A parallel 2Gops/s image convolution processor with low I/O bandwidth

Öwall, Viktor; Torkelson, Mats; Egelberg, Peter

Published in:
[Host publication title missing]

DOI:
10.1109/ASIC.1995.580688

1995

A Parallel 2Gops/s Image Convolution Processor with Low I/O Bandwidth

Viktor Öwall and Mats Torkelson
vikt@tde.lth.se and *torkel@tde.lth.se*
Dept. of Applied Electronics, Lund University, Box 118, 221 00 Lund, Sweden

Peter Egelberg
egel@agro.se
Agrovision AB, 223 70 Lund, Sweden

Abstract — A customized image processor for real time convolution of an image has been developed. Image convolution requires an extensive amount of calculation capacity and I/O communication which is hard to sustain with standard processors in real time. Therefore, a customized processor has been designed with a tailored architecture. The processors have a total sustained calculation capacity of >2G arithmetic operations/s at 20MHz clock frequency, surpassing that of TMS320C80 for this application due to the tailored architecture.

I. Introduction

The image convolution processor has been designed with the target application of grain quaility assessment [1] but is not restricted to this application. Image convolution is used to detect certain features of an image - in this case the grain - such as outline, color, lines, etc. One convolution detects one feature and if several features are of interest several convolutions have to be performed. Each convolution is computationally very intensive and sufficient calculation capacity is hard to achieve, at reasonable hardware cost, with standard Digital Signal Processors (DSPs) or computers. Therefore, an algorithm specific DSP with a tailored processor architecture has been developed. To allow parallel convolutions, each performing one feature detection, four processor cores are implemented on each chip and a chip select scheme is used to allow parallel calculations with up to sixteen chips. Each pixel value is used in several calculations and to reduce I/O communication a pixel memory bank is placed on chip allowing each pixel to be read only once.

The processor has been developed in a design environment for customized DSPs presented in [2]. No predefined processor cores have been used and a tailored architecture has been assembled with a general tool for hardware assembling [3], a DataPath Compiler (DPC). Besides transforming the processor specification into a netlist the DPC also generates a list of possible micro operations to be performed on the architecture. Image convolution is highly data intensive and consequently the processor cores are dataflow dominated and require a very simple controller. However, the memory bank requires extensive address processing and a corresponding increase in controller complexity. The design environment facilitates the algorithm to be specified in C and compiled into a microprogram. However, since the application requires a high calculation capacity the algorithm has been microprogrammed by hand to achieve an optimal solution. The microprogram has been simulated with a microcode simulator to obtain consistency checks with high level algorithm simulations. A control unit synthesizer, COMA [4], has been used to generate the controller hardware from the microprogram. COMA generates a complete controller together with interconnection specifications to datapath and I/O-units.

The processor has been designed and simulated in the Plessey Classic70000 cell library, which guarantees coherence with the fabricated circuit, using ≈150k gates and is awaiting processing. The design environment does not tie the design to a particular implementation technique or cell library and one processor core has been fabricated and successfully tested in a full custom process.

II. Two-dimensional Image Convolution

Two-dimensional convolution [5, 6], $h{**}x$, is performed by scanning the $K_1 \times K_2$ image, $x(k_1, k_2)$, with the $M_1 \times M_2$ kernel function, or pulse response, $h(k_1, k_2)$, figure 2. A value is calculated for each pixel according to

$$y(k_1, k_2) = \sum_{m_1} \sum_{m_2} x(k_1 - m_1, k_2 - m_2) h(m_1, m_2) \qquad (1)$$

where m_1 goes from $-(M_1 - 1)/2$ to $(M_1 - 1)/2$ and m_2 from $-(M_2 - 1)/2$ to $(M_2 - 1)/2$. This operation is performed for

$$\frac{M_1 - 1}{2} \leq k_1 \leq K_1 - 1 + \frac{M_1 - 1}{2} \qquad (2)$$

$$\frac{M_2 - 1}{2} \leq k_2 \leq K_2 - 1 + \frac{M_2 - 1}{2} \qquad (3)$$

and a filtered output image is produced.

0-7803-2707-1/95 $4.00 © 1995 IEEE

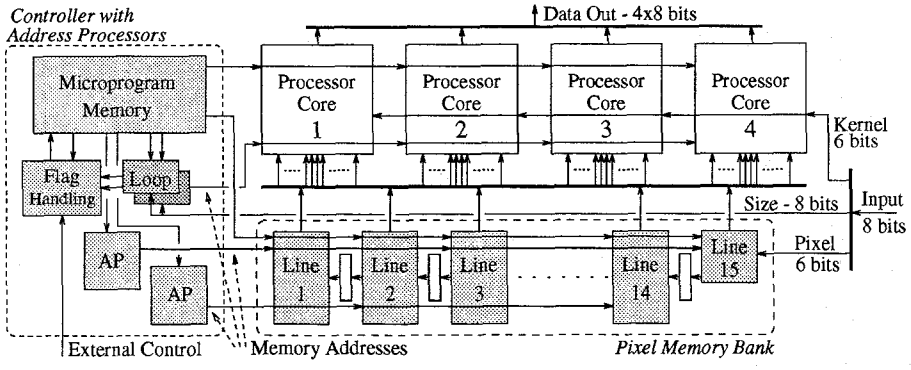

Figure 1: Block diagram of the image convolution processor.

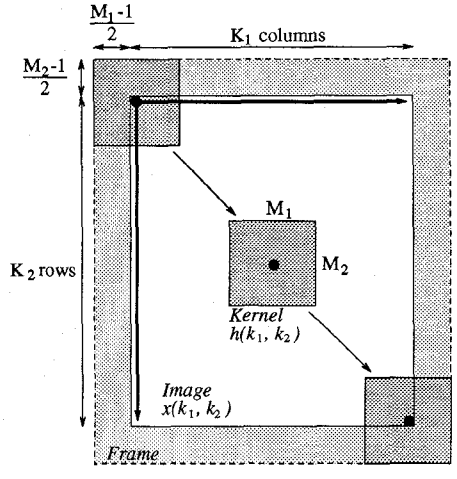

Figure 2: Convolution of an image by a kernel function.

The image is scanned from the upper left corner of the image, first horizontally and then vertically, and one convolution is completed when the kernel has reached the lower right corner of the image. To deal with border effects an image frame is added to the image data according to figure 2. The frame can be set to the background color of the image, or if the image is a sub-image to the values of the pixels in the adjoining sub-images. Additionally, by adding the frame the output image will have the same size as the input image. The size of an image is not fixed but a limitation is put by the processor to a width of 128 pixels and a number of lines of 255. To achieve powerful and versatile filtering the size of the kernel, $M_1 \times M_2$, is set to 15×15 and the kernel function can be changed rapidly.

III. Processor Architecture

The processor performs four convolutions in parallel and produces an output of four pixel operation values each 16th/clock cycle. The processor is divided into six main parts according to figure 1: four identical processor cores, a pixel memory bank, and a controller with address processors. Each processor core performs multiplications for one column of the kernel function, i.e. 15 multiplications, requiring 15 pixel and kernel values each clock cycle (cc). If these values are fed from input ports each clock cycle a high input bandwidth would be required. However, by studying figure 2 and 1 we see that each pixel, except the extreme corner pixels, are used in several calculations; a pixel in the center of the image is used in 15×15 pixel operations. Therefore, the pixel memory bank of figure 1 is used to store pixels values and reduce the input bandwidth. A single 8bits input bus is used for both kernel function, pixel values, and image size. However, kernel and pixel values only use the 6 least significant bits.

A. Processor Core

Each of the processor cores contains fifteen multipliers, handling one line of the kernel function, an adder tree, and an accumulator, figure 3. Each multiplier handles one line of the kernel function and the core architecture enables one pixel operation to be calculated in 16 clock cycles when the pipe is filled, one extra clock cycle for loop counter initialization. The multipliers are 6×6bits Booth multipliers producing a 12 bit output each clock cycle. The calculated values are added in a tree structure of adders and pipeline registers, 14 adders and 30 pipeline registers, and stored in an accumulator. The first adder has a width of 11 bits as the least significant bit of the multiplier output is truncated to reduce the width of the adders in the tree. In the tree structure the number of bits increases to avoid overflow in the adders, i.e. one bit each level. The accumulator has to add values from sixteen clock cycles and the final number of bits in the accumulator becomes 18. When a pixel operation value has been calculated the result is fed to the output register and the accumulator is reset. The inclusion of circuitry to detect how many pixel calculations are above a specified threshold is planned in a future version. This threshold circuitry would reduce the output bandwidth to one or a few values for each image.

When the pipe is filled 15 multiplications and 15 additions are performed in each processor core each clock cycle. The processor is designed for a clock frequency of 20MHz resulting in >2G arithmetic operations/s, corresponding to >2Gmac/s. The adders are of carry-ripple

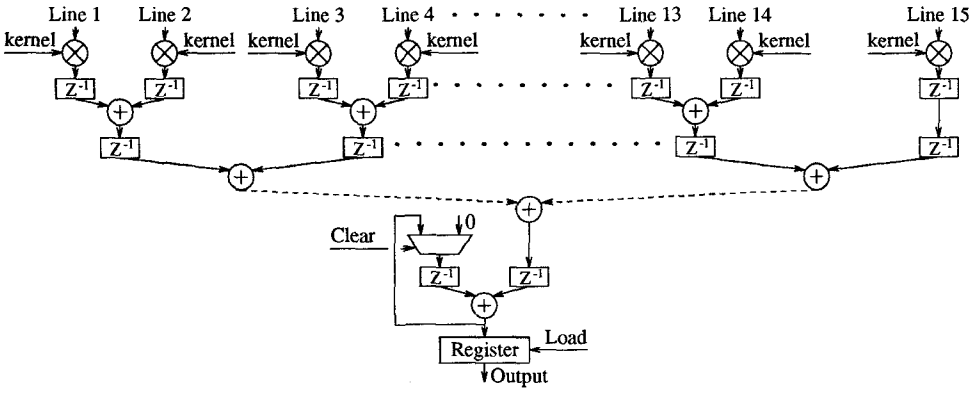

Figure 3: Schematic diagram of a processor core.

type, and the limiting delay is that of the 18 bit accumulator. A change to a faster type of adder, like carry-look-ahead or carry-select, will increase the possible clock frequency but this has not been an objective of the design.

B. *Kernel Function*

Each processor core requires 15 kernel values each clock cycle. Therefore, the kernel function is placed in distributed RAMs throughout the processor cores to decrease the number of data transfers. The kernel function is often used for several consecutive calculations while a change in kernel function should not be too time consuming to achieve versatility. Thus, each RAM is connected to an input port through an input register, figure 1. The processor structure could read one value each clock cycle and the speed at which a new kernel function can be changed depends on external circuitry. The current version of the processor is designed to read kernel and pixel values from a FIFO memory with a read cycle of three clock cycles at 20MHz. Consequently, the loading of new kernel functions for the processor cores requires $4\times(15\times15)\times3$ cc. A decrease in clock cycles could be achieved by either using a faster external FIFO or increasing the number of inputs.

C. *Pixel Memory Bank*

To fully utilize the processor capacity 15 pixel values have to be passed to each processor core each clock cycle. However, each pixel value is used in several calculations, figure 2. Therefore, a Pixel Memory Bank (PMB) is implemented on chip storing all pixel values to be used in consecutive calculation enabling each value to be read only once. To achieve this 14 complete lines and the first 15 pixel values of a 15th line have to be stored on chip, corresponding to the 15 lines of the kernel. These pixel values are stored in 15 line memories connected in series with intermediary registers, figure 1. The size of the line memories limits the width of the image while the number of lines is limited by the width of the input bus. These are set to 128 and 255 respectively not including the frame. The size of an image is read at the beginning of each convolution through the 8bits input port.

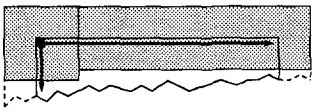

Figure 4: Initial filling of the pixel memory bank.

At the beginning of a convolution the PMB is filled according to figure 4 to allow the first pixel operation. Pixel values use the same FIFO and input port as the kernel values with the corresponding limitations in speed. As the kernel moves through the image only one pixel value is read for each pixel operation, one value is shifted between the line memories, and one value is discarded. Hereby, the input bandwidth is reduced from 15 pixels/cc to 1 pixel/16cc during the calculation phase of the microprogram. At the end of a line the first 14 pixel values of the next line are read at a maximum speed of 1 pixel/3cc.

IV. Microprogram and Controller

The processor cores require a very simple controller with just a single control signal while the PMB and the kernel RAMs require extensive address calculations and loop control. Therefore, a control unit synthesizer, COMA [4], has been used which synthesizes a complete controller from a microprogram. Two address processors calculate addresses in parallel to the PMB. However, by increasing the clock cycle count a single address processor could be used. Loop control is separated from the address calculations to increase the throughput of the processor, and two loop counters are used, figure 1.

By using COMA it is easy to change the microprogram if the specification changes and synthesize a new controller. Modifications of the microprogram could include a flexible kernel size and communication with faster external circuitry. COMA has a range of controller architectures to choose from; ranging from a simple FSM controller to a decomposed hierarchical controller structure. The size of the controller depends both on the implementation technique of the control logic, i.e. PLA, random logic, etc., and the structure of the microprogram. In the designed

processor in Plessey Classic70000 cell library the control logic has been implemented by synthesizing random logic from truth tables using the Synopsys software. The controller architecture resulting in the lowest gate count was a decomposed controller structure with separate microinstruction logic, generating control signals, and sequencing logic, handling subroutine addresses, figure 5. The address field of the microinstruction memory was divided into two separate parts, a subroutine and a program counter, which resulted in a lower gate count than that of a single address field [4]. A separate flag handling module was used to handle conditional statements and loops depending on both external signals and signals from loop counters.

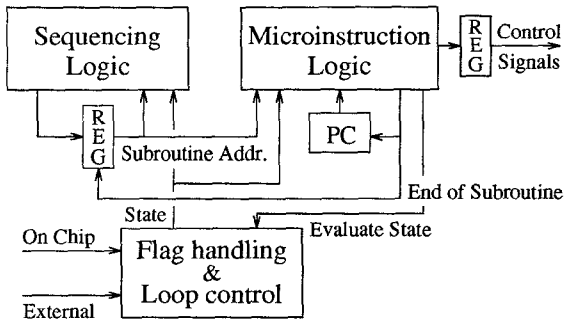

Figure 5: Controller architecture.

V. Results

The image convolution processor has been designed and extensively simulated in the Plessey Classic70000 cell library and awaits fabrication. However, the Plessey design system guarantees the fabricated circuit to coincide with simulations and the design is presented in this paper. The processor has been designed and simulated for a clock frequency of 20MHz. The useable gate count for the processor cores, the control structure, and registers in the pixel memory bank totals 42k gates while the final gate count depends significantly on the implementation of RAMs, i.e. the pixel and the kernel memories. In Classic70000 the final gate count become ≈150k gates.

As the design environment does not tie the design to a particular implementation technique or cell library, one processor core with kernel RAMs was fabricated and successfully tested in a full custom process, figure 6. Note the large number of required I/O-ports. A one micron standard CMOS technology was used and the circuit contains $>50\,000$ transistors. The fabricated processor core had 16 multipliers instead of 15 and the die area is $\approx 8 \times 6.5$ mm^2.

VI. Conclusion

An algorithm specific digital signal processor for image convolution has been designed for the Plessey Classic70000 cell library with a calculation capacity surpassing that of standard processors. Image convolution requires a high calculation capacity as well as a large amount of data for each calculation. Four tailored processor cores enables four convolutions to be performed in parallel with a total sustained processing capacity of >2G arithmetic operations/s at a clock frequency of 20MHz. Powerful and versatile filtering is achieved with a large kernel of 15×15 values and the use of an on-chip pixel memory bank decreases the input bandwidth and enables the use of a single input bus with preserved processing capacity.

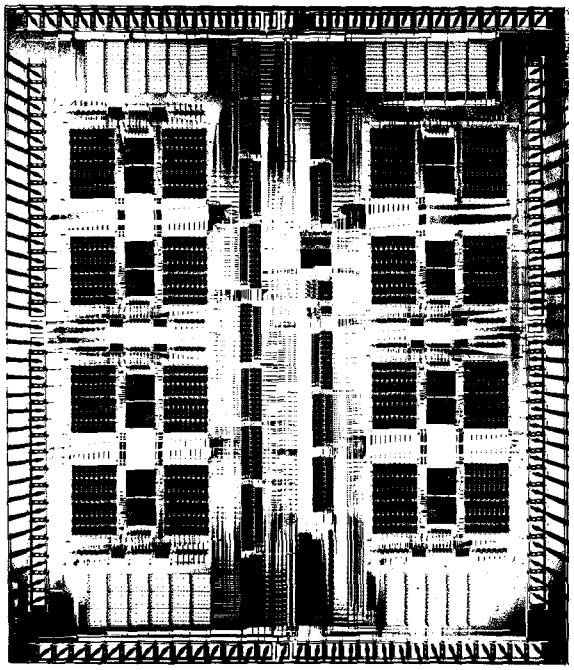

Figure 6: Die photo of the processor core.

References

[1] P. Egelberg, O. Månsson, and C. Peterson. "Assessing Cereal Grain Quality with a Fully Automated Instrument Using Artificial Neural Networks Processing of Digitized Color Video Images". In *Proc. of the SPIE's Int. Sympsium on Photonic Sensors & Controls for Commercial Applications, Boston*, November 1994.

[2] V. Öwall *et.al.* "A GSM Speech Coder Implemented on a Customized Processor Architecture". In *Proc. of the IEEE ISCAS*, pages 235–238, 1993.

[3] L. Brange and M. Torkelson. "A Basic CAD-tool for module generation". In *Proc. of ESSCIRC*, 1989.

[4] V. Öwall. *Synthesis of Controllers from a Range of Controller Architectures.* PhD thesis, Lund University, Sweden, December 1994.

[5] J. S. Lim. *Two-Dimensional Signal and Image Processing.* Prentice-Hall, 1990.

[6] W. K. Pratt. *Digital Image Processing.* John Wiley & Sons, Inc., 1991.