EXHIBIT 1041

K. CHEN, A. ASTROM and P.E. DANIELSSON, "PASIC: A Smart Sensor for Computer Vision," Pattern Recognition, 1990. Proceedings," 10th International Conference on pp. 286-291, 16-21 June 1990

PASIC. A Smart Sensor for Computer Vision

Keping Chen
LSI Design Center

Anders Åström and Per-Erik Danielsson
Department of Electrical Engineering

Linköping University, S-581 83 Linköping, Sweden

Abstract

The PASIC prototype chip contains 256 x 256 photo sensors, a linear array of 256 A/D-converters, two 256 8-bit shift registers, 256 bit-serial ALU, and a 256 x 128 bit-dynamic RAM. It is claimed to be a viable architecture for low-level vision processing. The processors operate in SIMD-mode at 20 MHz. When executing an edge-detection algorithm it is shown that the 256 processors are capable of an output rate of 3 Mpixel/second.

1. Introduction

The advancement of VLSI technology will inevitably give birth to integration of sensors and processors in systems for computer vision. The two overriding issues for such architectures are:

i) Efficiency when executing a core subset of low level algorithms.

ii) Adaptation of the architecture to various constraints of VLSI technology.

An early approach in this direction initiated at our university is LAPP [1]. This is now a commercially available chip which features a linear array of 128 photo sensors, thresholding units and simple processors for binary logic.

PASIC (Processor, A/D-converter, Sensor Integrated Circuit) is a prototype for a more general purpuse smart sensor. The general floor plan is shown by Fig. 1. Most of the area in this prototype is occupied by the photo sensors. Their data are read out, one row at a time, over 256 vertical lines which connect to a linear array of comparators. Together with some digital circuits these comparators form a linear array of Analog-to-Digital converters.

The remaining part of the chip is purely digital and contains two 8 bit wide 256 stage shift registers, 256 bit-serial processors and a 256 x 128 bit dynamic RAM. These parts will be described in the following sections. Throughout the design both analog and digital circuitry has been implemented in double metal 1.6 µm CMOS technology.

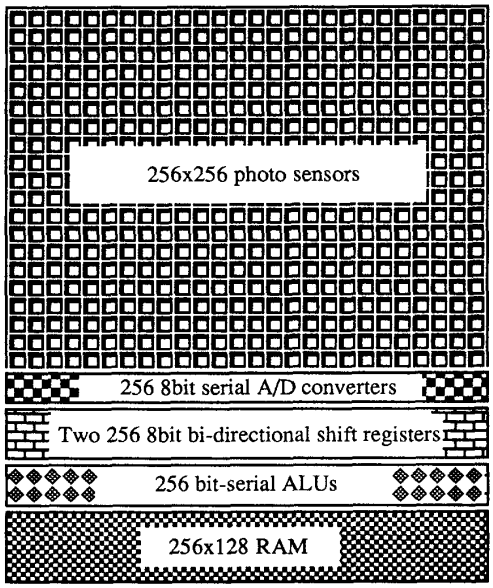

Figure 1 PASIC floor plan

2. The parallel A/D-converter

A conventional semiconductor video camera has to provide for high speed (10 MHz) transportation of tiny charge packages from the sensor array to the analog video output. In PASIC this transportation problem is greatly alleviated since it takes place at much lower rate, one row at a time in parallel. See Figure 2. For each new row of analog data the 256 comparators are comparing the individual signal levels to a common ramp signal generated in synchronism with stepping a counter. The eight bits of the counter are continuously copied to each of 256 8-bit registers up to the

CH2898-5/90/0000/0286$01.00 © 1990 IEEE

point where the ramp value exceeds the individual signal level. The 256 A/D-conversions are completed after a full ramp generation. With 20 MHz clock rate such an event takes place in 256 · 0.05 = 12.8 μs.

The accuracy in the present design is limited to 8 bit/pixel. It should be noted, however, that this precision can be traded for speed. The A/D-conversion can be speeded up by changing the slope of the ramp signal. For instance the conversion time for 256 x 256 pixels is reduced from 3276.8 μs for 8 bits to 1638.4, 819.2, 409.6, 204.8 and 102.4 μs respectively for 7, 6, 5, 4 or 3 bit accuracy. In many vision tasks such a programmability for speed-precision trade-off is very valuable.

The possibility to increase the signal-to-noise ratio by lowering the frame rate is self-evident.

The registers of Figure 2 are actually designed as cells in a 8 x 256 bit memory. See Figure 3. The read out to the memory bus is controlled by a 3-to-8 decoder.

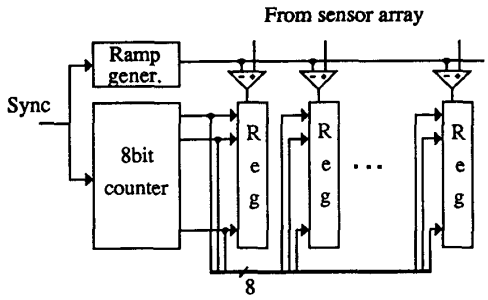

Figure 2 The parallel A/D-converter

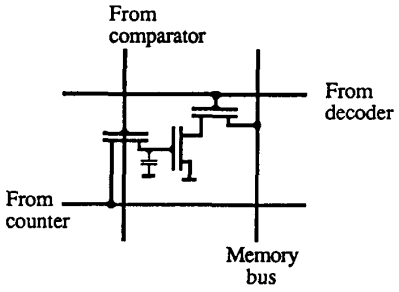

Figure 3 A bit position in the A/D converter registers

3. The memory bus architecture

We believe that convincing arguments for organizing the A/D-conversion of an image sensor a linear array were given in the previous section. By pure geometrical constraints it is then hard to avoid a linear array organization for the rest of the system. Putting 256 processors on a chip immediately requires the processors to be bit-serial devices and that they should work in SIMD-mode.

Many bit-serial processor architectures have been presented in literature, for instance CLIP[2], DAP[3], MPP[4], GAPP[5] and the Connection Machine [6]. The details of the GAPP is well known since it has been commercially available as a chip. The GAPP processor is very simple, having four one-bit registers per chip and logic consisting of a full adder/subtractor. Flexibility and generality of the design stems from a rather extensive multiplexor system that connects the ALU outputs, the register outputs, the memory output and the four NEWS neighbors to the memory and the four register inputs. In this network several simultaneous bit transfers can take place. For instance, one bit can be received from East, another transferred Westbound, the sum output stored in a memory location, carry saved in the C-register, input bit shifted into the communication register, all in the same cycle.

This seems very efficient but a closer examination reveals that processing speed is nearly always limited by one single factor, the memory bandwidth. In a typical two-input one-output operation like addition the execution time is therefore 3b + 1 cycles where b is the wordlength. We found it necessary to design a processor even simpler than the GAPP. In particular, we found it impossible to squeeze a GAPP type processor into the narrow 40 μm vertical slot, given by the pitch of the sensor columns.

In our VLSI single chip smart sensor, all units, the registers in the A/D converter, the bit-serial ALU array, the shift registers and the memory cells are hooked to the single-bit bus as shown in Figure 2. All these units communicate with the bus bitwise, that is, at one instance only one bit in each unit is input/output to/from the bus. The bit-serial ALU and the shift registers can be viewed as logic embedded in a memory and they appear as memory ports to the bus. The architecture where all computation units are memory-like and are centered around buses are referred to as a memory-bus organized architecture as shown in Figure 4. The control and addressing to each unit can be combined in a single address decoder broadcasting to all column of units. The instruction word (including both address and control signals) is 26 bit wide, which can be divided into five fields.

AD: 4bit A/D converter address/control
- AD=0–7 0–7th bit to Bus
- AD=8 data latch
- AD=9 count

SRA: 5bit shift register 1 address/control

SRB: 5bit shift register 2 address/control
- SR=0 shift right
- SR=1 shift left
- SR=2 data latch
- SR=3 circle right
- SR=4 circle left
- SR=5–12 Bus to 0–7th bit register
- SR=13–20 0–7th bit register to Bus

ALU: 4bit ALU input/output address/control
- ALU=0 B+C to Bus
- ALU=1 BC to Bus
- ALU=2 carry to Bus
- ALU=3 Bus to C
- ALU=4 AC+BC to Bus
- ALU=5 AC+BC to Bus
- ALU=6 Bus to B
- ALU=7 B to Bus
- ALU=8 Bus to B
- ALU=9 B⊕C to Bus
- ALU=10 Bus to A
- ALU=11 Bus to A
- ALU=12 Bus to A & carry feedback
- ALU=13 sum to Bus
- ALU=14 sum to Bus & carry feedback
- ALU=15 no operation

CMA: 8bit memory address/control
- MA=0 0 to Bus
- MA=1–127 1–127 bit to Bus
- MA=128 no operation
- MA=129–256 Bus to 1–127 bit

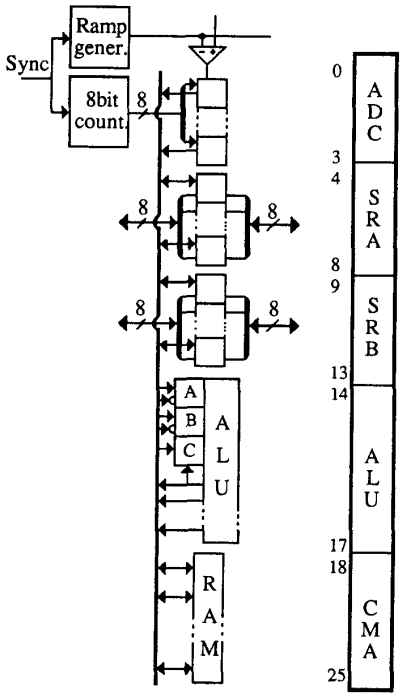

Figure 4 The architecture and instruction format of PASIC

The key element is the ALU. It has three one-bit registers: A, B and C that feed the logic. The logic is basically a full adder with SUM and CARRY output, activated by the addresses PA = 13, 14 and 2 respectively. In addition, however, the ALU-logic is tapped in six other places to make six more Boolean functions available for each given input A, B, C. With little extra hardware cost these functions greatly enhances the overall performance of the processor and in effect replaces bit-shuffling over multiplexors in GAPP. The number of Boolean functions is further extended by the ability to load the A- and B-registers with both inverted and non-inverted data from the bus. For instance, the CARRY output becomes a Borrow = $\overline{X}B + \overline{X}C + BC$ by inserting A := $\overline{X}$.

Not shown in Figure 4 is the Global OR function in PASIC. It is implemented as a wired OR combination of the C-register output from all 256 processors.

4. Algorithm implementation

Most low-level algorithms for image processing invol neighborhood operations. The necessary horizont neighbor communication is provided by the bi-directior shift registers. For grayscale (8-bit) images the obvio sequence of events is to load the shift registers over the b from the A/D-converter or the memory in eight cycl After one shift cycle, these 256 x 8 bits are available to t neighbors at left or right. Summing two 8-bit pixels fr neighboring columns in this manner takes 8 + 1 + 3·8 + = 34 cycles where 9 extra cycles stem from t communication. Note, however, that if the result is to used in a subsequent neighborhood operation we shou store the result into the shift register rather than t memory. The extra delay for neighborhood communicati is then virtually eliminated.

The memory in PASIC (present version) is too small allow for more sophisticated algorithms. In fact, since cannot harbor a full 256 x 256 gray level image we can o

implement algorithms in **bit-serial, row-parallel pipe-lined versions.** If an algorithm is specified using intermediate image buffers we have to reformulate the algorithm and execute all the stages of the algorithm row by row. See Fig. 5. We can hold a limited number of inputs and intermediate rows in the memory (128 bits per column). However, we have to produce one complete output row before we can accept a new row input from the image sensor.

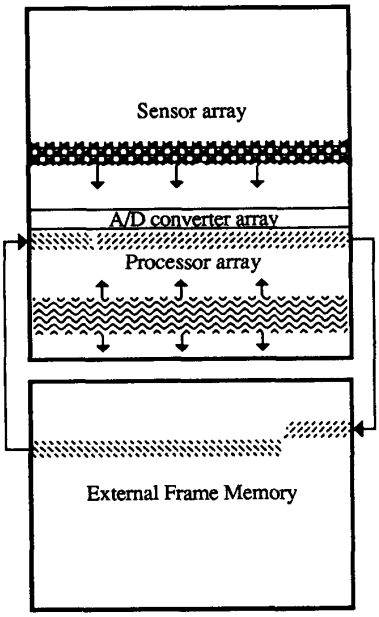

Figure 5 Row parallel pipe-lined processing

5. Application example: Edge detection

A rather traditional edge detection procedure is described by Figure 6. A cycle count for the various stages of this algorithm implemented in PASIC for a 256 x 256 8-bit image reveals the following. (See also Figure 4).

Input of one new row to memory takes **8 cycles**.

Median, 1 x 3 "vertical" neighborhood: Three 8 bit comparisons are necessary. Each comparison consists of one subtraction, 16 cycles, followed by one multiplexing operation using the $A\overline{C} + BC$ logic of the ALU to save the larger (and/or the smaller) of two numbers. The first multiplexing is a two-way one and takes 33 cycles, the following ones are one-way and takes 25 cycles each. The whole operation totals 16 + 33 + 2(16 + 25) = **131 cycles**.

Median, 3 x 1 horizontal neighborhood. The result of the previous operation was loaded into the shift registers as well as to the memory. The computation continues as for the vertical neighborhood except for 1 extra shift cycle to bring in the left neighbor and, later on, 2 extra cycles to bring in the right neighbor.
Total: 131 + 3 = **134 cycles.**

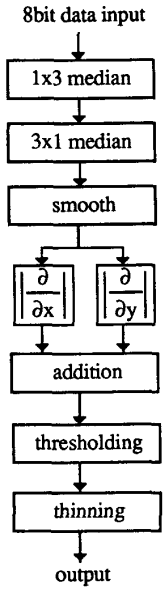

Figure 6 The edge detection algorithm

Smooth is performed by adding two horizontal neighbors followed by adding two vertical results. Input data for the smoothing stage is already in shift registers. Total execution time is 25 cycles for the first 8 bit add operation, 28 cycles for the following one which totals **53 cycles**.

Derivative magnitude in the x-direction begins with subtraction on 10 bit data that takes 31 cycles + 3 extra cycles to bring in the two most significant bits, which do not fit into the shift register. Then follows an addition with vertical neighbor to yield a 12 bit Sobel result f_x which takes 34 cycles. Finally, the magnitude is obtained by inverting the 11 LSB bits if the sign is negative. This can be implemented in 24 cycles by loading the sign bit in the A-register, a zero in the B-register and then streaming in the 11 LSB-bits while saving the SUM output from the ALU.

To compute the magnitude of the Sobel response in the y-direction is identical in complexity. The total Sobel

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.