

1. INTRODUCTION

In the past three years several multiple data path and pipelined digital signal processors have been introduced into the marketplace. This new generation of DSP's takes advantage of higher levels of integrations than were available for their predecessors. The **Tiger SHARC** processor is the newest and most power member of this family which incorporates many mechanisms like SIMD, VLIW and short vector memory access in a single processor. This is the first time that all these techniques have been combined in a real time processor.

The TigerSHARC DSP is an ultra high-performance static superscalar architecture that is optimized for tele-communications infrastructure and other computationally demanding applications. This unique architecture combines elements of RISC, VLIW, and standard DSP processors to provide native support for 8, 16, and 32-bit fixed, as well as floating-point data types on a single chip.

Large on-chip memory, extremely high internal and external bandwidths and dual compute blocks provide the necessary capabilities to handle a vast array of computationally demanding, large signal processing tasks.

2. DIGITAL SIGNAL PROCESSOR

Strictly speaking, the term “DSP” applies to any microprocessor that operates on digitally represented signals. In practice, however, the term refers to microprocessors specifically designed to perform digital signal processing tasks. Because most signal processing systems perform complicated mathematical operations on real-time signals, DSPs use special architectures to accelerate repetitive, numerically intensive calculations. For example, DSP architectures commonly include circuitry to rapidly perform multiply accumulate operations, which are useful in many signal-processing algorithms. Also, DSPs often contain multiple-access memory architectures that allow the processor to simultaneously load multiple operands. In addition, DSPs often include a variety of special memory addressing modes and program-flow control features designed to accelerate the execution of repetitive operations. Lastly, most DSP processors include specialized on-chip peripherals or I/O interfaces that allow the processor to efficiently interface with other system components, such as analog-to-digital converters and host processors.

Before going into the details of the Tiger SHARC architecture let us familiarize with a few architectural techniques which are the key elements of this new DSP.

1. VLIW- Very Long Instruction Word

VLIW points to the instructions that specify more than one concurrent operation in a single instruction. The following features characterize it: -

- Instruction width is quite large taking many bits to encode multiple operations.
- Rely on software to pack the collection of operation (compaction).
- In code with limited instruction parallelism, most of the instruction is wasted with no operation.

2. SIMD: - Single Instruction Multiple Data

A very important class of architectures in the history of computation, single-instruction/multiple-data machines is capable of applying the exact same instruction stream to multiple streams of data simultaneously. For certain classes of problems, e.g., those known as data-parallel problems, this type of architecture is perfectly suited to achieving very high processing rates, as the data can be split into many different independent pieces, and the multiple instruction units can all operate on them at the same time.

SIMD (Single-Instruction Stream Multiple-Data Stream) architectures are essential in the parallel world of computers. Their ability to manipulate large vectors and matrices in minimal time has created a phenomenal demand in such areas as weather data and cancer radiation research. The power behind this type of architecture can be seen when the number of processor elements is equivalent to the size of your vector. In this situation, component wise addition and multiplication of vector elements can be done simultaneously. Even when the size of the vector is larger than the number of processors elements available, the speedup, compared to a sequential algorithm, is immense. There are two types of SIMD architectures. The first is the True SIMD followed by the Pipelined SIMD. Each has its own advantages and disadvantages but their common attribute is superior ability to manipulate vectors.

The CPU can perform high-speed arithmetic operations within one instruction cycle because of its parallel and combinational architectural design. There are two execution units in the processor in order to facilitate the SIMD mode of operation of the processor. Only one execution unit is used in case of SISD operation. The SIMD mode is characterized by multiple instances of the same operation on different data.

3. Pipelining

The architectural problems faced in developing a high-performance pipeline for a DSP processor include most of the same challenges of traditional processor architecture, but a number of additional difficult challenges as well. As in any pipeline design, the tasks should be well balanced among the different stages.

The idea of pipelining comes from the flow of water through a pipe. The process of sending water is continued without waiting for the water in the pipe to be drained out. In this way a process is continued without caring about the completion of another process that follows it (literally there is relationship between these processes). This leads to the reduction in the critical path. The main advantage is that it either increases the clock speed (sampling speed) or reduces the power consumption at the same speed in a DSP system. By the increase in clock speed, it means that the same operation can be executed in lesser time as compared to a system without pipelining. For example fetching of an instruction in a processor and the execution if the process may be pipelined. The fetching of a next instruction is done before or within the time of execution of an operation.

Now let us check some of the architectural techniques used.

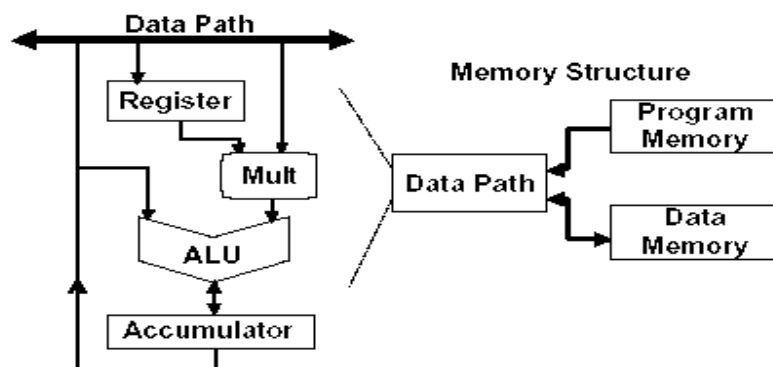
The first major architectural modification that distinguished DSP processors from the early GPPs (General Purpose Processors) was the addition of specialized hardware that enabled single-cycle multiplication. Another highly visible difference between DSP processors and GPPs lies in their memory structure. The structural details are given below.

3. ARCHITECTURAL TECHNIQUES

Von Neumann architecture

The processor architectures may be broadly classified into three groups. The first is the Von Neumann architecture. Von Neumann machines store program and data in the same memory area with a single bus. In a Von Neumann machine, an instruction contains the operation command and the address of data to be operated on (operand). Most of the general-purpose microprocessors use this architecture. It is simple in hardware implementation, but the data and program requires sharing a single bus.

Early DSP Architecture



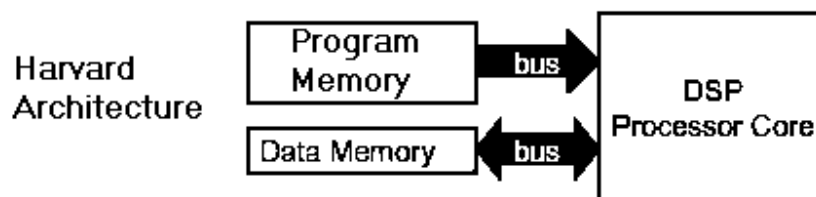
The main disadvantage of this architecture was that, when both data and programs share the same bus the system becomes slow. This is not admissible for a DSP processor that demands fastness in addition to accuracy. To overcome this handicap a new design was suggested by Howard Aiken in the 1940s during his work at the Harvard University.

Good DSP performance requires high memory bandwidth—higher than was supported on the conventional general-purpose microprocessor. To address the need for increased memory bandwidth, early DSPs developed different memory architectures that could support multiple memory accesses per cycle. The most

common approach (still commonly used) was to use two or more separate banks of memory, each of which was accessed by its own bus and could be read or written during every clock cycle. Often, instructions were stored in one memory bank, while data was stored in another. With this arrangement, the processor could fetch an instruction and a data operand in parallel in every cycle. Since many DSP algorithms (such as FIR filters) consume two data operands per instruction, a further optimization commonly used is to include a small bank of RAM near the processor core, as an instruction cache. When a small group of instructions is executed repeatedly the cache is loaded with those instructions, freeing the instruction bus to be used for data fetches instead of instruction

Harvard architecture

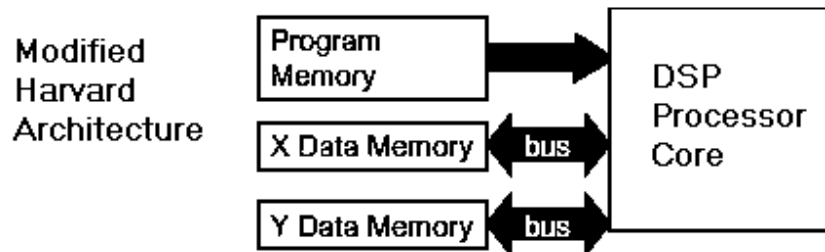
The design called as the Harvard architecture has separate buses for the data and program. Such a structure can enhance the performance because instruction and data can be fetched at the same time. Most present day DSPs use this dual bus architecture. The Harvard architecture is shown in its basic form in the figure.



The Harvard architecture is more commonly used in specialized microprocessors for real-time and embedded applications. However, only the early DSP chips use the Harvard architecture because of the cost.

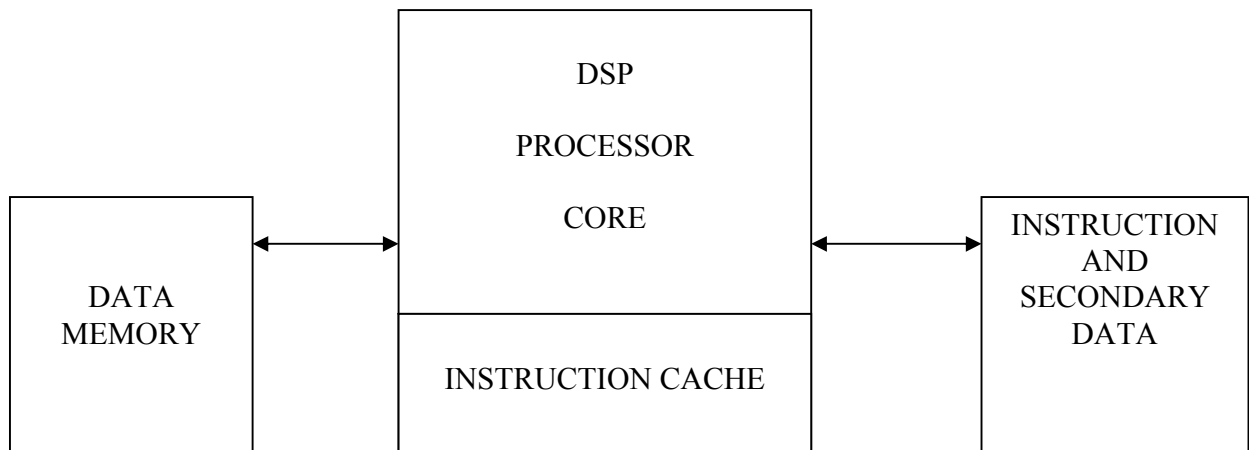
Modified Harvard architecture

There is one cost penalty with the Harvard architecture, which needs twice as many address and data pins on the chip because of using separate program and data memories. To balance between the cost and performance, modified Harvard architecture is used in most of the DSP, which uses single data and address bus externally but internally there are two separate busses for program and data. Multiplexing does the separation of program and data information. For one clock cycle program information flows on the pins, in the second cycle data follows on the same pins. The basic block diagram would be as shown below: -



SUPER HARVARD ARCHITECTURE

The most powerful architecture of DSP available now is the super Harvard architecture. Here also there is separate program and data memory. But the important thing is that the program memory can store secondary data. Also it has an instruction cache which also increases the speed. The general diagram is as shown.



The program memory can store secondary data and this greatly improves the speed. Here after an operation we can store the data if needed in the program memory itself.

4. The Tiger SHARC

The word “SHARC” implies Super Harvard ARChitecture. The SHARC architecture has been improved greatly and the most powerful DSP today known has been implemented by Analog Devices and due to the high performance it yields it is called “Tiger” SHARC.

The first implementation of the Tiger SHARC architecture is in a 0.25 micron, five level metal process at 150 MHz core clock speed. It delivers 900 MFlops (10 to the power 9 floating point operations per second) of single precision floating point performance or 3.6 GOPS of 16-bit arithmetic performance. It sustains an internal data bandwidth of 7.2 Gbytes /sec.

The TigerSHARC DSP provides leading edge system performance while keeping the highest possible flexibility in software and hardware development - flexibility without compromise. This concept will allow wireless infrastructure manufacturers to continue adapting to the evolving 3G standards while deploying a highly optimized and effective Node B solution that will realize significant overall cost savings.

For general purpose multiprocessing applications, TigerSHARC DSP's balanced architecture optimizes system, cost, power and density. A single TigerSHARC DSP, with its large on-chip memory, zero overhead DMA engine, large I/O throughput, and integrated multiprocessing support, has the necessary integration to be a complete node of a multiprocessing system. This enables a multiprocessor network exclusively made up of TigerSHARCS without any expensive and power consuming external memories or logic.

The ADSP-TS101S, the latest member of the TigerSHARC DSP family can execute 2.0 billion MACs per second while achieving the world's highest floating-point DSP performance. The TigerSHARC DSP's parallelism capabilities allow for up to four 32-bit instructions per cycle while an enhanced communication instruction set reduces some of the mountainous signal processing

functions associated with wireless down to a manageable level. The ADSP-TS101S also provides an unmatched level of both internal and external bandwidth that enable high computation rates and high data rate processing.

The combination of all the above mentioned features positions the TigerSHARC DSP as an excellent candidate for applications requiring extremely high throughput such as the channel decoding algorithms of wireless communications.

4.1 KEY FEATURES

1. STATIC SUPERSCALAR ARCHITECTURE.

- Eight 16-bit MACs/cycle with 40-bit accumulation.
- Two 32-bit MACs/cycle with 80-bit accumulation.
- Only one half a cycles required per Add, Compare, and Select (ACS) sequence for Viterbi algorithm.
- Add-subtract instruction and bit reversal in hardware for FFTs.
- IEEE-floating point compatible

2. HIGHLY INTEGRATED

- 6 Mbit on-chip SRAM
- Glueless multiprocessing
- Four Link Ports – 720 MBytes/sec transfer rate
- 64-bit external port – 720 MBytes/sec
- 14 DMA channels

3. FLEXIBLE PROGRAMMING IN ASSEMBLY AND C LANGUAGES

- User-defined partitioning between program and data memory.
- 128 general-purpose registers.
- Algebraic assembly language syntax.
- Optimizing C compiler.

- VisualDSP tools support.
- Single-instruction, multiple-data (SIMD) instructions, or direct issue capability.
- Predicated execution.
- Fully interruptible with full computation performance

The TigerSHARC DSP processes fixed- and floating-point data on a single chip. It executes 1.44 billion 40-bit MACs per second and achieves the world's highest Floating-point DSP performance.

The processor has got some more significant aspects like a register based load – store technique with super scalar dispatch mechanism in which instruction level parallelism is determined prior to the runtime under compiler and programmer control. It has got highly parallel, short vector oriented memory architecture and it supports multiple data types, including 32-bit IEEE single precision floating point and 16- bit fixed point with partial support for 8-bit fixed point. Tiger SHARC has an eight stage fully interruptible pipeline with a regular two cycle delay on all arithmetic and load/store operations and a 128-bit entry four way set – associative branch target buffer or BTB. It also has 128 architecturally visible, fully interlocked registers in four register files.

The figure shows a block diagram with major components of the architecture as well as the primary data buses. Each of the two computational blocks on the figure on the lower left (Comp Block X and Y or CBX and CBY) consists of a 32- entry general purpose register file, an ALU, multiplier and shifter. The two computational blocks constitute the primary data path of the processor. Each computational block has two 128 bit ports that connect to the three internal buses.

In the upper part of fig there are two integer units (JALU and KALU collectively called IALU). They functions as generalized addressing units; each one includes a 32 bit entry general purpose register file. Although used primarily for addressing the IALU also supports general integer arithmetic and pointer manipulation. One of the four primary masters (JALU, KALU, sequencer, or external port) produces addresses and one of the five slaves (CBX, CBY, M0, M1, M2) consumes them. Each data bus has an associated address bus, which for clarity not shown here. This figure shows three internal SRAM banks (M0, M1 and M2), each with a 128 bit connection to the bus system.

The sequencer appears in the figures upper left, along with a 128 bit entry, four way set-associative branch target buffer. The sequencer, two ALUs and the external port are the four masters of the internal bus system and supply addresses and control to the memory banks.

The three internal buses provide a direct path for instructions into the sequencer. They also provide two independent paths that may connect each memory block with each computation block, where a path can carry up to four 32 bit words per cycle.

The register file supplies up to 4 operands to the multiplier, ALU and shifter and accepts 3 operands. On the side of the memory interface, the register file can either accept 2 operands from 2 load operations or accept 1 load and provide 1 store operand.

4.3 SEQUENCING AND INSTRUCTION FLOW

We base the sequencing mechanism on a static superscalar approach in which 1 to 4 instructions execute each cycle in an instruction line. Code generation tools or a programmer determines the instruction level parallelism prior to runtime; the executable binaries contain this information (in other processors instruction lines are also referred to as instruction groups or bundles). An instruction line may contain from 1 to 4 instructions and the processor has a throughput of 1 instruction line per cycle. The idea of exposing instruction level parallelism to the compiler comes from the VLIW approach. However the sequencing in this DSP is a more general mechanism that avoids some of the shortcomings of conventional VLIW.

With the recent proliferation of processors that are statically scheduled by the compiler, VLIW has become so broad a term that it is virtually a synonym for any type of statically scheduled processor. However most of the processors are in fact significantly different from raw VLIW. As a matter of nomenclature, we refer to the particular type of sequencing implemented in the TigerSHARC as static superscalar, because it is a statically scheduled multiple issue processor. In addition, the sequencer supports the following three non VLIW mechanisms: fully interlocked register files; all computation and memory access instructions with a regular 2 cycle delay pipeline; and computation block instructions with optional SIMD capability in which a single instruction can be issued two units in parallel.

Interlocking register files (similar to general purpose superscalar processors) support a programming model that is functionally well defined at instruction line boundaries. That is, when the result of a computation or load is not available at the next instruction cycle, the processor stalls until that data becomes available. This contrasts with the VLIW model in which the compiler directly controls all aspects of scheduling, including instruction timing (that is

scheduled an operation before all its source operands are available results in wrong data used as input).

The Tiger SHARC architecture implements a model where the program specifies only, but the hardware dynamically resolves instruction timing.(There are also DSPs that are fully dynamically scheduled in which processor determines the instruction level parallelism at run time with some compiler assistance.).

The following are the benefits of interlocked superscalar like programming model:

- The processor supports a fully interruptible system, which is required in embedded real time processing, and significantly simplifies the implementation of interrupt system. A processor with no interlocks may be unable to interrupt the pipeline and holds all data that is in flight, as more than one data may be targeted to a register.
- The architecture supports code compatibility across different processor implementations without the need for recompilation, which conventional VLIW processors are unable to support.
- The architecture simplifies and enhances programmability.
- There are no wasted instruction slots due to vertical and horizontal no-ops, which in conjunction with SIMD results in high code density.

An additional aspect of the programming model related to code scheduling is that all computation block instructions, have a pipeline delay of exactly two cycles.(That is, the results of an instruction that executes at cycle ' i ' are always available at cycle 'i+2')

4.4 TYPES OF PARALLELISM

Tiger SHARC has three distinct forms of data parallelism:

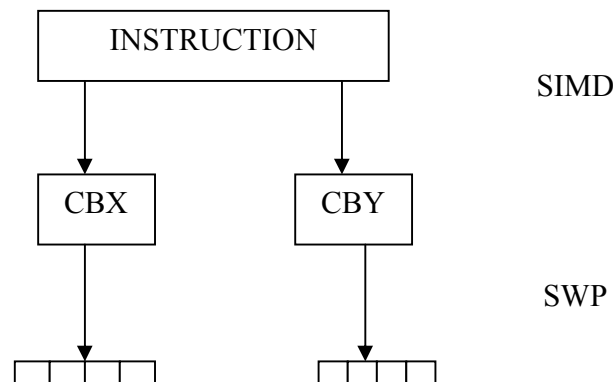
- SIMD and *subword* parallelism.
- Short vector memory architecture.
- Two cycle deep computational pipeline.

SIMD and SUBWORD Parallelism

This DSP implements SIMD dispatch by optionally issuing one computational instruction to both CBX and CBY computational blocks. All computational instructions are encoded with two bits that that determines whether the target computational box is CBX or CBY.

Subword parallelism is another distinct architectural technique. It increases the parallelism at the data element level by means of partitioning a processors data path and performing more than one parallel computation on a single word. This technique is also referred to as multimedia extensions or packed operations.

Subword parallelism is specialized form of SIMD. The Tiger SHARC makes use of both techniques, but in different situations. Generally computation at the 32 bit level is organized by SIMD only. However, computation at the 16 bit and the 8 bit is organized by SIMD and subword parallelism.



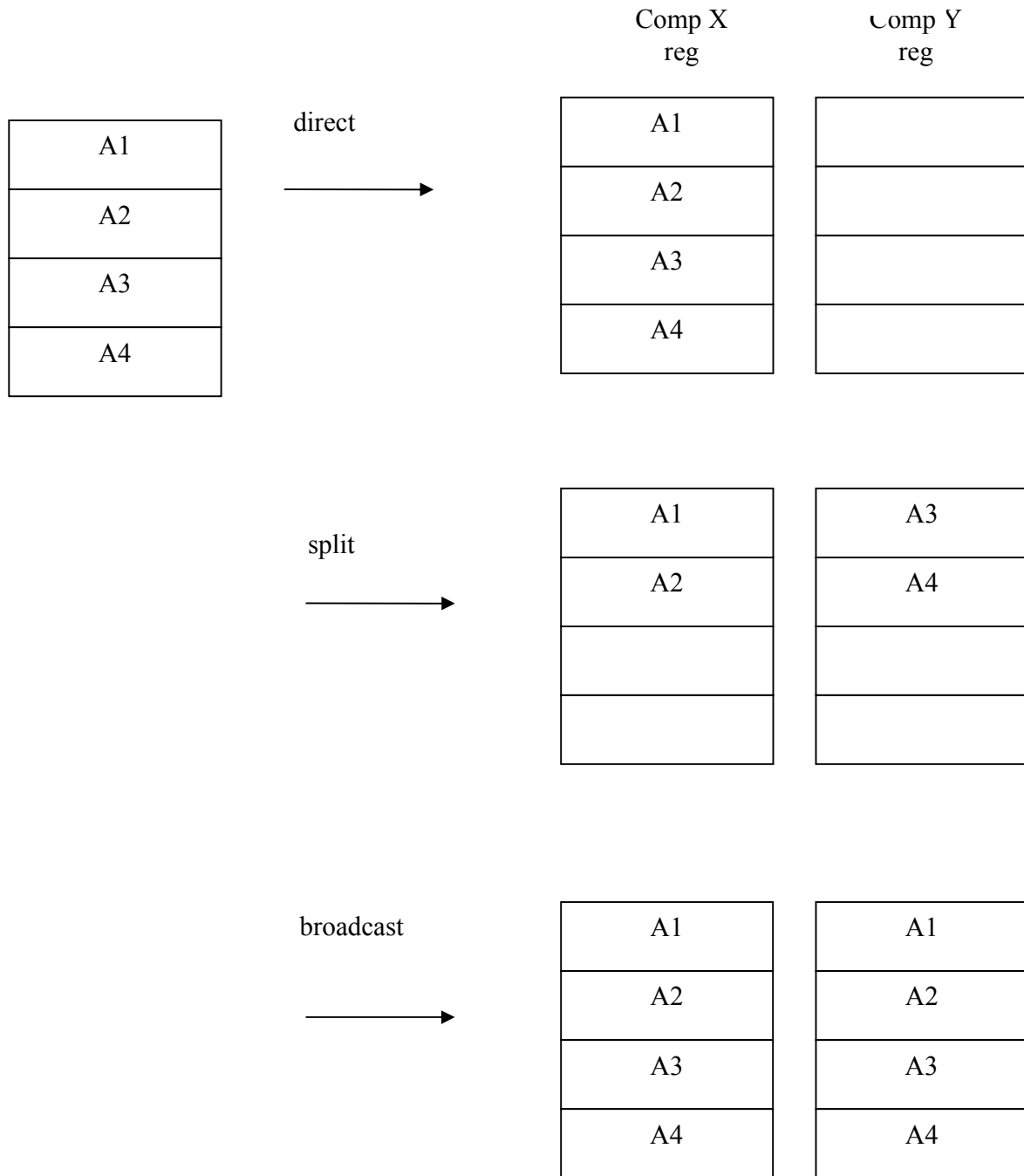
The figure shows conceptually the two way SIMD execution and four way subword parallelism.

SHORT – VECTOR MEMORY

To sustain high core computation rates, the memory architecture handles transactions that carry several words of data per access (Short Vector access). A memory transaction can carry from one to four words of consecutive data, with up to two simultaneous transactions per cycle. A memory access can move data from any of the three internal memory banks or to the memory blocks. The architecture supports three types of memory access. They are direct, split and broadcast.

In direct access the words fetched from the memory are directly loaded into registers of a single computational block. In the split type half of the words are loaded into registers of a one block and the other half into the second computational block. In the third type all the words are loaded into registers of both computational blocks.

DIRECT, SPLIT AND BROADCAST MEMORY ACCESS



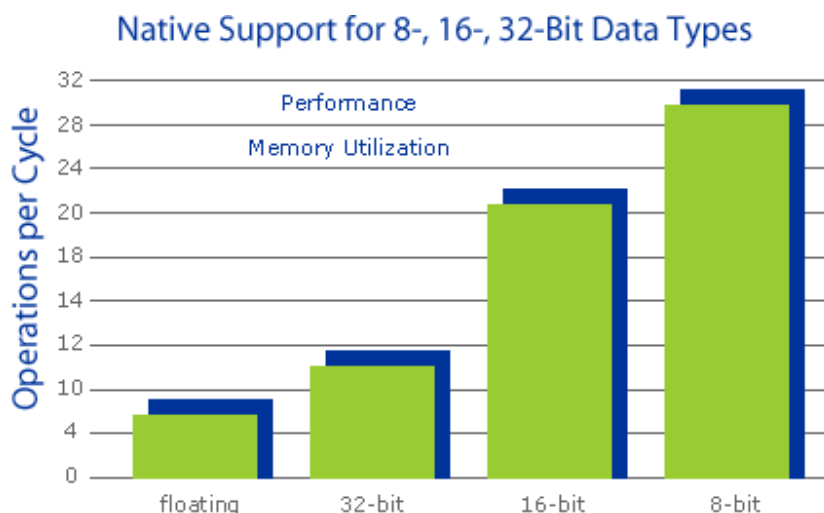
SUPERSCALAR STRUCTURE

This is not a type of parallelism but is also helpful in improving the speed of the processing. Here the instruction level parallelism is determined prior to the runtime by the hardware. So the tedious job of scheduling the instructions is reduced.

4.5 ARITHMETIC CAPABILITY

In addition to single and extended precision floating point support, the instruction set directly supports all 16 bit and 32 bit fixed point DSP, image and video data formats. They include fractional, integer, signed and unsigned data types. All fixed point formats support for saturation arithmetic.

The following figure shows the performance of TigerSHARC's performance in the case of 8, 16, and 32 bit floating as well as fixed point capability.



5. TYPICAL APPLICATIONS

3G Basestation

The capacity of communications applications to absorb processing bandwidth is enormous. Applications like xDSL, RAS modem servers and 3G base stations measure their performance requirements in GFLOPS. FPGAs and/or DSPs are the only way to succeed within sensible size and power consumption constraints. The continually evolving standards for mobile communications together with short project timescales make designing a new system difficult. Base station designers are also attracted to the TigerSHARC DSP by the presence of special assembler instructions that simplify the implementation of certain decoding and despreading algorithms.

RADAR

The next generations of military radars require large processing arrays to provide the capability to discern increasingly stealthy aircraft. Previous systems used arrays of SHARC DSPs, but bottlenecks in passing data between processors over link ports caused problems in larger systems. The TigerSHARC DSP has much higher bandwidth interprocessor links (4 x 200 Mbytes/s instead of 6 x 40 Mbytes/s) that allow large 2-D processor arrays to be built. The links can also interface the external I/O into the processor. This permits the TigerSHARC DSPs to concentrate on processing the data and sharing it with other TigerSHARC DSPs via the cluster bus.

Professional Audio

Professional audio equipment is moving into the digital arena but requires high data rates and long bit widths. Digital data (24 or 32 bits wide) may

arrive on each channel at up to 192 KHz, corresponding to almost 800 Kbytes/s. A large mixing console may have 128 incoming channels, and at least 80 output buses of mixed data. In addition, there will be monitoring outputs and other sundry signals, possibly adding up to 400 streams of input and output data. The incoming data on each channel must be preprocessed (filtered, equalized, etc.) before being mixed with the other incoming channels to create the outputs. The sensitivity of the human ear to time delays means that the incoming data cannot be buffered. Instead, each sample must be processed before the next sample arrives, creating a massive multiply/accumulate requirement. The TigerSHARC DSP is well suited to this sort of application, which demands a high I/O rate and significant processing. The lack of TigerSHARC DSP serial ports is not important as data typically arrives through custom digital channels in such equipment.

Biometrics Applications

In biometric applications like retinal scanning, finger print recognition and voice recognition we need high speed real time processing of data. Here for the sake of high quality results we can use Tiger SHARC.

6. CONCLUSION AND FUTURE SCOPE

The Tiger SHARC architecture solves a number of problems in very high computation by applying parallelism at the data element level and at the instruction level. The implementation of this architecture is intended to support applications that use floating point data types, for example, in radar, imaging, and medical computer graphics. The implementation will also support applications that use 16 bit fixed point data types, primarily telecommunication infrastructures. Tiger SHARC is the most powerful DSP processor up to date.

Analog Devices have already announced the release of the new member of the TigerSHARC family named TS-P36N which is quad SHARC processor capable of performing 9 GFLOPS. Also they are designing an octal processor named TS-V39 capable of performing 12 GFLOPS. The arithmetic capability of TigerSHARC is growing at high speed at the moment.

DSP processor architectures are evolving to meet the changing needs of DSP applications. The architectural homogeneity that prevailed during the first decade of commercial DSP processors has given way to rich diversity. Some of the forces driving the evolution of DSP processors today include the perennial push for increased speed, decreased energy consumption, decreased memory usage, and decreased cost, all of which enable DSP processors to better meet the needs of new and existing applications. Of increasing influence is the need for architectures that facilitate development of more efficient compilers, allowing DSP applications to be written primarily in high-level languages. This has become a focal point in the design of new DSP processors, because DSP applications are growing too large to comfortably implement (and maintain) in assembly language. As the needs of DSP applications continue to change, we expect to see a continuing evolution in DSP processors.