixels in a VGA image line) with filter width as parameter. The curves in Figure 6 show that beyond a certain degree of parallelism the saving in energy is very marginal. While the trend is the same, larger filter kernels benefit from increased number of PEs.

It should be noted that configurations with more processing elements (PEs) can handle increased throughput for the same algorithmic complexity. The increase is proportional to P since $P \leq N$ and the filter kernels can be fully parallelised over the pixels in an image line. The minimum number of processing elements needed to meet the real-time constraint is given by $P_{\min} = \lceil C_{\text{alg}} \times f_{\text{pixel}}/f_{\max} \rceil$, where C_{alg} is the algorithmic complexity in number of instructions, f_{pixel} the pixel rate, and $f_{\max}$ the maximum clock frequency of the processing elements (PEs). The clock frequency of the PEs can be increased further by optimisation and pipelining. While optimisation for speed leads to larger PE sizes and increases computation energy dissipation, the impact of pipelining on the SIMD scaling needs to be investigated further.

When throughput comes in the picture, power dissipation becomes a more convenient metric for comparison [16]. Figure 7 shows the power dissipation versus the number of PEs with performance (Per $f = N_{\text{ops}} \times f_{\text{pixel}}$) as parameter. The curves start with $P_{\min}$ computed for PEs designed to run at $f_{\max} = 50$ MHz. Increasing parallelism beyond $P_{\min}$ increases the chip cost (area) which has been traded for lower power dissipation through supply voltage and frequency scaling [17].

In Figure 8, the energy scaling factor is shown which has been used to generate the power dissipation curves of Figure 7. The energy scaling factor (6) has been derived from the CMOS propagation delay model given in [18]. The scaling factor starts with unity at the $P_{\min}$ corresponding to a given performance and decreases as the number of PEs increases. A threshold voltage of $V_{\text{th}} = 0.5$ V and maximum supply voltage of $V_{\text{dd\_max}} = 1.8$ V have been assumed in the plotted curves. To allow for noise margin, the lowest operating voltage has been set to $V_{\text{dd\_min}} = V_{\text{th}} + 0.2$ V:

$$\text{Eratio} = \left[\frac{V_{\text{th}} + (P_{\min}/P)^{2/3}(V_{\text{dd\_max}} - V_{\text{th}})}{V_{\text{dd\_max}}} \right]^2 \qquad (6)$$

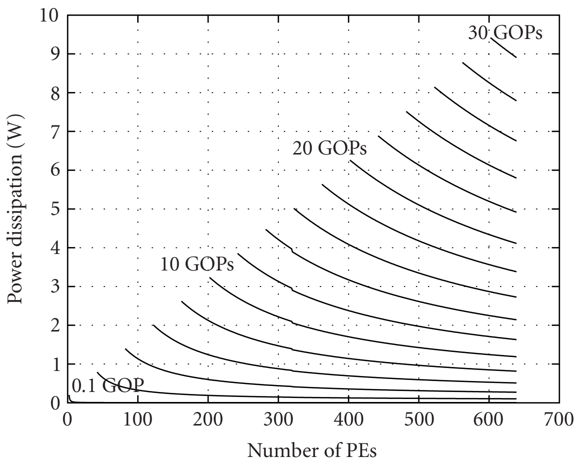

FIGURE 7: Impact of voltage scaling on power dissipation.

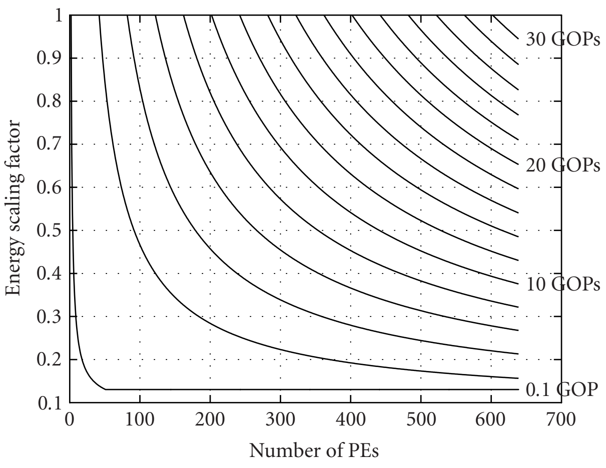

FIGURE 8: Energy scaling factor for varying computational demand.

5.2. SIMD area scaling

5.2.1. Scaling the linear processor array

Due to its regularity, the linear processor array (LPA) can be easily scaled according to the desired performance, cost, and power figures. Silicon area of the processor array can be given by $A_{\text{parray}} = P \times A_{\text{PE}}$, where A_{PE} is area of a single processing element. The array area scales linearly with the number of processing elements (P).

5.2.2. Scaling the line memories

The size of on-chip line memories is dictated by the algorithm to be executed and is independent of the number of PEs used in the SIMD configuration. From area point of view, the line memories do not scale; they only change in layout shape since more pixels would be allocated per PE as the number of PEs decreases. The silicon area contribution of the line memories becomes $A_{\text{lmem}} = N_{\text{lines}} A_{\text{line}}$.

5.2.3. Scaling the global controller

Like the line memories, the global controller also does not scale with change in the number of PEs. This is to be expected since, in the SIMD principle, the global controller is already a shared resource by all PEs. The global controller area is simply A_{gcon}.

5.2.4. Scaling the program memory

The program memory scaling depends on how the impact of loop overhead is addressed. When there are fewer PEs than the number of pixels in a line, algorithms need to be iterated over partitions of an image line. The overhead is associated to the instructions that control the iterations and can be considerable when the loop body is small. A straightforward way of reducing the loop overhead is to unroll the loop by replicating the algorithm code. This results in an increase in the program memory by an amount, that is, a function of the length of the unrolled code and the number of replications. The area of the program memory can be given by $A_{\text{pmem}} = N_{\text{ops}} N_{\text{unroll}} (1 + \gamma) A_{\text{instr}}$, where A_{instr} is the area of a single instruction and γ is the increase factor related to address-width expansion. Instead, one can use special loop control hardware in the global controller to avoid the cost of replicating codes.

5.3. Scaling case study

To summarize the SIMD scaling issue, we collect the components into one cost function described in terms of silicon area: $A_{\text{SIMD}} = A_{\text{parray}} + A_{\text{lmem}} + A_{\text{gcon}} + A_{\text{pmem}}$. Figure 9 shows how the SIMD area scales with the scaling in the number of PEs. The curves are offset by an amount equivalent to the line memory and global controller areas which do not scale. Since the size of the program memory is small relative to the other components, the relative impact of loop unrolling is minimal for large array sizes. For lower number of processing elements ($P < 200$), the area of the nonscaling components dominates. Under this condition, it is sensible not to scale down the number PEs so that the performance loss in the no loop-unrolling case can be compensated for. When combined with the power scaling curves given earlier, the area scaling curve provides a quantitative means for performance-driven SIMD scaling.

6. CONCLUSIONS

In this paper, we have motivated the use of a scalable programmable architecture for video processing in the various applications of smart cameras. It appears that the highly parallel nature of image processing algorithms allows to put the major part of the load on an SIMD type of processors. Another processor in the system has to be a general-purpose microcontroller, microprocessor, or DSP.

We have also shown that the SIMD architecture scales nicely with regard to performance, power consumption and silicon area (cost). All this, while the so-costly program suite remains equal. Exploiting parallelism saves energy and increases performance, but the gain starts to stabilize when a

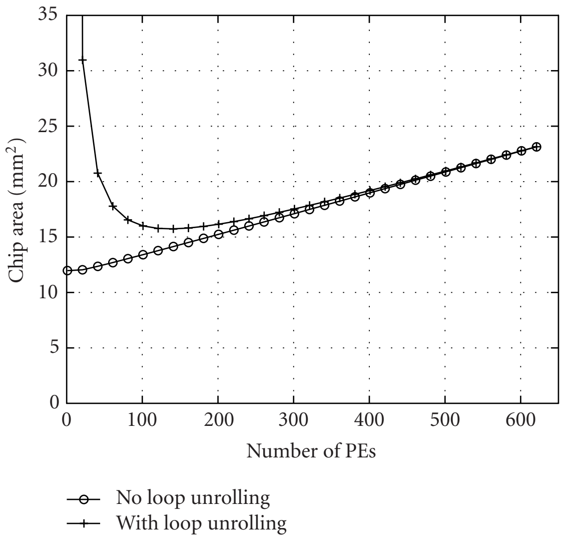

FIGURE 9: SIMD area scaling based on a 0.18 μm design data.

certain number of processors is used. The SIMD area increase is linear in the number of processors with an offset. For practical low-cost applications, a design at maximum silicon speed is preferred with a sufficient level of parallelism to obtain some power savings. When voltage scaling is used, the lowest power consumption for a given performance is achieved at the needed number of processors at the maximum silicon speed of the IC at the lowest voltage supply.

REFERENCES

[1] J. C. Gealow and C. G. Sodini, "A pixel-parallel image processor using logic pitch-matched to dynamic memory," *IEEE J. Solid-State Circuits*, vol. 34, no. 6, pp. 831–839, 1999.

[2] H. Yamashita and C. G. Sodini, "A 128 × 128 CMOS imager with 4×128 bit-serial column-parallel PE array," in *Proc. IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC '01)*, pp. 96–97, 436, San Francisco, Calif, USA, February 2001.

[3] A. A. Abbo and R. P. Kleihorst, "A programmable smart-camera architecture," in *Proc. Advanced Concepts for Intelligent Vision Systems (ACIVS '02)*, pp. 6–13, Ghent, Belgium, September 2002.

[4] R. P. Kleihorst, A. A. Abbo, A. van der Avoird, et al., "Xetal: A low-power high-performance smart camera processor," in *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS '01)*, vol. 5, pp. 215–218, Sydney, New South Wales, Australia, May 2001.

[5] T. H. Meng, "Wireless video systems," in *Proc. IEEE Computer Society Workshop on VLSI System Level Design*, pp. 28–33, Orlando, Fla, USA, April 1998.

[6] RoboCup, "The RoboCup web site," 2002, http://www.robocup.org/.

[7] P. P. Jonker, *Morphological Image Processing: Architecture and VLSI Design*, Kluwer Academic, Amsterdam, the Netherlands, 1992.

[8] W. Caarls, P. P. Jonker, and H. Corporaal, "Data- and task parallel image processing on a mixed SIMD-ILP platform using skeletons and asynchronous RPC," in *Proc. 5th PROGRESS Seminar on Embedded Systems*, pp. 27–34, Nieuwegein, the Netherlands, October 2004.

[9] P. P. Jonker, "Why linear arrays are better image processors," in *Proc. 12th IAPR International Conference on Pattern Recognition*, vol. 3, pp. 334–338, Jerusalem, Israel, October 1994.

[10] D. W. Hammerstrom and D. P. Lulich, "Image processing using one-dimensional processor arrays," *Proc. IEEE*, vol. 84, no. 7, pp. 1005–1018, 1996.

[11] R. P. Kleihorst, H. Broers, A. Abbo, et al., "An SIMD-VLIW smart camera architecture for real-time face recognition," in *Abstracts of the SAFE & ProRISC/IEEE Workshops on Semiconductors, Circuits and Systems and Signal Processing*, Veldhoven, the Netherlands, November 2003.

[12] Trimedia Technologies, "portfolio," 2002, http://www.trimedia.com/.

[13] R. K. H. Fatemi, R. P. Kleihorst, H. Corporaal, and P. P. Jonker, "Real-time face recognition on a smart camera," in *Proc. Advanced Concepts for Intelligent Vision Systems (ACIVS '03)*, pp. 222–227, Ghent, Belgium, September 2003.

[14] R. Manniesing, R. P. Kleihorst, A. van der Avoird, and E. Hendriks, "Power analysis of a general convolution algorithm mapped on a linear processor array," *Journal of VLSI Signal Processing*, vol. 37, no. 1, pp. 5–19, 2004.

[15] R. P. Llopis and K. Goossens, "The Petrol approach to high-level power estimation," in *Proc. International Symposium on Low Power Electronics and Design*, pp. 130–132, Monterey, Calif, USA, August 1998.

[16] A. A. Abbo, R. P. Kleihorst, V. Choudhary, and L. Sevat, "Power consumption of performance-scaled SIMD processors," in *Proc. 14th International Workshop Power and Timing Modeling, Optimization and Simulation (PATMOS '04)*, vol. 3254, pp. 532–540, Santorini, Greece, September 2004.

[17] R. W. Brodersen, A. Chandrakasan, and S. Sheng, "Design techniques for portable systems," in *Proc. IEEE 40th International Solid-State Circuits Conference (ISSCC '93)*, pp. 168–169, San Francisco, Calif, USA, February 1993.

[18] S.-W. Sun and P. G. Y. Tsui, "Limitation of CMOS supply-voltage scaling by MOSFET threshold-voltage variation," *IEEE J. Solid-State Circuits*, vol. 30, no. 8, pp. 947–949, 1995.

Richard P. Kleihorst received the M.S. and Ph.D. degrees in electrical engineering from Delft University of Technology, the Netherlands, in 1989 and 1994, respectively. In 1989, he worked at the Philips Research Laboratories in Eindhoven, the Netherlands, on fuzzy classification techniques for video-speed optical character recognition. From 1990 to 1994, he worked as a Research Assistant, investigating the application of order statistics for image processing in the Laboratory for Information Theory, Delft University of Technology, the Netherlands. In 1994, he joined the VLSI Design Group, Philips Research Laboratories, Eindhoven, the Netherlands. He worked on single-chip MPEG-2 encoding, embedded compression techniques, and parallel image processing. Currently he focuses on programmable architectures for real-time high-performance computer vision. During 2002–2004, he was a Committee Member of the IEEE International On-Line Testing Symposium, and since 2003, he has been a Committee Member of the Advanced Concepts for Intelligent Vision Systems Conference. At present, his interests include digital video processing, with emphasis on coupling expertise between DSP and silicon implementations.

Anteneh A. Abbo received his B.S. degree in electrical engineering from Addis Ababa University, Ethiopia, his M.S. degree in electronic engineering from Eindhoven University of Technology, and his Ph.D. degree from Delft University of Technology, the Netherlands. Since 1999, he has been working at Philips Research Laboratory in Eindhoven as a Senior Scientist. His research interests include signal processing architectures and IC design methodology.

Vishal Choudhary received his B.E. degree in electronics engineering from the S.G.G.S College of Engineering and Technology, Nanded, India, and the M.Tech. degree in VLSI design tools and technology from the Indian Institute of Technology, New Delhi, India. Since 1999, he has been working as a Research Scientist in the Group Digital Design and Test, Philips Research Labs in Eindhoven, the Netherlands. His research interests include dynamic power management for SoC, low power VLSI design, and reconfigurable computing architectures.

Harry Broers received the M.S. degree in electrical engineering from the University of Twente, the Netherlands, in 2000. In 2000, he started to work on image processing for industrial applications at the Philips Centre for Industrial Technology in Eindhoven, the Netherlands. He worked as a Research Scientist, investigating the application of machine vision for assembly systems in the Industrial Vision Department. Since 2003, he has studied face detection methods, automotive applications, and parallel and real-time image processing architectures. Currently, he focuses on architectures for compact intelligent camera systems and image-based human machine interfacing. Furthermore, he is responsible for the image-based measurement systems of the Philips RoboCup Team that competes in an international scientific robotic soccer competition. At present, his interests include image-based perception for intelligent autonomous systems, and compact intelligent camera systems, with emphasis on miniaturization and real-time behavior.