ABSTRACT

Title of Thesis: A Common Architecture for Processing Data from Thin Film Sensor Arrays Tailored to 3-D Applications

Rajasekhar Jammalamadaka,
Master of Sciences, 2006

Thesis directed by: Professor Neil Goldsman
Department of Electrical and Computer Engineering

Thin film sensor arrays on ICs are studied as an example of 3-D integration. Various thin film sensor arrays are studied, with emphasis on systems integratable on silicon. 3-D integration offers the chance to have all the data from the sensors in the sensor arrays available to the processing layer at once. The data from the sensors is read vertically directly into the processing layer. With the motivation to identify the needs of various thin film sensors and to exploit the available parallelism in data, thin film sensors are categorized into two - those with similar sensors in the arrays or those with different sensors in the sensor array. A common architecture to address both the categories is proposed, designed and implemented. Such a generic processing layer design also helps independant development of sensor arrays (of either categories) and finally deposited on such a generic processing layer.

Also any commercial off the shelf generic microprocessor can not be used for such a project. The reasons being that the generic microprocessors were not designed keeping in mind the vertical integration and hence parallelism in data.

Nor are they designed to handle the specific needs of sensor arrays. The modSIMD processor (proposed and implemented in this thesis) exploits the 3-D integration aspects and the needs of sensor arrays. modSIMD stands for a modified SIMD (Single Instruction Multiple Data) architecture. In the common addressing mode, it behaves like a SIMD processor and works on various data elements parallely. In the specific addressing mode, it behaves like a SISD (Single Instruction Single Data) processor.

First the advantages of 3-D integration are studied. The 3-D system architecture is compared to a 2-D system architecture with similar processing architecture. The 3-D architecture is seen to dissipate more power but is more efficient in area and speed. Then the processing architecture is put to test with applications from both the categories. It outperforms an ARM microprocessor in SIMD applications, thus exploiting the parallelism in data. modSIMD also works on applications from both the categories, thus developing such a common architecture is made possible.

A Common Architecture for Processing Data from Thin Film Sensor Arrays Tailored to 3-D Applications

by

Rajasekhar Jammalamadaka

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Master of Sciences
2006

Advisory Commmittee:
Professor Neil Goldsman, Chair/Advisor
Professor Martin Peckerar
Associate Professor Bruce Jacob

This dissertation is dedicated to my parents.

ACKNOWLEDGEMENTS

I am grateful to Dr. Neil Goldsman, who has been tremendously patient with me as an advisor. I also would like to thank the ECE Dept and specifically the Technical Staff in the department for constantly upgrading the systems and software and keeping them uptodate. I would like to thank Zeynep Dilli, fellow graduate student ECE Dept, University of Maryland.

And to all those of my friends, who know you deserve to be acknowledged - be you in 120 Westway or in AID, a big thank you for being there with me, for all these years. A family away from family, I am glad I had all of you with me during my graduate studies.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

Integrated circuits came into being in the late 1960's. Microprocessors built of ICs became popular from the 1980's. With huge leaps in technology and following the Moore's Law, transistor count on an IC has been doubling with more and more powerful processors being built. With limits in technology - both going close to the physical limits as well as memory access issues and heat dissipation problems, the industry is looking at various fields for growth. Research is going on a variety of processes, the leader of them being nanotechnology.

The silicon industry, meanwhile is studying approaches to best use their existing technology in silicon engineering for growth. This has led to a lot of investment into the multi-core technologies. Though the core might clock at a lower frequency than a single core, the actual gains could be about 30-50% (for a dual core processor) compared to the single core. This does require specialized programming to exploit the dual cores or the multi-cores.

Some research groups are also investigating the use of 3-D technology to enhance technology. The circuitry in the processors is all two dimensional. The depth

of a transistor is a few microns, on a substrate of silicon which is possibly a couple of hundreds of microns thick. With the device dimensions reaching nanoscale and thereby reaching the physical limits in increasing the speed of the devices, researchers have started exploring materials in various forms. One of such interesting avenues is 3-D processors or processors which exploit the third dimension.

Memory access has always been a problem for processors. The development of the hard disks has been slower than the development in processors. Thus computer architects had come up with a variety of schemes to overcome these issues, some of which include a bigger on-chip cache, reliable branch prediction and various bus protocols to transfer the required data faster to the core for processing. 3-D integration also promises to provide faster data access to the processing core through vertical interconnects.

1.1 3-Dimensional Processors

A 3-Dimensional processor can be achieved in a couple of ways. One way is to stack up various blocks of a processor, one over the other. Another way, and reported in literature for photosensors is to use a thin film layer on an ASIC [8]. Sensors for various types are available on thin films compatible with silicon. A simple 3-D thin film sensor array on an IC is depicted in Figure 1.1.

In the following few sections, a few important developments in related fields is briefly presented. The core idea of the thesis is developed and presented towards the end of this chapter.

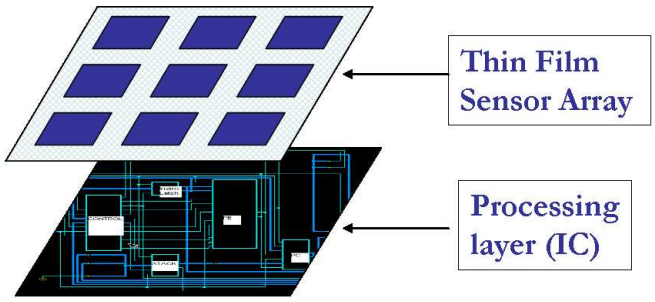


Figure 1.1: A 3-D Thin Film Sensor Arrays on IC

1.2 Multi-processing, multi-core chips?

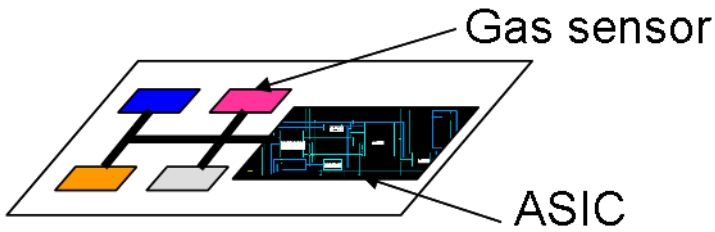
Architectural improvements in single-core microprocessors that take advantage of continually increasing transistor budgets, such as superscalar execution and deeper pipelines, have improved performance tremendously over the years. However these techniques have seen diminishing returns recently. As a result, server manufacturers are now using their additional transistors to produce multi-core chip multiprocessors (CMPs), that sidestep these diminishing returns by executing the many threads typically available in server tasks in parallel across the cores of a CMP.

Traditionally multi-processors have been used for high end servers, doing specialized tasks. With imporvements in silicon technology, a shift to chip multiprocessing has been seen.

1.3 Sensors

Parallely sensors in silicon have gained a lot because of this development in technology. A variety of sensors for images/video, pressure, chemical and magnetic sensors have been developed in silicon and devices have been miniaturized [1]. Sensors of

various types mentioned have been built adjoining to processing circuitry. A structure as depicted in Figure 1.2 for gas sensors is apt for various other sensors too.



Thin film sensors have been researched with great interest. Silicon thin films, circuits on SOI substrates or thin films on silicon compatible systems are of particular interest, as systems which exploit the investments already made by a huge number of firms in silicon can be used to build these sensor systems [13][11][17].

1.4 Thin Film Sensor Arrays (TFSAs) on CMOS circuits a solution?

Traditionally data from the sensors is fed into a processor and then processed. This transfer of data is also a bottleneck. Architecturaly this has been approached with on-chip cache and deep pipelines.

A parallel sensor array can feed a lot of data into a processor. Pre-processing the data just to transfer the necessary items requires a lot of processing power. For example to process and get a color image for a 640*480 pixel array takes about 300 Million Operations Per Second (MOPS). Since the pixel array (on a thin film) can feed the data to the processing layer directly underneath it, parallely, one can build a multi-processing processing layer to process color images. This is just one example as shown in Fig 1.1.

The applications of TFSA on silicon circuits are many. They can be broadly categorized into two, depending on the sensors in the sensor arrays. The crux of the thesis is to identify the commonalities in these structures and develop a common, generic processing layer to process data from the TFSAs.

1.4.1 Similar sensors in the TFSA

All TFSAs which have similar sensors in the sensor array (Figure 1.3) - tactile sensors or photosensors or thermal sensors. Applications of such an array would be live finger print scanning (tactile sensors), images (photosensors and thermal sensors).

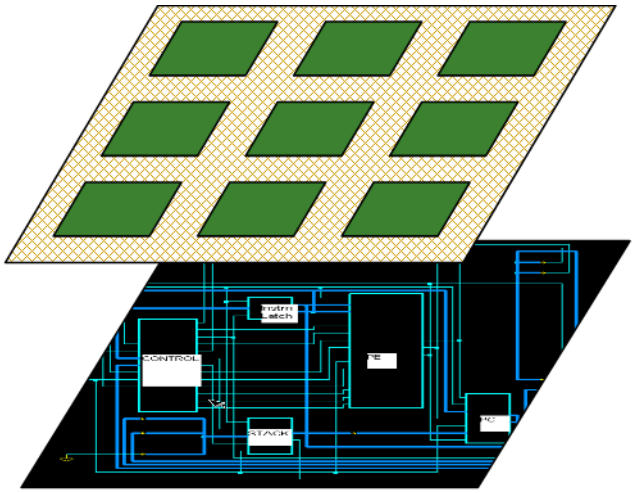


Figure 1.2: Similar sensors in TFSA

1.4.2 Differnt sensors in the TFSA

An array of sensors, each sensor different from another can also be created, technology permitting on top of a silicon processing layer (as visulaized in Figure 1.4). Gas/chemical sensors are dissimilar sensors, which compare the results of all the

sensors to identify a particular chemical or gas.

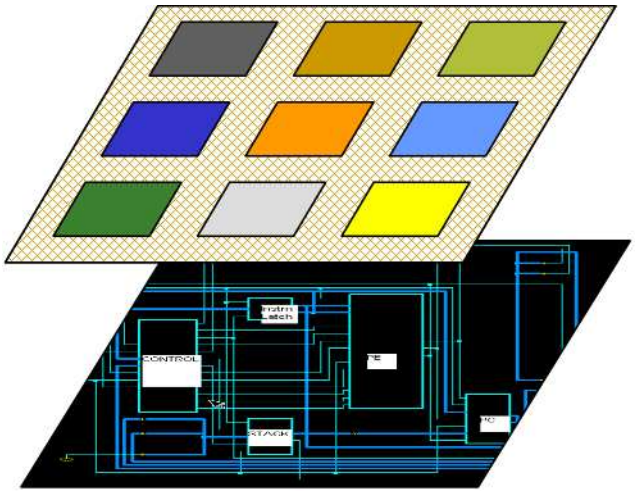


Figure 1.3: Different sensors in TFSA

Also, for deep space exploration (either Mars reconnaissance orbiter or Pluto reconnaissance orbiter) or battlefield reconnaissance - one can build a sensor system with a variety of sensors (tactile - to measure pressure, gas/chemical - to check for chemicals, photo and thermal - for images, etc) on top of a single processing layer. Such a sensory system would not only provide information about the surroundings but also be light and extremely compact.

1.5 Thesis overview

The purpose of the thesis is to find and present various technologies that can be part of the two layers as depicted in Figure 1.1. The processing layer should be able to work on both categories of applications as described in the earlier section.

- Identify the processing needs of various thin film sensor arrays (TFSAs)on silicon

- Identify, design and implement an architecture for the processing layer, that can work with many of the thin film sensor arrays

The contributions of this thesis can be enumerated as follows:

1. Established that a common processor can be used for various Thin Film Sensor Arrays (TFSA)

2. Proposed and implemented a generic processor architecture for processing information from many different TFSAs: for both, arrays with same sensor elements, or arrays with different sensor elements

The remainder of the report is organized as follows

1. In chapter 2, sensors are defined and various sensors for different physical and chemical properties are identified. TFSAs which can be integrated with silicon processes, alongwith their processing needs are identified.

2. In chapter 3, multiprocesser taxonomy is discussed and an architecture for the processing layer is proposed.

3. Chapter 4 details the implementation of the processing layer - the design methodology, the design and the results.

4. Chapter 5 discusses the applications put to test on the architecture. A summary of the findings is included in this chapter.

Chapter 2

Thin Film Sensor Arrays on Silicon

A variety of sensors can be integrated with the CMOS processes. Photosensors, UV detectors, near infra red and infra red detectors, magnetic detectors, pressure sensors (piezoelectric), and chemical sensors can be made on silicon. Some have a direct effect on silicon, others can be used with materials which can be easily integrated with silicon processes have an effect.[1]

With advancement in technology, most of the sensors (as sensor arrays) can be put on a single chip and processing can be done next to the sensors. A variety of thin film sensors are now available.

2.1 Signals and Sensors

A wide variety of systems like computers, oscilloscopes, door locks, clinical thermometers, satellites, word processors, etc - can be termed together as "'informaion processing systems"'. Any system which processes information is termed as an information processing system.

A sensor is a device that receives a signal or stimulus (as heat or pressure or light or motion etc.) and responds to it in a distinctive manner. Or a sensor can be defined as an electronic device used to measure a physical quantity (such as temperature, pressure or loudness) and convert it into an electronic signal of some kind. Since a signal is a form of energy, sensors can be classified according to the type of energy they detect.

The various forms of energy are: Electromagnetic radiant energy, Gravitational energy, Mechanical energy, Thermal energy, Electrostatic and Electromagnetic energy, Molecular energy, Atomic energy, Nuclear energy and Mass energy.

In brief, the electromagnetic radiant energy is related to electromagnetic radiowaves, microwaves, infrared, unltraviolet, visible light, x-rays and gamma rays. The gravitational energy concerns with the gravitational attraction between a mass and earth or other bodies. Mechanical energy pertains to displacements, flows of materials where as the thermal energy concerns the kinetic energy of atoms and molecules. The electrostatic and eletromagnetic energies relate to the electric and magnetic fields, the currents and the voltages. Molecular energy is the energy related to the bonds which hold the atoms together in a molecule where as the atomic energy pertains to the energy between the nucleus and the eletrons in an atom. Nuclear energy is the energy which keeps the nuclei together and the mass energy is as described by Einstein.

For either sensing or processing information, practically the energy groups can be classified into six different 'main' energy domains. The most important physical properties of the six signal domains are mentioned in the Table 2.1. Gravitational

and mechanical energy have been clubbed under mechanical signal domain, eletrmagnetic waves are considered under radiant signal domain. Molecular and atomic energy are brought together under the chemical signal domain. Nuclear and mass energy have been left out of consideration.

Signal domain	Physical Properties
Radiant signals	Light intensity, wavelength, polarization, phase, reflectance, transmittance
Mechanical signals	Force, pressure, torque, vacuum, flow, volume, thickness, mass, level, position, displacement, velocity, acceleration, tilt, roughness, acoustic wavelength and amplitude
Thermal signals	Temperature, heat, specific heat, entropy, heat flow
Electrical signals	Voltage, current, charge, resistance, inductance, capacitence, dielectric constant, electric polarization, frequency, pulse duration
Magnetic signals	Field intensity, flux density, moment, magnetization, permeability
Chemical signals	Composition, concentration, reaction rate, toxicity, oxidation-reduction potential, pH

Table 2.1: The six signal domains

Specifically in this report, devices which can be integrated into complex silicon processes and if possible identify any thin film sensors built using the aforementioned physical properties in Table 2.1 are studied. In order to improve the characteristics of a sensor, a signal processing circuit maybe combined with a sensor and such devices are refered to as smart sensors. Compatible technologies with silicon processes like polymer films for gas sensors or ZnO films for pressure sensors are also considered.

The physical and chemical effects in silicon for the non-electrical signal domains are presented in Table 2.2. The following sections, the study of the various signal domains and sensors in thin films if available will be presented.

Signal Domain	Self-generting effect	Modulating effect
Radiant	Photovaltaic effect	Photoconductivity, Photoelectric effect
Mechanical	Acoustoelectric effect	Piezoresistivity, Lateral photovaltaic effect
Thermal	Seebeck effect, Nernst effect	Thermoresistive effect
Magnetic		Hall effect, Magnetoresisance, Suhl effect
Chemical	Galvano effect	Electrolytic conduction

Table 2.2: Physical and Chemical effects in Silicon

2.2 Mechanical Signal Domain

Mechanical signals can be directly converted into electrical energy using mecahnical sensors. Sensor for mechanical signals are used to measure (a) motion relatd measurands - position, displacement, velocity, flow, speed of rotation, etc and (b) force related measurands - weight, pressure, acceleration, torque, strain, vibration, etc.

2.2.1 Mechanical sensors as arrays

Capacitive tactile sensor arrays have been reported in literature, which are used for robotic applications.[13] Structural information about an object, such as orientation, size and required force pattern can be obtained by such sensor arrays. Tactile sensor arrays have also been reported for live finger print scanning tools.[10]

2.3 Thermal Signal Domain

Most applications of conventional electronic circuits, the temperature sensitivity of silicon devices is undersiable. Hence a lot of research has gone into understanding the temperature sensitivity of silicon devices. This knowledge can be used to design silicon-based temperature sensors.

Silicon and silicon devices can be used to measure a host of thermal effects like

the Seebeck effect, the Thompson effect, etc. Also flow sensors and vacuum sensors can be built using silicon temperature sensors. In this section, the discussion will be focused on thermal infra-red sensors.

In thermal infra-red detectors, the radiation is absorbed by the material, which generates phonons and causes the lattice to heat up; this change in the lattice temperature is converted into a change in the electronic properties of the substrate. Although thermal infrared sensors are slower than their photon detectors, since thermal infrared sensors allow a wider spectral range, they have been shown to be very suitable for passive intrusion alarms. Also arrays of thermal infrared sensors can be used for the construction of infrared spectrometers.

Silicon thermocouples can be structured to make efficient thermal infrared detectors. Either a suitable material can be deposited on silicon substrate for the thermocouple or silicon could be used as one material in the thermocouple. An example for the former type is discussed [11]. A thin membrane of SiN is PECVD (Plasma Enhanced Chemical Vapor Deposition) on silicon substrate. A platinum thin film is used as the thermodetector. Detailed analysis of such a device is available [11].

2.4 Magnetic Signal Domain

Many materials show a change in their resistivity on the application of a magnetic field. This is called the magnetoresistance effect. Silicon smart sensors built in the magnetic signal domain exploit this effect. Arrays of very small magnetic sensors

on a single chip can be used to detect very small magnetic fields with very high spatial resolution. These sensors can be deposited on active silicon substrates thereby facilitating on-chip signal processing and multiplexing. GMR sensor arrays with frequency independent sensitivity offers improvements in speed, depth, and resolution in eddy-current testing. [12]

2.5 Chemical Signal Domain

Chemical sensors detect the presence of specific chemicals or classes of chemicals depending on how a specific chemical reacts. Silicon technology for chemical sensors has been widely researched because it permits mass production and makes low-cost devices possible. A definite subset of the chemical sensors are the bio-chemical sensors (which detect biological macromolecules).

A disadvantage of chemical sensors and also biochemical sensors is that they are not only sensitive to one chemical measurand but usually respond to many, thus leading to the creation of sensor arrays to identify specific measurands. Though a disadvantage, this provides an interesting challenge to our project.

A variety of cross-reactive chemical and biochemical sensors are available. The measurands include gases to biological macromolecules [19].

Miniature polymer-based chemical gas sensor array on silicon using micromachining technology, (sensors are polymer-carbon black composite films) swell reversibly and cause a resistance change upon exposure to a wide variety of gases. Since the composite film sensors are not specific to any one gas, an array of these

sensors, each with a different sensing film, is used to identify gases and gas mixtures through the pattern response of the array. The polymer composite film is not specific to any one particular gas, as compared to conventional chemical sensors which work for only one analyte. Used in an array, with each sensor containing a different polymer composite film, gases and gas mixtures can be identified by the pattern response of the array. This allows a much more general-purpose chemical gas sensor capable of broadly detecting and identifying various constituents. Each of the polymer sensor films responds differently to each of the gas vapors. By integrating the responses from the sensor array, a unique pattern or signature is produced for each chemical gas vapor. An example of such a sensor array is described in [17].

Pattern recognition techniques such as data clustering, least-squares fit, etc can be used to identify individual gases [18]. Applicationss of such sensor arrays include environmental monitoring (such as detecting the presence and concentration of toxic or otherwise dangerous gases that may come from spills and leaks), quality control and industrial monitoring, particularly in such industries as food processing, perfume, beverage and other chemical products.

The response of the polymer-carbon black sensors is shown to increase proportionally as the concentration of the gas vapor increases [17], thus it is not only possible to identify the gases but also the concentration of the gases. Also the charateristic resistance change of the film was not affected by the reduction in sensor area, thus opening up the door for miniaturization.

2.6 Radiant Signal Domain

Photosensors are used in digital cameras, head mounted kits, etc for taking pictures and analyzing the pictures. Once the picture is taken, various techniques are used for either object recognition or motion detection.

Photodiodes or photogate sensor arrays operating in the visible and near infra red region can be used for image detection such as pattern recognition, position sensing and for image tracking and in many other applications.

The basic physics of either CMOS sensors or the thin film sensors is the same. Incident photons on a silicon surface generates an electron-hole pair. It is the photo-electric effect in action. The energy of a photon is directrly related to its frequency and inversely proportional to its wavelength.

2.6.1 CMOS Cameras

The CMOS imager takes advantage of the mature CMOS technology, which can be very cheap with volume production and the accumulated experience on processing, reliability, yield etc. Each pixel can be addressed randomly through the column and row decoder or multiplexer. This readout method enables the electronic windowing, pan, zoom and sleeping mode operation for multimedia application.

Two types of CMOS sensors are passive pixel sensors and active pixel sensors.

1. Passive Pixel Sensors (PPS) were the first image-sensor devices used in the 1960s. In passive-pixel CMOS sensors, a photosite converts photons into an electrical charge. This charge is then carried off the sensor and amplified.

These sensors are small-just large enough for the photosites and their connections. The problem with these sensors is noise that appears as a background pattern in the image. To cancel out this noise, sensors often use additional processing steps.

2. Active-pixel sensors (APS) reduce the noise associated with passive-pixel sensors. Circuitry at each pixel determines what its noise level is and cancels it out. It is this active circuitry that gives the active-pixel device its name. The performance of this technology is comparable to many charge-coupled devices (CCDs) and also allows for a larger image array and higher resolution.

2.6.2 Thin film phototransistors

Image sensors in TFA technology employ thin-film detectors based on multilayer structures of hydrogenated amorphous silicon (a-Si:H) and its alloys. The thin-film system of a TFA sensor is deposited onto the completed ASIC wafer in a Plasma Enhanced Chemical Vapor Deposition (PECVD) cluster system. The PECVD process is based on the decomposition of a gaseous compound near the substrate surface. Amorphous silicon layers are fabricated using the process gas silane (SiH4) at substrate temperatures between 150 C and 200 C, which inherently leads to the formation of a silicon-hydrogen alloy. The hydrogen atoms in a-Si:H prevent the formation of dangling bonds, therefore the mid-gap defect density is decreased. The a-Si:H material properties are considerably better than those of pure amorphous silicon (a-Si), which is indeed useless for electronics because of its extremely low carrier

mobility.

Due to its higher absorption coefficient in the relevant spectral range and its maximum spectral response for green light, amorphous silicon is more qualified for visible light detection than crystalline silicon. Moreover, the a-Si:H deposition sequence is adaptable to the specific requirements of an application. With a suitable layer sequence it is possible to distinguish three or more colors within the same pixel [16][8].

2.7 Processing needs of Thin Film Sensor Arrays (TFSAs)

1. Photodetectors, thermal sensor arrays, tactile (pressure) sensor arrays and magnetic sensor arrays output an image. Image processing algorithms like edge detection, thresholding, etc are done on the data to get the information (Table 2.3).

Sensor array	Algorithms for processing data	Applications
Photodetector	edge detection, motion detection, etc	cameras
Tactile	edge detection, thresholding	live finger print scans
Thermal	edge detection, thresholding	staring arrays
Magnetic	edge detection	eddy current testing

Table 2.3: TFSAs, Algorithms and Applications

2. Chemical and gas sensor arrays output data, which needs to be individually compared to each other.

Thus, the processing layer should be able to exploit 3-D integration, specifically the availability of the data from all the sensor elements at once and exploit the parallelism in data processing. The processing layer should be able to implement

image processing algorithms as well as work be able to work on each sensor output independantly as required.

Chapter 3

Survey of various architectures

A study of the various processing architectures that can be used for the different silicon sensors in use is presented in this chapter. Computer architecture broadly covers three aspects of computer design - the instruction set architechture, organization and the hardware [2]. Combining the silicon sensors identified in Chapter 2 with silicon processors would give a system which can be entirely manufactured in the present fabs. In this chapter, multiprocessor taxonomy is explored and an architecture for the processing layer is presented.

3.1 Flynn's taxonomy of multi-processors

To start with a taxonomy for multi-processors is presented here. Flynn's taxonomy, the most popular in multi-processors is presented here. The taxonomy as presented here can also be found in detail in [2].

- SISD: Single instruction stream, single data stream *This is a uniprocessor.*Conventional single processor computers are classified as SISD systems. Each arithmetic in-

struction initiates an operation on a data item taken from a single stream of data elements.

- SIMD: Single instruction stream, multiple data stream *A single instruction is executed on multiple data sets at the same clock cycle.* SIMD machines have one instruction processing unit, sometimes called a controller and indicated by a K in the PMS notation, and several data processing units, generally called D-units or processing elements (PEs)

- MISD: Multiple instruction stream, single data stream *No such machine has been built yet.*

- MIMD: Multiple instruction stream, multiple data stream *Multiple instructions are executed on different streams of data (on different processors), making this a generic multiprocessor.* The category of MIMD machines is the most diverse of the four classifications in Flynn's taxonomy. It includes machines with processors and memory units specifically designed to be components of a parallel architecture, large scale parallel machines built from "off the shelf" microprocessors, small scale multiprocessors made by connecting four vector processors together, and a wide variety of other designs. With the continued improvement in network communication and the development of software packages that allow programs running on one machine to communicate with programs on other machines, users are even starting to use local networks of workstations as MIMD systems.

3.2 Memory organization of Multi-processors

Memory organization plays a big role in multi-processing as it defines how memory is addressed and accessed by each processor. No matter how fast one makes the processing unit, if the memory cannot keep up and provide instructions and data at a sufficient rate there will be no improvement in performance. The main problem that needs to be overcome in matching memory response to processor speed is the memory cycle time, the time between two successive memory operations.

Processor cycle times are typically much shorter than memory cycle times. When a processor initiates a memory transfer at time t_0, the memory will be 'busy' until $t_0 + t_m$, where t_m is the memory cycle time. During this period no other device - I/O controller, other processors, or even the processor that makes the request - can use the memory since it will be busy responding to the request.

Solutions to the memory access problem have led to broadly two kinds of parallel systems. In one type of system, known as a shared memory system, there is one large virtual memory, and all processors have equal access to data and instructions in this memory. The other type of system is a distributed memory, in which each processor has a local memory that is not accessible from any other processor. Cache coherence becomes an essential part for the memory organization in a multi-processor.

3.2.1 Shared memory systems

Shared memory refers to a (typically) large block of RAM that can be accessed by several different central processing units (CPUs) in a multiple-processor computer system.

The problem with shared memory systems is that the many CPUs need fast access to memory and will likely cache memory. Whenever one cache is updated with information that may be used by other processors, it needs to immediately update the shared memory, otherwise the different processors will be working with different data. Thus a cache coherence protocol has to be used.

3.2.2 Distributed memory systems

Distributed memory means that in a multi-processor system each processor has its own memory. This requires that computional tasks have to be distributed on the different processors for processing. After the processing the data has to be reassembled.

3.2.3 Distributed Shared memory systems

Distributed Shared Memory (DSM) refers to hardware implementations, in which each node of a cluster has access to a large shared memory in addition to each node's limited non-shared private memory. As in the case with shared memory systems, a distributed shared memory system also needs to have a cache coherence protocol implemented.

3.3 The Instruction Set Architecture

An Instruction Set Architecture (ISA) is a specification of the set of all binary codes (opcodes) that are the native form of commands implemented by a particular CPU design. The set of opcodes for a particular ISA is also known as the machine language for the ISA. It describes the aspects of a computer architecture visible to a programmer, including the native datatypes, instructions, registers, addressing modes, memory architecture, interrupt and exception handling, and external I/O.

Some of the categories of ISA of interest are discussed in the following subsections.

3.3.1 The Reduced Instruction Set Computing (RISC) architecture

The RISC architecture favors a smaller and simpler set of instructions that all take about the same amount of time to execute. The RISC philosophy is to make smaller instructions, implying fewer of them, and thus the name "reduced instruction set". Code is implemented as a series of these simple instructions, instead of a single complex instruction that has the same result.

Some examples of the RISC architecture processors are the DEC Alpha, Freescale and IBM's PowerPC, ARM processors, MIPS processors, Sun SPARC and ULTRA-Sparc's.

3.3.2 The Complex Instruction Set Computing (CISC) architecture

In the CISC architecture each instruction can indicate several low-level operations, such as a load from memory, an arithmetic operation, and a memory store, all in a single instruction. The instruction set is designed to support high-level languages by providing "high-level" instructions such as procedure call and return, loop instructions such as "decrement and branch if non-zero" and complex addressing modes to allow data structure and array accesses to be combined into single instructions. Generally, the more complex the instruction set, the greater the overhead of decoding any given instruction, both in execution time and silicon area.

Examples of the CISC architecture machines include Intel's Pentium processor, AMD's x86 based architecture. x86 is a CISC architecture instruction set.

The continuous evolution of both CISC and RISC designs and implementations have rendered the definitions less meaningful. Nevertheless the above are a broad categorization of the instruction set architectures. To develop an instruction set for the desired application (as discussed in earlier chapters), it is also important to understand a couple of instruction set architectures, which may or may not be completely categorized as RISC or CISC.

3.3.3 Digital Signal Processor (DSP)

A DSP is a specialized processor to process digital singals mostly in real time. DSPs usually have an instruction set optimized for rapid signal processing. Some characteristics of the instruction set of DSPs include, multiply-accumlate operations,

modulo addressing and bit-reversed addressing. Floating point DSPs are used for scientific computing.

3.4 Designing an Architecture for processing data from a sensor array

For most of the TFSAs identified in Chapter 2, the basic processing of the data involves image processing algorithms. For chemical and gas sensors, the processing needs to take place on one sensor data at a time. Exploiting the advantages of 3-D integration, by which the data is available for all the sensors in the sensor array at once, a parallel processing architecture can be developed for image processing algorithms. This has to be suitably modified for chemical and gas sensor arrays.

3.4.1 Identifying a processor architecture and a memory system

The parallel data from the sensors is intended to be processed independantly for each sensor. There might be at most nearest neighbor communication. A SIMD machine is the preferred solution. The control unit is responsible for fetching and interpreting instructions. When it encounters an arithmetic or other data processing instruction, it broadcasts the instruction to all PEs, which then perform the same operation.

One of the advantages of this style of parallel machine organization is a savings in the amount of logic. Anywhere from 20% to 50% of the logic on a typical processor chip is devoted to control, namely to fetching, decoding, and scheduling instructions. The remainder is used for on-chip storage (registers and cache) and

the logic required to implement the data processing (adders, multipliers, etc.). In an SIMD machine, only one control unit fetches and processes instructions, so more logic can be dedicated to arithmetic circuits and registers.

In literature, a variety of SIMD processors have been designed and discussed [9]. A SIMD architecture is a preferred solution for various image processing tasks. Also if each PE has a local memory, the data can be sought from the sensor array, for all the elements at once. The memory can also be extended to store the data from the nearest neighbors. Thus, a distributed memory system is the choice.

3.4.2 A SIMD Architecture

SIMD stands for Single Instruction, Multiple Data streams. A SIMD processor has one control unit, one program counter and multiple functional units or processing elements (PE). The PEs execute the program, which is decoded by the control unit on their respective data items. The data is understood to be in blocks, and the same instruction to be executed on each and every data item. Executing the program on all the data items parallely in different PEs, rather than processing them serially in one PE speeds up the process considerably.

The SIMD architecture contains a single contol unit(CU) with multiple processor elements(PE) acting as arithmetic units. The arithmetic units (PEs) are slaves to the control unit. They cannot fetch or interpret any instructions. They are merely a unit which has capabilities of addition, subtraction, multiplication, and division. Each PE has access only to its own memory. If a PE needs the information contained in a different PE, it must put in a request to the CU and the CU

must manage the transferring of information. The applications of SIMD include image processing, 3D rendering, video and sound applications, speech recognition, networking, and DSP functions.

3.4.3 Modifications in SIMD architecture for processing needs of dissimilar sensors in the sensor array

Example of dissimilar sensors in the TFSA are either chemical or gas sensors. Data for all the sensors in a chemical or gas sensor array can be acquired at once. But processing needs to take place, one at a time. A MIMD architecture would increase the size of the processing layer by a lot. By uniquely and randomly being able to identify each PE in a SIMD architecture, this bottleneck can be overcome. Also, data needs to be shared across PEs and hence memory needs to be addressed uniquely too.

Thus, a modified SIMD (modSIMD) architecture which in a common addressing mode, behaves like a SIMD architecture and in a specific addressnig mode, behaves like a SISD processor suits all the TFSAs discussed in the thesis. The proposed architecture is shown in Figure 3.1. PE_ID is the processing element ID.

3.5 The modSIMD architecture

- A Programmable processor: 32 bit Instruction,16 bit data, non-pipelined processor
- Processor architecture: One control unit, one program counter, multiple PEs

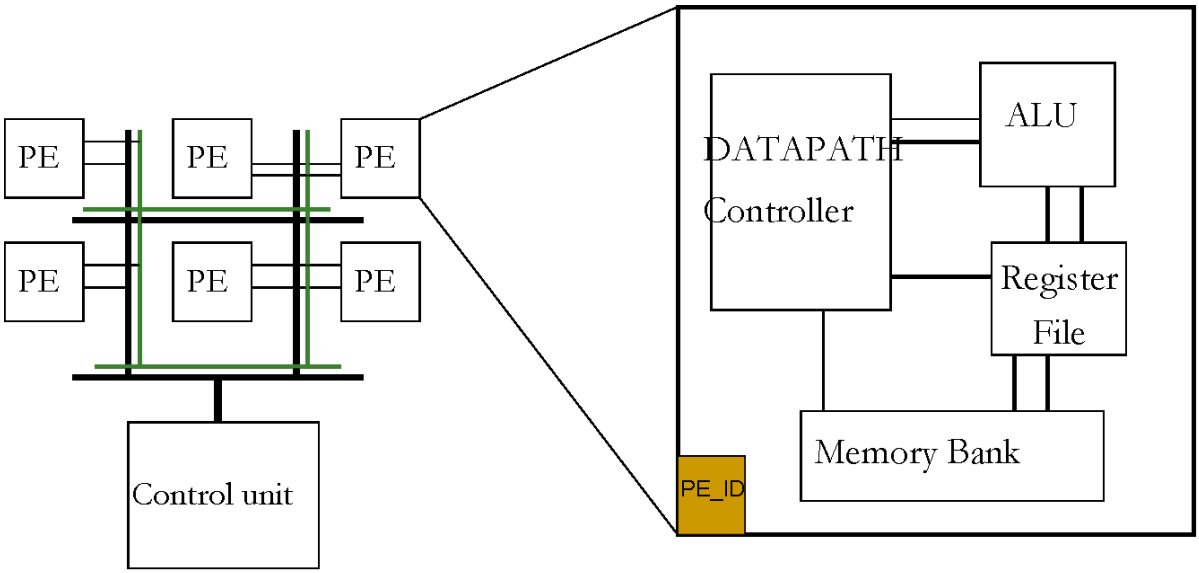


Figure 3.1: Proposed Architecture

In common addressing mode, it is a Single Instruction Multiple Data parallel processor, for processing data from all sensors at once. In specific addressing mode, it is a Single Instruction Single Data processor, each processing element (PE) identified by a PE_ID. For processing data from each sensor separately

- Memory: Local to each PE, 64 addressable memory elements in each PE
- Bus: Data bus is 16 bit wide to communicate with other processing elements and read out

3.6 Why modSIMD?

A generic commercial off the shelf processors might not be able to address all the needs of thin film sensor arrays. These generic microprocessors are neither designed to exploit 3-D integration nor are they designed to handle the needs of sensor arrays.

3-D integration provides the oppurtunity to drastically reduce the data bot-

tlenecks in a microprocessor system, by making the data available through vertical interconnects. Also, as noted in the earlier chapter, many TFSAs with similar sensors in the array, do the same processing on all the sensory data. This can be exploited by moving to a SIMD architecture.

The proposed modSIMD architecture exploits the parallelism in data. By being able to address each of the PEs independantly, the modSIMD architecture can also work with different sensors in the TFSAs. The sensor layer can be developed and fabricated independantly and fit onto the modSIMD architecture, making this a very versatile architecture to build a variety of 3-D integrated TFSAs.

3.7 TFSAs on a modSIMD processing layer

The envisioned TFSA on a modSIMD processing layer architecture is depicted in Figure 3.2.

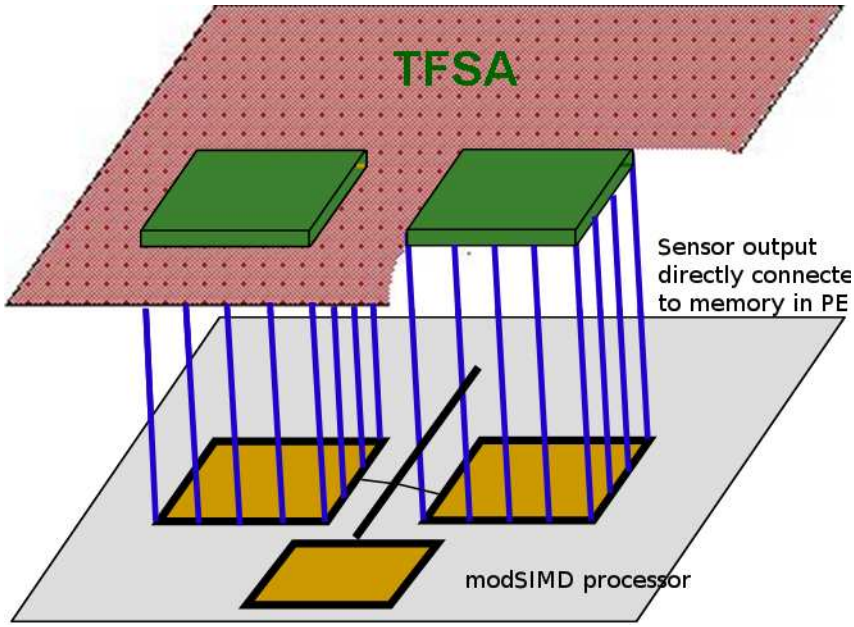


Figure 3.2: TFSA on modSIMD architecture

The sensor outputs are directly connected to the memory in the PEs of the modSIMD processor. Also the nearest neighbors' (sensors) output is also stored in the memory of each PE. Similar addressing schemes for the PE and TFSA can be exploited to achieve this. Each PE in the modSIMD processor is hardcoded with an individual address for individual sensor processing if necessary (specific addressing mode). In the common addressing mode, all PEs also respond to a particular common address for SIMD type operations on the data recieved from the sensors.

Chapter 4

Design and Implementation

A variety of thin film sensors on silicon can be used to build an equal variety of smart silicon sensors as described in chapter 2. Chapter 3 described the proposed modSIMD architecture. This chapter will detail the design methodology used and the implementation of the modSIMD processor.

4.1 Design Methodology

A digital design is a top down approach. First, a behavioral model of the intended chip is created. Then a behavioral model of each of the modules is created. A hardware description language is used to describe each of the modules. In electronics, a hardware description language or HDL is any language from a class of computer languages for formal description of electronic circuits. It can describe the circuit's operation, its design, and tests to verify its operation by means of simulation. In this case, verilog is the choice. All HDLs have object oriented programming support. The design is divided into modules, whose instances behave like the chip.

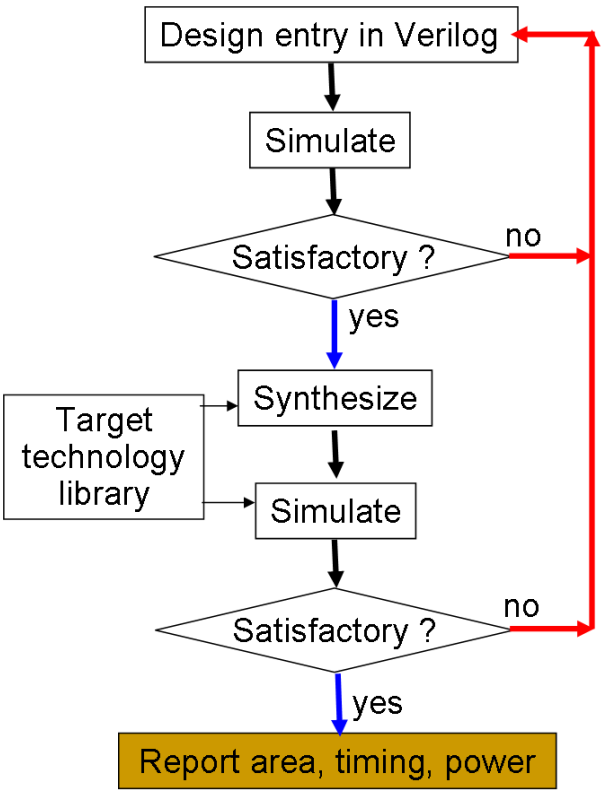


Figure 4.1: Digital Design Flow

Each module, is again sub-divided in a similar manner and all functions are coded in HDL. But for the HDL to be converted into circuits, the code has to be written into a register transfer level - RTL. The design flow is simplistically shown in Figure 4.1

4.1.1 Validation

The design flow of a chip is described in Fig 4.1. VHDL/Verilog code is created for the idea about a chip. Since both languages are modular, a chip can be sub-divided into various modules (could be SRAM, buses, logic units, etc) and designed separately. Tests in verilog or VHDL are written to simulate the behavior of the various units and as a full chip itself. This is followed by synthesis, where the

complier generates a gate level design for the chip. After synthesis the chip needs to be validated according to the logic and timing constraints. Again this is done at unit level as well as at the full chip level. If any of the tests fail here, because of any bugs in the code, that particular unit is tested and code is re-written. If there is a bug at the full chip level, the bug is isolated to a unit the unit is then further tested, updated and put back into the full chip design. After validation, the chip design proceeds to placement and routing. The chip is validated again. Finally a gds map of the chip is generated and mask generation data is collected and the chip is sent to the fab.

Validation is a necessary part in designing and fabricating a chip. Once the RTL code for designing a chip is written, the code needs to be tested for various errors, etc. Testbenches can be written in Verilog and VHDL, but one cannot test different kinds of scenarios. To generate all kinds of scenarios in data to test the chip extensively, other tools need to be used. Either the data can be generated using script languages like perl or python or languages like VERA or e can be used. The advantage of using these languages is to exploit the features since they are specifically built for validation. They include random traffic generation on the various buses in the design or random data in the various kinds of memory units in the design, and so on and so forth.

To test a chip conclusively before it is released to the market is an impossible task. But failing to do a basic minimum amount of testing before sending it for fabrication might be costly. Non functional part can be spotted, errant buses or the dont cares skipped during the design could be tested for any interference with

the design documents. As the example quoted below shows, validation is also an important part of engineering. In the challenger explosion, the O rings were never tested at temperatures the launch had taken. It cost seven lives. Improper validation or lack of led to the recall of the first Intel Pentium processor. Hence validation comes off as a necessary step to check the chip design as thoroughly as possible and to check that it adheres to the design document. Also it ensures that all the parts are functional and perform as expected. Validation is also an important step in the design process. To be able to validate a piece of code, one needs to understand the various design requirements, the standards used, the protocols used in the design (communication protocols) if need be.

4.1.2 Synthesis

Synthesis is the process of converting a digital design written in a hardware description language (HDL) into a low-level implementation consisting of primitive logic gates. Synthesis tools optimize and compile the (RTL) design as per specified constraints and map to target devices as per the design libraries.

4.2 The modSIMD processor

The processing layer underneath as discussed earlier is a modified SIMD architecture. The input is stored in the memory of each "processing elements" (PE)(described later) directly. This is done when the global reset of the processing unit goes high.

The addressing is 4 bits to identify the processing unit and 6 bits to locate the

particular location in the processing element (PE). Each processing element requires data for 30 sensors. Each instruction fed is concurrently processed in ten PEs in the system. Additional PEs can be added - the trade off being silicon real estate.

Detailed description of the various units follows in the rest of the chapter.

4.3 Instruction word

The processor takes a 32 bit instruction. The instruction word is structured as follows. (Figure 4.2)

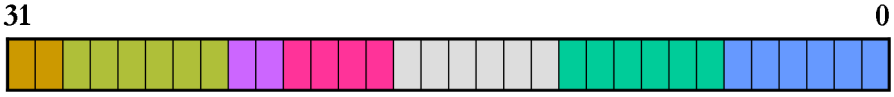


Figure 4.2: Instruction Word

Bits [31:30] and Bits [23:22] are decoded in the control unit. The control unit determines if the instruction is an ALU operation or an immediate operation. Bits [29:24] determine the ALU operation. These bits are directly passed to each processing element which contains the ALU. The bits [21:18] contain the PE ID - processing element ID. The addresses for the operands are contained in remainder of the bits. Bits [17:12] contain address of Operand A, [11:6] contains the address of output register (register C) and [5:0] determines the address of Operand B.

4.4 Processing Element (PE)

The processing unit is the work horse of the CPU system underneath.(Figure 4.3)Each processing element has a ALU, a memory bank, a data-path controller and a couple

of register files to work on the data. Adding as many processing units as required helps work on various data items in parallel. Each PE can be addressed individually. The PEs work in two modes, a common addressing mode and a specific addressing mode. The number of processing elements in the processor will determine the parallelism the system as a whole will enjoy.

Each processing element also has an unique identifier to allow random access to a particular element. This gives the user more control for any specific processing element and use that processing element to run some specialized routines on that particular set of data items.

The PE_ID in the instruction determines which PE operates on the instruction. In the common addressing mode, PE_ID is set at '1111' - and all the PEs perform the operation as shown in Figure 4.4. The instrcution for the same looks like follows

```
Instrn = 32'b00000000001111000000000010000000010
```

The instruction decodes to an addition of elements of Address A (000000) and Address B (000010) and the result is put into Address C (010000). Since the PE_ID is 1111, all the processing elements perform the addition.

Each PE can also be individually addressed in the specific addressing mode as shown in Figure 4.5. In this case, the addressed PE performs the operation, while the rest of the PEs donot react. The instruction word for the example cited in Figure 4.5 looks as follows

```
Instrn = 32'b00000000000001000000000010000000010
```

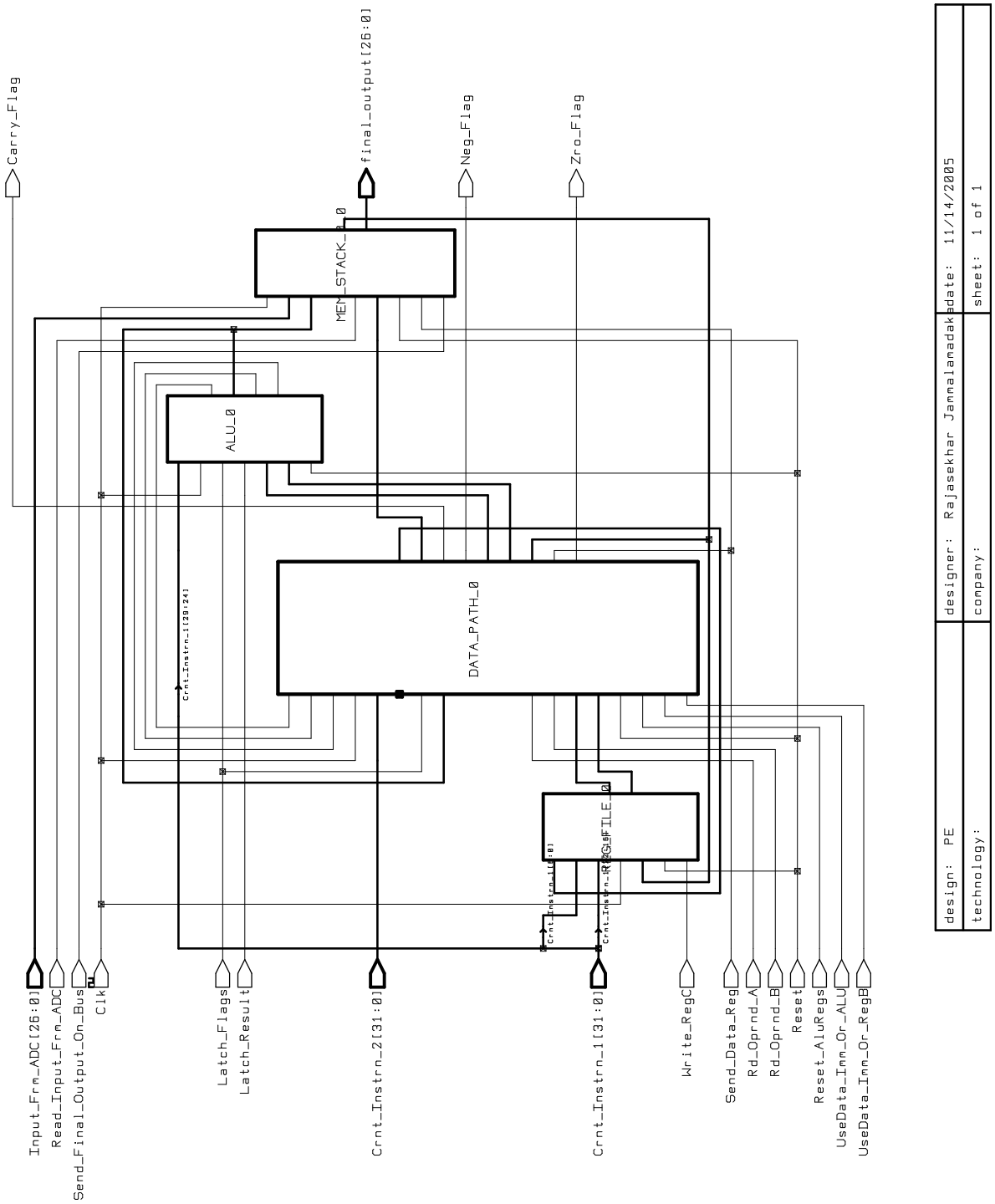


Figure 4.3: Processing Element Schematic

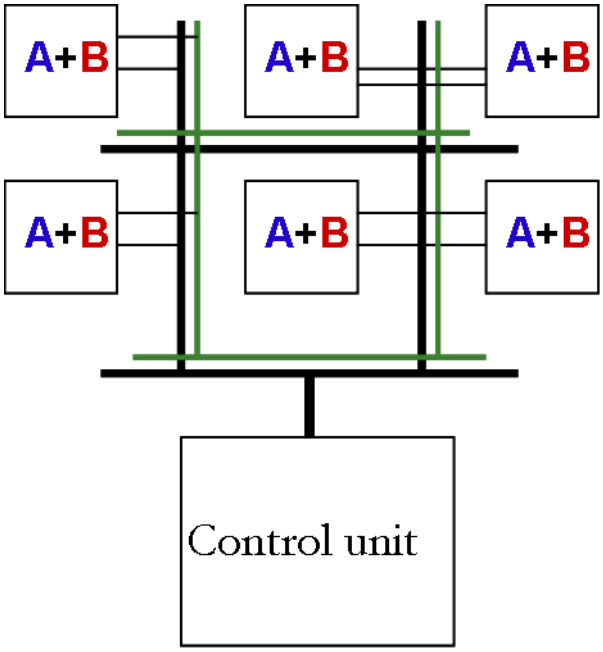


Figure 4.4: Common Addressing Mode

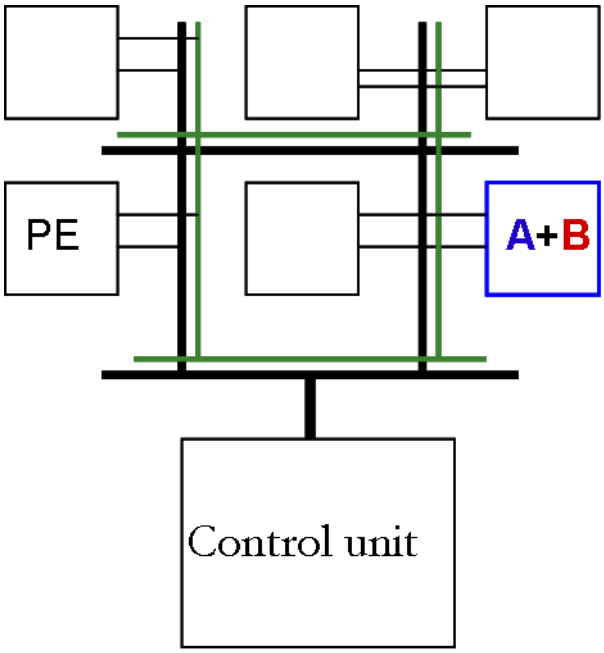


Figure 4.5: Specific Addressing Mode

The PE_ID is identified as '0001' and only that PE performs the addition of the operands as described earlier.

4.4.1 The do_process signal

The PE needs to communicate with its constituents (ALU, datapath, etc) once it decodes whether a particular instruction has to be executed in that PE or not. This is done with the do_process signal as shown in Figure 4.6.

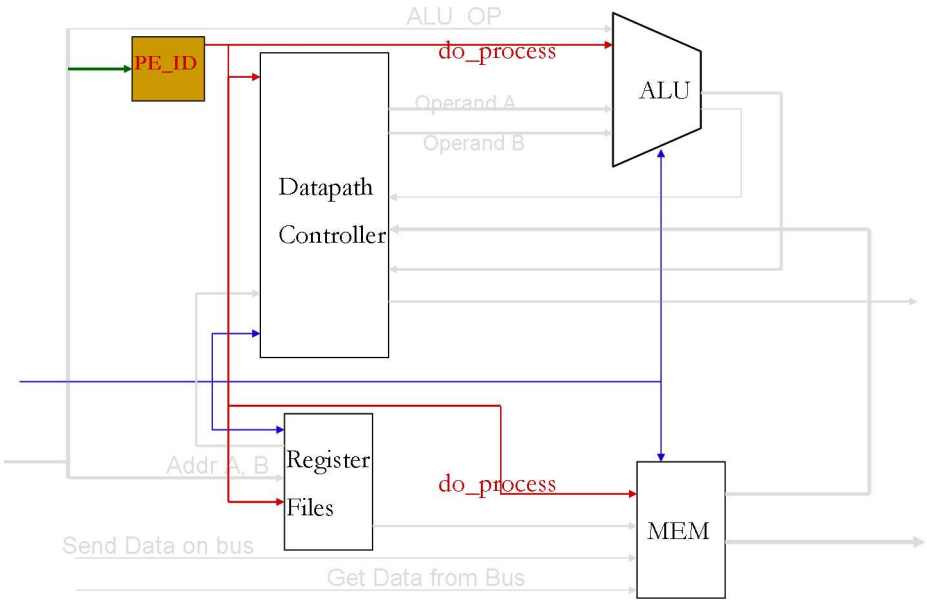


Figure 4.6: The do_process signal in a PE

The do_process signal high, indicates that the operation has to be executed in the PE. This has also been verified by simulation with rtl as well as synthesized code as shown in Figure 4.7.

In Figure 4.7, the instruction initially had the PE_ID = '1111' for which, all the PEs (A_PE, B_PE and H_PE shown), have a do_process signal high. After a few clock cycles, the instruction was changed to PE_ID = '0001', as seen, only do_process signal for A_PE is high and the rest are all low.

The sub-units of the processing elements are discussed in the next few sections.

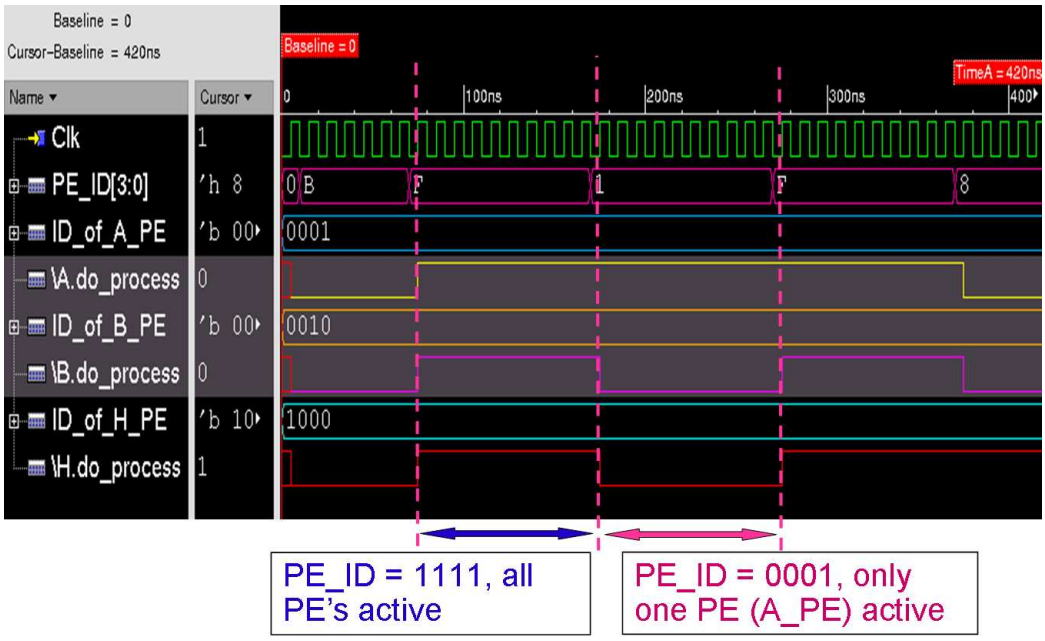


Figure 4.7: The do_process signal simulation

4.5 The ALU

Each processing element has a local ALU which performs the arthimetic and logical operations on the operands. Apart from addition and substraction, the ALU also performs some bit wise operations. The ALU operands are described in Appendix A. The ALU decodes the signal, ALU_OP, recieved from the datapath controller and acts on the operands. The datawidth is 16 bits.

The basic change in the ALU design from a typical ALU is the signal "'do_process"'. As noted earlier in this section, the do_process signal basically identifies the operation to be performed by a particular ALU. In case the do_process signal is low, the ALU latches to the previous results and doesnot act on the new operands. If the do_process signal is high, the ALU performs the operations on the operands and the new result is latched.

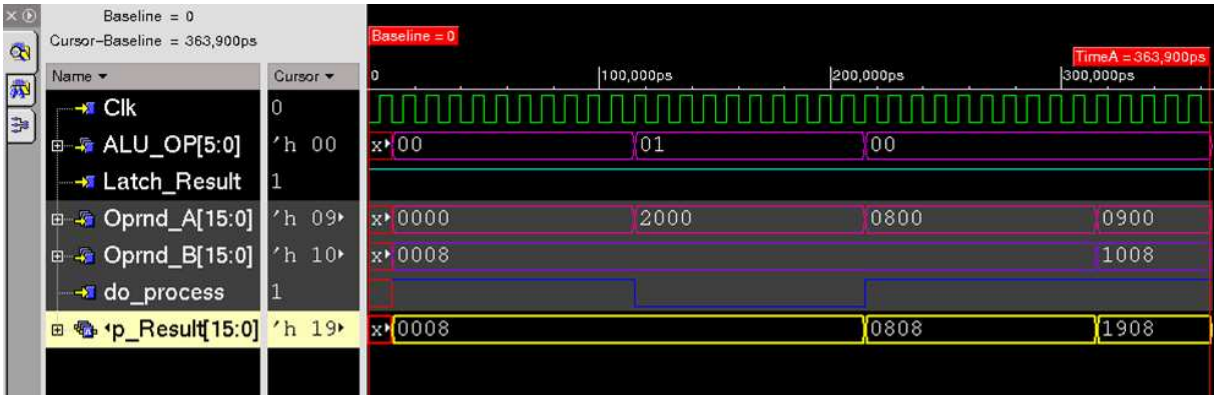


Figure 4.8: The ALU Simulation

Figure 4.8 shows the simulations of the ALU. A simple addition was performed on the operands A and B. In the next step, the "'do_process"' signal was forced low, the output can be seen to be latched to the previous result. Again, the "'do_process"' signal was changed to high, with the operands A and B changing. This leads to a different result as shown in the waveform. The simulation was done on the rtl model as well as the synthesized model. Same results were observed.

4.6 REG_FILE - the register files

Each processing element has a register file unit, which stores the immediate values of the operands and the data for the ALU. The do_process command determines if the register file will act on the rest of the commands or not, similar to as described in the ALU. The inputs for the register file unit are Clk, Reset, Addr_A, Addr_B, Addr_C, Write_RegC, RegPort_C. The outputs are Reg_PortA and Reg_PortB. The address of operand A and operand B are fed to the register file which fetches their values from the memory bank and sends them to the ALU. The ALU also sends in the data to be written into the Addr_C which is used to store the result. The

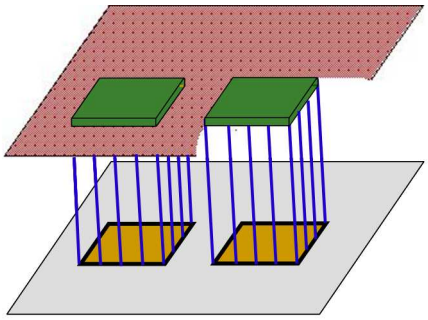


Figure 4.9: Data Acquisition

Write_RegC command from the control unit, triggers this.

4.7 Data Acquisition and Memory

The memory is designed to be local to a processing element. Every memory item can be individually addressed. The address of a particular memory element is simply the PE_ID, local memory address - hence a 10-bit address.

Exploiting the full advantage of 3-D integration, the data from the sensors can be acquired through the vertical interconnects as shown in Figure 4.10. Each memory element can be associated with a particular sensor in the sensor array for data acquisition. Also during data acquisition, the data from the nearest neighbors can be stored locally in the PE memory bank, thus eliminating much usage of the data bus.

The read-out mechanism is serial. Since each memory element can be addresssed invidually, the memory unit puts the particular memory element on the data bus on "'send_data_on_bus"' instruction from the control unit. The PE_ID is

verified and particular memory unit puts the data on the bus.

4.8 Control Logic

There is just one control unit for the whole chip. The control unit decodes the instructions and sets the necessary signals to the various processing units. Rd_Oprnd_A, Rd_Oprnd_B and Write_RegC are the flags set for ALU operations. The UseDataImm flags are set, if the data is to be used immediately. The control unit also waits for the "'process_done"' flags from each of the PEs to be high, to go to the next instruction. The control loop is shown in Figure 4.11.

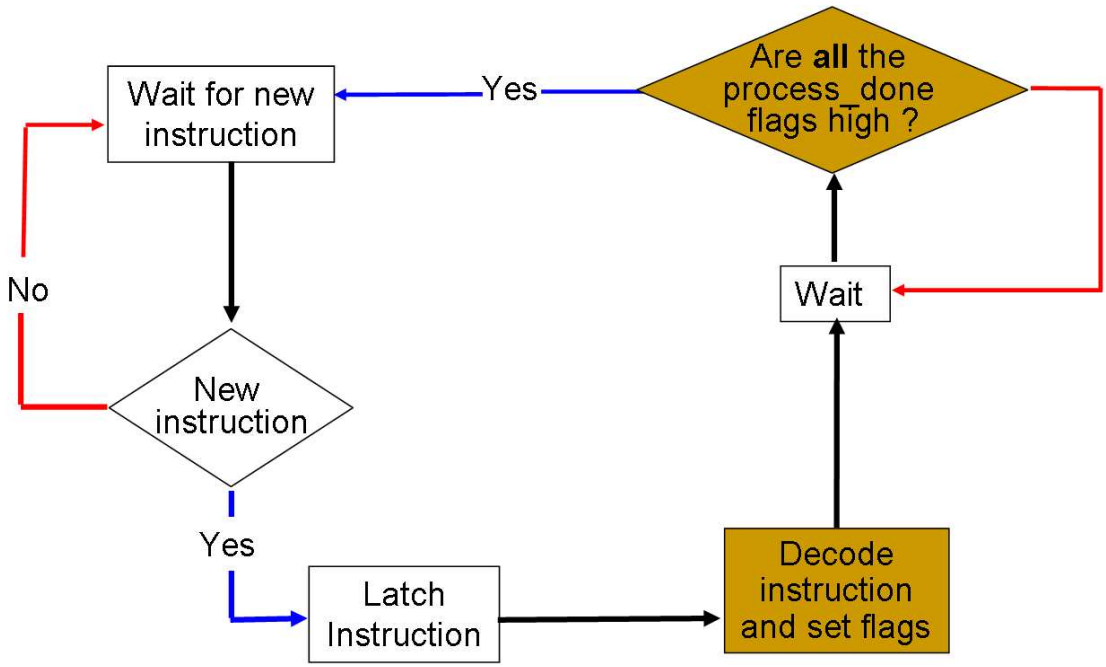


Figure 4.10: The Control Loop

The schematic of the control unit is shown in Figure 4.12

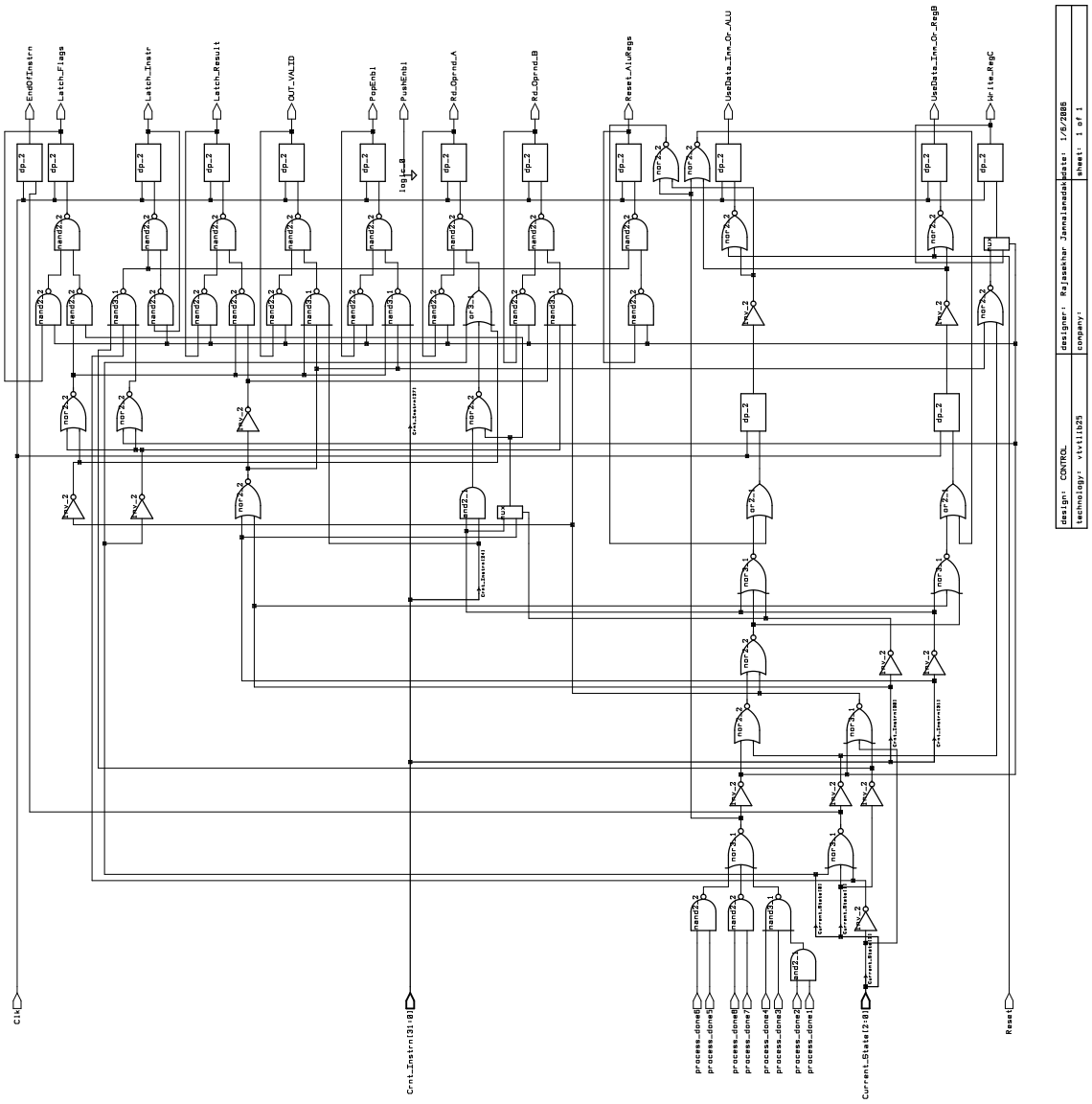


Figure 4.11: The Control Unit Schematic

4.9 Determining the number of PEs in a generic processor

The following designs were synthesized for the following cases to handle a 10X10 sensor network:

1. Just one processing element
2. 10 processing elements
3. 100 processing elements

All the three designs were synthesized using three different design libraries. The three different technologies of the libraries were

- 0.6μm - Available from MOSIS and Michigan State University
- 0.25μm - Available from Virginia Tech
- 0.13μm - Available from University of Texas at Dallas

The number of processing elements is a key to the design of the processor (Figure 4.13). Moreover, all the processing elements are individually addressable. So, only one processor can be asked to perform a certain task.

4.9.1 One processing element

Designing a single processing element processor is similar to designing a regular RISC processor. Care had to be taken to add more memory to read the data from all the sensors at once. Though all the data is available at once, the processor would act on one data item at a time. Thus to work on 100 data items (for a 10X10 array),

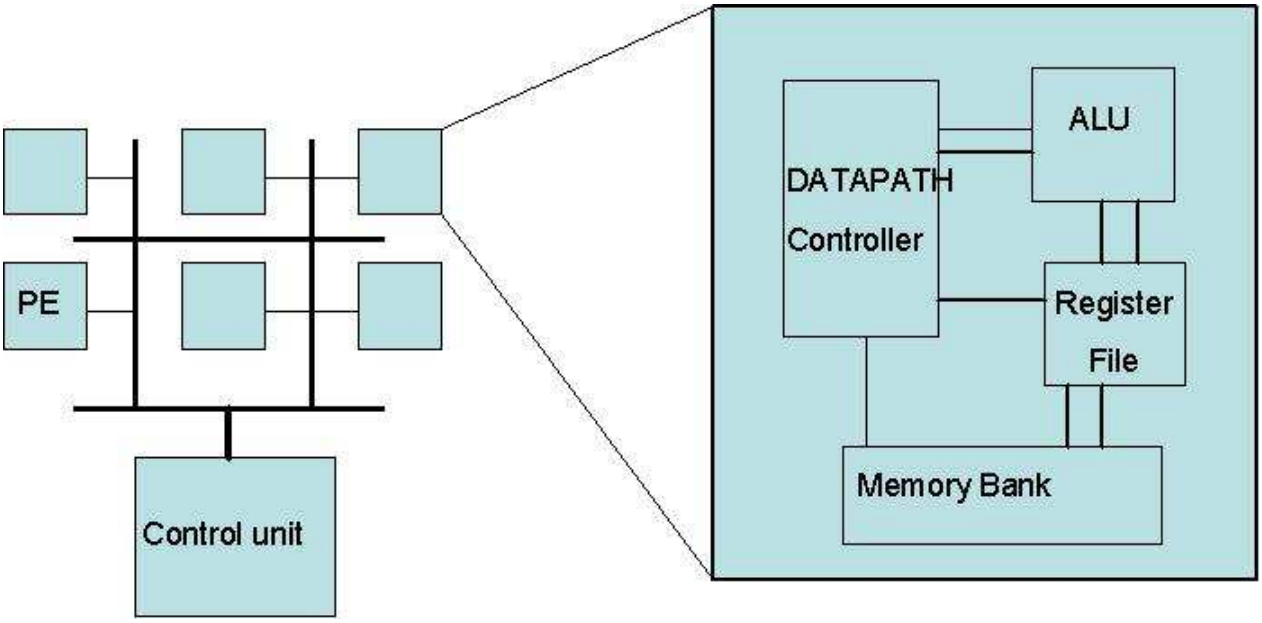


Figure 4.12: modSIMD Processor

it would work the same piece of code 100 times. A schematic of the developed design is shown in Figure 4.14.

4.9.2 100 processing elements

100 processing elements to process a 10X10 sensor array, with every processing element working on the same bit of code on different data items. The memory requirement in this case is for the local sensor data, and the data for nearest neighbors.

4.9.3 10 processing elements

10 processing elements can execute the same code, on ten different data items and looped to do so ten times. This can be done easily as each processing element can either be chosen to work on a row or a column of sensors from the sensor arrays. Thus the memory requirement for such a task would increase to getting all the

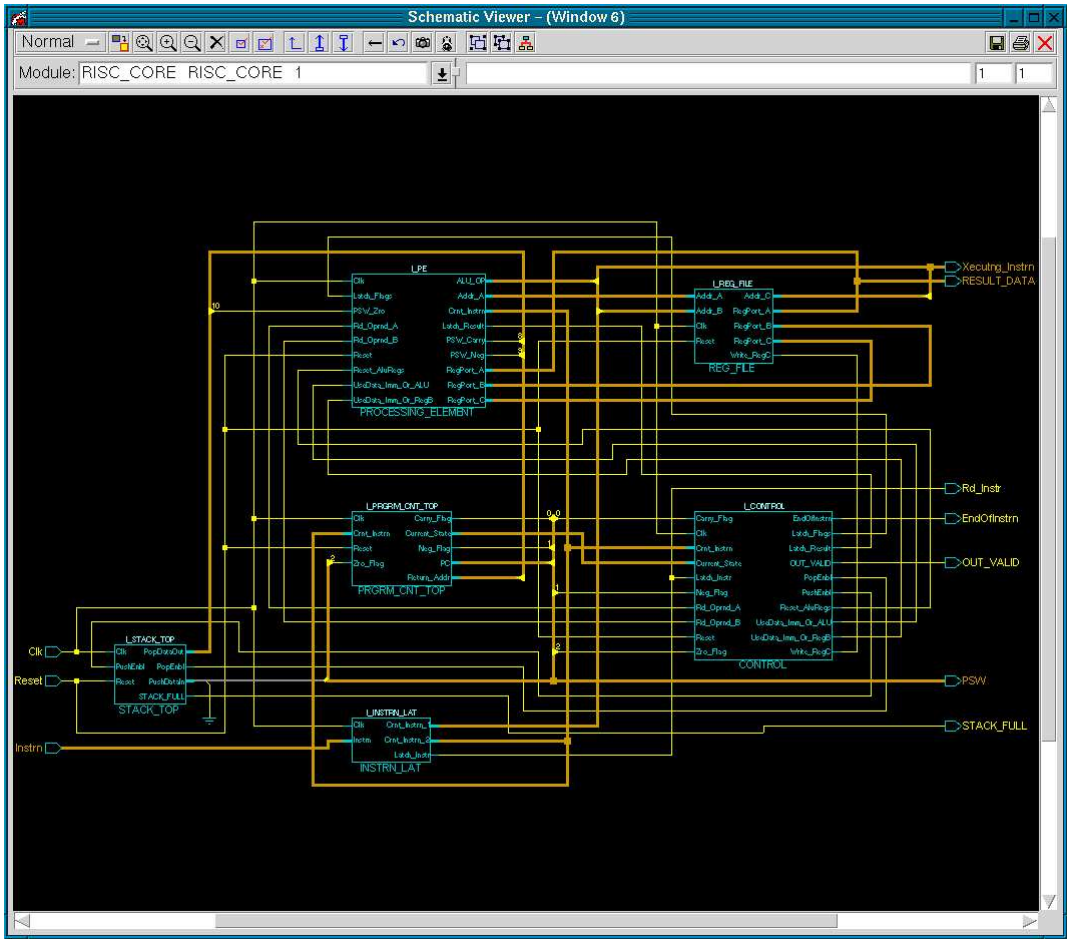


Figure 4.13: Single Processing Element

nearest neighbors, about 30 elements. Such a design is called the block parallel design. [8]

4.10 Results

The area of the above designs are tabulated in Table 4.1 The designs were synthesized in 0.6μm, 0.25μm and 0.13μm technology. The areas are as follows in millimeter square:

All the designed processors had 16 bit data elements.

Number of processors	0.60μm tech	0.25μm	0.13μm
1 processing element	0.37	0.18	
10 processing elements	3.67	1.22	0.89
100 processing elements	38.28	12.54	8.94

Table 4.1: Areas of the synthesized designs

4.10.1 Discussion

The area increases directly with the number of processing elements, across technology. The control block remains the same in all the designs. The memory needed also changed from design to design and hence the area shows a near linear growth.

Chapter 5

Results, applications and conclusions

The implementation of the modSIMD architecture and various modifications of it, reported in this chapter, was done in verilog and synthesized using Synopsys Design Analyzer [6] in 0.25μm process. The simulator used was Synopsys VCS verilog simulator. The standard cell libraries for the 0.25μm process are available from Virginia Tech University [21], [22]. The Design Analyzer tool has features to estimate power, critical path and area based on the data from the standard cell library which is given as an input. The critical path is defined as the longest electrical path in the synthesized design. The maximum delay across such a path determines the minimum time at which the system can be clocked. The voltage level is set as 3.3 V and temperature is set as 25^{O}C. Unless specified, all the implementations used the above specifications.

5.1 Results

The area, power and critical path estimates for the 10 PE implementation, 3-D system are shown in Table 5.1. The sensor array on the sensor layer is assumed to be the same area as the processing layer. Additionally 10 vertical interconnects were assumed for which the area estimates are not available. But in a 3-D implementation, it is assumed that the vertical interconnects occupy the same area on both the processing layer and the sensing layer.

The Design Analyzer tool gave the following results for the 10 PE implementation with the above mentioned design files as inputs:

Area	$1.22mm^2$
Power	190 mW
Critical Path (without ADC)	1.4 ns

Table 5.1: Results for modSIMD with 10PE

5.2 Comparisons

The lack of a generic processing layer architecture for such an application, makes it difficult for a relevant comparison. Application Specific ICs (ASICs) are too specific and modSIMD generic design is bound to have a larger area and power consumption than any of the ASICs. Though the functionality of modSIMD indeed happens to be better. Alternately, any off the shelf generic microprocessor might not be best suited for these tasks for the reasons previously noted. Since it is more likely to pick one of the generic microprocessor architectures while designing a generic processing

layer as a starting point, comparisons with ARM processor have also been made.

Before proceeding to comparing the modSIMD 3-D system architecture with a generic architecture, it has to be determined if a 2-D implementation is better or a 3-D implementation is beter. Thus, the first comparisons are done between a 3-D system implementation and a 2-D system implementation. Both the 2-D system and the 3-D systems used the same modSIMD architecture to maintain similarity in the processing architecture for the comparisons.

5.3 Comparison of 2-D vs 3-D modSIMD for image applications

Image applications as noted earlier in the thesis are the most common applications. In this section, the advantages of 2-D implementation of an architecture to a 3-D implementation of an architecture taking image applications (edge detection) as an example is studied. The modSIMD architecture is used in two configurations - one a 2-D implementation, where the photosensor array is assumed to be on the same die, alongside with the processing architecture. And in the second configuration a 3-D system architecture is assumed.

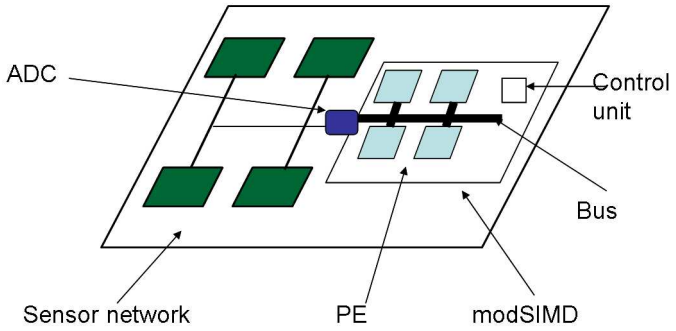


Figure 5.1: modSIMD 2D System Architecture

The 2-D structure is shown in Figure 5.1. The sensor array is connected to 10 processing units in the modSIMD architecture. The data is fed to the processing elements with a bus - 26 bits wide, 10 bits for the address and 16 bits of data. An ADC converts the signal from the sensor and digitizes it and feeds it to the processor. Every processing element decodes the address and the identified processing element picks up the data.

The 3-D structure as shown in Figure 5.2 reads data from all the sensors at once in a block. Each processing element is also assumed to have an ADC.

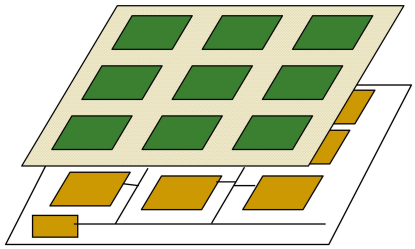


Figure 5.2: modSIMD 3D System Architecture

An edge detection algorithm (Sobel edge detection) is discussed in section 5.6.1. This algorithm was simulated on the 2-D architecture and the 3-D architecture. A flash ADC as discussed in [20] was assumed. It is easy to compare as the technology in the processor design and the ADC is the same 0.25μm technology.

Each edge detection program needs 10 instructions per pixel of processing (Section 5.6.1 and Appendix A). Each instruction works on 10 data items at a time, hence each instruction performs 10 operations. For 100 pixels, the number of operations is 1000. Also, the analog to digital conversion needs to be done only once for 100 sensors in the array. Once the digital data is avaliable, it can be stored and moved in the processing layer using other instructions if necessary. Thus the

number of instructions required to process an edge detection program is 100.

Number of pixels in design: 10 X 10 = 100

Time for processing one sample in ADC:

$$t_{ADC} = 2.17 \text{ ns [20]}$$

Area for each ADC:

$$A_{ADC} = 0.191 \text{ mm}^2 \text{ [20]}$$

Number of samples to be processed by ADC:

$$s_{total} = 100$$

Design Analyzer Estimates:

Time for processing one instruction in a 3-D system:

$$t_{3-D} = 1.4 \text{ ns}$$

Area for the 3-D processing layer:

$$A_{3-D,proc} = 1.22 \text{ mm}^2$$

Time for processing one instruction in a 2-D system:

$$t_{2-D} = 1.57 \text{ ns}$$

Area for the 2-D processing layer:

$$A_{2-D,proc} = 2.213 \text{ mm}^2$$

For a 3-D system, 10 ADCs covert the sensory information (from 100 sensors)

Number of read instructions:

$$nRead_{3-D} = 10 \text{ (1 read operation per ADC)}$$

Time for 10 ADC conversions:

$$t_{ADC,3-D} = t_{ADC} * nRead_{3-D} = 21.7 \text{ ns. (10 ADCs, processing 10 samples)}$$

Time for 100 instructions:

$$t_{3-D,100} = t_{3-D} * 100 = 140 \text{ ns}$$

(Since each instruction is excuted in 10 PEs at a time, the number of instructions to be executed is only 100)

Time for one edge detected image in a 3-D system:

$$t_{3-D,total} = t_{ADC,3-D} + t_{3-D,100} = 161.7 \text{ ns}$$

Area for the 3-D system:

$$A_{3-D,system} = A_{2-D,proc} + (A_{ADC} * 10) = 3.132 \text{ mm}^2$$

Area of the sensing array: $A_{SA} = A_{3-D,system} = \mathbf{3.132\ mm^2}$

For a 2-D system, a single ADC converts all the sensory information (from 100 sensors)

Number of read instructions:

$$nRead_{2-D} = 100$$

Time for 100 ADC conversions:

$$t_{ADC,2-D} = t_{ADC} * nRead_{2-D} = 217 \text{ ns (single ADC)}$$

Time for 100 instructions:

$$t_{2-D,100} = t_{2-D} * 100 = 157 \text{ ns}$$

Time for one edge detected image in a 2-D system:

$$t_{2-D,total} = t_{2-D,ADC} + t_{2-D,100} = 374 \text{ ns}$$

Area for the 2-D system $A_{2-D,system}$ = Area of sensor array + Area of processing layer + Area of an ADC

$$= A_{SA} + A_{2-D,proc} + A_{ADC}$$

$$= 3.132 \text{ mm}^2 + 2.213 \text{ mm}^2 + 0.191 \text{ mm}^2$$

$$A_{2-D,system} = 5.536 \text{ mm}^2$$

Thus the area for a 2-D system comes to 5.536 mm^2. The time for the read operations is covered in the time to convert the analog signals to digital signals.

Table 5.2 provides the results comparing a 2-D system with a single ADC and a 3-D system with ten ADCs. The area, time and power were estimated by the Synopsys Design Analyzer tool. The systems were synthesized in 0.25μm technology.

	2-D system (single ADC)	3-D system
Area (mm^2)	5.536	3.132
Power (mW)	202.1	270.9
Time taken for one edge detected image (ns)	374	161.7
Number of ADCs	1	10
Number of PEs	10	10

Table 5.2: Comparisons between 2-D architecture (single ADC) and 3-D architecture

5.3.1 Discussion on 2-D system architecture (single ADC) and 3-D system architecture

As the results shown in Table 5.2 indicate, there is an advantage of going to a 3-D system. Though the power dissipation is higher than the 2-D architecture, the 3-D system can exploit speed and parallelism. Also the fill factor, in this thesis defined as the photosensor area plus the sensor circuitry area compared to the total area is better for a 3-D system as the processing layer in a 3-D system is below the sensing area. The fill factor as defined above for a 2-D system is 56%.

$$\text{Fill factor} = \frac{sensingarea}{totalarea} * 100$$

$$= \frac{A_{SA}}{A_{2-D,system}} * 100$$

$$\text{Therefore, fill factor} = 56\%$$

The bottleneck in the 2-D system happens to be the ADC. A single ADC converts the analog signals from each sensor and feeds the bus. To output an edge detected image as a 3-D architecture (in 161.7 ns), the processing takes 157 ns, so the ADC need to process 100 samples in 0.047 ns.

Time taken by a 3-D system = $t_{3-D,total}$ = 161.7 ns

Time taken for 100 instructions in a 2-D system = $t_{2-D,100}$ = 157 ns

Time left for analog to digital conversion of 100 samples: $t_{(3-D)-(2-D)}$ = 4.7 ns

Worst delay of the ADC required: $t_{ADCreqd} = \frac{t_{(3-D)-(2-D)}}{s_{total}}$ = 0.047 ns

Number of read instructions required in a 2-D system = $nRead_{2-D}$ =100

Time required for the read instructions: $t_{2-D,read,100} = t_{2-D} * nRead_{2-D}$ = 157 ns

Minimum total time, in which such a 2-D system can do an edge detection: $t_{2-D,min}$ = $t_{2-D,read,100} + t_{2-D,100}$ = 314 ns

$t_{2-D,min} > t_{3-D,total}$. The 3-D system is better than a 2-D single ADC system. The 3-D system implemented exploits the advantages of 3-D integration. The data from 10 sensors can be read at one time, parallely. Each sensor data is then converted in one ADC (10 ADCs in total) and then processed in 10 processing elements in parallel. In the 2-D system with a single ADC, the time for the data to be read by the processing elements is the bottleneck.

The following subsection covers another possible scenario - row parallel ADCs for a 2-D structure and compares the simulation results with a 3-D system archictecture as shown in Figure 5.2.

5.3.2 An ADC per row in 2-D system architecture

A column of ADCs converts analog signals row wise and feed the processing elements as shown in Figure 5.3 and the results are compared in Table 5.4

The calculations for the 3-D system remain the same as described earlier. This

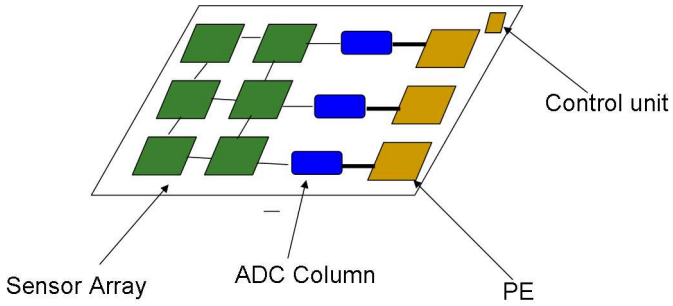


Figure 5.3: modSIMD 2D System Architecture with an ADC column

modified 2-D system architecture implementation, addresses the concern of the time required to read the data (by the PEs). A row-wise ADCs are included. So for a 10 X 10 sensor array, a column of 10 ADCs is placed and they directly feed the data into different processing elements as shown in Figure 5.3. The calculations for the same are as follows:

For a modified 2-D system, an ADC per row converts the sensory information (from a 10 X 10 sensor array).

Number of read instructions:

$$nRead_{2-D,10ADC} = 10$$

Time for 100 ADC conversions:

$$t_{2D,10ADC} = t_{ADC} * nRead_{2-D,10ADC} = 21.7 \text{ ns (10 ADCs)}$$

Time for 100 instructions:

$$t_{2-D,10ADC,proc} = t_{2-D} * 100 = 157 \text{ ns (for edge detection)}$$

Time for one edge detected image in a 2-D system:

$$t_{2-D,10ADC,system} = t_{2-D,10ADC} + t_{2-D,10ADC,proc} = 178.7 \text{ ns}$$

Area for the 2-D system:

$$A_{2-D,10ADC,system} = \text{Area of sensor array} + \text{Area of processing layer} + \text{Area of 10 ADCs}$$

$$= A_{SA} + A_{2-D} + (A_{ADC} * 10) = 3.132 \text{ mm}^2 + 2.213 \text{ mm}^2 + 1.91 \text{ mm}^2$$

$$= 7.036 \text{ mm}^2$$

Thus the area for a 2-D system comes to 7.036 mm^2

	2-D system	3-D system
Area (mm^2)	7.036	3.132
Power (mW)	274.1	270.9
Time taken for one edge detected image (ns)	178.7	161.7
Number of ADCs	10	10
Number of PEs	10	10

Table 5.3: Comparisons between modified 2-D architecture (an ADC per row) and 3-D architecture

As the results indicate, the performance of the modified 2-D system architecture with an ADC is per row is closer to the 3-D architecture. The distinguishing factor is the fill factor, which in the case of a modified 2-D architecture with a column of ADCs falls below 50%. The heat dissipated by such a system is closer to the 3-D system. Based on the fill factor, 3-D architecture is better than the 2-D architecture.

$$\text{Fill factor} = \frac{sensingarea}{totalarea} * 100$$

$$= \frac{A_{SA}}{A_{2-D,10ADC,system}} * 100$$

Therefore, fill factor for a 2-D system with 10 ADCs = 44.51%

The fill factor for a 3-D system depends on the area occupied by the 10 vertical interconnects. In the design implemented, only 10 vertical interconnects are needed and the rest of the sensing area is used for the sensor arrays.

5.3.3 Conclusions from 2-D implementations and 3-D implementation

Two different 2-D system implementations were compared to the 3-D implementation. In both the cases, the 3-D system implementation is determined to be better than any of the 2-D implementation. The 3-D system exploits the availability of data at once and also utilizes the area of the top sensing layer for sensing. Thus the fill factor in the 3-D system is greater than any of the 2-D systems discussed. In the 3-D system, only 10 vertical interconnects to transfer the data are required. The analog to digital conversion and the processing happens in a layer below the sensing layer. The power dissipation in a 3-D system, is more than a single ADC 2-D system but is similar to 10 ADC 2-D system. The area in a 3-D system is optimally used for processing needs as well as sensing needs. The 3-D system also happens to be faster than any of the 2-D systems.

5.4 Comparison with LARS II

LARS II or Local autoadaptiver sensor is an imaging chip used in automotive applications. It has 368 X 256 pixels [14]. modSIMD architecture was scaled up to

cover as many pixels and the following results were obtained (Fig 5.4). modSIMD estimates from Design Analyzer and LARS II numbers from [14].

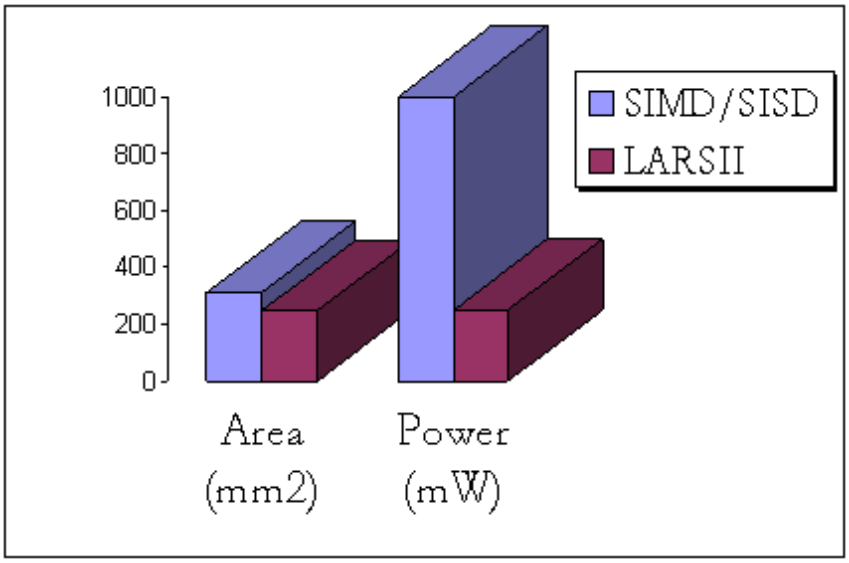


Figure 5.4: Comparing LARS II and modSIMD

From Figure 5.4, it is observed that the area of modSIMD is larger than LARS II and the power consumption (of modSIMD) is much higher than LARS II, as was expected. LARS II imagers compare the pixel values of the adjoining pixels and adjust the value of the center pixel to obtain an image.

5.5 Comparison with ARM Processor

An ARM processor is a low end RISC microprocessor [23]. The RTL code for the RISC processor was obtained from University of Texas at Austin. The ARM processor is a 32 bit instruction, 32 bit data pipelined RISC processor. The multiplier unit in the ARM was disabled. Figure 5.5 shows the configuration in which the ARM processor was simulated to obtain the results.

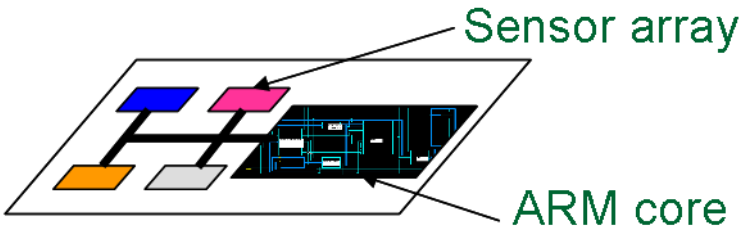


Figure 5.5: ARM and Sensor Array

The ARM processor was also synthesized in the 0.25μm technology. The comparison are made to a modSIMD(10) - the archtecture with 10 PEs.

	modSIMD processor	ARM processor system
Area (mm^2)	1.22	2.02
Power (mW)	190	129
Time taken for 10 additions (ns)	3.35	11.7
Time taken for 10 comparisons (A¿B)(ns)	3.35	10.2

Table 5.4: Comparisons between modSIMD(10) and ARM processors

As shown in Table 5.4, the modSIMD processor dissipates 50% more heat than an ARM processor. The modSIMD processor performs a single operation on 10 data points as opposed to the ARM processor which operates only on one data point at a time. This is seen clearly in the comparison to do 10 additions or 10 comparisons.

5.6 Application 1: Same sensor in TFSAs: Image convolutions and the Sobel Edge Detector

Image processing algorithms are necessary not just for photosensors but also for a variety of sensors associated with the various signal domains. Some examples include

- Optical pressure sensor heads

- Bacteriorhodopsin-Silicon Photosensor
- Infrared photodetectors
- Optical vapor sensing arrays (Polymer deposited)

One of the basic algorithms in image processing is edge detection. It showcases how image algorithms work. Moreover many of the algorithms used to process data from parallel sensors uses techniques like convolutions etc, Sobel Edge Detection program makes it easy to see the parallels in various techniques.

Convolution is a simple mathematical operation which is fundamental to many common image processing operators. Convolution provides a way of 'multiplying together' two arrays of numbers, generally of different sizes, but of the same dimensionality, to produce a third array of numbers of the same dimensionality. This can be used in image processing to implement operators whose output pixel values are simple linear combinations of certain input pixel values.

In an image processing context, one of the input arrays is normally just a greylevel image. The second array is usually much smaller, and is also two dimensional (although it may be just a single pixel thick), and is known as the kernel.

The convolution is performed by sliding the kernel over the image, generally starting at the top left corner, so as to move the kernel through all the positions where the kernel fits entirely within the boundaries of the image. Each kernel position corresponds to a single output pixel, the value of which is calculated by multiplying together the kernel value and the underlying image pixel value for each of the cells in the kernel, and then adding all these numbers together. Convolution

can be used to implement many different operators, particularly spatial filters and feature detectors.

5.6.1 Sobel Edge Detection

The Sobel operator performs a 2-D spatial gradient measurement on an image and so emphasizes regions of high spatial gradient that correspond to edges. It consists of two kernels as described below.

-1	0	$+1$
-2	0	$+2$
-1	0	$+1$

Table 5.5: Gx

$+1$	$+2$	$+1$
0	0	0
-1	-2	-1

Table 5.6: Gy

5.6.2 Psuedo code for Edge Detector

Case 1: 10 PEs modSIMD

```
For i = 1:10
GetData(adjoining elements);
DoEdgeDetection(i);
End //end for loop
```

The for loop goes from 1 to 10 for each processing element in the 10X10 pixel array. The code for edge detection (the routine DoEdgeDetection) is provided in Appendix A.

Case 2: ARM Processor

```
For i = 1:100
GetData(adjoining elements);
DoEdgeDetection(i);
End //end for loop
```

The for loop has to go from element 1 to 100 for a 10X10 pixel array, for an ARM processor. In a same sensors in the TFSA case, where the same operations are executed on all the data points, the modSIMD processor exploits the SIMD mode of operation, thus giving the necessary speed up.

5.7 Application 2: Different sensors in the sensor array: The chemical sensor array

As opposed to arrays of same detectors, chemical arrays of many detectors allow us to do multiple checks on a given reading. It would be desirable to provide some cross checking in the sensory array itself. This in itself represents a totally different computational challenge. These hybrid arrays of sensors is not very uncommon.

A single processing element based design would be considerably slow. A 100 processing element processor, wouldnt necessarily be fast.

Also the processor needs to implement some sophisticated signal processing tools available. Such tools include correlated sampling approaches, repeated tests for a given analyte, cross checking, etc.

A simple example. Suppose *Analyte A* and *Analyte B* react similarly to sen-

sor 1. And *Analyte A* reacts to sensor 2 in the sensor array but *Analyte B* doesnt respond to sensor 2.

```
PE_ID = 1111

GatherData();

PE_ID = 0010;

ProcessData(address0);

StoreData(address1);

ProcessData(address2);

StoreData(address3);

CompareData(address1, address3);

End;
```

'PE_ID 0001'(processing element attached to 'sensor 1') narrows the chemicals to either 'Analyte A' or 'Analyte B'. 'PE_ID 0010' can now determine if the measurand is 'Analyte A' or 'Analyte B'. Thus, being able to randomly address the processing elements, aids biochemical, chemical and gas sensor arrays.

Except for the `GatherData` part, both the ARM and the modSIMD(10) take similar number of cycles to execute the rest of the code. Since the modSIMD(10) acquires data for all the sensors at once and also since the PEs acquire data for their nearest neighbors, it is faster than ARM processor.

5.8 Drawbacks and Future Work

The proposed architecture dissipates 50% more heat than an ARM processor. The speed of operations comes from exploiting the 3-D integration, all the data from sensor arrays is available at once and performing operations in parallel on them, which was intended to be exploited. Moreover 32-bit processing might not be required (in the ARM case) and the savings in area could be utilized as shown in the modSIMD architecture.

Pipelining can be added to the architecture. Data from 10 sensors is processed in one PE, as the data is being read, the processor element can start processing the data it has already recieved.

Read-out mechanism in the proposed architecture is serial. A column-wise or row-wise or vertical read-out mechanism can be envisioned. For a column-wise or row-wise read out, the PEs can be organized in an appropriate way. This leads to the possibility of more number of pins on the chip. For a vertical read-out mechanism, the biggest block could be the area required for the vertical interconnect.

A floating point unit can be added to the architecture, to make it a more powerful processing tool.

5.8.1 Future Work - Heat Chip

The layout of a heat chip designed by Zeynep Dilli (graduate student, ECE Department, University of Maryland, College Park) is shown in Figure 5.6. The chip was manufactured in Mosis. Using an array of diodes (16 X 16 array of diodes) and

resistors, various parts of the chip are heated in intervals of 100 ms. The diodes carry different currents in the diode array and thus heat up the chip differently.

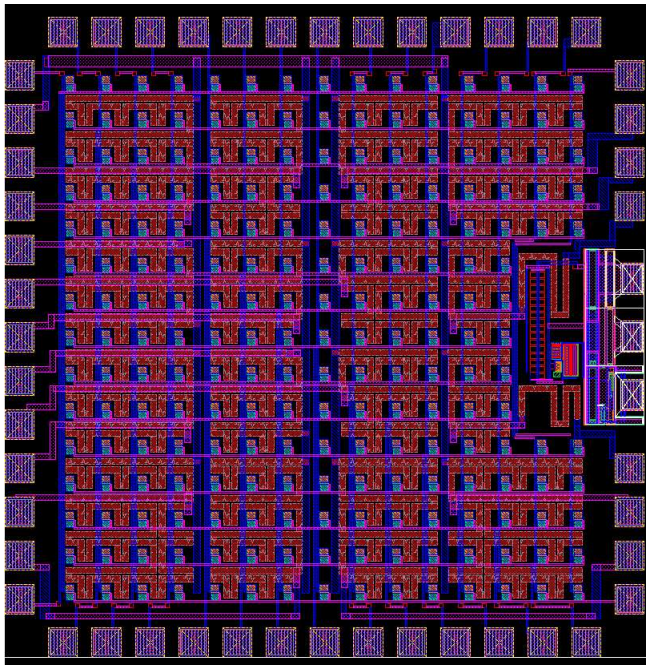


Figure 5.6: Layout of a Heat Chip, by Zeynep Dilli

The heating pattern can be seen in Figure 5.7.

modSIMD processor with heat sensors as the sensor array can be employed to determine the heat arrays. modSIMD architecture with appropriate TFSA can be used to sense appropriate microscopic or even smaller physical changes, thus increasing the understanding of materials greatly. Applications of these include not only heat changes but also microscopic pressure changes, microscopic structural faults and in identifying very low concentration chemicals in a small area.

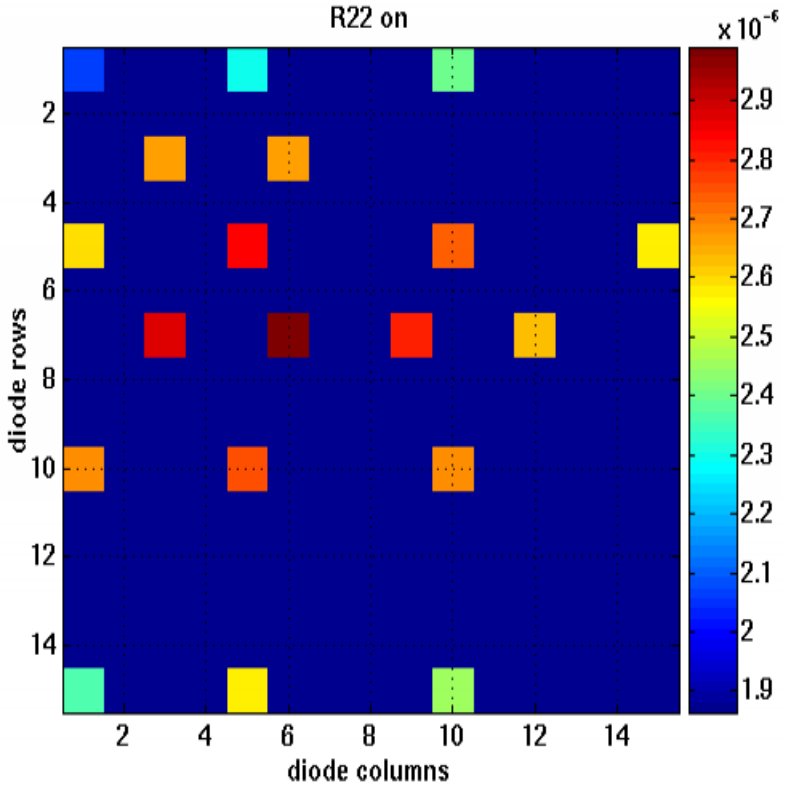


Figure 5.7: Heating Pattern Across the Chip

5.9 Conclusions

As shown earlier, the 3-D architecture has more advantages than the 2-D architecture. The 3-D architecture dissipates more power than a 2-D architecture, but is faster and occupies lesser area. The 3-D architecture for thin film sensors can be used for a variety of physical signals. Many sensor arrays use image processing tools to process data into information as shown in Table 2.3. Thus a processing layer, as proposed and implemented in this thesis, is a valuable addition. The modSIMD architecture not only exploits the advantage of SIMD mode on sensor arrays which can use SIMD type processing but it also works on dissimilar sensor arrays.

The processor for image processing or any other sensors on top can be designed

independently in the architecture presented. The thin-films can be developed and deposited on the proposed processor. From the case studies of edge detection and chemical sensors, a 10 processing element processor turns out to be more versatile. It is also a cost effective in silicon area.

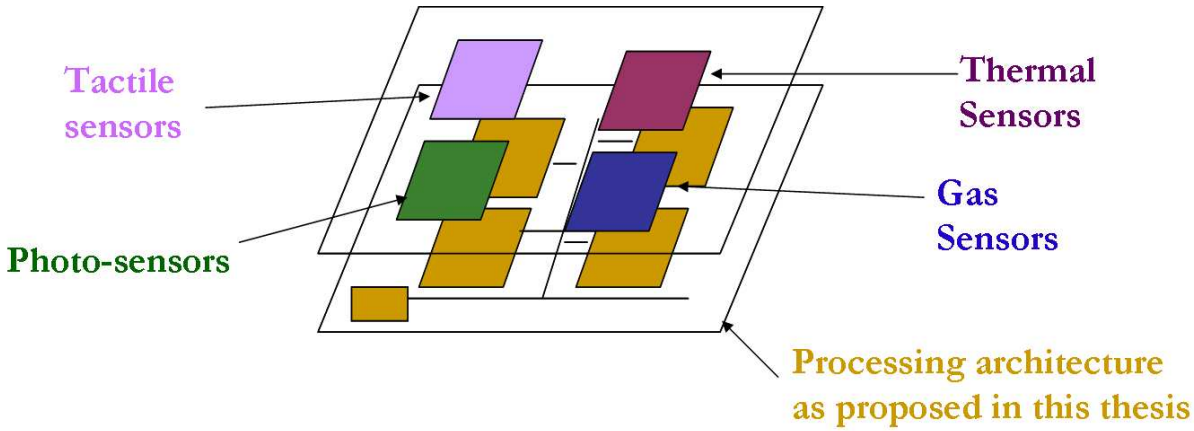


Figure 5.8: 3-D TFSA on modSIMD

Figure 5.8 gives the future view of Thin Film Sensor Arrays on IC. The proposed architecture can handle a variety of thin film sensor arrays on the same processing layer. By appropriately addressing all the processing elements, a processor design to process many thin film layers photodetectors, magnetoresistance sensors, biochemical sensors, gas sensors or thermal sensors a 3-D integrated thin film sensor arrays on IC (processor) can be built. Such a device can find application in wide ranging fields - be it astronomical exploration or war zone or in a wind tunnel.

3-D integration of devices provides an exciting new field of research as multicore architectures are explored and lesser power dissipation is sought. Off chip memories increase power dissipation and with multicore architectures, either in SIMD or MIMD, there will be a lot of need for data, data caches, instructions to be fed to

various cores. Thus future direction of reducing power dissipation may well might be 3-D integration.

Appendix A

ALU Operations and Code for Edge Detection

A.1 ALU Operations

The following is a list of ALU operations in the modSIMD processor.

`‘OP_ADD: Result = Oprnd_A + Oprnd_B;`

Adds the operands A and B and stores in result.

`‘OP_A, ‘OP_Ap, ‘OP_App:  Result = Oprnd_A;`

Copies operand A into the result.

`‘OP_SUB: Result = Oprnd_A - Oprnd_B;` Subtracts operand B from operand A and puts into result.

`‘OP_LEFT_SHIFT: Result = Oprnd_A << 1;`

Bit wise left shifts operand A, useful in division by 2.

`‘OP_RIGHT_SHIFT: Result = Oprnd_A >> 1;`

Bit wise right shifts operand A, useful for multiplication by 2.

`‘OP_ALL_ZEROS: Result = 16’b0000000000000000;`

Fills in the result with all zeros.

`OP_A_AND_B: Result = Oprnd_A & Oprnd_B;

Bit wise ands operand A and operand B.

`OP_notA_AND_B: Result = Oprnd_A & Oprnd_B; And

`OP_B: Result = Oprnd_B;

`OP_notA_AND_notB: Result = Oprnd_A & Oprnd_B;

`OP_A_XNOR_B: Result = (Oprnd_A Ôprnd_B); `OP_notA: Result = Oprnd_A;

`OP_notA_OR_B: Result = Oprnd_A | Oprnd_B;

`OP_A_AND_notB: Result = Oprnd_A & Oprnd_B;

`OP_A_XOR_B: Result = Oprnd_A Ôprnd_B;

`OP_A_OR_B: Result = Oprnd_A | Oprnd_B;

`OP_notB: Result = Oprnd_B;

`OP_A_OR_notB: Result = Oprnd_A | Oprnd_B;

`OP_A_NAND_B: Result = (Oprnd_A & Oprnd_B);

`OP_ALL_ONES: Result = 16'b1111111111111111; Fills in the result with all ones.

A.2 Code for Edge Detection

The following routine is executed in parallel over different PEs.

The code should be read as:

Operation, A, C, B

Where operation is the ALU Operation, A is the address of operand A, C is the destination address and B is the address of operand B.

OP_RIGHT_SHIFT 000010, 100010

OP_RIGHT_SHIFT 000100, 100100

OP_RIGHT_SHIFT 001000, 101000

OP_RIGHT_SHIFT 000110, 100110

OP_ADD 000001, 100001, 000011

OP_ADD 100011, 100011, 100010

OP_ADD 000111, 100111, 001001

OP_ADD 100111, 101001, 101000

OP_SUB 100010, 100101, 101000

Appendix B

Other aspects of 3-D integrations

This appendix highlights the research directions in Analog to digital conversion (ADC), vertical interconnects, etc aspects of 3-D integration. It is hoped that this will complete the reference list for a 3-D integrated TFSA on silicon solution.

B.1 Analog to Digital Conversion

The electrical signals generated from conversion of the above signal domains to electrical are processed digitaly. This requires building ADC circuitry. Almost all the sensor arrays discussed above have used the $\Sigma\Delta$ modulator. An asynchronous $\Sigma\Delta$ modulator is a better choice as the number of analog components that must be matched for reducing fixed pattern noise is minimized [8].

B.2 Vertical integration of thin films on silicon

A variety of thin films can be deposited on silicon substrate. Vertical interconnects will connect the processing core to the sensor arrays. Vertical integration has been

reported in literature (Figure B.1).

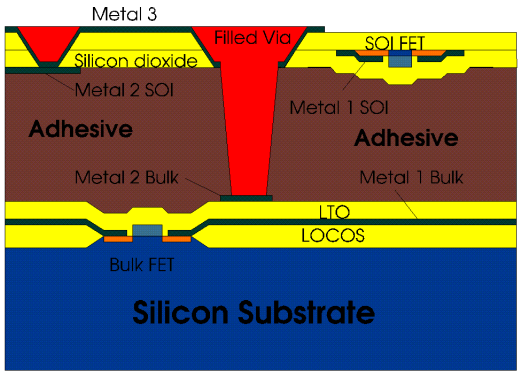


Figure B.1: 3-dimensional photodetector

Figure B.2 shows vertical integration of a photosensing thin film over a processing substrate. Electrical through-wafer interconnects (ETWI) connect devices between both sides of a substrate are critical components for microelectromechanical systems (MEMS) and integrated circuits (IC), as they enable three-dimensional (3-D)structures and permit new packaging and integration geometries. For the highest interconnection density, area interconnections with electrical through-wafer interconnects (ETWI), as opposed to peripheral interconnects, are required. An example of ETWI with piezoresistive cantilever beam arrays is discussed in [24].

Figure 5.2 shows an SEM image of the ETWI from [24].

B.3 Power Dissipation in 3-D processors

3-D via technology provides savings in power for same rates of data transfer (between sensors and processor(s)), higher data transfer rates than comparable 2-D processing systems. Primarily, the 3-D vias have parasitic capacitences of the order of femtofarads thus greatly reducing the power consumption. A 3-D imager and a

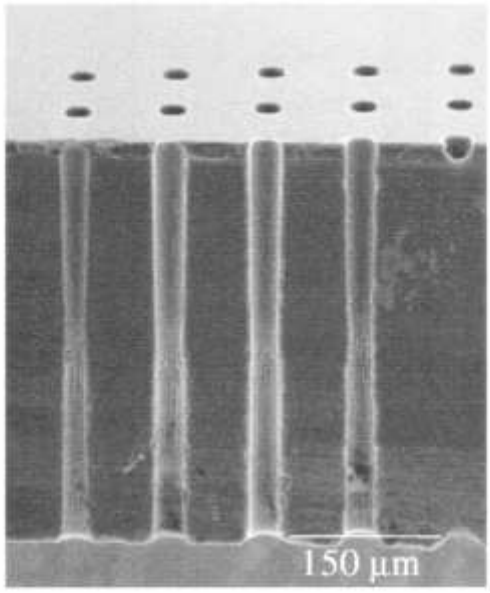


Figure B.2: Through Wafer Interconnect

comparable 2-D imager system are compared in [8].

B.4 Data Transfer Strategies

Data transfer has been seen as the bottleneck for many applications. Data transfer from memory to hard disk specifically. A variety of devices are available. A study of Serial-ATA architecture is presented in Appendix C.

Appendix C

The SATA architecture for Data Transfer

Data, digital music, video - all need storage devices. With digital photography, home video making, digital music, live TV recording and other video games, storage of data at home has come a long way. With the growing demands of data transfer from the processor to the storage device, there is a need to understand and improve the bus speeds and data bandwidth of these devices.

Some storage devices are directly attached to the motherboard of a computer - hard disk in most computers. Some storage devices are on the network and not necessarily directly attached to the motherboard of a computer. A variety of data transfer protocols have been hence developed to transfer data to and from these storage devices to the processor for processing. In this study, we will discuss the serially attached ATA devices (SATA devices) as ATA devices are predominantly used in home computing.

SATA devices are directly attached to the motherboard of a computer. In the case above, for desktop computing - the devices predominantly in use are the SATA devices. In the next few pages, we will discuss the working of a SATA device and

contrast it with the parallel ATA controllers found in most of the desktops these days.

C.1 SATA Devices

Serial ATA is an evolutionary replacement for the Parallel ATA physical storage interface. Parallel ATA is the primary internal storage interconnect for the desktop, connecting the host system to peripherals such as hard drives, optical drives, and removable magnetic media devices. Parallel ATA is an extension of the original parallel ATA interface introduced in the mid 1980's and maintains backward compatibility with all previous versions of this technology. The latest revision of the Parallel ATA specification supports up to 100Mbyte/sec data transfers. Further update to the parallel bus architecture provides up to 133Mbytes/sec - was recently finalized.

Serial ATA is the next generation storage interconnect, designed to replace parallel ATA technology. Serial ATA is the proactive evolution of the ATA interface from a parallel bus to a serial bus architecture. This architecture overcomes the electrical constraints that are increasing the difficulty of continued speed enhancements for the classic parallel ATA bus. Serial ATA will be introduced at 150Mbytes/sec. Though Serial ATA will not be able to directly interface with legacy Ultra ATA hardware, it is fully compliant with the ATA protocol and thus is software compatible.

C.2 Some of the advantages of Serial ATA versus Parallel ATA:

1. Bus design: Parallel ATA Bus Design: The bus design maintains a backward compatibility with all previous ATA revisions, using the standard 16-bit wide parallel data bus and 16 control signals across a 40-pin connector. Serial ATA Bus Design: The bus has just 7 pins. SATA uses a single signal path to transmit data serially (bit by bit) and a second serial path to return receipt acknowledgement to the sender. Each of these signals is a 2-wire differential pair. Control information is transmitted either as short predefined bit sequences that are distinguishable from data, in packet format, or out-of-band signaling (control signals sent using on/off signal pulses).

2. Bandwidth Parallel ATA Bandwidth: 100Mbytes/sec Serial ATA Bandwidth: 1500Mbits/sec using 8b/10b encoding, thus effective bandwidth is 150 Mbytes/sec

3. Electronic Design Constraints Optimization of any high-speed digital bus design in fact requires careful consideration of analog design issues. Undesired analog effects associated with parallel data busses such as crosstalk, ground bounce, ringing, and clock skew have become major design constraints.

   Parallel ATA: Crosstalk - parallel busses where multiple adjacent lines may be switching in the same direction at the same time and inject a noise voltage onto a victim signal. Ground bounce is most problematic when several signals switch at the same time or when using high-speed drivers

   Serial ATA: Serial ATA uses low voltage differential signaling. With this ap-

proach, each data "signal" is in fact transmitted over two lines which carry equal and opposite versions of the signal. The receiver then decodes the signal based on the differential voltage between these lines. The "common-mode" voltage, or the voltage the lines use as a DC reference plus any noise injected equally into both lines, is rejected at the receiver. This common-mode voltage may change over time, though the variations above a certain frequency may be injected into the receiver as noise.

4. Clocking Parallel ATA: the clock or data strobe signal is generated at the source and sent with the data - this is also referred to as non-interlocked clocking Serial ATA: The clock is embedded in the data strobe itself.

C.3 Overview of Serial attached ATA devices

- 7-pin connector cable
- Control Registers on device as well as the host
- A copy of registers on host side
- Device is indirectly controlled over SATA bus

The host writes into the control registers and the data transfer begins. To maintain software compatibility the device registers initiate the transfers. But unlike parallel ATA devices, the SATA devices don't have many control lines over which the device can signal the host. Hence, a copy of the registers on the device is maintained on the host side too which is updated periodically.

A SATA controller also handles various functions like the Open Systems Interconnection model (for TCP/IP etc).

BIBLIOGRAPHY

[1] S. Middelhock and S. A. Audet, *Silicon Sensors*, Academic Press.

[2] John L. Hennesy and David A. Patterson, *Computer Architecture - A quantitative approach*, Morgan Kaufmann Publishers.

[3] Samir Palnitkar, *Verilog HDL*, Prentice Hall.

[4] David R. Smith and Paul D. Franzon, *Verilog styles for Synthesis of Digital Systems*, Prentice Hall.

[5] Zimmermann, *Integrated Silicon Optoelectronics*, Springer.

[6] *http://www.synopsys.com*

[7] *http://www.cadence.com*

[8] Lisa G. McIlrath and Paul M. Zavracky, An Architecture for Low-Power Real Time Image Analysis Using 3D Silicon Technology, In *Proceedings SPIE AeroSense Symposium*, Orlando, FL, April 1998.

[9] P. Dudek and Peter J. Hicks, "'A General-Purpose Processor-per-Pixel Analog SIMD Vision Chip"', *IEEE Transactions on Circuits and Systems-I:Regular papers*, vol 52, No.1, Jan 2005.

[10] Satoshi Shigematsu, Hiroki Morimura, Yasuyuki Tanabe, Takuya Adachi, and Katsuyuki Machida, "'A Single-Chip Fingerprint Sensor and Identifier"', in *IEEE Journal of Solid State Circuits*, Vol. 34, No. 12, December 1999.

[11] Pontus Eriksson, Jan Y. Andersson, and Goran Stemme, "'Thermal Characterization of Surface-Micromachined Silicon Nitride Membranes for Thermal Infrared Detectors"', in *IEEE Journal of Microelectromechanical Systems*, Vol. 6, NO. 1, March 1997.

[12] C. H. Smith, R. W. Schneider, T. Dogaru, and S. T. Smith, "'Eddy-current testing with GMR Magnetic Sensor Arrays"', presented at the *Quantitative Nondestructive Evaluation Conference*, Green Bay, WI , July 28, 2003.

[13] Yong Xu,Yu-Chong Tai,Adam Huang and Chih-Ming Ho,"'IC-Integrated Flexible Shear-Stress Sensor Skin"', in *IEEE Journal of Microelectromechanical Systems*, Vol. 12, No. 5, October 2003.

[14] Stephan Benthien, Tarek Lule, Bernd Schneider, Michael Wagner, Markus Verhoeven and Markus Bohm, "'Veritcally Integrated Sensors for Advanced Imaging Applications"', in *IEEE Journal of Solid State Circuits*, Vol. 35, No.7, July 2000.

[15] Cristina Nicolescu and Pieter Jonker, "'Parallel Low-Level Image Processing on a Distributed-Memory System"', in *Parallel and Distributed Processing: 15 IPDPS 2000 Workshops*, Cancun, Mexico, May 2000. Proceedings

[16] G. de Cesare, G. Masini and F. Palma, "'Modeling and Realization of a High-Gain homojunction a-Si:H Bulk Barrier Phototransistor"', in *IEEE Transactions on Electron Devices*, Vol. 43, No. 8, August 1996.

[17] Frank Zee and Jack W. Judy, "'Micromachined polymer-based chemical gas sensor array"', *Sensors and Actuators* B 72 (2001) pg 120-128.

[18] P. C. Jurs, G. A. Bakken, and H. E. McClelland, "'Computational Methods for the Analysis of Chemical Sensor Array Data from Volatile Analytes"', in *Chemical Reviews* 2000, 100, pp 2649-2678.

[19] Keith J. Albert, Nathan S. Lewis, Caroline L. Schauer, Gregory A. Sotzing, Shannon E. Stitzel, Thomas P. Vaid, and David R. Walt, "'Cross-Reactive Chemical Arrays"', in *Chemical Reviews* 2000, 100, pp 2595-2626.

[20] D. Lee, J. Yoo, K. Choi, and J.Ghaznavi, "Fat Tree Encoder Design for Ultra-High Speed Flash A/D Converters", in *45th IEEE Midwest Symposuim on Circuits and Systems*, Vol. 2, pp. 87-90, 2002

[21] J. B. Sulistyo, J. Perry, and D. S. Ha, "Developing Standard Cells for TSMC 0.25um Technology under MOSIS DEEP Rules", Department of Electrical and Computer Engineering, *Virginia Tech, Technical Report VISC-2003-01*, November 2003

[22] Jos. B. Sulistyo and Dong S. Ha, "A New Characterization Method for Delay and Power Dissipation of Standard Library Cells", *VLSI Design 15 (3)*, pp. 667-678, 2002

[23] $http://arm.com/documentation/ARMProcessor_Cores/index.html$

[24] Eugene M. Chow,Venkataraman Chandrasekaran, Aaron Partridge,Toshikazu Nishida, Mark Sheplak, Calvin F. Quate, and Thomas W. Kenny, "'Process

Compatible Polysilicon-Based Electrical Through-Wafer Interconnects in Silicon Substrates"', in *IEEE Journal of Microelectromechanical Systems*, Vol. 11, No. 6, December 2002