Architecture and Design of an Associative Processor Chip for Image Processing and Computer Vision

Project Thesis

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

Leonid Yavits

SUBMITTED TO THE SENATE OF THE TECHNION - ISRAEL INSTITUTE OF TECHNOLOGY

AV 5754 HAIFA JUNE, 1994

This project paper was carried out in the faculty of Electrical Engineering under the supervision of Dr. Ran Ginosar

Acknowledgment

I would like to thank Dr. Ran Ginosar for teaching me most of the practical things I know about VLSI, computer architectures, and general research methods; Prof. Smil Ruhman for his many suggestions and help with different hardware and software aspects of this project; Dr. Avidan Akerib for funding this project, theoretical ideas, encouragement and support.

In addition, there are many people who have aided me in this work. I would like to thank Goel Samuel, for his technical support, that really made this project possible; Amir Morad and Nitzan Poiliz for taking a part in practical design; Eial Avreh, Tal Keren Zvi and Rafi Frid for their great help in all technical aspects, including chip definition, logic verification and circuit design.

Table of Contents

Abstract

Associative processing systems draw a growing interest in the field of computer vision. The ASP100 associative processor chip was designed primarily for implementation of compact, easily extendible modular SIMD parallel processing systems for real-time image processing and computer vision tasks. The processor architecture, including 1K word associative memory array, parallel image I/O buffer and peripheral associative mechanisms, were designed and implemented using VLSI CAD tools. The associative memory array consists of two parts, one of them implementing the main processing and storage unit, while the second one serving as an image I/O buffer operating in parallel with processing of images. The peripheral associative mechanisms include a fast response counting circuit, a select first circuit and a some/none response circuit. Chip design and simulation combined two different techniques: semi-custom (cell based) and full-custom. The design of basic cell of CAM array is described including detailed analysis of design goals and technology limitations. Physical (power consumption, silicon area, clock generation) and design (interface between full custom and cell-based parts, multilevel simulation) considerations were examined in detail. The ASP100 performance is estimated while performing some low-level and mid-level vision tasks.

List of symbols

VLSI	Very Large Scale Integration
CAD	Computer aided design
SIMD	Single Instruction Multiple Data
MIMD	Multiple Instruction Multiple Data
CAM	Content Addressable Memory
FIFO	First In First Out
I/O	Input/Output
ARTVM	Associative Real Time Vision Machine
CAAPP	Content Addressable Array Parallel Processor
MPP	Massively Parallel Processor
ASP	Associative String Processor
CAPRA	Content Addressable Processor/Register Array
ASP100	Associative processor chip
ARRAY	Main CAM array of the ASP100
FIFO	the ASP100 image I/O buffer
TOP	the ASP100 control unit
BOTTOM	the ASP100 output unit
SIDE	the ASP100 processing array
PE	Processing Element
ALU	Arithmetic Logical Unit
APE	Associative Processing Element
VDD	Positive voltage supply (+5V)
VSS	Ground (0V)
VHDL	VHSIC Hardware Description Langauge

List of figures

List of tables

1. Theoretical background

This thesis deals with the design and hardware implementation of a highly parallel associative computer for image processing and computer vision basing on an Associative Real Time Vision Machine (ARTVM) that was introduced and developed by Ruhman and Akerib [1]. There are three basic problem areas that must be integrated to produce an effective machine design: computer vision, computer architecture and engineering. The computer vision area is necessary to determine the desired facilities of a machine. The computer architecture area determines what machine structure should be selected and what basic techniques should be implemented to meet the requirements set by computer vision area. The engineering problem deals with limitations of current fabrication technology and determines what can actually be built within these limitations. The present thesis is focused on computer architecture and engineering areas. The practical goals are the design of a computer microarchitecture, logic design and simulation, circuit design and simulation, CAM array design, and at last re-evaluation and re-design of the ARTVM architecture and its software model. This results in VLSI implementation of the associative processor chip (designated as ASP100), that serves as a basic functional block for the ARTVM.

1.1 Content addressable memory and associative processing

The Content Addressable Memory (CAM), also called an Associative Memory is defined as a store whose registers are not identified by their name or position but by their content [2]. Three basic operations are defined for a CAM array: *compare*, *write* and *read*. A CAM is accessed by broadcasting a data value to all memory cells. This operation activates logic in each of the memory cells, that compares the data stored in the cell to the pattern being broadcast. If the values match, then the cell is selected. In specific CAM that the ASP100 associative processor is based on, there is a mechanism that makes it possible to read and write contents of the selected cell. Compare, write and read operations can be performed either on whole data stored in a cell or on a subset of they as well. There are two registers usually associated with CAM. The COMPARAND register stores the search value. The MASK register stores the

mask pattern, that is broadcast along with the search value such that only not masked bits (fields) are activated in every CAM cell.

There are some additional mechanisms also provided by CAM. A Count Tag mechanism allows counting the number of CAM cells selected by compare operation. It is usually either very slow or, if fast, it requires expensive hardware support. The Select First mechanism chooses the 'first' of the selected CAM cells. This is essential for reading the CAM contents if more than one cell was selected by compare. The RSP mechanism indicates whether any or none of the cells was selected. In the ASP100, these mechanisms are implemented as one-dimensional logic arrays.

Two main primitives of an associative processor based on CAM are COMPARE and WRITE. Since this combination implements relation '*if* condition *then* action', all logical and arithmetic functions can be performed. All operations of an associative processor are both word- and bit-parallel, that makes possible parallel execution of many image processing and computer vision algorithms and thus provide the desired performance.

1.2 SIMD parallel architectures for image processing

The quantity of data in an image is too great and the processing required is too complex for a serial machine, in a reasonable amount of time. Image processing and computer vision applications need parallel processing in some form. Using well-known Flynn's taxonomy [3], a vision machine should be of SIMD, MISD or MIMD class. The MISD one can be eliminated because usually image processing application can not be effectively mapped onto such a structure. The question is whether SIMD or MIMD is better. Theoretical ideal solution is some combination of these structures and it is beyond. However, it is known [2] that SIMD vision machines do provide the greatest speed for low level image processing tasks. More specifically, it should be an architecture in which there are a large number of processing elements (PE), each of which operates on a single pixel or a group of pixels. The PEs in such machine are arranged in some kind of network, that allows fast communication between neighbors. The networks used are as follows:

- Two way (linear connection between neighboring PEs arranged in a linear array).

- Four way (PEs are arranged in a square grid; every PE connects to its north, west, south and east neighbor). This type is also called mesh network.
- Eight way (PEs are arranged in a square grid; every PE connects to all its nearest neighbors).

The main difficulty in implementation of such an architecture is VLSI technology limitations. The most important ones are chip size, I/O rate, number of chips per board, number of boards, power dissipation of a single chip and the whole system and operation frequency. Even today, using high density and low power silicon technologies, there is still impossible to integrate such SIMD machine on a single chip, for processing images of reasonable size and rate. There are some machines that come close enough to providing the desired combination of features and even have been actually built. Architectures 1 to 4 are examples of earlier massively parallel SIMD image processors, while 5 to 8 are examples of the latest designs in this area.

The first of these machines is CLIP-4. This is the fourth version of CLIP machine series constructed in 1973-1975 at University College, UK [2]. The CLIP-4 is a 96 by 96 array of bit-serial PEs, each with 32 bits of memory, connected by an eight way mesh. The destination of the CLIP-4 was processing of binary images. The only feedback mechanism provided by CLIP-4 was fast response count circuit. One of the interesting features of this machine is the separation of image processing and image input/output (I/O) that was implemented by a set of shift registers.

The second design is Massively Parallel Processor (MPP), built for NASA by Goodyear Aerospace Group [4]. This machine is one of most advanced earlier parallel image processors. It contains 16,384 PEs arranged in a 128 by 128 four way connected mesh. Each MPP PE is a bit serial processor with 1024 bits of memory. The only feedback mechanism in this machine is some/none responder. There is no response counter circuit. The limitation of this machine is relatively small array size, so images are processed region by region.

The third machine is Content Addressable Array Parallel Processor (CAAPP) designed by Weems in 1984 [2]. Most of the goals towards design of a real time associative SIMD image processor defined by Weems are still valid now. These design goals are:

1. Host driven architecture
2. Image I/O in a video frame time
3. 512 by 512 array of PEs (PE per pixel)

4. Square grid communications
5. Linear array communications
6. bits memory per PE
7. Bit serial PE
8. PE includes ALU performing logic and arithmetic operations
9. Instantaneous Some/None report back
10. Fast response count and Select First operation

According to the design goal set, the CAAPP consists of 512 by 512 array of bit-serial PEs connected by four way square mesh. The CAAPP machine was divided into 64 boards, 64 chips per board, 64 PEs (configured as 8 by 8 mesh) per chip. The interchip communication is implemented by bit serial shift network. Thus all PEs in a chip can be configured as two way linear array, such that only two bounding PEs have an external bit-serial connection. The estimated performance of this machine is in range 1 to 10 GOPS.

The fourth machine is LUCAS designed by Fernstrom, Kruzela and Swensson in 1986 [5]. This LUCAS is a linear array of 128 PEs. Every PE contains 4K CAM, ALU and Select First circuit. Every PE of the LUCAS can be directly connected to seven different PEs. One interesting feature of this machine is a built-in perfect shuffle-exchange network, dedicated FFT calculation. The LUCAS can perform matrix multiplication of 128 by 128 matrix (using 8 bit fixed point multiplication) within 0.5 sec.

The fifth machine is an Associative String Processor (ASP) designed by Aspex company [20,21]. This machine is organized as linear array of ASP modules connected by a communication network. Every ASP module contains string of ASP PEs connected by an internal communication network and data interface unit that contains a vector data buffer and data exchanger. Every PE consists of 70 bit CAM, one bit ALU and control logic. The ASP performance is 10 - 100 GOPS, while performance/cost ratio is about 10 MOPS per $1,000.

The sixth design is an associative real time vision machine (ARTVM) designed by Ruhmam, Scherson and Akerib in 1985-1993. The ARTVM is a full associative processor. It contains 256K lines CAM, fast count circuit, select first circuit and RSP (some/none) circuit, linear shift network for communication between PEs (a PE is associated with each CAM line) and

image I/O buffer (dedicated for load/dump images simultaneously with processing).

The seventh architecture called CAPRA was designed by Grosspietch and Reetz in 1992-1993. The main feature of this machine is three level memory organization. Every PE is associated with RAM line, CAM line, and content addressable processor/register (CAPR), that contains 4 bit fully parallel ALU. Some interesting feature of CAPRA is that a photo sensor, an A/D converter and the processor are integrated on the same piece of silicon.

The eighth architecture, the GLiTCH Associative Processor was designed by Storer and Duller in 1987-1993. The machine is partitioned into a number of dies, each of them containing 64 PEs and image I/O buffer. Every PE contains 64 bit CAM and one-bit ALU. The data communication network is implemented as a linear shift register, one bit of which is associated with each PE. There is a some/none response circuit. A 64 PE test chip designed using 2μ m CMOS combined 90K gates. Power dissipation was 1W and operation frequency was 20MHz.

The common feature of these machines (both those that have been actually built and those only evaluated) is that none has been dedicated for general use, like personal computers. The common limitations of a massively parallel SIMD vision machine are:

1. Technology restriction

Silicon technology of recent years did not provide the desired level of integration thus there was not possibility to allocate a reasonable number of PEs on a single chip. Addition problems caused by partitioning of the system are necessity to simplify a communication network due to chip pad limitation, decreasing of operation frequency, complex control and high synchronization requirements.

2. Physical size

In order to process an image of reasonable size (256 by 256 or 512 by 512) in real time mode, number of PE in an associative machine should be 128K - 256K. Number of PEs per chip can be between 1K to 4K using current

silicon technology. Number of chips per board can be among 8 to 32. It means the number of boards in a system is 4 to 64. An example of system integration is CAAPP that was divided into 64 boards, 64 dies per board.

3. Power dissipation

A massively parallel SIMD machine achieves its best performance while all image data are processed in word-parallel manner. In some type of machines processing is restricted to bit-serial. Nevertheless, associative processing could be word- and bit-parallel. It means that activity ratio of an associative machine on-chip hardware is much higher than that of a regular processor. Thus, power consumption of a SIMD machine chip is quite high. The second factor is a high number of such chips in the system. Result power consumption could be among hundred to 1K watts.

4. Cost

The cost of a massively parallel SIMD vision machine is highly affected by all technical limitations. Thus, high power consumption and number of chip pins define the package type, while package cost is a great part of a chip cost. Chip size and number of I/O pads define the board size affecting the overall cost of system. Special software, control circuits, some peripheral and interface mechanism are components of high systems cost.

The follows design issues were selected for comparative analysis of the SIMD vision architectures:

1. Performance

The main purpose of all architectures described above is meeting real time image processing and computer vision requirements. This purpose is achieved by implementation of massively parallel computation, on all image data simultaneously. There are some machines employing *PE per pixel* strategy, while the others utilize a single PE for processing of multiple pixels. Obviously, the first group provides better performance due to higher parallelism level. From the other hand, there are three different groups of architectures, employing various types of processing element. First group is based on associative memory only, without a dedicated hardware for data

processing. The second one utilizes one-bit arithmetic/logic unit serving for data processing in every PE. The third group does not conceptually differ from the second one, employing a bit parallel ALU. Actually, most SIMD vision architectures belong to group 1 and 2 or 1 and 3, using both associative memory and dedicated ALU in every PE. The best performance is obviously achieved by the third group architectures, while the full associative machine is probably the 'slowest' one. This result stays true while considering regular arithmetic/logical operations. Nevertheless, the associative machine is inherently both word-parallel and bit-parallel processor, therefore it can overcome other architectures while performing bit-parallel algorithms (like mathematical function calculation via look-up tables).

2. Integration

This point is very important due to enormous physical size of a SIMD vision machine. Unfortunately, most advanced architectures have most irregular structure, therefore integration level of such machines is very small. Moreover, memory is much denser than random logic, therefore full memory based architecture has inherently higher integration level (even using a special CAM cell) rather than ALU-based one. There is no explicit information about integration level of different SIMD machine design, therefore the complete quantitative analysis is not possible. Nevertheless, since integration level actually defines the cost of the system, some comparisons can be made basing on the cost information. Thus, the most advanced Associative String Processor cost is estimated at ≈$1,000,000 ,while the ASP100 based ARTVM one is estimated at ≈100,000 (for prototype, icluding NRE).

3. Power consumption

As mentioned above, all SIMD vision machines have very high power consumption due to both high parallelism and high activity factor. Obviously, power consumption due parallelism is essential for every SIMD architecture. On the other hand, power consumption of a single PE differs for various architectures. Generally, it is proportional to the performance. Thus, the bit-parallel ALU PE has highest consumption (including dissipation in memory array during load/store operations and in random logic during calculations), while the associative PE probably has the lowest power consumption.

1.3 Goals towards design of an associative real time vision machine

Given all constraints and limitations claimed in the previous section, the designed goals should be selected very carefully. The main purpose of design goal selection is definition a strategy that allows both achieving the desired features and keeping resulting design within reason taking into account the current technology state. Some goals stay the same as were defined by Weems, while others modify to meet technology advancing and destination variation.

The first goal is that the machine be driven by a host processor. The preferred host could be a personal computer (PC), such that the vision machine can probably be integrated within extended PC system. The host would communicate with a dedicated controller that will direct the vision machine operations.

The purpose of the vision machine is processing of an image or sequence of images in real time. This requires image I/O at video frame rate (1/30 sec) at least. In most massively parallel vision machine architectures there is a dedicated mass storage device (shift register block), that implements an image I/O buffer and operates in pipeline with processing an image. Thus, the second design goal is video rate image I/O.

The third goal is an efficient communication between PEs. The most useful communication networks are, as mentioned above, linear two way network, four and eight way mesh and 'one to some' network. Some machines implement special purpose networks like shuffle-exchange one. There is a clear trade-off between throughput of such a network to its implementation complexity, like internal chip routing and board routing. Thus the eight way network is probably most attractive from throughput point of view. However, until the machine can be integrated within a single chip, the number of chip external connections is unreasonably large. From the other hand, two way network is most simple for implementation but its throughput would not satisfy some image processing tasks. One possible solution is separation of inter-chip and intra-chip communication and using two different communication rates. However, this does not solve communication problem for low level vision algorithms, in which the same processing is carried out on all image date at the same rate.

According to this, the goal is to provide as high throughput as possible under limitation of chip pad number.

The fourth design goal is one PE per each pixel. The *PE per pixel* concept was first used in CAAPP. It became to a basic design philosophy since *PE per pixel style* of programming is conceptually much simpler than that of *group of pixels per PE*. In addition, this approach does simplify a microarchitecture of a chip and shorten design time.

The fifth goal is a full associative processing. Most SIMD vision machine PEs have some arithmetic/logical unit. Usually, it serves as accelerator, or 'co-processor' for arithmetic and logic operations. The consideration to avoid this mechanism is that such machine already has extremely high performance due to highest level of parallelism, and it is rather important to reduce the size and power consumption than to increase the performance. The practical reason for this is a fact that a memory (even associative one) is much dense and consumes less power during operations than random logic.

The sixth goal is an optimal trade-off between bit-serial and bit-parallel processing. This approach allows using bit parallel processing whenever it is possible. Most associative arithmetic and logical operations are carried out in bit-serial manner. Therefore, activity of the CAM array is quite low. Nevertheless, there is a great challenge to develop new algorithms or to adapt the known ones to let them to use the CAM in bit-parallel manner. An example to such 'associative' approach is mathematics function calculation. In an ordinary processor, such function is calculated by a sequence of arithmetic operations, using series. In associative processor, a function of n-bit argument can be calculated in 2^n cycles using 'look-up table' approach, with no arithmetic calculation.

The seventh goal is an appropriate selection of the CAM width (wordlength). According to recent researches, 32 bit width is sufficient for most black-white image processing applications. The goal of the current design is ability to process color images at first, and ability to implement some high level computer vision algorithms at second. This requires 48 - 64 bits per CAM line.

The eightieth design goal is integration of image I/O buffer and processor itself into a common CAM array. This approach is based on ability of CAM to

operate as two-dimensional memory, that allows fast exchange between buffer and processing array. This leads to an addition goal, that is building buffer with a variable width. This allows very high utilization of the CAM storage. For example, if the machine processes the color images, then buffer width is maximal (24). Otherwise, buffer width is 8, and rest of the array space is used for processing.

The ninth goal is to implement all peripheral mechanisms associated with a CAM, namely fast response count circuit, select first circuit and RSP (some/none) circuit. Introduction of these circuits allows fast implementation of classic associative techniques, like searching for patterns, sorting, look-up tables, histogram calculation etc..

The tenth goal is arranging the machine architecture and design in most suitable for VLSI design manner. This means including considerations like instruction set, chip physical size, power consumption, layout and floorplan, number of pads and so on into all stages of top to bottom design.

2. The ASP100 - associative computer chip for image processing and computer vision.

The ASP100 is an associative processing chip. It is intended to serve as part of the Associative Real Time Vision machine, most likely in an array of multiple ASP100 chips, and in conjunction of a dedicated controller chip.

The ASP100 consists of an associative memory array of 1K72× bits, peripheral circuitry, image FIFO I/O buffer, and control logic. The ASP100 runs under a special software dedicated for efficient implementation of arithmetic-logical operations and data transmission using associative techniques. The top-level architecture of the ASP100 is described in section 2.1. Software model and instruction set are discussed in section 2.2. Detail description of the ASP100 microarchitecture and building blocks is given in section 2.3. Some important design limitations and restriction are discussed in section 2.4. Section 2.5 describes design of the CAM array, including design goals and considerations, basic cell classification and implementation, design of response count, select first and RSP circuits. At last, section 2.6 describes the ASP100 multilevel simulation.

2.1 Top-level architecture

The top-level view of the ASP100 is shown in Figure 2.1. ARRAY is the main CAM processing array. FIFO is the image I/O buffer, that serves for image load/dump. The ARRAY and the FIFO are integrated into a common CAM MATRIX. SIDE is the side array, consisting of the TAG register, the tag logic, the response count circuit (tag count), the select first circuit, the RSP (some/none response) circuit, the line drivers and sense amplifiers, and the shift network. TOP consists of the mask and comparand registers, control circuits and column drivers. BOTTOM contains the output register and sense amplifiers. The following Table 2.1 contains the partial pin list of the ASP100 chip.

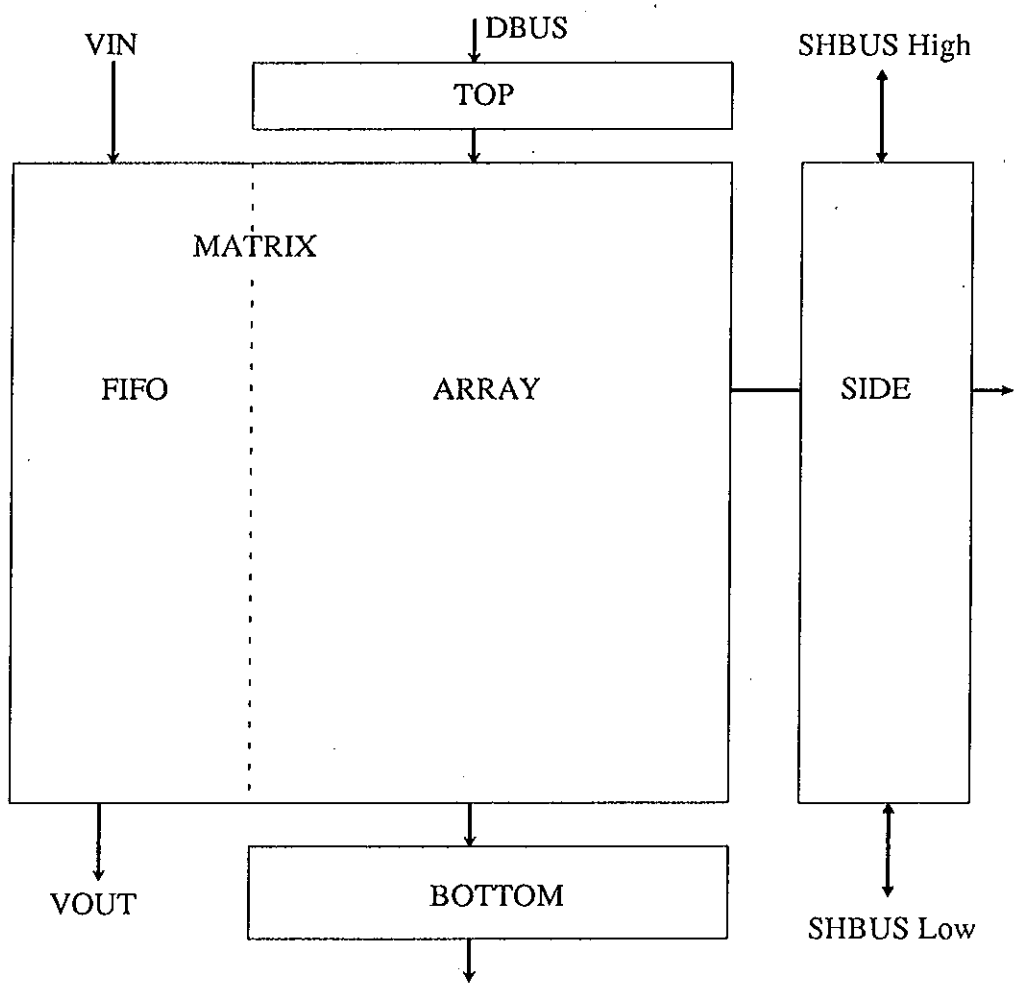


Figure 2.1: The ASP100 top level view

ציור 2.1 רמת תכנון עליונה של א'ס'פ'100

Table 2.1 Pinout of the ASP100 chip

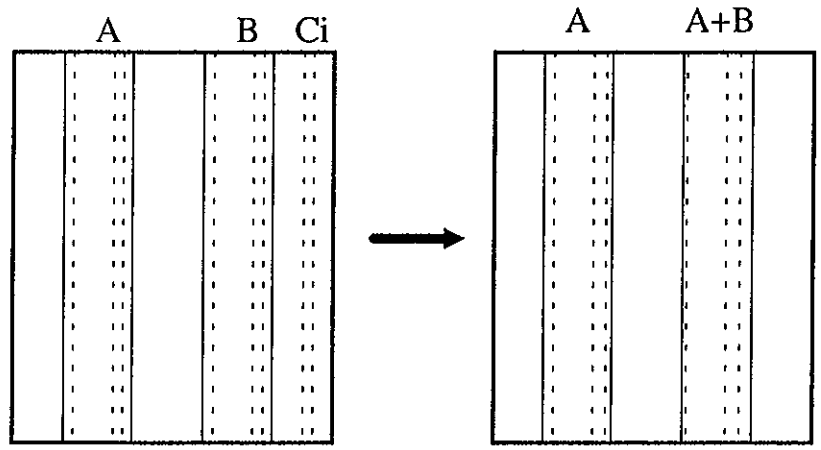
Pin Name	Description	I/O	# pins
DBUS	Data bus, serves CAM read out and Data in	I/O	32
VIN	Video In Bus	I	24
VOUT	Video Out Bus	O	24
SHBUS	Shift network In/Out	I/O	32
CTAG	Count Tag Serial Output	O	1
RSP	RSP (response exists)	O	1
CLK	First Clock (25 MHz)	I	1
DCLK	Delayed Clock (25 MHz)	I	1
FFUL	FIFO Full (Image I/O Completed for the chip)	O	1
FIRSTIN	Initialize select first circuit	I	1
LIN	Shift Up Enable	I	1
HIN	Shift Down Enable	I	1
WE	Write Enable	I	1
SETAG	Set Tag	I	1
FIRCNTEN	Connects tag to select first and response count circuits	I	1
FENB	Enable FIFO I/O	I	1

2.1.1 Processing ability

The processing core of the ASP100 associative computer chip is a classical bit- and word-parallel CAM array. The main associative primitives of the ASP100 are compare and write operations. During write operation, the data stored in the comparand register is written into selected fields. During compare operation, the data stored in every CAM line are compared with a pattern being broadcast along by the comparand register. If the patterns match, an appropriate bit of the tag register is set high. In both write and compare operations, the active fields

are selected both by a value stored in mask register and a value stored in tag register. The selection based on tag register creates mechanism of conditional write depending on results of compare, that actually allows implementation of an arbitrary logical/arithmetic operation [6]. Obviously, this operation is executed in word- and bit-parallel manner for a great amount of data. Computational power of the ASP100 can be shown using example of vector addition operation, depicted in Figure 2.2. This operation is executed in word-parallel and bit-serial manner. Before execution, the vector operands are located in two CAM fields (A and B). The result sum is written into field B (destructive addition). A carry vector generated during operation is stored in an additional CAM column. In every operation step, only three bit slices (of A and B respectively, and current carry slice) are activated. Then, contents of selected columns are compared with sequence of input combinations (according to Table in Figure 2.2), and sum-carry pattern is written to result field in matching lines. An advantage of destructive addition is that input combination that does not change value stored in result field (see Table in Figure 2.2) does not have to be checked. This can provide significant reducing of computation time, but order of comparisons becomes important. One bit binary addition involves four compare-write cycles [11]. Both compare and write ASP100 instructions occupy a single clock cycle then execution time of n-bit addition is $8n$. This means the ASP100 can add up 1K 8-bit operands within 64 clock cycles.

There are additional computational facilities based on peripheral associative circuits. Thus, the response count mechanism is very useful for mass addition-accumulation operations, that run over all data stored in CAM. Examples of such operations are histogram calculation, since the result value equals to number of logic '1's stored in the tag register, and mean square error calculation, since the result is an accumulated sum of values allocated in a certain CAM field. The RSP some/none mechanism is useful for control of iterative computational processes, in which a break condition occurs if RSP returns zero. The select first mechanism provides a serial access to selected CAM lines if there is necessity to read them.



CAM array during addition CAM array after addition

A	C	B	C	A+B	order
0	0	0	0	0	-
0	0	1	0	1	-
0	1	0	0	1	1
0	1	1	1	0	2
1	0	0	0	1	4
1	0	1	1	0	3
1	1	0	1	0	-
1	1	1	1	1	-

Figure 2.2: Associative addition

ציור 2.2 חיבור אסוציאטיבי

2.1.2 Data communication

Data communication problems in massively parallel SIMD machines were considered and discussed in the previous sections. According to ASP100 design goals, the data communication network has to provide a maximal flexibility and speed data transmission under constraint of chip pin number and internal/external routing complexity. Obviously, in actual chip design this constraint is a fuzzy one. The number of pins provided by available packages (supplied by ASIC fabs) reaches 400-500. For reasonable amount of PEs per chip (1-2K), even full four way mesh (requiring about 200 external connections) becomes feasible. Nevertheless, the internal/external routing factor (that is usually defined as ratio between area occupied by wires to area occupied by devices) is too high for such kind of communication network.

Unfortunately, we found no quantitative criterion, that can aid in configuring an optimal network a priori. Thus, the design of communication network was based on practical reason and trial and error. In the ASP100 communication network (designated as shift network) every PE (actually, a bit of the SIDE array associated with the CAM line) is connected to its south and north nearest neighbors and to its south and north neighbors located by b positions apart. Formally, every j-th PE ($j \in 1,..,N$, where N is the SIDE array length) is connected to $(j-1)$-th, $(j+1)$-th, $(j-b)$-th and $(j+b)$-th PEs (overflows are sent off chip). Image is normally loaded into the ASP100 by raster scan. Therefore the line neighboring pixels are physically allocated in neighboring PEs. Column neighboring pixels are spaced by N positions. Thus the ASP100 shift network can transmit data in line direction within a single cycle, while transmission in column direction takes (N/b) cycles. If $N=b$, the ASP100 shift network becomes a regular four way mesh. Design experience shows that optimal values of b (under constraints we defined earlier) are 16 or 32.

2.2 Software model

Software model of a real time associative vision machine contains full set of operations defined for classical CAM. The basic CAM primitives (group 4 of Instruction Set, see Table 2.2) support arithmetic and logical operations. An additionally defined *read* instruction (group 5 of Instruction Set) is dedicated to support a random access to the CAM. *Count Tag* and *First Select* instructions (group 5 of Instruction Set) activate fast response count and select first circuit. There is no instruction activating the RSP circuit, which is instantaneous. *Shift Tag* instructions (group 3 of Instruction Set) activate the shift network supporting data transmission. *Set* and *Let* instructions (group 1 of Instruction Set) serve for load of search and mask values to comparand and mask registers respectively. All instructions of this group can affect only one sector (see Section 2.3.1. for definition of sector). Exclusive instructions operate on specified sector clearing two others. These instructions were defined to guarantee reducing number of active CAM columns decreasing peak power consumption of the chip. *SetTag* and *ResetTag* instructions (group 2 of Instruction Set) set TAG register high or low respectively. *Configure FIFO* instruction (group 5 of Instruction Set) defines the I/O buffer area disconnecting it from the processing array. All ASP100 instructions except instructions of group 5 have a single cycle execution time. In the ASP100 Instruction Set there are three types of instructions:

1. Instruction without operand
2. One operand instruction (operand is sector number)
3. Two operand instruction (first operand is sector number, second is 24-bit data word)

The ASP100 instruction format is shown in Figure 2.3. It contains generally one bit for NOP, seven OpCode bits and 24-bit for second operand. The first operand if needed is included in OpCode field. The actual ASP100 instruction set is shown in Table 2.2

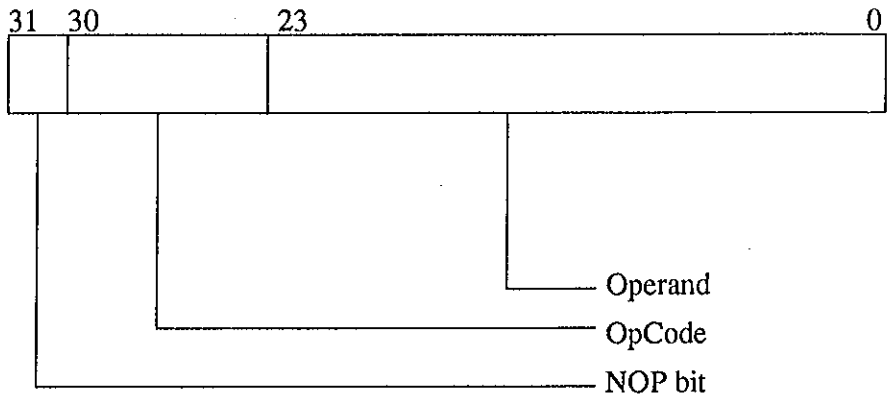


Figure 2.3: The ASP100 Instruction format

ציור 2.3 צורת פקודה של א'ס'פ'100

In the table, d(n) is a n-bit argument, n=24, and s(2) is a 2-bit sector number.

Table 2.2 The ASP100 instruction set

No.	Instruction	Format	Cycles
Group 1			
1	Load Mask	LM s(2),d(24)	1
2	Load Comparand	LC s(2),d(24)	1
3	Load Mask and Comparand	LMC s(2),d(24)	1
4	Load Mask, Clear Comparand	LMCC s(2),d(24)	1

5	Load Mask, Clear Comparand Both Exclusive	LMCCXX s(2),d(24)	1
6	Load Comparand, Set Mask	LCSM s(2),d(24)	1
7	Load Mask Exclusive	LMX s(2),d(24)	1
8	Load Comparand Exclusive	LCX s(2), d(24)	1
9	Load Mask, Set Comparand	LMSC s(2),d(24)	1
10	Set Mask Exclusive	SMX s(2)	1
11	Set Comparand Exclusive	SCX s(2)	1
Group 2			
12	Reset Tag	RESETAG	1
13	Set Tag	SETAG	1
Group 3			
14	Shift Up	SHUP	1
15	Shift Down	SHDN	1
16	Long Shift Up (16 places)	LGUP	1
17	Long Shift Down (16 places)	LGDN	1
Group 4			
18	Compare	COMPARE	1
19	Write	WRITE	1
Group 5			
20	Read	READ s(2)	3
21	Count Tag	COUNTAG	31
22	First Select	FIRSEL	23
23	Configure FIFO	CONFIFO d(2:0)	1
Group 6			
24	No Op	NOP	1

Current ASP100 instruction format allows parallel execution of instruction of different groups. According to this, instructions of either group 1 and 3 or groups 1,2 and 4 can be carried out in parallel. Examples of such 'horizontal' instructions can be {LMCXX (operand 1, operand 2); SETAG; WRITE} and {LMCXX (operand 1, operand 2); SETAG; COMPARE} fitting the ARTVM theoretical software model [11].

2.3 Microarchitecture and operation

2.3.1 ARRAY-FIFO (MATRIX) architecture

The MATRIX is a CAM array of 1024 by 72 associative processing elements, logically organized in three columns of 24 elements each, and physically split into three blocks of 342×72. Logic column of 1024 by 24 elements is designated as sector. Therefore, there are three sectors, 0 to 2. The ARRAY part of matrix operates as main processing array. Sectors 0 and 1 of the MATRIX belong to the ARRAY only. Sector 2 of the MATRIX is reconfigurable as follows:

1. All 24 bits serve as FIFO (total ARRAY width is 48).
2. 16 bits FIFO, 8 bits ARRAY (total ARRAY width is 56).
3. 8 bits FIFO, 16 bits ARRAY (total ARRAY width is 64).

The first configuration is used for processing of RGB images, the second one is used for stereo vision, and the third one used for black and white image processing. Integration of ARRAY and FIFO allows better utilization of the CAM array area and improves the chip layout.

The associative processing element (APE) is a CAM basic cell. It consists of three parts:

Storage device;
Write device;
Match device.

There are three vertical incident buses and four horizontal incident buses in the associative processing element:

Vertical:	Bit Line (BL)
	Inverse Bit Line (IL)
	Mask Line (MASK)
Horizontal:	Write Line (WL)
	Match Line (ML)
	VDD
	VSS (GND)

The Storage device consists of two cross coupled CMOS invertors. The Write device implements the logical AND of MASK and WL. The Write device purpose is supporting both vertical and horizontal masking. Therefore, only cells selected by both MASK and TAG registers are affected by write operation. The Match device implements a dynamic EXCLUSIVE OR (XOR). Its goal is implementation of COMPARE operation. The schematic circuit diagram of the associative processing element is shown in Figure 2.4. More detailed description on modes of operation, design goals and considerations of CAM basic cell design is given in Section 2.5. The MATRIX interconnection structure is depicted in Figure 2.5.

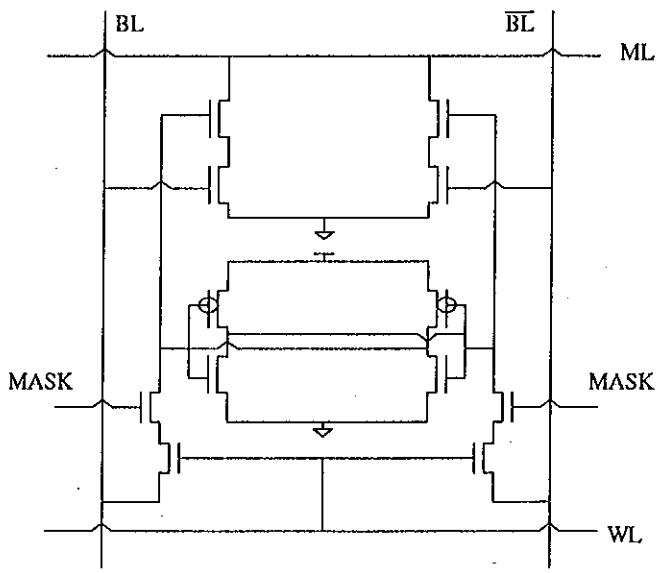


Figure 2.4: The associative processing element

ציור 2.4 יחידת עבוד אסוציאטיבית

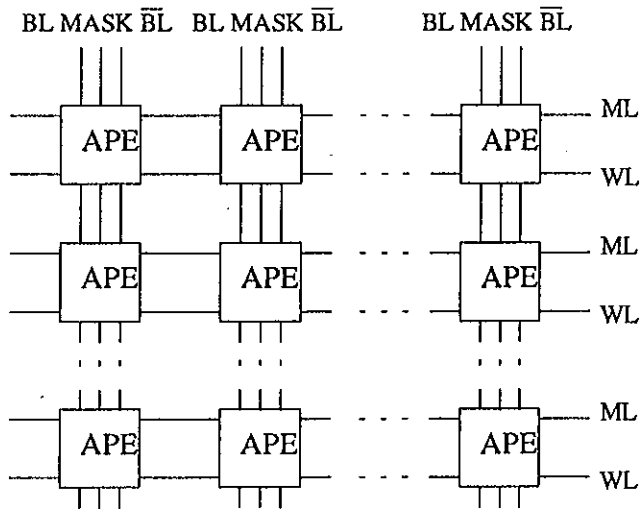


Figure 2.5: The MATRIX interconnection structure
ציור 2.5 מבנה חיבורים פנימים של MATRIX

The FIFO is designed to input and output image data in parallel with computations carried out on the ARRAY. It consists of a reconfigurable matrix of 1024×[24 or 16 or 8] APEs, three columns of 1024 Bi-directional Switches each, and Address Generator (Figure 2.6). The corresponding section of the Comparand register (in TOP) serves as the FIFO input register, and the corresponding section of the Output Register (in BOTTOM) serves as the FIFO Output Register. The FIFO Controller FC resides in TOP.

The Address Generator consists of a shift register and implements a sequential addressing mode. It selects the currently active FIFO word line during image I/O.

The FIFO has two modes of operation, IMAGE I/O Mode and IMAGE EXCHANGE Modes. The bi-directional Switches (one column of the three) disconnect the FIFO from the ARRAY in IMAGE I/O Mode (see below) and connects the FIFO to the ARRAY in IMAGE EXCHANGE Mode, creating a combined array of APEs. The Input and Output Registers serve as buffer registers for the image I/O.

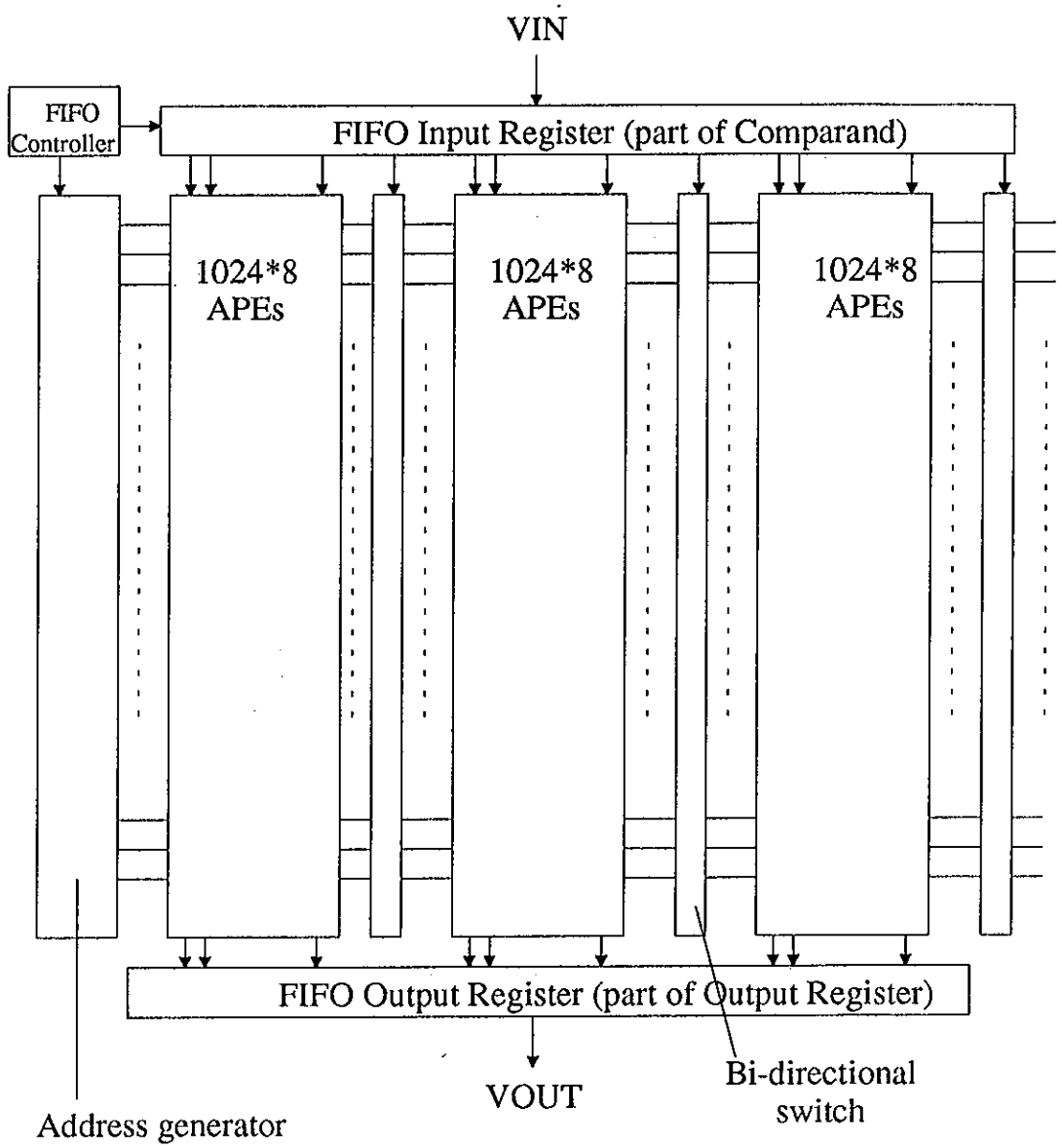


Figure 2.6: The FIFO

IMAGE I/O Mode

In IMAGE I/O Mode, a new image is read into the FIFO, while the processed (preceding) image is written out. The I/O activity is performed asynchronously of the computation in the rest of the chip.

The basic operation of IMAGE I/O mode is carried out as follows. The pixel at the VIN pins is entered into the FIFO Input Register (the FIFO section of the comparand register). The Address Generator enables exactly one word line.

The corresponding word is read into the FIFO Output Register (the FIFO section of the Output Register), and through it directly to the VOUT pins, in an operation similar to Read execution. Subsequently, the word in the FIFO Input Register is written into the same word, similar to a Write execution. The VOUT pins are 3-state. They are enabled and disabled internally as needed.

This sequence of operations is carried out in a loop 1024 times.

Multiple ASP100 chips can be chained together with an 'Enable FIFO'/FIFO full' (FENB/FFUL) chain, where the first ASP100 receives the FENB from an external controller, the FFUL of each ASP100 is connected directly to the FENB input of the next chip, and the last FFUL goes back to the controller.

IMAGE EXCHANGE Mode

In IMAGE EXCHANGE Mode, the image previously loaded into the FIFO is transferred into the ARRAY for subsequent processing, and the previously processed image from the ARRAY is transferred to the FIFO for subsequent output. These transfers are carried out via the TAG register of the SIDE block by a sequence of COMPARE and WRITE instructions, as follows:

IMAGE IN

A destination bit slice of the ARRAY is masked by MASK register and is then reset by a chain of {SETAG; LC (0); WRITE} instructions (which can all be executed in one cycle). A source bit slice of the FIFO is masked by the MASK register. The contents of the bit slice are passed to the TAG register as a result of the COMPARE operation. The destination bit slice is masked again and then the contents of the TAG register are passed to the destination bit slice by a {SCX; WRITE} operation. In summary, the following three cycles are employed:

```
FOR (all bit slices of FIFO)
        {
        LMCCXX (sector 0/1/2, destination ARRAY bit) ; SETAG; WRITE;
                /* reset destination bit slice in the proper array sector */
        LMC (sector 2, source FIFO bit); COMPARE;
```

```
        /* copy source slice (FIFO sector) to TAG */
    LMC (sector 0/1/2, destination ARRAY bit); WRITE;
        /* copy TAG to destination in the proper sector */
    }
```

IMAGE OUT

This operation is carried out in exactly the same way as IMAGE IN, except that a destination bit slice is allocated in the FIFO while a source bit slice is allocated in the ARRAY. Note that IMAGE EXCHANGE operation requires two different fields in the ARRAY (a first field for allocation of a new image, and a second one for temporary storage of the processed image). The two operations (IMAGE IN and IMAGE OUT) can be combined in one loop.

2.3.2 SIDE Architecture

The SIDE block contains the TAG register, the shift network (containing NEAR neighbor connections and FAR neighbor connections), the response counter, the select first circuit, the RSP circuit, and the required horizontal bus drivers and sense amplifiers.

The TAG register consists of a column of 1024 TAG_CELLs. The TAG_CELL is shown in Figure 2.7. The TAG register is implemented by D flip flop (FF) with an asynchronous set input and non-inverse output. The input is selected by means of an 8-input multiplexor, with the following inputs: FarNorth, NearNorth, FarSouth, NearSouth, MatchLine (via sense amp), TAG (feedback loop), GND (for tag reset), and select first circuit output.

The NEAR neighbor connections interconnect the TAG_CELLs in an up/down shift register manner to nearest neighbors. They are typically employed for neighborhood operations along an image line, since pixels are stored consecutively by lines. The FAR connections interconnect TAG_CELLs 16 apart. They are typically used for neighborhood operations between image lines. See Figure 2.8.

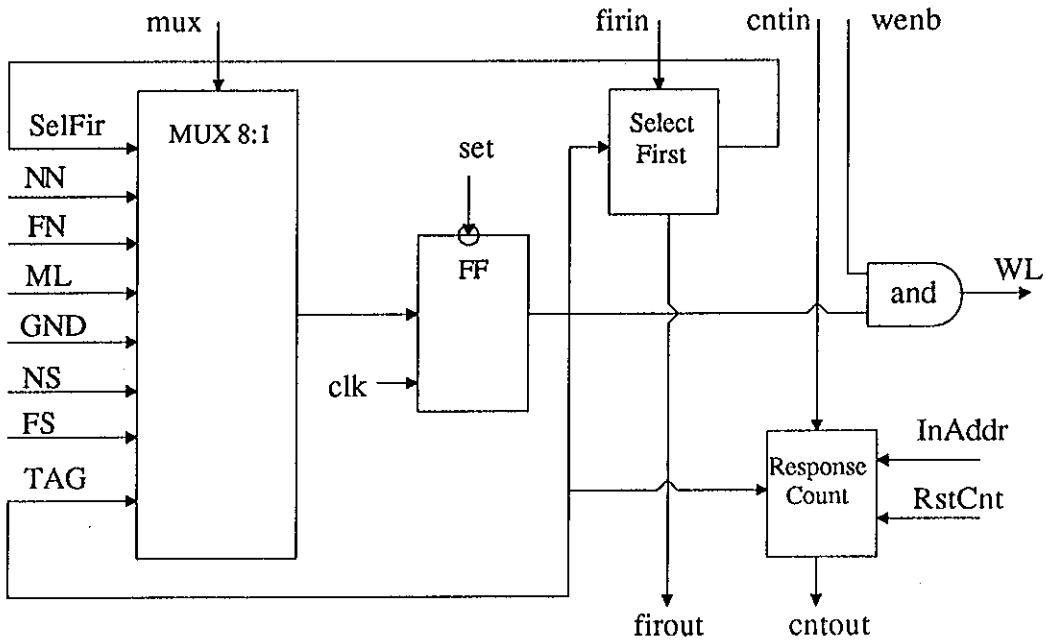


Figure 2.7: The TAG_CELL

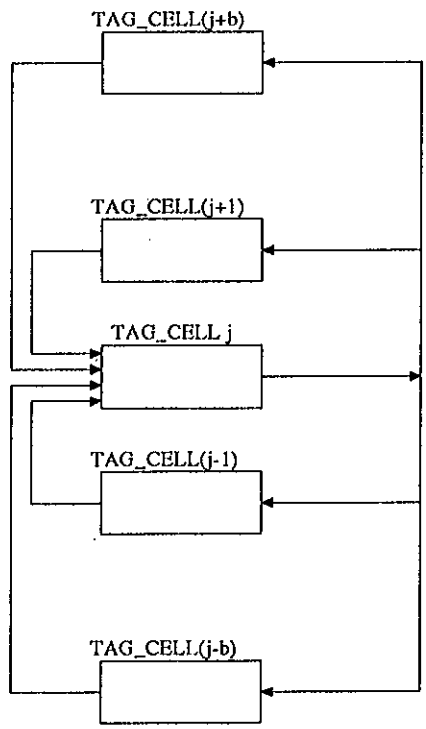


Figure 2.8: The Shift Network

ציור 2.8 רשת הזזה

In this design, b=16

The following instructions affect TAG: SETAG, SHUP, SHDN, LGUP, LGDN, COMPARE, FIRSEL.

The response count circuit counts the number of TAG_CELLs containing '1'. It consists of three iterative logic arrays of 1342× cells each. Every logic array consists of 18 pipeline stages, 19 cells each. See Figure 2.9. The side inputs of the counter are fed from the TAG register outputs. The count circuit operates in a bit-serial mode, starting with the least significant bits. The operation is executed within two phases: initial and iterative. Both phases are pipelined, starting with the topmost pipeline stage and propagating downwards. In the initial phase (it occupies a single clock cycle), the adder section inputs are connected trough the mux to outputs of TAG register (respin). In the iterative phase (it occupies $\lceil \log_2 19 \rceil = 5$ cycles) the adder section inputs are connected to the FFs outputs. In each cycle, the carry bits are retained in the FF for the next cycle, and the sum is propagated down to the next stage. The outputs of all three blocks are added by a summation stage, which generates the final result in a bit-serial manner. Response count circuit is activated by the COUNTAG instruction.

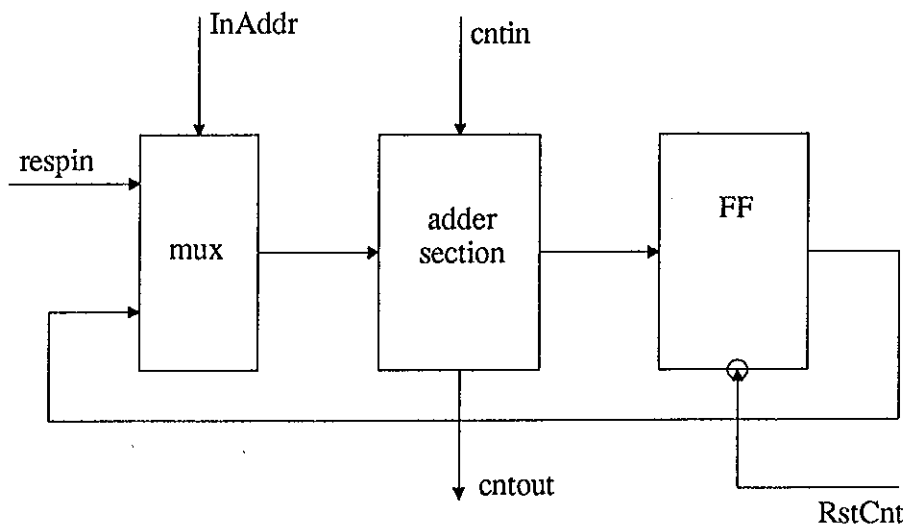


Figure 2.9 a: Response count circuit cell

ציור 2.9א' תא של מנגנון לספירת תגובות

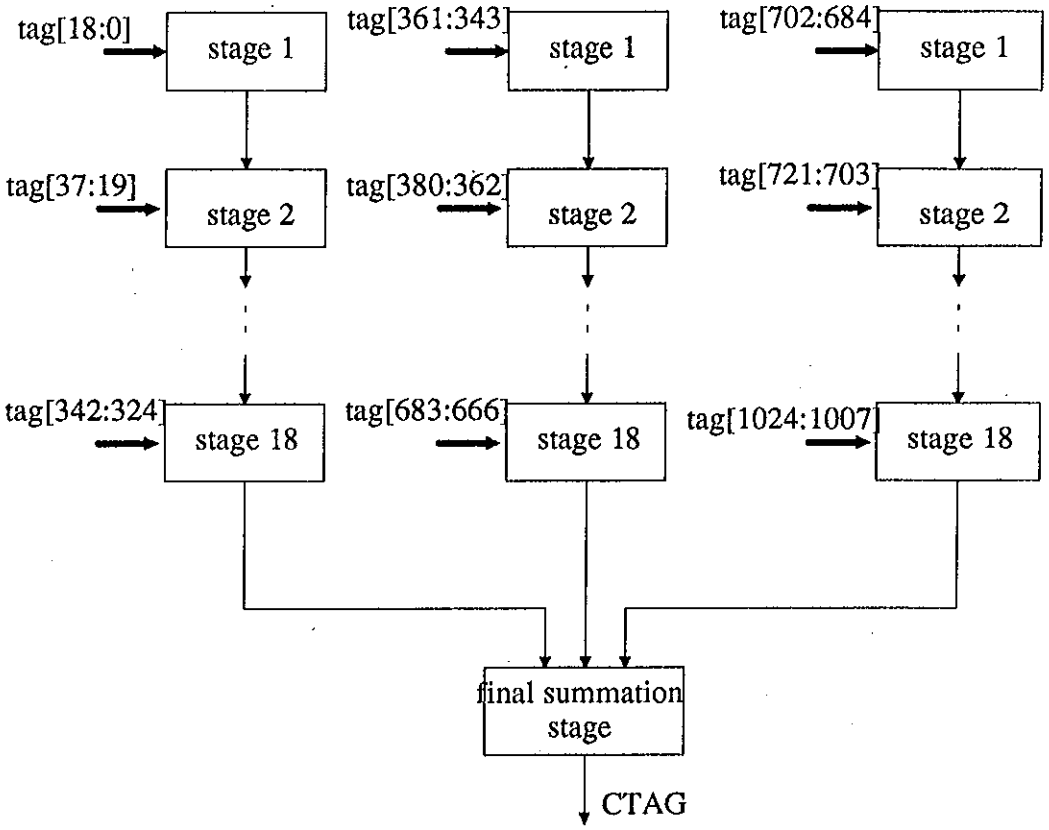


Figure 2.9 b: Response count circuit pipeline

ציור 2.9ב' צינור של מנגנון לספירת תגובות

The select first circuit finds the first TAG_CELL containing '1', and resets all other TAG_CELLs to '0'. It is activated by the FIRSEL instruction. The beginning of the chain is fed by external initializing signal ('first in'), and if it is LOW then all TAG_CELLs are reset to '0'. This is intended to chain the FIRST-SELECT operation over all ASP100 interconnected together, and the OR of the RSP outputs of the lower-numbered ASP100 should be input into 'first in'.

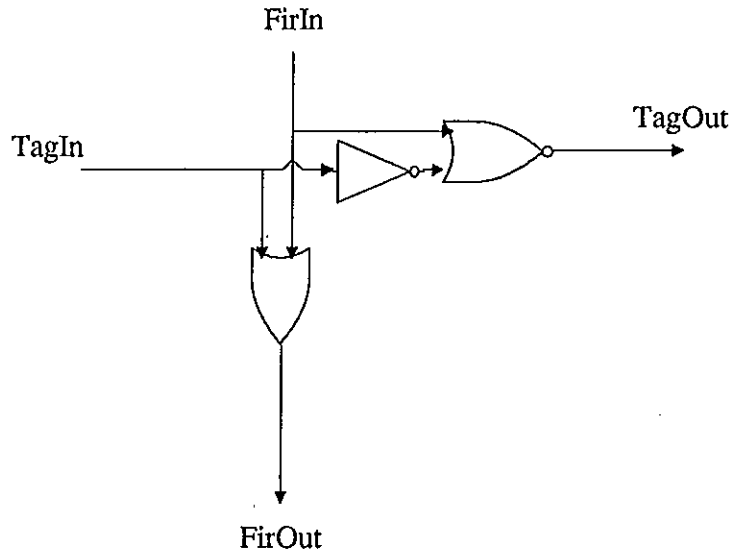


Figure 2.10: Select First Circuit

ציור 2.10 מנגנון בחירת הראשון

About 60% of the SIDE unit hardware belong both to select first and response count circuits. These circuits respond every change in TAG register but outputs of these circuits are read during COUNTAG and FIRSEL instructions only. This means power can be saved either if these circuits are disconnected from the power supply lines or their inputs are disconnected from TAG outputs whenever the circuits are not active. The second technique is much simple while saving about 95% of power consumption of the circuits (in VLSI CMOS process only 5% of power consumption is not due to charge/discharge of capacitance). The TAG outputs can be disconnected from the select first and response count circuits by pulling FIRCNTEN control input to '0'.

The some/none responder (RSP) circuit generates '1' on the RSP output pin after COMPARE instruction, if there is at least one '1' TAG value. This output is registered in order to keep the response until next compare occurs.

The sense amplifier of the Match line is shown in Figure 2.11. The sense amplifier is implemented by an unbalanced inverter with enable inputs. The enable inputs (Penb and Nenb) allow disconnecting the Match line sense amplifier from the power supply lines in order to save power whenever the sense amplifier is not in use. The Match line sense amplifier design is described in detail in Section 2.5

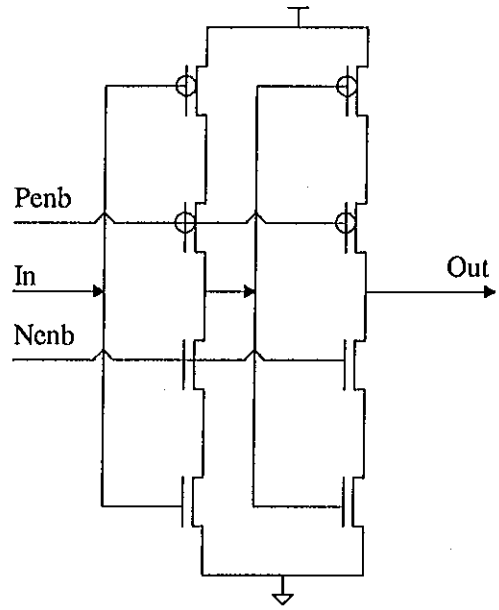


Figure 2.11: ML Sense Amplifier

ציור 2.11 מגבר הפרש של קו התאמה

2.3.3 TOP Architecture

The TOP block consists of the COMPARAND and MASK registers and their respective logic, and the vertical bus drivers. See Figure 2.12.

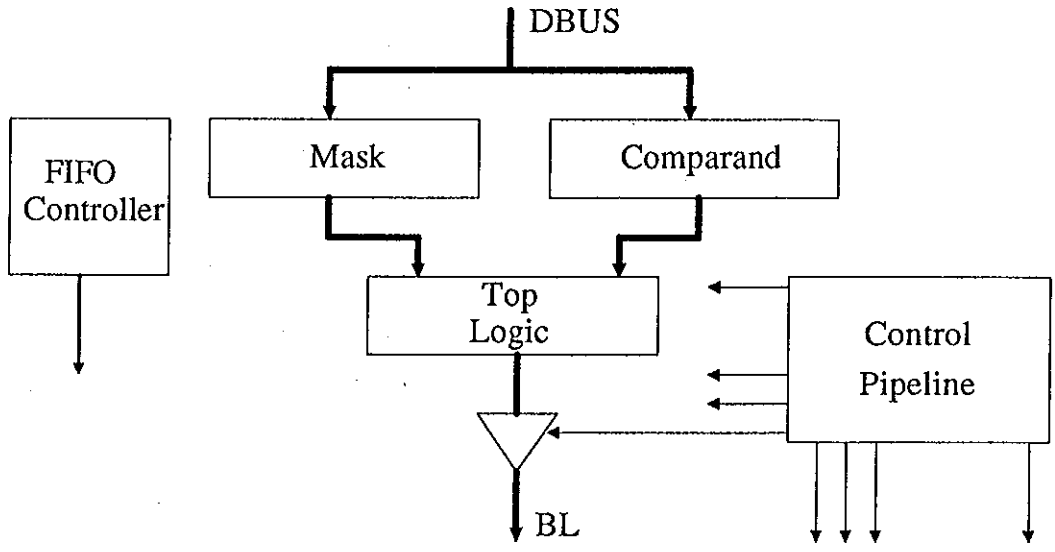


Figure 2.12: The TOP

The COMPARAND register contains the word that is compared against the ARRAY. It is 72 bits long, and is partitioned according to FIFO configuration. It is affected by the group 1 instructions. As mentioned above in Section 2.2, all these instructions affect only one of the three sectors at a time, according to the sector bits. The exclusive instructions also clear the comparand bits of the non-addressed sectors. The FIFO section of the COMPARAND operates differently, as described above in Section 2.3.1.

The MASK register masks (by '0') the bits of the COMPARAND which are to be ignored during comparison and write. The Bit Lines and Inverse Bit Lines of the masked bits are kept at '0' to conserve power. It is affected by group 1 instructions. Like in COMPARAND, the relevant instructions affect only one sector at a time, whereas exclusive ones also clear the mask bits of the non-addressed sectors. The FIFO section of the MASK operates differently, as described above in Section 2.3.1.

The 24 bit data input from DBUS (the operand) are pipelined by three stages, so as to synchronize the operand and the corresponding control signals.

2.3.4 BOTTOM Architecture

BOTTOM contains the Read sense amplifiers, the Output Register, the RSP Output Logic and Response Count Output logic. See Figure 2.13.

The Read sense amplifier is shown in Figure 2.14. Since the ARRAY is physically organized in three columns, the output of the three Read sense amplifiers must be resolved. A Sense Amplifier Resolve unit (see Figure 2.13) selects which column actually generated the output, as follows:

READ:	Select the column whose RSP is true.
FIFO OUT:	Select the column in which the address token is in.

The Output Register is 72 bits long. 8 or 16 or 24 bits serve the FIFO and are connected to the VOUT pins. On READ operation, one of the three sectors (according to the sector bits) is connected to 24 bits of SHBUS via a multiplexor. See Figure 2.14.

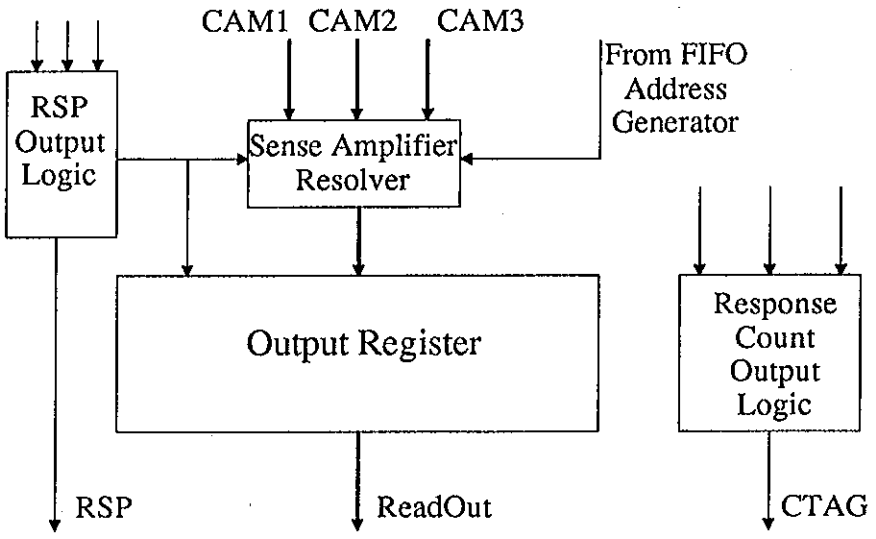


Figure 2.13: BOTTOM

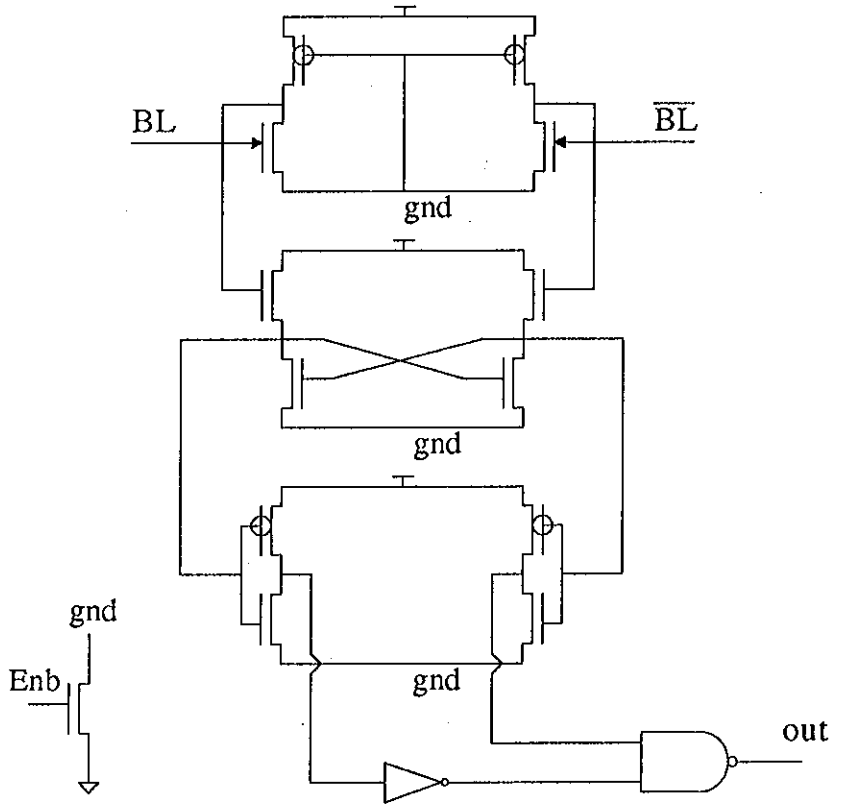


Figure 2.14: BL Sense Amplifier

ציור 2.14 מגבר הפרש של קוי נתונים

SHBUS multiplexor connects bits [15:0] (bit 0 is LSB) of the Output Register to SHBUS[15:0], and bits [23:16] of the Output Register to SHBUS[23:16].

The SHBUS I/O buffers control whether the SHBUS is connected as input or output, and are controlled by special control signal.

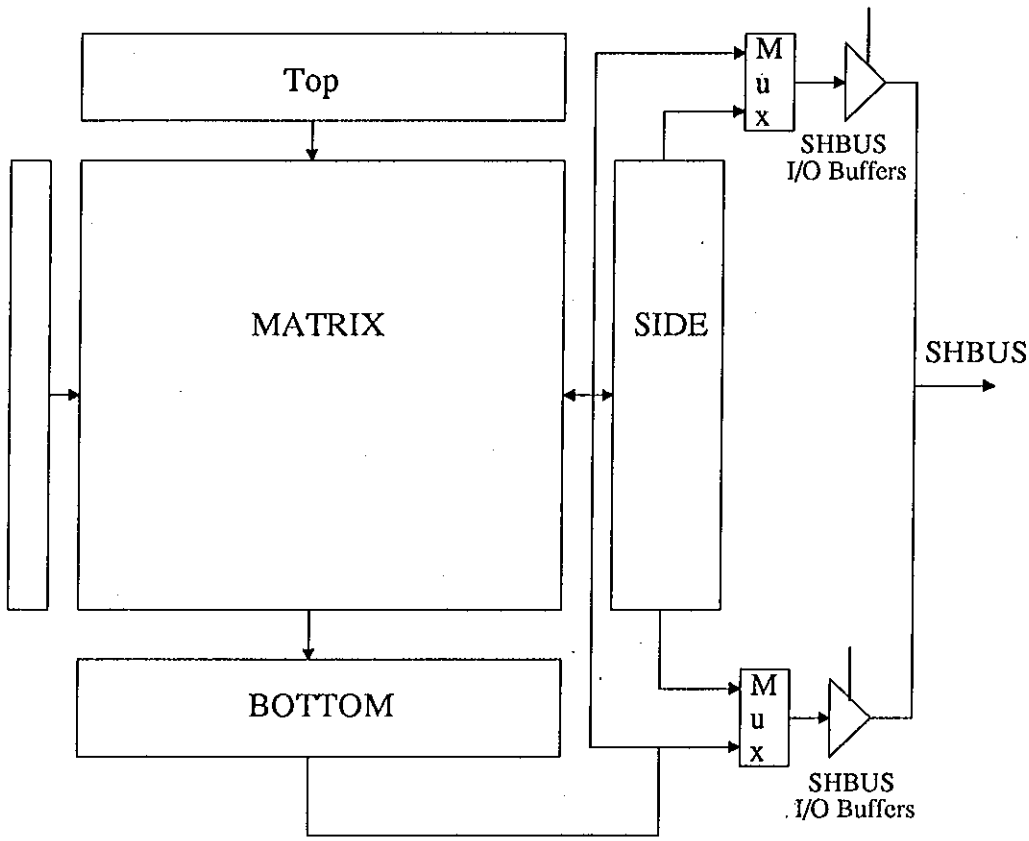


Figure 2.15: Read Output connection to SHBUS

ציור 2.15 חיבור יציאת קריאה ל SHBUS

2.3.5 ASP100 pipeline and microcontrol notes.

The ASP100 is controlled by means of an external microcoded state machine, and it receives the decoded control lines. The external microcontroller allows horizontal microprogramming with parallel activities.

The combined operation of APS100, its microcontroller, and the external controller is organized in a five-stage instruction pipeline, consisting of the following stages: Fetch, Decode, microFetch, Comparand, and Execute. In the Fetch stage, the instruction is fetched from the external program memory, and

is transferred over the system bus to the IR. In the Decode stage, the instruction (from the IR) is decoded by the microcontroller and stored in the μIR. In the μFetch stage, the control codes are transferred from the external μIR, through the input pads, into the internal μIR. In the Comparand stage, parts of the execution that affect the Comparand register are carried out, and the control codes move from the internal μIR to the internal μIR2. In the Execute stage, the execution in the ARRAY and other parts takes place. See Figure 2.16.

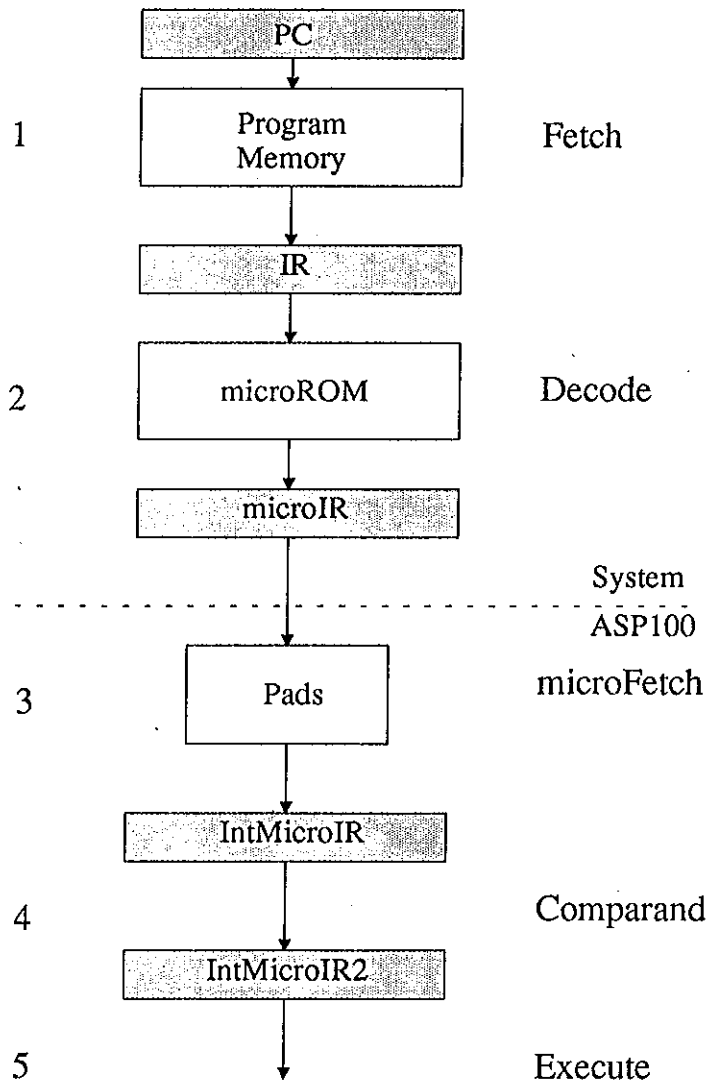


Figure 2.16: ASP100 Instruction Pipeline

ציור 2.16 צינור פקודות של הא'ס'פ'100

Branches are handled by the external controller, and are interpreted as NOP by the APS100 microcontroller. Similarly, non-ASP operations are treated as NOPs. Instructions which execution time is longer than one cycle (READ,

COUNTAG, and FIRSEL) are divided to sets of one cycle micro instructions by compiler. Therefore, all code words being fetched from the program memory have one cycle execution time.

2.3.6 System configuration

As mentioned before, the ASP100 is implemented to serve as part of the Associative Vision Computer. An arbitrary number of the ASP100 chips can be easily added to the system, according to the following interconnection algorithm:

1. DBUS of every ASP100 chip is connected to the system DBUS
2. VIN bus of every ASP100 chip is connected to the system VIN bus
3. VOUT bus of every ASP100 chip is connected to the system VOUT bus
4. Control signals of every ASP100 chip are connected to the system control bus
5. SHBUS[15:0] of the succeeding chip is connected to SHBUS[31:16] of the preceding chip
6. SHBUS[31:16] of the succeeding chip is connected to SHBUS[15:0] of the preceding chip
7. FENB input of the succeeding chip is connected to FFUL output of the preceding chip
8. FIRSTIN input of the succeeding chip is connected to RSP output of the preceding chip
9. CTAG output of every ASP100 chip is connected to system parallel summation unit
10. RSP output of every ASP100 chip is connected to OR circuit generating the system RSP signal

The symbolic interconnection scheme of the Associative Vision Computer is depicted in Figure 2.17.

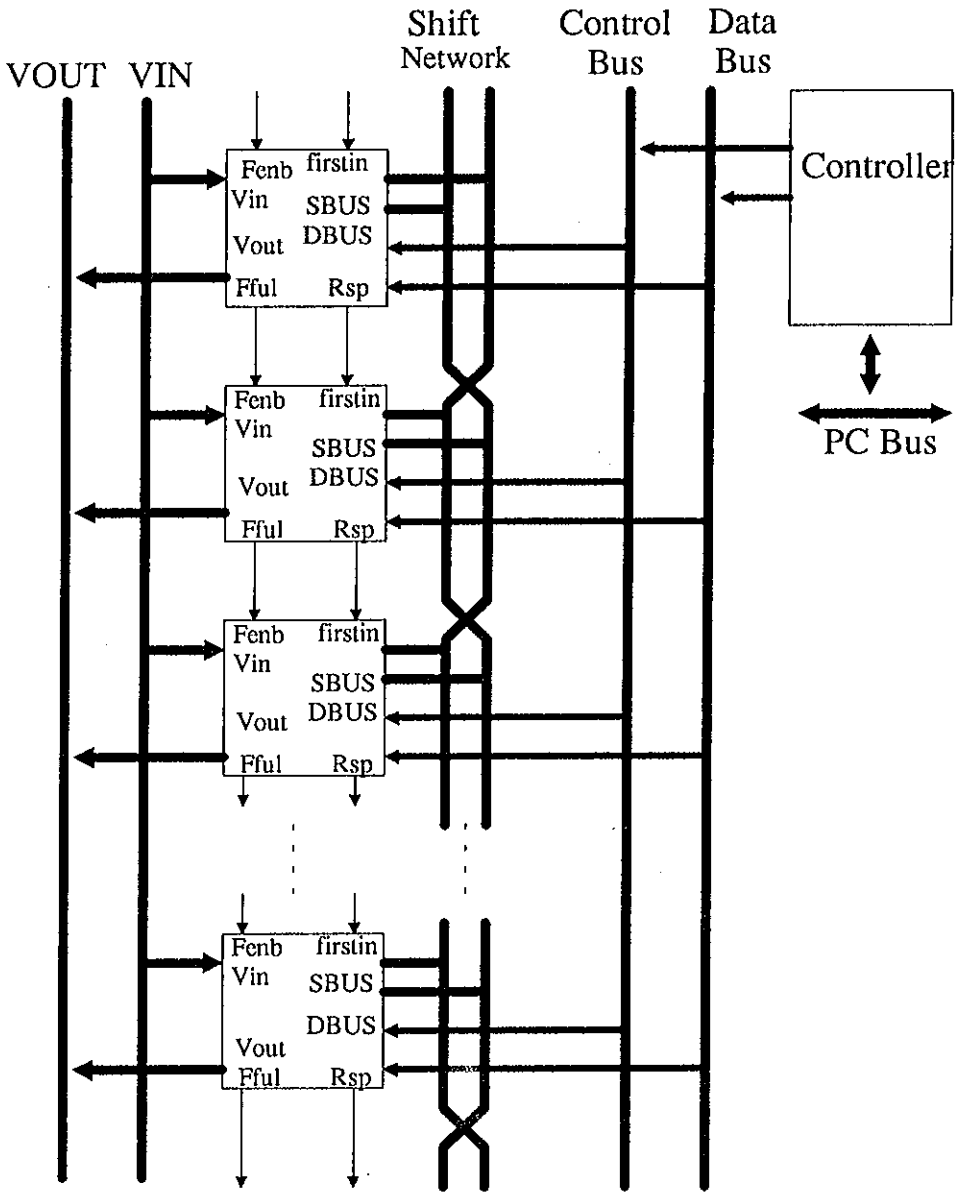


Figure 2.17. System configuration

ציור 2.17 מבנה מערכת

2.4 ASP100 design considerations and restrictions

2.4.1 Fabrication technology and CAD tools

The following issues were considered while choosing fabrication technology:

1. Density;
2. Speed;
3. Power dissipation;
4. Availability (technology that is provided by most of VLSI fabs);
5. Standard cell libraries that are obtainable for this technology.

According to these considerations, 0.8μ two-metal one-poly CMOS process has been used.

Two design methods were employed. Semi-custom (cell based) design was used for peripheral digital logic, and full custom design was used for the associative memory array, sense amplifiers and some/none response (RSP) circuit.

The great advantage of first method is that the cells have already been designed and characterized. Thus, design efforts have to be invested in area of logic design. In this case, there is almost full identity between logic model behavior and actual chip behavior. In contrast, in full custom design great part of effort is invested in area of generation, simulation, characterization and verification of the base hardware cells. Obviously, every such design step increases the likelihood of fault. In order to increase reliability of the design, we use the cell based design wherever it is possible. Thus, the SIDE, the TOP and the BOTTOM were designed using elements from standard cell libraries. The MATRIX was built and simulated as a full custom unit. Standard libraries given by a vendor are represented by some phantom cells. This means we can not have the full set of characteristics of the cells, but a number of models (abutment model for place and route, behavioral model for simulation, RC model for timing verification).

Implementation of both cell based and full custom design techniques requires using VLSI CAD tools that can integrate those methods. The ASP100 was designed using COMPASS Design Automation system. The specific tools that were employed are:

1. Layout Editor - used for drawing of custom layout;
2. Extractor - used for circuit extraction;
3. SPICE and ELDO - used for circuit simulation;
4. LVS - used for layout verification;

5 Logic Assistant - used for logic design;
6. VHDL Assistant - used for RTL logic simulation;
7. Mixed-mode simulator - used for gate level simulation;
8. ChipCompiler - used for place and route;
9. NetCompare - used for chip layout verification;
10. Mainsail Compiler - used for behavioral modeling of the CAM array for logic simulation
11. Timing Verifier - used for static timing analysis and verification

Integration of cell based and full custom design has a number of specific features, as follows:

1. Combination of synchronous control and data path (TOP, SIDE and BOTTOM) with asynchronous CAM array (MATRIX). The biggest difficulty of this combination is that most of the signals entering the MATRIX must be spike-free. Appearance of hazards on control signals may cause unrecoverable faults and even destroy operation completely. Hazard on the Match line precharge and discharge signals may fail COMPARE operation. Hazards on Write lines may initiate a false write operation, while hazards on Bit (Inverse Bit) lines may cause both compare and write mismatch. Since the cell based design is intended to be synchronous, there are no automatic tools aiding generation of spike-free logic units. To fix this problem, latching of the relevant MATRIX inputs was employed.

2. Interface between full-custom part and cell based part. Obviously, most of circuit characteristic parameters differ for these parts. First, no satisfactory information about signal levels is available from library vendor. The signal level in ASIC design is represented by logic values like '1' (logical one), '0' (logical zero), '1z' (precharged floating), '0z' (discharged floating), 'u' (unknown) etc., and there is no indication of the actual value. Second, there is no information about actual strength of standard cell buffers driving the full-custom blocks. This kind of information is essential for accurate full-custom circuit design and characterization. A proper solution to improve interface is using SPICE model of 'boundary' standard cells. In this design, the appropriate models were provided by the vendor after special request.

3. Simulation. The cell based units are simulated using behavioral models that are available for every standard cell. The full custom unit needs SPICE simulation. The SPICE simulation is carried out within two levels. First level is circuit simulation that validates the formal behavior. Second level is SPICE simulation of 'extracted' circuit, i.e., simulation with actual values of resistance and capacitance and parasitic devices. This simulation yields exact timing parameters (input/output capacitance, time delay, output driver power, signal ramp etc.) These parameters can be passed from SPICE model to a behavioral model of the circuit. This behavioral model can be processed by logic simulator. In the ASP100 case, introduction of timing parameters into behavioral model is essential for logic simulation accuracy. The ASP100 simulation is described with more details in Section 2.6

2.4.2. Power consumption

As mentioned above, there are two reasons that the ASP100 power consumption is expected to be higher than that of a regular processor. First, the ASP100 is a large processor, containing 1024 independent processing elements and peripheral circuits. Second, the activity factor (defined as ratio of the number of operative gates to overall number of gates) is higher than that of a regular processor because of high parallelism of operations. The ASP100 contains some independent power consumers, as following:

MATRIX

Power consumption of the CAM array is determined by type of operation (write, read, compare), number of active bit slices and number of active CAM lines.

1. Write. In this operation, new data are written into selected APEs. Most power is dissipated while switching Bit lines and Inverse Bit lines, and it obviously depends on the number of both active CAM columns and CAM lines. The worst case occurs while all CAM lines are active. The number of active lines is data dependent parameter and there is no possibility to affect it by program modification. On the other hand, the number of active columns depends on the algorithm and therefore can be controlled by a program. In order to restrict a power spike in the system, the number of

simultaneously active ARRAY (excluding FIFO) columns is limited to 24. This purpose is achieved by introduction of exclusive instructions.

2. *Compare.* In this operation, power is dissipated while precharging Match line, switching Bit and Inverse Bit lines and discharging Match lines (if mismatch occurs). Desired value of precharge current is defined by precharge device width. The worst case occurs while all CAM lines are active.
   The Match line sense amplifier is a CMOS device, and hence it does not consume a static power. Moreover, it is enabled only during compare operation.

3. *Read.* In this operation, power is dissipated in Bit and Inverse Bit lines while precharging and in Read sense amplifiers. The Read sense amplifier consumes both static and dynamic power components only while enabled. It is completely shut down by VSS (ground) line disconnection during execution time except read operation.

SIDE

TAG register is affected by WRITE, COMPARE and shift instructions only. The FF state changes only during COMPARE, shift and FIRSEL instructions. Otherwise, it is disabled and hence the power consumption decreases (to about 40% of consumption while enabled). The select first and response count circuits are normally disconnected from FF, thus their states do not change while SETAG and COMPARE instructions are executed. However, as mentioned above, they are not disconnected from supply lines. These circuits are affected by FIRSEL and COUNTAG instructions only.

TOP and BOTTOM

The TOP logic is affected by Group 1 instructions (load Mask and Comparand) and CONFIFO instruction. The BOTTOM logic is affected by READ and CONFIFO instructions. In both TOP and BOTTOM, all inactive registers are disabled (consuming about 40% of power while enabled).

I/O power consumption

The ASP100 image I/O bus (namely VIN and VOUT) operates with 50% activity factor (image I/O is carried out as sequence of FIFO READ/WRITE operations, while output buffers are activated during READ). The load capacitance of VOUT output bus is about 150 pF (eight or sixteen ASP100 chips are connected to the same system bus). This turns the VOUT drivers the largest I/O power consumer of the processor. The activity of the ASP100 external shift network (SHBUS) is highly algorithm and data dependent. For neighborhood operations like convolution, it is typically 30-40% (see Section 3, Convolution implementation example). The load capacitance of SHBUS is about 15-20 pF.

At 25MHz operation frequency, the ASP100 core power consumption is about 2-2.5W, while I/O power consumption is about 1-1.5W. The overall ASP100 average power consumption is 3.5W. Both power consumption of the cell-based designed units (TOP, SIDE and BOTTOM) and I/O power consumption were estimated using COMPASS Power Estimation Tools, basing on average consumption of all basic cells, activity factor and operating frequency. The power consumption of the full-custom designed units was obtained by calculating both peak and average current drawn by those units under 'worst case' conditions (Vdd=5.5 V, Process=N Fast, P Fast, Temperature=0°C). The estimation was done using SPICE simulation.

2.4.3. Chip Area

The ASP100 chip area is defined by an area of full custom block and an area of standard cell block. The full custom block area depends on the CAM array size and layout density. The standard cell block area depends on type of cell library used and routing between cells and blocks. There are two types of COMPASS cell libraries: high density cell library and high performance one. In current ASP100 design, the high density library is used. Routing factor depends on wire material and number of metal layers used for interconnection. Unfortunately, the COMPASS high density cell library uses polysilicon for inter-cell connections. It decreases utilization of metal 2 and thus decreases the routing factor. Typical cell routing factor in the standard cell part of the ASP100 is about 1.1 (it means that about 53% of area are occupied by wires).

2.4.4. Clock generation

The ASP100 uses two overlapping clocks called *clk* and *dclk*. Both clk and dclk are of the same duration, and dclk is delayed by a quarter of a period relatively to clk (see Figure 2.18). Numbers of implementations of a clock generator circuit were considered. The clk signal load is about 120pF, while dclk signal one is about 50pF. Unfortunately, the COMPASS standard cell libraries do not provide buffers with sufficient output gain. The only solution within cell-based design area is using the very high speed COMPASS clock pads. Implementation of this technique requires connecting the output of the clock generator circuit to the input of the pad. Thus, the simplest solution is using an external clock generator.

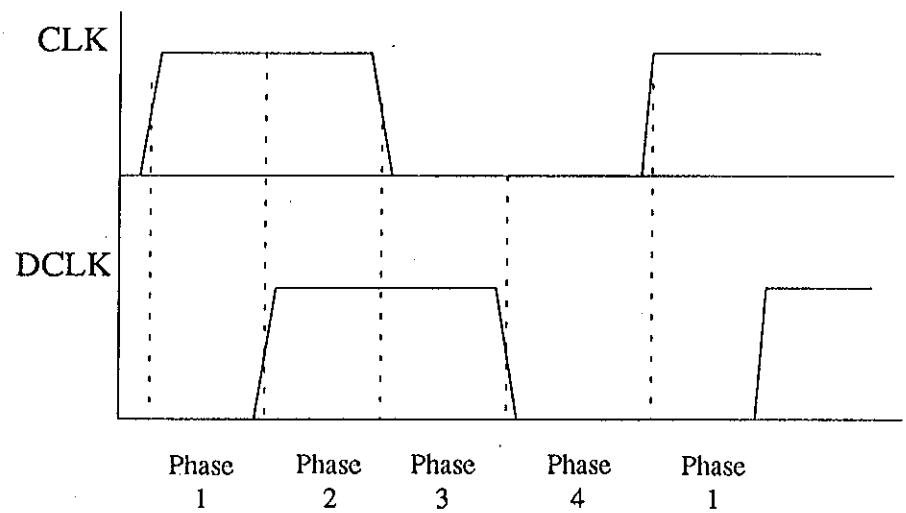


Figure 2.18: The ASP100 clock generation

ציור 2.18 מחולל שעון של הא'ס'פ'100

Utilization of two clock generators allows four-phase operation (using positive and negative edges of both clock sources, such that there is a clock event every quarter of a period). Four-phase operation is used to match the CAM array performance and overall system performance. The system performance is affected by the SIMD machine timing limitations (defined by number of boards, number of chips per board, number of shared buses, connectivity etc.). The system frequency was estimated to be 20-25 MHz. On the other hand, most basic CAM events (like precharge/discharge of Match line, precharge/discharge

of Bit line, write operation) take about 6-10 ns. According to this, some of the ASP100 instructions were implemented using four phases, as follows:

WRITE. The WriteEnable signal is active during first two phases. The Bit lines and Inverse Bit lines are valid during the first through third phases. During the fourth phase, the Bit and Inverse Bit lines are driven low.

COMPARE. The Precharge signal is active during the first phase. The Bit and Inverse Bit lines are driven low during the first phase and valid during the second through fourth phases.

FIFO READ. The Bit and Inverse Bit lines are precharged during half a period (two phases). Then they stay floating until end of cycle. The FIFOWriteEnable signal is active during the third and the fourth phases.

SETAG. This instruction uses asynchronous set input of the FF, then it should be certainly separated from clock event that the FF samples at. In this case, SETAG signal is active during second and third phases.

2.5 Design of content addressable memory array

The CAM array is the main processing and storing block of the ASP100. It obviously takes the largest part of the chip area and power consumption. Moreover, the basic CAM cell (APE) is repeated 72K times in this design. The Match line sense amplifiers and RSP circuit elements are repeated 1K times. This creates a very high motivation in selection of design goals and very high accuracy in CAM cell and peripheral circuit design.

2.5.1 Design requirements and goals

The following issues are most significant for a CAM cell design:

1. Reliability of information store. A datum once written into a CAM cell will never disappear until new write to this cell.

2. Reliable operations. This means the CAM cell operations are not to depend on

- condition of floating nodes
- values of node capacitance
- device sizes
- clock duration
- noise level

3. Silicon area. The cell area is most affected by the following factors:
   - using of P-channel devices (needs N well)
   - number of transistors
   - number of incident lines
   - number of contacts

4. Power consumption. Due to extremely high parallelism of operation, power dissipation of a single APE is most important design constraint. For every CAM operation, power consumption of the APE should be minimal. Full CMOS design of the APE allows avoiding static power consumption. The AC power consumption (charging and discharging of capacitors) depends mostly on Bit line, Inverse Bit line and Match line capacitance. Anyway, the APE should have as a low capacitance as possible.

5. Simple and reliable control. The APE operations should be independent (or least dependent) on order of operations, signal overlap, clock skew and so on.

6. Yield considerations. There are some design techniques that are not claimed as necessary design rules, but using them provides better final yield. These techniques include polysilicon and metal-2 superposition rules, via and diffusion (active area) superposition rules, distance between wide metal line to a regular width metal line and etc..

The CAM cell design goals are summarized in Table 2.3:

Table 2.3.: The CAM cell design goals

• Static store
• Minimal area on silicon
• Lowest power consumption
• Reliable execution

- Least parasite capacitance on signal propagation lines
- Simplest control
- Yield considerations affecting the CAM layout design

2.5.2 Static CAM cell design

There are three issues identifying a CAM cell, as follows:

1. Store type
2. Match logic
3. Write logic

The static CAM cell is based on a static memory element. The main disadvantages of static store are a high number of transistors (four) and the necessity of both p-channel and n-channel devices, that increases the silicon area. The static memory element is implemented by two cross-coupled inverters.

The Match logic is dedicated to support compare operation. The function of the Match logic is to compare the contents of the cell against the value on bit lines and to return 1 if they match and 0 otherwise. The Match logic has to provide the operation independent on result of comparison in other CAM cells. Since result of compare operation is logic AND of match results returned by active cells, the Match logic should not affect the whole result while the cell is masked (not active). There were two types of Match logic found.

The first type is Dynamic restoring XOR (see Figure 2.19.a). It is the full pull-down tree implementing XOR of the cell contents and values on Bit line and Inverse Bit line. During compare operation, both Bit line and Inverse Bit line are forced to zero and Match line is precharged to logic '1'. Then, the ***inverted*** match pattern is forwarded to bit lines of active cells. Both Bit line and Inverse Bit line of inactive cells stay low. If there is at least one cell that the pattern does not match the contents, then the Match line is discharged trough this cell. According to Figure 2.19.a, the storage device output is connected to gate of nearest to Match line transmission gate. This configuration avoids charge sharing during the second stage of the compare operation, while either Bit line or Inverse Bit line is forced to '1'.

The second type of Match logic is Dynamic steering XOR (see Figure 2.19.b). In this scheme, the XOR function is implemented by two n-channel devices. The enhancement device operates as isolator. During compare operation, both Bit line and Inverse Bit line are forced to '1' and Match line is precharged to '1'. Then, the match pattern is forwarded to bit lines of active cells. Both Bit line and Inverse Bit line of inactive cells stay high. The disadvantage of the second type is that Match line can not be driven to zero (the lower bound is threshold voltage). Obviously, both Match logic types need precharge mechanism and sense amplifier in every CAM line. The common disadvantage of dynamic Match logic that decreases the design reliability is that Match line is floating until the end of compare cycle.

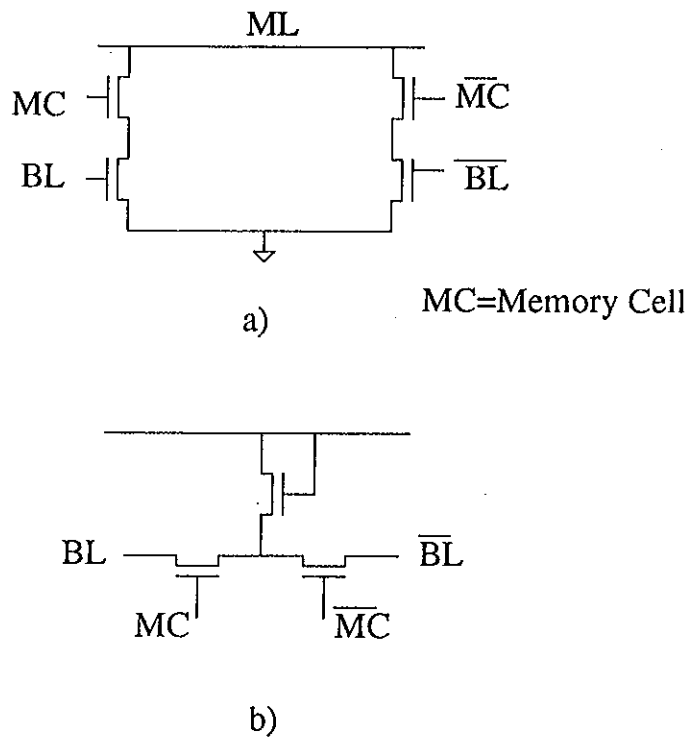


Figure 2.19: Match logic types

ציור 2.19 צורות מנגנוני התאמה

The Write logic is dedicated to support write operation. Three different types of Write logic are depicted in Figure 2.20.a,b,c. In scheme *a*, the Write line is forced to '1' and then Bit line value is written into the cell. Scheme *a* is the simplest Write logic, but it does not support masked write operation. This problem is solved in scheme *b*. In this scheme, the Bit (Inverse Bit) line value is

written into the cell if both Write line and Inverse Bit (Bit) line are high. This means that one of the memory element nodes is always floating during write operation, while the opposite one is forced to '0'. The masked write operation is carried out while both Bit line and Inverse Bit line are kept low. The disadvantage of this scheme is that parasitic dynamic store created at node A (see figure 2.20.b) may be dangerous during masked write. The Write logic that was implemented in the ASP100 design is the scheme *c*. The disadvantages of this scheme are introduction of a special Mask line and relatively high resistance on write/read path. In regular write operation, both Write line and Mask line are forced to '1', and Bit line value is written into the cell. Both nodes of the memory elements are driven (no floating nodes). In masked write, the Mask line stays low. During read operation, both Bit line and Inverse Bit line are precharged to '1' and then both Mask line and Write line are forced to '1'. If the CAM cell stores '1' ('0') then Bit line (Inverse Bit line) stays high while Inverse Bit line (Bit line) is discharged trough the cell. The Read sense amplifier responds once the difference of Bit line and Inverse Bit line voltages achieves its sensitivity level.

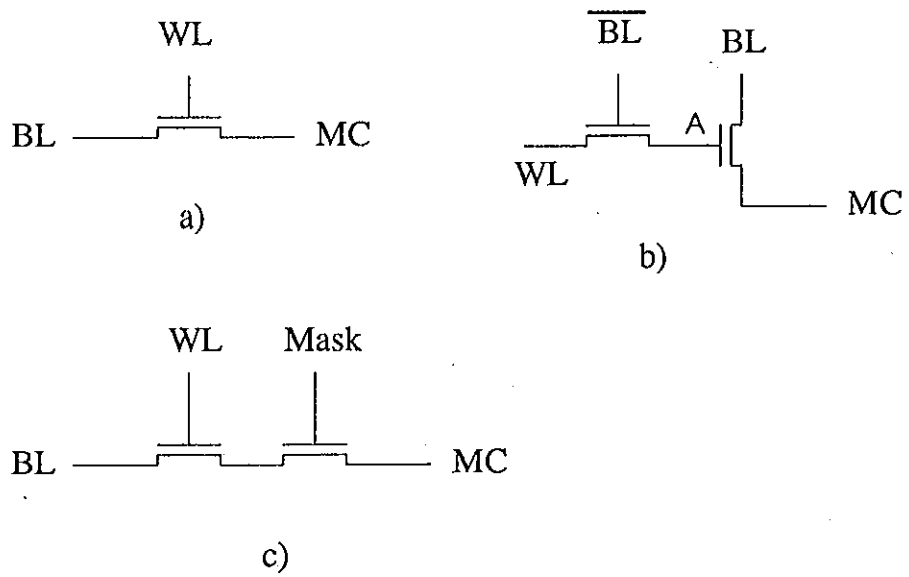


Figure 2.20: Write logic types

ציור 2.20 צורות מנגנוני כתיבה

The ASP100 CAM cell is depicted in Figure 2.4.

2.5.3 Non-static CAM cells

The ASP100 APE employs the static information storage. Actually, there are two other design options:

1. Self-refreshing or pseudo-static. Here data storage occurs trough making use of the capacitance on the transistor gates. This charge will decay in time, but by asserting a control signals the cell can be configured to behave as a static storage device and restore the data, although at the expense of increased power consumption.

2. Dynamic. Similar to the one transistor RAM cell, this design is compact, but relies on external logic to sense and refresh the data signals.

As mentioned before, the static data storage is the most rugged (in the silicon area sense) circuit option. In opposite, the pseudo-static is smaller, but requires the data to be regularly refreshed. However, its self-refreshing capability means that all the cells can be refreshed simultaneously. The dynamic cell structure offers the potential for highest packing density. However, the need to regularly refresh the cell individually involves more complex support logic and a significant time penalty. More important, since both direct and inverse data should be held in each cell to support the Match logic, two dynamic storage circuits per CAM cell would be needed. This means there is probably no big difference between density of pseudo-static and dynamic CAM.

Pseudo-static CAM cell is depicted in Figure 2.21. [18]. During the write operation, WL is forced to '1' and then nodes L and R get value of BL and Inverse BL respectively. In masked write operation, both BL and Inverse BL are forced to '1'. Thus, the refresh operation occurs. During read, both BL and Inverse BL are precharged to '1' and then WL is forced to 1. To sense the BL and Inverse BL voltage difference, an external sense amplifier is needed. The compare operation is similar to that of static CAM cell. The advantage of pseudo-static storage over static one is decreasing of the number of transistors (two instead of four) and that the transistors both are of n-type (it means N-well is eliminated). The great disadvantage of such pseudo-static cell is large power consumption. In an associative vision machine, about 85% of the CAM cells are masked during write operation, while write instruction appears with

probability ≈0.45 [19]. This means that 85% of the CAM cells will consume static power during half a running time.

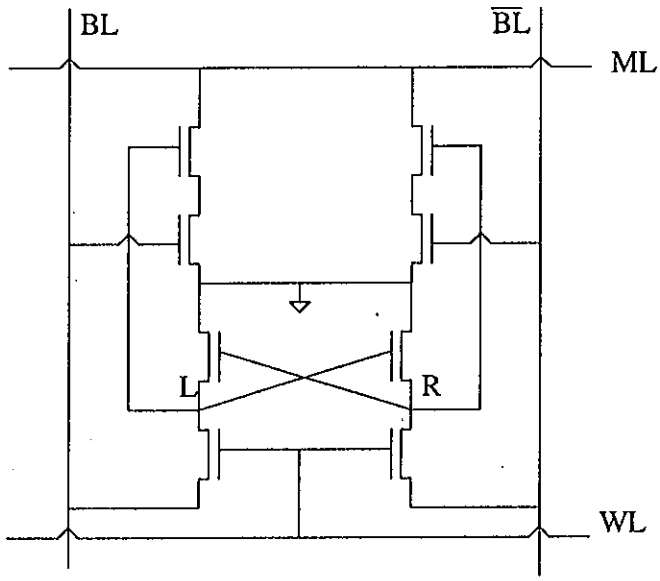


Figure 2.21: The pseudo-static CAM cell

ציור 2.21 תא פסאודו-סטטי של זכרון ממוען כתובות

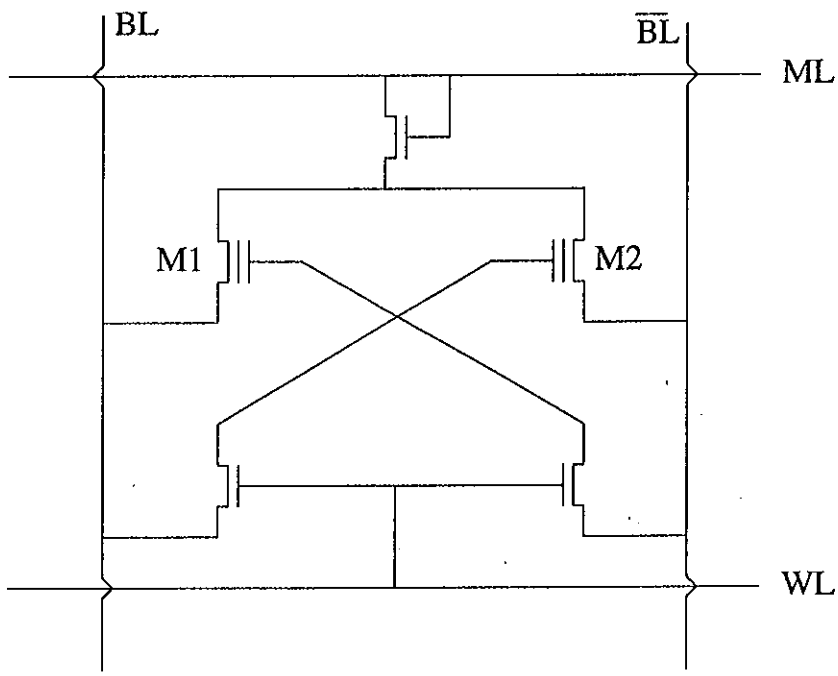


Figure 2.22: Dynamic CAM cell

ציור 2.22 תא דינמי של זכרון ממוען כתובות

Dynamic CAM [7] cell is depicted in Figure 2.22. In this cell, the information storage is implemented by floating gate n-type CMOS transistors (See Figure 2.22, M1 and M2). Note, that dynamic steering XOR scheme was implemented for Match logic, while M1 and M2 both used for information storage and for matching. The great advantage of this cell is small number of transistors (five only, in opposite to eight of pseudo-static CAM cell and ten to twelve of static CAM). The basic operations (write, read and compare) are carried out in the same way as in already described CAM cells. An important difference is that the dynamic CAM cell does not support masked write.

2.5.4 CAM cell comparative analysis

The following tables contain results of comparative analysis of four static CAM cells composed employing different Write and Match logic schemes, and the pseudo-static CAM cell described above.

Table 2.4 The CAM cell description
(according to Figures 2.19, 2.20)

Cell name	Write logic	Match logic
Static Cell 1	c	a
Static Cell 2	c	b
Static Cell 3	b	b
Static Cell 4	b	a
Pseudo Static	a	a

Table 2.5 The comparative analysis of CAM cells

Cell name	Cell 1	Cell 2	Cell 3	Cell 4	Pseudo
size, λ	71*73	70*73	75*74	75*76	61*58
area, μ^2	829	820	885	910	565
gate account	12	11	11	12	8
number of interconnection lines	3*2	3*2	2*2	2*2	2*2

total Bit Line resistance, Ohm	355	345	390	390	287
total ML (WL) resistance, Ohm	190	190	195	205	145
total Bit Line capacitance, pF	7.3	7.2	7.7	7.7	6.4
total ML capacitance, pF	2	2	2.1	2.2	1.6
total WL capacitance, pF	1.9	1.8	2	2	1.4
CAM array size, *mm*×*mm*	9.5×9	9.3×9	10×9.5	10×9.7	8×7
BL buffer peak power, W	0.98	0.97	0.64	0.64	0.7
BL buffer average power, W	0.034	0.033	0.02	0.02	0.024
Total average power write, W	0.55	0.55	0.39	0.39	5.1
total average power compare single, W	0.46	0.56	0.56	0.46	0.42
total average power compare all, W	0.51	0.72	0.72	0.51	0.51
write time, nsec	9	12	20	14	12
precharge time, nsec	14	5.4	6.4	8.1	6.5
discharge time, nsec	28	33	33	30	28

For static cells, the cell 2 (write scheme c, match scheme b) based CAM has the minimal physical size, while the cell 1 (write scheme c, match scheme a) based CAM has better timing and power consumption parameters. As we also can see, the pseudo static cell based CAM has the best area parameters, while power consumption during write operation is extremely high. The ASP100 chip exploits the cell 1 based CAM array.

2.5.5 Peripheral circuitry design

Three peripheral circuits were implemented in the ASP100 full-custom design. They are:

1. Match line sense amplifier and precharge circuit
2. RSP (some/none responder) circuit
3. Read sense amplifier

The Match line sense amplifier is depicted in Figure 2.11. The reason to implementation of some sense circuit is acceleration of compare. The 'critical path' of compare operation is discharge of Match line that is carried out trough a single cell in the worst case. Obviously, the Match line discharge time is limited by Match line capacitance and pull down transistor size. The estimated worst-case Match line discharge time is 30 to 40 ns, that obviously slows down the operation of the 25MHz vision machine. The main limitation of Match line sense amplifier is power consumption. The reasons to this are number of Match line sense amplifiers (1024 per chip !) and their activity (according to operation statistics, the COMPARE instruction appears with probability of 0.4-0.45 [19]). This requires the Match line sense amplifier to consume no static power and minimal dynamic power. The Match line sense amplifier is implemented by unbalanced inverter with high trip-point (about 3.5V instead of usual 2.5V). The amplifier returns zero once the Match line value decreases to 3.5V (it takes about 12-16 ns). The lower bound of trip-point is determined by timing requirements, while the upper one is limited by noise margin. To reduce power consumption, the Match line sense amplifier is connected to VSS supply line (enabled) only for two clock phases (half a period) of the compare cycle. Match line precharge circuit is implemented by pull-up device connecting match line to VDD supply line during precharge phase. The size of precharge device is determined by timing requirements (less or equal to a single clock phase) and is restricted in order to reduce power consumption during precharge.

The RSP (some/none responder) circuit is shown in Figure 2.23. The RSP circuit is implemented as wired NOR. Before responding, the RSP line is precharged to '1'. Then, every bit of TAG register is connected to appropriate pull-down device of the RSP circuit. If at least one TAG bit is high, the RSP line is discharged. Otherwise, it stays high. To accelerate the response, a sense amplifier (similar to Match line sense amplifier) is employed. The RSP precharge device is implemented by pull-up transistor controlled by the ASP100 internal control unit.

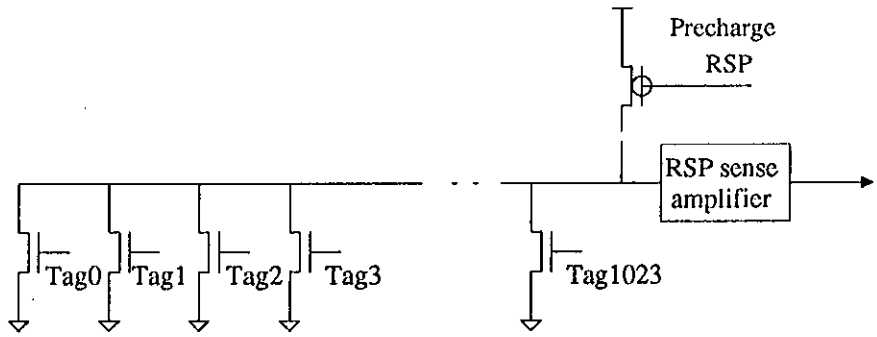


Figure 2.23: The RSP circuit

ציור 2.23 מנגנון התגובה

The Read sense amplifier consists of differential pair, level shifters and some output logic (see Figure 2.14). The transistor sizes were selected to sense 0.6-0.8 V difference between Bit line and Inverse Bit line values. This level is determined by noise margin consideration. To reduce power consumption, the Read sense amplifier is connected to VSS supply line (enabled) for two clock phases of a read cycle.

2.6 ASP100 simulation

Simulation is the main tool for functional (logic) and timing verification of the design. Since the ASP100 has been designed using both full-custom and cell based design, two different simulation levels were used. First is circuit simulation, applied for full-custom unit verification. Second is logic simulation, dedicated for verification of cell based blocks. To complete the system simulation, results of circuit simulation should be interpreted in logic simulation in some manner. For this purpose, a behavioral model of the CAM array was created and included in logic simulation. The physical parameters used by the model were obtained from circuit simulation. For both types of logic simulation (VHDL and gate-level), the input stimuli were generated by software simulator/compiler [19].

2.6.1 VHDL simulation

The VHDL simulation is the high level logic simulation. The simulator input is a set of behavioral models of the standard cells, automatically generated from

the netlist, and behavioral model of the CAM array. The purpose of VHDL simulation is functional validation. In order to simplify the debugging process, unit time delay approach was used. It means that time delay of every standard cell is a single time unit. Obviously, this method does not verify timing. Nevertheless, it clarifies the internal processes and logic dependencies. It also aids in finding out some problems, that gate-level simulator would smooth as result of RC modeling (like glitch generators). The VHDL simulation was performed by the ASP corp. using the VHDL format netlist of the ASP100 standard cell part, supplied by us.

2.6.2 Gate level simulation

The main tool for complete system verification is gate level simulator. Unlike the VHDL simulator, there is no behavioral modeling (except the CAM array model). The gate level simulator uses a structural modeling, i.e., every design unit is simulated at its gate structure level. There are some issues that are verified during gate level simulation:

1. Timing. The gate level simulator exploits RC models of standard cells. Both RC modeling and prediction of interconnection capacitance make possible timing verification even before Place and Route. Obviously, the final timing results can be obtaining from post-route gate level simulation (simulation of extracted netlist).

2. Ramp (slope) delay. Ramp delay is a signal latency period due to transaction. Using the gate level simulation, it is possible to verify a ramp delay at every chip node. Detection of nodes with long ramp delay is very important: First, long ramp delay may cause some timing problems (like creation of false critical paths). Second, it obviously increases power consumption by keeping the CMOS gates at some mid voltage value (between '0' and '1') for a relatively long duration.

3. Fault coverage. The ASP100 does not contain standard cells that have an automatic test generation facility. Nevertheless, we were able to generate high fault covering tests using toggle feedback. The toggle feedback reports about nodes that were not driven high and/or low. It actually gives the

designer an immediate report about fault coverage of the test and thus simplifies test generation.

2.6.3 Circuit simulation

The main tool for circuit simulation is SPICE. The circuit simulation was carried out within two stages. In the first stage, the logic transistor level circuits were simulated. The second stage was simulation of extracted layout. Actually, the most important extracted information is the inter-node and inter-layer capacitance. The COMPASS Circuit Extractor extracts no resistance neither parasitic devices. The following approaches were implemented during circuit simulation:

1. Functional simulation. We simulate the actual CAM array operations (compare, read, write, FIFO write and read) rather than some complicated events on different circuits. We believe this technique aids in better integration and reliability of simulation.

2. Simulation accuracy. An appropriate simulation accuracy can be achieved by correct circuit modeling and making right assumptions. According to our short experience in this area, we tried to make as fewer assumptions as possible, and simulate the circuits without simplification. Obviously, there is no possibility to run SPICE simulation of circuit containing about 300,000 devices (number of columns × number of rows × number of transistors per APE). Nevertheless, using the ELDO circuit simulator, we simulated groups of the CAM array rows and columns (up to 10,000 transistors). The entire CAM array row was simulated using SPICE (about 1,000 transistors). For the last case, simulation time is about half an hour per clock period running on SUN 10-31 workstation.

3. Worst case simulation. For every simulation, we used the worst case parameters that were relevant for the simulation. Timing simulations were carried out at highest temperature (120°C), lowest voltage (4.6V) and Slow N, Slow P model parameters. Power simulations were carried out at lowest temperature (0°C), highest voltage (5.5V) and Fast N, Fast P model parameters. Moreover, some extreme operation assumptions were usually made. An example is Match line precharge simulation. The regular time

period for this operation is 10ns (one clock phase). We assumed the worst case of *clk* to *dclk* skew and signal delay and then simulated the precharge operation within 6ns period.

3. Image processing and computer vision applications

According to operation statistics, the maximal performance of the associative processor can be achieved implementing low-level vision applications. The reason is that the associative processor machine has very high activity while executing arithmetic-logical operations (they are carried out in massively word-parallel manner). Unfortunately, the activity decreases dramatically performing data communication operations, except neighborhood operations while all image data are transmitted in exactly the same way. Actually, the CAM array is never activated during communication operations. Nevertheless, still the associative vision machine can successfully perform some mid-level image processing and computer vision tasks. As we will see, the 'bottle neck' of these applications is data communication rather than arithmetic-logical operations. An efficient way to face with the problem is further development of the SIMD machine communication networks issue. To exhibit and analyze the ARTVM performance, we consider implementation of some low and mid-level image processing and computer vision tasks. The operating frequency, the instruction length, the shift network parameters and other estimation tools were taken from the actual VLSI design and simulation. The image is assumed to be arranged linearly in the CAM array, line by line according to video scan. The Long Shift is of 16 position length. The operating frequency is 20MHz. The instruction lengths are derived from the Table 2.2.

The following low-level image processing algorithms were estimated:

1. Histogram

The associative nature of the ARTVM and its relatively fast response counter mechanism make histogram evaluation very suitable. It consists of short loop repeated for every gray level. The COMPARE instruction tags all pixels of the level and then COUNTAG instruction counts them up. There is no data

communication in this algorithm (except of course the image I/O that is carried out in parallel with processing). The CAM array is active for about 7% of running time, while the response count circuit is active during 92% of the running time. For 8-bit pixel image, the histogram evaluation takes about 350μsec.

2. Convolution

In this algorithm, the image data of wordlength M are initially placed in a data field. The filter vector of length P and coefficient wordlength K is applied as operand from the external controller, one element at a time. The result is accumulated in a partial result field. After every multiplication, the data field is shifted by one word position (for convolution in line direction, it is shift by 1, while for column direction, it is shift by N, where N is an image length). For 2D non-separable kernel, an optimal shift strategy is $P(P-1)$ short shifts and $(P-1)$long shifts. The width of the partial result field is obviously larger than the data field width, therefore every addition has to be followed by carry propagation to the end of the partial result field. The string recoding multiplication algorithm provides an average multiplication time of $\frac{M \cdot K}{3}$ one-bit additions [8,9]. According to this, the convolution time for $P \times P$ kernel is as follows:

$$T_C \approx P^2 \cdot (\frac{M \cdot K}{3} + M + K + \log_2 P) \cdot t_{FA} + (P-1) \cdot \frac{N}{16} \cdot M + P \cdot (P-1) \cdot M \qquad (*)$$

where t_{FA} is one bit addition time (usually 8 cycles). The first component of the equation is due to arithmetic operations, while the second and the third ones are due to communication operations. According to (*), the convolution time for 3× 3 kernel, for 8 bit precision pixels and coefficients and for image size 512×512 is about 100 μsec.

In convolution evaluation, the data communication occupies about 33% of the running time, while arithmetic operations take about 66% . This ratio is almost independent on kernel size. The component of arithmetic calculation increases with increasing of data precision.

3. Canny edge detection

This algorithm [10] has three stages:

1. Convolution with directional derivatives of Gaussian filter $\frac{-n}{\sqrt{2\pi\sigma^3}} e^{\frac{-n^2}{2\sigma^2}}$
2. Non-maximum suppression
3. Threshold with hysteresis

The second stage selects as edge candidates the pixels in which the gradient magnitude is maximal. The test can be carried out in 8 directions (using 3×3 neighborhood). To determine if maximal, the gradient value at each pixel is compared with those on either side of it. Thresholding with hysteresis eliminates weak edges that may be due to noise but continues the strong edges as they become weak. In order to differ these edges, two threshold values (low and high) are used. Candidates with value between low and high are considered as edges if they can be connected to a pixel above high trough a chain of pixels above low. This process is iterative and involves curve propagation. The iteration break condition is returned by the RSP mechanism. For 100 iterations of curve propagation and 3×3 convolution kernel, the Canny edge detection takes about 300 μsec. The ratio between data communication and arithmetic-logical operation is quite similar to that of convolution.

4. Optical flow

Optical flow assigns to every pixel of an image a velocity vector that describes its motion across the visual field. The Horn & Schunk [11] algorithm for optical flow computation is an iterative process, estimating new velocity vector at each iteration as sum of the old average velocity and some innovation value. This innovation is estimated as linear combination of weighted pixel values in 3×3 neighborhood. Thus, this algorithm is very suitable for the ARTVM. An associative implementation of this algorithm contains two steps, initial and iterative. In the initial step, the constant parameters used in the iterative step are computed. The data communication occupies about 10-15% of the running time, while arithmetic and look-up table operations take about 85-88%. For 32 iterations using 8-bit data precision, the optical flow computation takes about 10 msec.

5. Contour tracing and labeling

This algorithm is carried out in two steps. The preparation step labels each contour point (obviously, a previous edge detection is needed) with its x-y

coordinates. The main process is iterative and operates on a 3×3 neighborhood. Every contour point checks each of 8 neighbors and adopts its coordinate label if it is smaller then its own. This process is iterative and it continues until all labels remain unchanged leaving each contour identified by its lowest coordinate. The point of lowest coordinates in each contour is the only one retaining its original label. These points can be counted to obtain the number of contours in an image. The data communication and arithmetic operations occupy both about half a running time. For 100 iterations and 8 bit data precision, the contour labeling time is about 1.5 msec.

6. Hough Transform

The Hough transform can detect a curve whose shape is described by parametric equations, like a straight line or a circle section, even if there are gaps in the curve. To implement the transform, each pixel in an image is transformed to a locus in the parameter space. After splitting the parameters into suitable ranges (quantization), a histogram is generated giving the distributions of locus points in the parameter space. Occurrence of an object curve is signed by a distinct peak in the histogram. Matching straight lines in an image (using 64-point histogram evaluation) takes about 5 msec. Matching circles takes a little bit more due to more complex parameter space transformation.

List of references

1. Akerib A., Ruhman S., "Real Time Associative Vision Machine", *Artificial Intelligence and Computer Vision*, pp 441-453, 1991.

2. Weems C., "Image processing on a content addressable array parallel processor", *Ph.D. Thesis*, Computer and information science, University of Massachusetts, USA, 1984.

3. Flynn M., "Some computer organization and their effectiveness", *IEEE Transactions on Computers*, v. 21, no. 9, pp. 948-960, 1972.

4. Batcher K, "Bit serial parallel processing systems", *IEEE Transactions on Computers*, v. 31, no. 5, pp. 377-384, 1982.

5. Fernstrom C., Kruzela I., Svensson B., *LUCAS. Associative array processor*, Springer-Verlag, 1986.

6. Foster C., *Content addressable parallel processors*, Van Nostrand Reinhold, 1976.

7. Watabe Kiyoto, Shinohara Hirofumi, Eimori Takahisa, Arima Hideaki, Ajika Natsuo, Yuichi Nakashima, Shinichi Satoh, "Content addressable memory device", *U.S. Patent* no. 5,051,948, 1991.

8. Hwang, *Computer arithmetic. Principles, architecture and design*, John Wiley & Sons, 1979.

9. Swartzlander (ed), *Computer arithmetic*, Benchmark papers in Electrical Engineering and Computer Science, V. 21.

10. Canny J., "Computational Approach to edge detection", *IEEE Transactions on PAMI*, pp. 679-698, 1986.

11. Akerib A., "Associative real time vision machine," *Ph.D. Thesis*, Computer Science, Weizman Institute of Science, Israel, 1992.

12. Scherson I., "Multioperand associative processing and application to tomography and computer graphics," *Ph. D. thesis*, Computer Science, Weizman Institute of Science, Israel, 1983.

13. Shooman W., "Parallel computing with vertical data," *AFIPS conf. proc., EJCC*, v. 18, pp. 111-115, 1960.

14. Gonzalez R., Wintz P., *Digital signal processing*, Addison-Wesley, 1987.

15. Rao K., Yip P., *Discrete Cosine Transform. Algorithms, Advantages, Applications,* Academic Press, 1990.

16. Bluestein L., "A linear filtering approach to the computation of the discrete Fourier transform," *IEEE Trans. Audio Electroacouts.*, AU 18, pp. 451-455, 1970.

17. Blahut R., *Fast algorithms for digital signal processing*, Addison-Wesley, 1985.

18. Jones S., "Design, selection and implementation of a content addressable memory for a VLSI CMOS chip architecture," *IEE Proc. E*, v.35, n.3, pp. 165-171, 1988

19. The ASP100 Software Simulator., ASP corp., 1994

20. Lea R., "ASP: A cost-effective parallel microcomputer," *IEEE Micro*, pp. 10-29, October 1988.

21. Krikelis A., "Benchmarcking the ASP for computer vision," *SPIE*, v. 1360, *Visual Communication and Image processing*, pp. 92-103, 1990