Last revision: 8 / 25 / 03

Programmable Digital Signal Processor (PDSP): A Survey

Surin Kittitornkun and Yu Hen Hu

University of Wisconsin - Madison
Department of Electrical and Computer Engineering
1415 Engineering Drive
Madison, WI 53706
hu@engr.wisc.edu

1. Introduction

Programmable digital signal processors (PDSPs) are general-purpose microprocessors designed specifically for digital signal processing (DSP) applications. They contain special instructions and special architecture supports so as to execute computation-intensive DSP algorithms more efficiently.

PDSPs are designed mainly for embedded DSP applications. As such, the user may never realize the existence of a PDSP in an information appliance. Important applications of PDSPs include modem, hard drive controller, cellular phone data pump, set-top box, etc.

The categorization of PDSPs falls between the general-purpose microprocessor and the custom-designed, dedicated chip set. The former have the advantage of ease of programming and development. However, they often suffer from disappointing performance for DSP applications due to overheads incurred in both the architecture and the instruction set. Dedicated chip sets, on the other hand, lack the flexibility of programming. The time to market delay due to chip development may be longer than the program coding of programmable devices.

1.A A Brief Historical Scan of PDSP Development

1.A.1 1980s to 1990s

A number of PDSPs appeared in the commercial market in the early 1980s. Intel introduced the Intel2920 around 1980 featuring on-chip A/D (analog-to-digital) and D/A (digital-to-analog) converters. Nonetheless, it had no hardware multiplier and was difficult to load program parameters into the chip due to the lack of digital interface. In nearly the same time, NEC introduced the NEC MPD7720. It is equipped with a hardware multiplier, and is among the first to adopt the Harvard architecture with physically separate on-chip data memory and program memory. Texas Instrument introduced the TMS320C10 in 1982. Similar to the MPD7720, the 'C10 adopts the Harvard architecture and has a hardware multiplier. Furthermore, the 'C10 is the first PDSP that can execute instructions from off-chip program memory without performance penalty due to off-chip memory input/output (I/O). This feature brought PDSPs closer to the microprocessor/microcontroller programming model. In addition, the emphasis on development tools and libraries by Texas Instrument led to wide spread applications of PDSP. The architectural features of several representative examples of these early PDSP chips are summarized in Table 1.

Table 1. Summary of characteristics of early PDSP

Model	Manu-facturer	Year	On-chip Data RAM	ON-chip Data ROM	On-chip program RAM	Multiplier
A100	Inmos		---------	---------	---------	4,8,12,16
ADSP2100	Analog Device	1986	---------	---------	---------	16x16→32
DSP16	AT&T		512x16	2Kx16	---------	16x16→32
DSP32	AT&T	1984	1Kx32	512x32	---------	32x32→40
DSP32C	AT&T	1988	1Kx32	2Kx32	---------	32x32→40
DSP56000	Motorola	1986	512x24	512x24	512x24	24x24→56
DSP96001	Motorola	1988	1Kx32	1Kx32	544x32	32x32→96
DSSP-VLSI	NTT	1986	512x18	---------	4Kx18	(18-bit) 12E6
Intel2920	Intel	1980	40x25	---------	192x24	------
LM32900	National.		---------	---------	---------	16x16->32
MPD7720	NEC	1981	128x16	512x13	512x23	16x16→31
MSM 6992	OKI.	1986	256x32	---------	1Kx32	(22-bit) 16E6
MSP32	Mitsubishi		256x16	---------	1Kx16	32x16->32
MB8764	Fujitsu		256x16	---------	1Kx24	
NEC77230	NEC	1986	1Kx32	1Kx32	2Kx32	24E8→47E8
TS68930	Thomson		256x16	512x16	1Kx32	16x16->32

TMS32010	TI	1982	144x16	---------	1.5Kx16	16x16→32
TMS320C25	TI	1986	288x16	---------	4Kx16	16x16→32
TMS320C30	TI	1988	2Kx32	---------	4Kx32	32x32→32E8
ZR34161 VSP	Zoran		128x32	1Kx16	---------	16-bit vector eng.

In these early PDSPs, DSP-specific instructions such as MAC (multiply-and-accumulate), DELAY (delay elements), REPEAT (loop control), and others flow control instructions are devised and included in the instruction set so as to improve both programmability and performance. Moreover, special address generator units with bit-reversal addressing mode support have been incorporated to enable efficient execution of the fast Fourier transform (FFT) algorithm. Due to limitation of chip area and transistor count, the on-chip data and program memories are quite small in these chips. If the program cannot fit into the on-chip memory, significant performance penalty will incur.

Later, floating-point PDSPs, such as TMS320C30, Motorola DSP96001 appeared in the market. With fixed-point arithmetic as in early PDSPs, the dynamic range of the intermediate results must be carefully monitored to prevent overflow. Some reports estimated that as much as one third of the instruction cycles in executing PDSP programs are wasted on checking the overflow condition of intermediate results. A key advantage of a floating-point arithmetic unit is its extensive dynamic range. Later on, some PDSPs also included on-chip DMA (direct memory access) controllers, as well as dedicated DMA bus that allowed concurrent data I/O at the DMA unit, and signal processing computation in the CPU.

1.A.2 1990s to 2000

In the this decade, a few trends of PDSPs emerged:

a) Consolidation of PDSP market

Unlike 1980s where numerous PDSP architectures have been developed, the 1990s is remarked by a consolidation of the PDSP market. Only very few PDSPs are now available in the market. Notably, Texas Instrument's TMS320Cxx series captured about 70% of the PDSP market toward the end of this decade. Within this family, the traditional TMS320C10/20 series has evolved into TMS320C50 and has become one of the most popular PDSPs. Within this TMS family, TMS320C30 was introduced in 1988 and its floating-point arithmetic unit has attracted a number of scientific applications. Other members in this family that are introduced in the 1990s include TMS320C40, a multiprocessing PDSP, and TMS320C80, another multiprocessing PDSP designed for multimedia (video) applications. TMS320C54xx and TMS320C6xx are the recent ones in this family. Another low cost PDSP that has emerged, as a popular choice is the Analog device's SHARC processor. These modern PDSP architectures will be surveyed in later sections of this chapter.

b) DSP Core Architecture

As the feature size of digital integrated circuit continues to shrink, more and more transistors can be packed into single chip. As such, it is possible to incorporate peripheral (glue) logics and

supporting logic components into the same chip in addition to the PDSP. This leads to the notion of *System on (a) Chip (SoC)*. In designing a SoC system, an existing PDSP core is incorporated into the overall system design. This design may be represented as a VHDL (Very High-Speed Integrated Circuit Hardware Description Language)/Verilog core, or a netlist format. A PDSP that is used in this fashion is known as a *processor core* or a *DSP core*.

In the 1990s, many existing popular PDSP designs have been converted into DSP cores so that the designers can design new applications using familiar instruction sets, or even existing programs. On the other hand, several new PDSP architectures are being developed and licensed as DSP cores. Examples of these DSP cores, that will be reviewed in section 4, include Carmel, R.E.A.L., StarCore, and V850.

c) Multimedia PDSPs

With the development of international multimedia standards such as JPEG [PeMi93] image compression, MPEG video coding [Mitch97], and MP3 audio, there is an expanding market for low-cost, dedicated multimedia processors. Due to the complexity of these standards, it is difficult to develop a multimedia processor architecture without any programmability. Thus, a family of multimedia enhanced PDSPs, such as MPACT, TriMedia, TMS320C8x, and DDMP have been developed. A key feature of these multimedia PDSPs is that they are equipped with various special multimedia related function units, for instance, the YUV to RGB (color coordinates) converter, the VLC (variable-length code) entropy encoder/decoder, or the motion estimation unit. In addition, they facilitate direct multimedia signal I/O, bypassing the bottleneck of a slow system bus.

d) Native Signal Processing with Multimedia Extension Instructions

By native signal processing (NSP), the signal processing tasks are executed in the general-purpose microprocessor, rather than a separate PDSP coprocessor. As their speed increases, a number of signal processing operations can be performed without additional hardware or dedicated chip sets. In the early 90s, Intel introduced the MMX (MultiMedia eXtension) instruction set to the Pentium series microprocessor. Since modern microprocessors have long internal word length of 32, 64 or even extended 128 bits, several 8-bit or 16-bit multimedia data samples can be packed into a single internal word to facilitate the so called *subword parallelism*. By processing several data samples in parallel in single instruction, better performance can be accomplished while processing especially multimedia streams.

1.A.3 Hardware Programmable Digital Signal Processors - FPGA

An FPGA (Field Programmable Gate Array) is a software configurable hardware device that contains (i) substantial amount of uncommitted combinational logic; (ii) pre-implemented flip-flops; and (iii) programmable interconnections between the combinational logic, flip-flops, and the chip I/O pins. The downloaded configuration bit stream programs all the functions of the combinational logic, flip-flops, and the interconnections. While not most efficient, an FPGA can be used to accelerate DSP applications in several different ways [Knapp95]:

1. An FPGA can be used to implement a complete application-specific integrated circuit (ASIC) DSP system. A shortcoming of this approach is that current FPGA technology does not yield most efficient hardware implementation. However, FPGA

implementation has several key advantages: (a) time to market is short, (b) upgrade to new architecture is relatively easy, and (c) low volume production is cost-effective.

2. An FPGA can act as a coprocessor to a PDSP to accelerate certain specific DSP functions that cannot be efficiently implemented using conventional architecture.
3. Furthermore, an FPGA can be used as a rapid prototyping system to validate the design of an ASIC and to facilitate efficient, hardware-in-the-loop debugging.

1.B Common Characteristics of DSP Computation

1.B.1 Real-time computation

PDSPs are often used to implement real-time applications. For example, in a cellular phone, the speed of speech coding must match that of normal conversation. A typical real-time signal processing application has three special characteristics:

(a) The computation cannot be initiated until the input signal samples are received. Hence the result cannot be pre-computed and stored for later use.

(b) Results must be obtained before a pre-specified deadline. If the deadline is violated, the quality of services will be dramatically degraded and even render the application useless.

(c) The program execution often continues for an indefinite duration of time. Hence the total number of mathematical operations need to be performed per unit time, known as *throughput*, becomes an important performance indicator.

1.B.2 Data flow dominant computation

DSP applications involve stream media data types. Thus, instead of supporting complex control flow (e.g. context switch, multithread processing), a PDSP should be designed to streamline data flow manipulation. For example, special hardware must be designed to facilitate efficient input and output of data from PDSP to off-chip memory, to reduce overhead involved in accessing arrays of data in various fashions, and to reduce overhead involved in execution of multilevel nested DO loops.

1.B.3 Specialize arithmetic computation

DSP applications often require special types of arithmetic operations to make computation more efficient. For example, a convolution operation

$$y(n) = \sum_{k=0}^{K-1} x(k)h(n-k)$$

can be realized using a recursion

$$y(n) = 0;\ \ y(n) = y(n) + x(k)*h(n-k),\ k = 0, 1, 2, \ldots, K-1$$

For each k, a multiplication and an addition (accumulation) are to be performed. This leads to the implementation of MAC instruction in many modern PDSPs:

$$R4 \leftarrow R1 + R2 * R3$$

Modern PDSPs often contain hardware support of the so-called *saturation arithmetic*. In saturation arithmetic, if the result of computation exceeds the dynamic range, it is clamped to either to the maximum or the minimum value. That is, $9 + 9 = 15$ ($01001_2 + 01001_2 = 01111_2$) in 2's complement arithmetic. Therefore, for applications that saturation arithmetic is applicable, there will be no need to check for overflow during the execution. These special instructions are also implemented in hardware. For example, to implement a saturation addition function using 2's complement arithmetic without intrinsic function support, we have the following C code segment:

```
int sadd(int a, int b) {
int result;
       result = a + b;
       if (((a ^ b) & 0x80000000) == 0) {
              if ((result ^ a) & 0x80000000) {
                     result = (a < 0) ? 0x80000000 : 0x7fffffff;
              }
       }
       return (result);
}
```

However, with a special _sadd intrinsic function support in TMS320C6x[Texas98c], the same code segment reduces to this single line:

```
result = _sadd(a,b);
```

1.B.4 Execution control

Many DSP algorithms can be formulated as nested, indefinite Do loops. In order to reduce the overhead incurred in executing multilevel nested loops, a number of special hardware supports are included in PDSPs to streamline the control flow of execution.

A. Zero-overhead hardware loop – A number of PDSPs contain a special REPEAT instruction to support efficient execution of multiple loop nests using dedicated counters to keep track of loop indices.

B. Explicit instruction level parallelism (ILP) – Due to the deterministic data flow of many DSP algorithms, ILP can be exploited at compile-time by an optimizing compiler. The led to several modern PDSPs to adopt the very long instruction word (VLIW) architecture to efficiently utilize the available ILP.

1.B.5 Low power Operation and Embedded System Design

A. The majority of applications of PDSPs are embedded systems, such as disk drive controller, modem, cellular phone. Thus, many PDSPs are highly integrated and often contain multiple data I/O function units, timers, and other function units in a single chip packaging.

B. Power consumption is a key concern in the implementation of embedded systems. Thus, PDSPs are often designed to compromise between conflicting requirements of high-speed data computation, and low power consumption [Bork99]. The specialization of certain key functions allows efficient execution of the desired operations using high degree of parallelism while holding down the power source voltage and overall clock frequency to conserve energy.

1.C Common Features of PDSPs

1.C.1 Harvard Architecture

A key feature of PDSPs is the adoption of a *Harvard memory architecture* that contains separate program and data memory so as to allow simultaneous instruction fetch and dada access. This is different from the conventional *Von Neuman* architecture where program and data are stored in the same memory space.

1.C.2 Dedicated Address generator

Address generator allows rapid access of data with complex data arrangement without interfering the pipelined execution of main ALUs (arithmetic and logic units). This is useful for situations such as two-dimensional (2D) digital filtering, and motion estimation. Some address generators may include bit-reversal address calculation to support efficient implementation of FFT, and circular buffer addressing for the implementation of infinite impulse response (IIR) digital filters.

1.C.3 High bandwidth Memory and I/O controller

To meet the intensive input and output demands of most signal processing applications, several PDSPs have built-in multichannel DMA channels and dedicated DMA buses to handle data I/O without interfering CPU operations. To maximize data I/O efficiency, some modern PDSPs even include dedicated video and audio codec (coder/decoder) as well as high-speed serial/parallel communication port.

1.C.4 Data Parallelism

A number of important DSP applications exhibit high degree of data parallelism that can be exploited to accelerate the computation. As a result, several parallel processing schemes, SIMD (Single Instruction Multiple Data) and MIMD (Multiple Instruction Multiple Data) architecture have incorporated in the PDSP. For example, many multimedia enhanced instruction sets in general-purpose microprocessors (e.g. MMX) employed subword parallelism to speed-up the execution. It is basically a SIMD approach. A number of PDSPs also facilitate MIMD implementation by providing multiple interprocessor communication links.

2. Applications of PDSP

In this section, both real-world and prototyping applications of PDSPs are surveyed. These applications are divided into three categories: communication systems, multimedia, and control/data acquisitions.

2.A Communications systems

PDSPs have been applied to implement various communication systems. Examples include Caller ID (using TMS320C2xx [TexasE97]), cordless handset, and many others. For voice communication, an acoustic-echo cancellation based on the normalized least mean square (NLMS) algorithm for hands-free wireless system is reported in [Texas97]. Implemented with a TMS320C54, this system performs both active-channel and double-talk detection. A 40-MHz

TMS320C50 fixed-point processor is used to implement a low bit rate (1.4 Kbps), real-time vocoder (voice coder) [YLZ98]. The realization also includes both the decoder and the synthesizer. A telephone voice dialer [PaRo96] is implemented with a 16-bit fixed-point TMS320C5x PDSP. It is a speaker-independent speech recognition system based on the hidden Markov model algorithm.

Modern PDSPs are also suitable for error correction in digital communication. A special Viterbi Shift Left (VSL) instruction is implemented on both the Motorola DSP56300 and the DSP56600 PDSPs [Taipa98] to accelerate the Viterbi decoding. Another implementation of the ITU V.32bis Viterbi decoding algorithm using a TMS320C62xx is reported by [Yiu98]. Yet another example is the implementation of the U.S. digital cellular error-correction coding algorithm, including both the tasks of source coding/decoding and ciphering/deciphering, on a TMS320C541 evaluation module [Chish94].

Digital baseband signal processing is another important application of PDSPs. A TMS320C25 DSP-based GMSK (Gaussian Minimum Shift Keying) modem for Mobitex packet radio data communication is reported in [Resw96]. In this implementation, transmitted data in packet form is level-shifted and Gaussian-filtered digitally within the modem algorithm so that it is ready for transmitter baseband interface, either via D/A converter or by direct digital modulation. Received data at either baseband or intermediate frequency (IF) band from the radio receiver is digitized and processed. Packet synchronization is also handled by the modem, assuring that the next layer sees only valid Mobitex packets.

System prototyping can be accomplished using PDSP due to its low cost and ease ofprogramming. A prototype of reverse channel transmitter/receiver for asymmetric digital subscriber line (ADSL) algorithm [Gottl94] is implemented using a floating-point DSP TMS320C40 chip clocked at 40 MHz. The program consisted of three parts: synchronization, training, and decision directed detection.

Navigation using the Global Positioning System (GPS) has been widely accepted for commercial applications such as electronic direction finding. A software-based GPS receiver architecture using TMS320C30 processor is described in [KSJ96]. The ‘C30 is in charge of signal processing tasks such as correlation, FFT, digital filtering, decimation, demodulation, and Viterbi decoding in the tracking loop. Further investigation on the benefits of using a PDSP in a GPS receiver with special emphasis on fast acquisition techniques is reported in [DaVi98]. The GPS L1 band signal is down-converted to IF. After A/D conversion, the signal is processed by a dedicated hardware in conjunction with algorithms (software) on a PDSP. Functions that are fixed and require high speed processing should be implemented in dedicated hardware. On the contrary, more sophisticated functions that are less time-sensitive can be implemented using PDSPs.

For defense system application, a linear array of TMS320C30 as the front-end and a Transputer™ processor array as the back-end for programmable radar signal processing are developed to support the PDDR (Point Defense Demonstration Radar) [AEDR91]. The input signal is sampled at 10 MHz to 16-bit, complex-valued samples. The PDSP front-end performs pulse compression, moving target indication (MTI), and constant false alarm (CFA) rate detection.

2.B Multimedia

2.B.1 Audio Signal Processing

The audible signals cover frequency range from 20 to 20,000 Hz. PDSP applications to audio signal processing can be classified into three categories according to the qualities and audible range of the signal [LeTo98]: professional audio products, consumer audio products, and computer audio multimedia systems. The DSP algorithms used in particular products are summarized in the table below.

Table 2. DSP algorithms for audio applications

Professional Audio Products	***DSP Algorithms Used***
Digital Audio Effects Processors *(Reverb, Chorus, Flanging, Vibrato Pitch Shifting, Dyn Ran. Compression¼)*	Delay-Line Modulation/Interpolation, Digital Filtering (Comb, FIR¼)
Digital Mixing Consoles Level Detection, Volume Control	Filtering, Digital Amplitude Panning,
Digital Audio Tape (DAT)	Compression techniques: MPEG,
Electronic Music Keyboards Physical Modeling	Wavetable/FM synthesis, Sample Playback
Graphic and Parametric Equalizers	Digital FIR/IIR filters
Multichannel Digital Audio Recorders	ADPCM, AC-3
Room Equalization	Filtering
Speaker Equalization	Filtering
Consumer Audio Products	***DSP Algorithms Used***
CD-I	ADPCM, AC-3, MPEG
CD Players and Recorders	PCM
Digital Amplifiers/Speakers	Digital Filtering
Digital Audio Broadcasting Equip.	AC-3, MPEG...
Digital Graphic Equalizers	Digital Filtering
Digital Versatile Disk (DVD) Players	AC-3, MPEG...
Home Theater Systems	AC-3, Dolby ProLogic, THX
{Surround-Sound Receivers/Tuners}	DTS, MPEG, Hall/Auditorium Effects
Karaoke	MPEG, audio effects algorithms
Satellite (DBS) Broadcasting	AC-3, MPEG
Satellite Receiver Systems	AC-3,
Computer Audio Multimedia Systems	***DSP* Algorithms used**

Sound card	ADPCM, AC-3, MP3, MIDI, …
Special purpose head sets	3D Positioning (HRTFs),

MP3 (MPEG-I Layer 3 audio coding) has achieved the status of the most popular audio coding algorithm in recent years. The PDSP implementation of MP3 decoder can be found in [RoLu98]. On the other hand, most synthesized sounds such as those used in computer gaming are still represented in the MIDI [YDG98] format. It can be seen that PDSPs are good candidates for the implementation of these audio signal processing algorithms.

2.B.2 Image/Video processing

Existing image and video compression standards such as JPEG, and MPEG are based on the DCT (discrete cosine transforms) algorithm. The upcoming JPEG 2000 image coding standard will also include coding algorithms that are based on the discrete wavelet transform (DWT). These standards are often implemented in modern digital cameras and digital camcorders where PDSPs will play an important role. An example of using the TMS320C549 to implement a digital camera is reported in [IGGL99] where the PDSP can be upgraded later to incorporate the upcoming JPEG 2000 standard.

Low bit rate video coding standards include the ITU H.263+/H.263M and MPEG4 simple profile. In [BRWT99], the potential applications of TMS320C54x family chips to implement low-power low-bit rate video coding algorithms are discussed. On the other hand, decoding of MPEG-II broadcasting grade video sequences using either the TMS320C80 [BMMOP96] chip or the TMS320C6201 [CKITT99] chip has been reported.

Medical imaging has become another fast growing application area of PDSPs. Reported in [CJL97] is the use of TMS320C3x as a controller and on-line data processor for processing magnetic resonance imaging (MRI). It can perform real-time dynamic imaging such as the cardiac imaging, angio-graphy (examination of the blood vessels using x-rays following the injection of a radio-opaque substance), and abdominal imaging. Recently, an implementation of real-time data acquisition, processing, and display of un-gated cardiac movies at moderate video rates of 20 frames per second using PDSPs was reported in [MIEB99].

2.B.3 Printing

Current printer consists of embedded processors to process various formats of page description languages (PDL) such as PostScript. In [GaTh99], a PDSP is used to interpret the PDL code, and to create a list of elements to be displayed, and to estimate the time needed to render the image. Rendering is the process of creating the source pixel map. In this process, a common source map is 600x600 pixels per square inch, with four colors for each pixel, and eight bits for each color. Compression is used to store the output map while rendering and screening cannot be completed within the real-time requirement. This phase involves JPEG compression and matrix transformations and interpolations. Depending on the characteristics of the screened image and the storage memory available, the compressed image may be either lossless or lossy. Decompression of the bit-mapped image occurs in real-time as the compressed image is fed to the print engine. The screening process converts the source pixel map into the appropriate

output format. Since the process must be repeated for all pixels, the number of calculations is enormous for a high-resolution color image especially in real-time.

2.B.4 SAR Image Processing

Synthetic aperture radar (SAR) signal processing possesses a significant challenge due to its very large computation and data storage requirements. A sensor transmits pulses and receives the echoes in a direction approximately perpendicular to the direction of travel. The problem becomes 2D space-variant convolution using range-Doppler algorithm where all the signals and coefficients are complex numbers with a precision of at least 16 bits. A heterogeneous architecture, vector/scalar architecture is proposed and analyzed [MIC96]. The vector processor (using Sharp LH9124 for FFTs) and the scalar processing unit (using 8 SHARC 21060's connected in a mesh network) are chosen based on performance, scalability, flexibility, development cost and repeat cost evaluation criterion. The design is capable of processing SAR data at about 1/10 of the real-time rate.

2.B.5 Biometric Information Processing

Handwritten signature verification, one of the biometric authentication techniques is cheap, reliable and, non-intrusive to the person being authorized. A DSP Kernel for online verification using the TMS32010 with a 200Hz-sampling rate is developed [DDNSZ95]. The authentication kernel comprises of personalized table and some general-purpose procedures. This verification method can be part of a variety of entrance monitoring and security systems.

2.C Control and Data acquisition

As expected, PDSP has found numerous applications in modern control and data acquisition applications as well. Several control applications are implemented using Motorola DSP56000 PDSPs that function as both powerful microcontrollers and as fast digital signal processors. Its 56-bit accumulator (hence the code name 56xxx) provides 8-bit extension registers in conjunction with saturation arithmetic to allow 256 successive consecutive additions without the need to check for overflow condition or limit cycles. The output noise power due to round-off noise of the 24-bit DSP56000/DSP56001 is 65,536 times less than that for 16-bit PDSPs and microcontrollers. Design examples include a PID (proportion, integration, and derivative) controller [StSo], and an adaptive controller [Rena].

Another example of DSP system development is the Computer Assisted Dynamic Data Monitoring and Analysis System (CADDMAS) project developed for the U.S. Airforce and NASA [SKB98]. It is applied to turbine engine stress testing and analysis. The project makes use of TMS320C40 for distributed-memory parallel PDSP. An application-specific topology interconnects 30 different systems with processor counts varying from 4 to 128 processors. More than 300 sensors are used to measure signals with sampling rate in excess of 100 KHz. Based on measured signals, the system performs spectral analysis, auto- and cross-correlation, tri-coherence, etc.

2.D DSP Applications of Hardware Programmable PDSP

There are a variety of FPGA implementation examples of specific DSP functions such as: FIR (finite impulse response) digital filter [Gosli95], DFT/FFT (Discrete Fourier Transform/Fast

Fourier Transform) processor [Dick96], image/video processing [SVMJ95], wireless CDMA (Code Division Multiple Access) rake receiver [ShLe2000], and Viterbi decoding [Gosli95].

16-Tap FIR digital filter – A distributed arithmetic (DA) implementation of a 16-tap finite impulse response digital filter has been reported in [Gosli95]. The DA implementation of the multiplier uses look up tables (LUTs). Since the product of two n-bit integers will have 2^{2n} different results, the size of the LUT increases exponentially with respect to the word length. For practical implementation, compromises must be made to trade additional computation time for smaller number of LUTs.

CORDIC based radar processor – The improvement of FPGA CORDIC arithmetic implementation is studied further in [Andr98]. The iteration process of CORDIC can be unrolled so that each processing element always performs the same iteration. Unrolling the processor results in two significant simplifications. First, shift amounts become fixed and can be implemented in the wiring. Second, constant values for the angle accumulator are distributed to each adder in the angle accumulator chain and can be hardwired instead of requiring storage space. The entire processor is reduced to an array of interconnected adder-subtractors, which is strictly combinatorial. However, the delay through the resulting circuit can be substantial but can be shortened using pipelining without additional hardware cost. A 14-bit, 5-iteration pipelined CORDIC processor that fits in half of an Xilinx XC4013E-2 runs at 52 MHz. This design is used for high throughput polar to Cartesian coordinate transformations in a radar target generator.

DFT/FFT – An FPGA based systolic DFT array processor architecture is reported [Dick96]. Each processing element (PE) contains a CORDIC arithmetic unit, which consists of a series of shift and add to avoid the requirement for area consuming multiplier. The timing analyzer *xdelay* determines the maximum clock frequency to be 15.3 MHz implemented on a Xilinx XC4010 PG191-4 FPGA chip.

Image/video signal processor – In [SVMJ95], the implementation of an FPGA-augmented low-complexity, video signal processor was reported. This combination of ASIC and FPGA is flexible enough to implement four common algorithms in real-time. Specifically, for 256×256×8 pictures, this device is able to achieve the following frame rates:

Table 3. Achievable frame rate of four different image processing operations

Algorithms	***Frames/sec***	***Latency (ms)***
7 x 7 Mask 2D Filter	13.3	75.2
8 x 8 Block DCT	55.0	18.2
4 x 4 Block Vector Quantization at 1/2 bpp	7.4	139.0
One level wavelet transform	35.7	28.0

CDMA rake receiver – A CDMA rake receiver for a real-time underwater data communication system has been implemented using four Xilinx XC4010 FPGA chips [ShLe 2000] with one multiplier on each chip. The final design of each multiplier occupies close to 1000 CLBs (configurable logic blocks) and is running at a clock frequency of 1 MHz.

Viterbi Decoder – Viterbi decoding is used to achieve maximum likelihood decoding of a binary stream of symbols. Since it involves bit-stream operations, it cannot be efficiently implemented using word-parallel architecture of general-purpose microprocessors or PDSPs. It has been reported [Gosli95] that a Xilinx XC4013E-based FPGA implementation of a Viterbi decoder achieves 2.5 times processing speed (135 ns versus 360 ns) compared to a dual PDSP implementation of the same algorithm.

3. Performance measures

Comparison of the performance between PDSP and general-purpose microprocessors, between different PDSPs, as well as between PDSP and dedicated hardware chip sets is a very difficult task. A number of factors contribute to this difficulty:

1. *A set of objective performance metrics is difficult to define for PDSPs.* It is well known that with modern superscalar instruction architecture, the usual metrics such as MIPS (millions instruction per section), FLOPS (floating-point operations per second) are no longer valid metrics to gauge the performance of these microprocessors. Some PDSPs also adopt such architecture. Hence a set of appropriate metrics is difficult to define.
2. *PDSPs have fragmented architecture.* Unlike general-purpose microprocessors that have converged largely to similar data format (32 bits, or 64 bits architecture), PDSPs have much more fragmented architecture in terms of internal or external data format and fixed-point versus floating-point operations. The external memory interface is varied on platform by platform basis. This is due to the fact that most PDSPs are designed for embedded applications, and hence cross-platform compatibility is not of a major concern between different manufacturers of PDSPs. Furthermore, PDSPs often have specialized hardware to accelerate a special type of operations. Such specialized hardware makes the comparison even more difficult.
3. *PDSP applications are often hand-programmed with respect to a particular platform*. The performance of cross-platform compilers is still far from realistic. Hence, it is not meaningful to run the same high-level language benchmark program on different PDSP platforms.

Some physical parameters of PDSPs are summarized in the following table.

Table 4. Physical performance parameters

Parameters	***Units***
Maximum clock frequency	MHz
Power consumption	Absolute power, watts (W) , power(W)/MIPS
Execution throughput, both peak and sustained	MIPS, MOPS (million operations/sec), MACS (#MAC/sec), MFLOPS
Operation latency	Instruction cycles
Memory access	Clock cycles
– Bandwidth	MB/s (Megabytes per second)

– Latency	Clock cycle
– Input/Output	No. of ports

Usually peak MIPS, MOPS, MFLOPS, MAC/s, MB/s for particular architecture are just the product of instructions, operations, floating-point operations, multiply-accumulate operations, memory access in bytes executed in parallel multiplied by maximum clock frequency respectively. They can be achieved instantaneously in real applications at certain clock cycle and somehow misleading. From a user's perspective, the ultimate performance measure is the "execution time" (wall clock time) of individual benchmark.

Recently, efforts have been made to establish benchmark suites for PDSPs. The proposed benchmark suites [BJER98, LPM97] can be categorized into kernel and application levels. They can be classified into general DSP and multimedia/graphic. Since each kernel contributes to run time of each application at some certain percentage of run time and each application may contain more than single DSP kernel, conducting benchmark tests at both levels gives more accurate results than just raw number of some DSP kernels. A number of DSP benchmarks are summarized below.

Table 5. Examples of DSP Benchmarks

Level	***Algorithm/Application Names***
Kernel	
• General	FFT/IFFT, FIR, IIR, matrix/vector multiply, Viterbi decoding, LMS (least mean square) algorithm
• Multimedia/graphic	DCT/IDCT, VLC (variable length code) decoding, SAD
Application	
• General	Radar [BJER98]
• Multimedia/graphic	MediaBench [LPM97], G.722, JPEG, Image [BJER98]

4. Modern PDSP Architectures

In this section, several modern PDSP architectures will be surveyed. Based on different implementation methods, modern PDSPs can be characterized into PDSP chip, PDSP core, multimedia PDSPs and NSP instruction set. The following aspects of these implementation approaches are summarized in terms of three general sets of characteristics:

1) program (instruction) execution,

2) datapath, and

3) physical implementation.

Program execution of PDSP is characterized by processing core (how PDSP achieves parallelism), instruction width (bits), maximum number of instructions issued, address space of program memory (bits). Its datapath is concerned with number and bit width of datapath, pipelining depth, native data type either fixed-point or floating-point, number of ALUs, shifters, multipliers, and bit manipulation units as well as their corresponding data precision/accuracy, and data/address registers. Finally, physical characteristics to be compared include maximum clock frequency, typical operating voltage, feature size and implementation technology, and power consumption.

4.A PDSP Chips

Some of the recent single-chip PDSPs are summarized in the table 6 below:

Table 6 Summary of recent single-chip PDSPs

Family Name		DSP16xxx	SHARC	TMS32054xx	TMS32062xx	TMS32067xx	TriCore
Model number		DSP16210	ADSP21160	TMS320VC5421	TMS320C 6203	TMS320C 6701	TC10GP
Company		Lucent	Analog Device	Texas	Texas	Texas	Infineon
Processing core		VLIW	Multiproc./SIMD	Multiprocessor	VLIW	VLIW	Superscalar
Instruction	Width (bits)	16 & 32	32	16 & 32	256	256	16 & 32
	Maximum Issued	1	4	2	8	8	2
	Address Space (bits)	20	32	-	32	32	32
Datapath	Number of Datapath	2	2	2	2	2	3
	Width of Datapath (bits)	16	32	16	32	32	32
	Pipeline Depth	3	3	-	11	17	4
	Data Type	Fixed-point	Floating-point	Fixed-point	Fixed-point	Floating-point	Fixed-point
Functional Units	ALUs	2 (40b)	2	2(40b)	4	4	1
	Shifters	2	2	2(40b)	2	0	-
	Multipliers	2 (16bx16b)	2	2(17bx17b)	2	2	2 (16bx16b)
	Address Generator	2	2	2x2	ALUs	ALUs	1
	Bit Manipulation Unit	1 (40b)	Shifter	2(40b)	Shifter	Shifter	1
Program Control	Hardware loop	Y	Y	Y	N	N	Y
	Nesting levels	2	-	2	2	2	3
On-chip Storage	Data Registers	8	2x16	2x2	2x16	2x16	16
	Width (bits)	40	40	40	32	32	32
	Address Registers	21	2x8	2x8	-	-	16
	Width (bits)	20	32	-	-	-	32
Performance	Maximum Clock (MHz)	150	100	-	300	167	66
	Operating Voltage (Volts)	3.0	-	1.8	1.5	1.8	2.5
	Technology	CMOS	-	CMOS	CMOS(15C05)	CMOS(18C05)	CMOS
	Feature Size (micron)	-	-	-	0.15	0.18	0.35
	Power Consumption	294mW@100MHz	-	162mW@100MHz	-	-	-

4.A.1 DSP16xxx [Bier97]

The Lucent DSP16xxx achieves ILP from parallel operations encoded in a complex instruction. These complex instructions are executed at a maximum rate of one instruction per clock cycle. Overheads due to small loops can be eliminated by embedding up to 31 instructions following the DO instruction. These embedded instructions can be repeated a specified number of times without additional overhead. Moreover, high instruction/data I/O bandwidth can be achieved from a 60-Kword (120-Kbyte) dual-ported on-chip RAM, a dedicated data bus, and a multiplexed data/instruction bus.

4.A.2 TMS320C54xx [Texas99]

Characterized as a low power PDSP, each TMS320C54xx chip is composed of two independent processor cores. Each core has a 40-bit ALU including a 40-bit barrel-shifter and two 40-bit accumulators, a 17-bit × 17-bit parallel multiplier coupled with a 40-bit adder to facilitate single-cycle MAC operation. The C54 series is optimized for low-power communication applications. Therefore, it is equipped with a compare, select, and store unit (CSSU) for the add/compare selection of the Viterbi operator. Loop/branch overhead is eliminated using instructions such as `repeat`, `block-repeat`, and `conditional store`. Interprocessor communication is carried out via two internal 8-element first-in-first out (FIFO) register.

4.A.3 TMS320C62x/C67x [Texas99a, Texas99b, Sesh98]

TMS320C62x/C67x is a series of fixed-point/floating-point, VLIW-based PDSPs for high performance applications. During each clock cycle, a *compact instruction* is fetched and decoded (decompressed) to yield a packet of 8 32-bit instructions that resemble those of conventional VLIW architecture. Compiler performs software pipelining, loop unrolling, “If” conversion to a predicate execution. Furthermore, a number of special-purpose DSP instructions called *intrinsic functions* can be accessed by programmers from high level language such as C. This feature helps ease the programming task and improve the code performance.

4.A.4 ADSP 21160 SHARC [Analog99]

Analog Device’s ADSP21160 SHARC (Super Harvard ARChitecture) contains two PEs, both using a 40-bit extended precision floating-point format. Every functional units in each PE is connected in parallel, and perform single-cycle operations. Even though its name is abbreviated from Harvard architecture, its program memory can store both instruction and data. Furthermore, SHARC doubles its data memory bandwidth to allow simultaneous fetch of both operands.

4.A.5 TriCore [TriCore99]

TriCore TC10GP is a dual-issued superscalar load/store architecture targeted at control oriented/DSP oriented applications. Even though its instructions are mixed 16/32 bits wide for low code density, its datapath is 32 bits wide to accommodate high-precision fixed-point and single-precision floating-point numbers.

4.B PDSP CORES

In the table below, we compare the features of four DSP cores reported in the literature:

Table 7 Summary of PDSP cores

Family Name		Carmel	R.E.A.L.	StarCore	V850
Model number		DSP 10XX	-	SC140	NA853C
Company		Infineon	Philips	Lucent & Motorola	NEC
Processing core		VLIW	VLIW	VLIW	RISC
Instruction	Width (bits)	24 & 48	16 & 32	16	16 & 32
	Maximum Issued	1/2	2	6	-
	Address Space (bits)	23	-	32	26
Datapath	Number of Datapath	2	2	-	-
	Width of Datapath (bits)	16	16	16	16
	Data Type	Fixed-point	Fixed-point	Fixed-point	Fixed-point
Functional Units	ALUs	2 (40b)	4 (16b)	4	1 (32b)
	Shifters	1(40)	1 (40b)	ALU (40b)	1 (32b)
	Multipliers	2 (17bx17b)	2 (16x16b)	ALU (16bx16b)	1 (32bx32b)
	Address Generator	1	2	2	-
	Bit Manipulation Unit	shifter	-	ALU	Shifter
On-chip Storage	Data Registers	16+6	8	16	32
	Width (bits)	16/40	16	40	32
	Address Registers	10	16	24	-
	Width (bits)	16	-	32	-
Performance	Maximum Clock (MHz)	120	85	300	33
	Operating Voltage (Volts)	2.5	2.5	1.5	3.3
	Technology	CMOS	-	CMOS	Titanium-Silicide
	Feature Size (micron)	0.25	0.25	0.13	0.35
	Power Consumption	200mW@120MHz	-	180mW@300Mhz	-

4.B.1 Carmel [Carmel99,EyBi98]

One of the distinguishing features of Carmel is its configurable long instruction words (CLIW) that are user-defined VLIW like instructions. Each CLIW instruction combines multiple predefined instructions into a 144-bit long superinstruction as shown below.

```
CLIW name (ma1, ma2, ma3, ma4) {                    // CLIW reference line
      MAC1 || ALU1 || MAC2 || ALU2 || MOV1 || MOV2     // CLIW def
}
```

Programmers can indicate up to four execution units plus two data moves according to the position of individual instruction within the long CLIW instruction. However, up to four memory operands can be specified using `ma1` through `ma4`. The assembler stores 48-bit reference line in program memory and 96-bit definition in a separate CLIW memory (1024 x 96

bits). In addition to CLIW, a specialized hardware is provided to support Viterbi decoding. Nearly all instructions can use predicated execution by two conditional-execution registers

4.B.2 R.E.A.L. [KLMW98]

Similarly R.E.A.L. PDSP core allows users to specify a VLIW-like set of application specific instructions (ASI) to exploit full parallelism of datapath. Up to 256 ASI instructions can be stored in a look-up table. These instructions are activated by a special class of 16 bit instructions with 8-bit index field. Each ASI instruction is 96 bits wide and of the predicated form below.

```
Cond (3) || XACU (11) || YACU (10) || MPY1 (3) || MPY0 (3) || ALUs (62) || DSU (2) || BNU (2)
ASI [if(asi_cc)]alu3_op,alu2_op,alu1_op,alu0_op,mult1_op][,mult0_op][,dr_op][,xacu_op][,yacu_op];
ASI [if(asi_cc)]alu32_op, alu10_op [,mult1_op][,mult0_op][,dr_op][,xacu_op][,yacu_op];
ASI [if(asi_cc)]lfsr [,mult1_op][,mult0_op][,xacu_op][,yacu_op];
```

Each ASI starts with 3-bit condition code followed by 11-bit X ALU opcode, 10-bit Y ALU, 3-bit Multiplier 1 and 0's opcodes, 62-bit operands, etc. In addition to user-defined VLIW instruction, R.E.A.L. allows Application specific eXecution Units (AXU)s to be defined by the customer which can be placed anywhere in the datapath or address calculation units. It is targeted application is a GSM baseband signal processor.

4.B.3 StarCore [Star99, WoBi98]

StarCore is a joint development between Lucent and Motorola for wireless software handset configurable terminals (radios) of the third generation wireless systems. It's expected to operate at low voltage down to 0.9 V. A Fetch set (8-word instruction set) is fetched from memory. Program sequencing (PSEQ) unit detects a portion of this set to be executed in parallel and dispatched to the appropriate execution unit. This feature is called variable length execution set (VLES). StarCore achieves maximum parallelism by allowing multiple address generation and data ALUs to execute multiple operations in a single cycle. StarCore is targeted at speech coding, synthesis and voice recognition.

4.B.4 V850 [NEC97]

The NEC NA853E is a five-stage pipeline RISC (Reduced Instruction Set Computer) core suitable for real-time control applications. Not only is instruction set a mixture of 16 and 32 bits wide but also includes intrinsic instructions for high-level language support to increase the efficiency of object code generated by compiler and to reduce the program size.

4.C Multimedia PDSPs

Multimedia PDSPs are designed specifically for audio/video applications as well as 2D/3D graphics. Some of their common characteristics are

1. Multimedia input/output (I/O): This may include ports and codec (coder/decoder) for video, audio, as well as super VGA graphics signals

2. Multimedia-specific functional units such as YUV to RGB converter for video display, variable length decoder for digital video decoding, de-scrambler in TriMedia [Philips99a] and motion estimation unit for digital video coding/compression in Mpact2 [Kala98, Purc98]
3. High speed host computer/memory interfaces such as PCI bus and RAMBUS DRAM interfaces
4. Real-time kernel or operating system for MPACT and TriMedia respectively
5. Support of floating-point and 2D/3D graphic

Examples of multimedia PDSPs include MPACT, TriMedia, TMS320C8x, and DDMP (Data-Driven Multimedia Processor) [TMI99]. Their architectural features are summarized in the following table.

Table 8 Summary of Multimedia PDSPs

Family Name		DDMP	MPACT	TMS320C8x	TriMedia
Model number		-	MPACT2/6000	TMS320C82	TM1300
Company		Sharp	Chromatic	Texas	Philips
Processing Core		Dataflow	VLIW	Multiprocessor	VLIW
Instruction	Width (bits)	72	81	32	16 to 224
	Maximum Issued	8	2	3	5
	Address Space (bits)	-	-	32	32
Datapath	Width of Datapath (bits)	12	72	32	32
	Floating-Point Precision	NA	Both	Single	Both
Functional Units	ALUs	2	2	3	7
	Shifters	-	1	3	2
	Multipliers	2 ALUs	1	3	2
	Address Generators	4	-	3	2
On-chip Storage	Data Registers	4 Accumulators	512	48	128
	Width (bits)	24	72	32	32
Performance	Maximum Clock (MHz)	120	125	60	166
	Operating Voltage (volts)	2.5	-	-	2.5
	Technology	CMOS(4 metal)	-	CMOS	CMOS
	Feature Size (micron)	0.25	0.35	-	0.25
Power	Consumption	1.2W@120 MHz	-	-	3.5W@166MHz

4.D NSP: Native Signal Processing

NSP is referred as the use of extended instruction sets in a general-purpose microprocessors to process signal processing algorithms. These are special-purpose instructions that often operate in a different manner than the regular instructions. Specifically, multimedia data formats usually are rather short (8 or 16 bits) compared to the 32, 64, 128-bit native register length of modern general-purpose microprocessors. Therefore, up to 8 samples may be packed into a single word and processed simultaneously to enhance parallelism at the subword level. Most NSP instructions operate on both integer (fixed-point) and floating-point numbers except Visual Instruction Set (VIS) [Sun97] which supports only fixed-point number. In general, NSP instructions can be classified into the following categories [Lee96]:

- Vector arithmetic and logical operations whose results may be vector or scalar.
- Conditional execution using masking operations.
- Memory/cache access control such as cache pre-fetch to particular level of cache, non-temporal store, etc. as well as masked load/store
- Data alignment and subword rearrangement i.e. permute, shuffle, etc.

Most NSP instruction set architectures exhibit the following features:

1. NSP instructions may share the existing functional units of regular instructions. As such, some overhead is involved when switching between NSP instructions and regular instructions. However, some NSP instruction sets have separate, exclusive execution units as well as register file.
2. Saturation and/or modulo arithmetic instructions are often implemented in hardware to reduce the overhead of dynamic range checking during execution as illustrated in section 1.B.3.
3. To exploit subword parallelism, manual or human optimization of NSP based programs is often necessary for demanding applications such as image/video processing, and 2D/3D graphics.

Common and distinguishing features of available NSPs are summarized alphabetically as follows.

Table 9. Summary of Native Signal Processing Instruction Sets

Name		AltiVec	MAX-2	MDMX	MMX/3D Now	MMX/SIMD	VIS
Company		Motorola	HP	MIPS	AMD	Intel	Sun
Instruction set		Power PC	PA RISC 2.0	MIPS-V	IA32	IA32	SPARC V.9
Processor		MPC7400	PA RISC	R10000	K6-2	Pentium III	UltraSparc
Fixed-Point (Integer)	8-bit	16	NA	8	8	8	8
	16-bit	8	4	4	4	4	4
	32-bit	4	NA	NA	2	2	2
Floating-point	Single Precision	4	2	2	2	4	Na
Fixed-Point Register File	Size	32x128b	32x64b	32x64b	8x64b	8x64b	32x64b
	Shared with	Dedicated	Integer Reg.	FP Reg.	Dedicated	FP Reg.	FP Reg.
Fixed-Point Accumulator	Size	NA	NA	192	NA	NA	NA
Arithmetic	Unsigned Saturation	Y	Y	Y	Y	Y	Y
	Modulo	Y	Y	Y	Y	Y	Y
Inter-Element Arithmetic	Multiply-Acc	4	NA	4	2	2	NA
	Fixed-point MAC Precision	32+=(16x16)	-	48+=(16x16)	32+=(16x16)	32+=(16x16)	-
	Compare	Y	N	Y	Y	Y	Y
	Min/Max	Y	N	Y	N	Y	Y
	Floating-Point Multiply-Acc	4 single	2 single	2 single	2 single	4 single	N

	Floating-Point Min/Max	Y	N	N	Y	Y	N
Intra-Element Arithmetic	Sum	Y	N	N	Y	Y	N
	Floating-Point Sum	Y	N	N	Y	N	N
Type Conversion	Pack	Y	Y	Y	Y	Y	Y
	Unpack	Y	Y	Y	Y	Y	Y
	Permute	Y	Y	Y	-	N	-
	Merge	Y	Y	Y	Y	Y	Y
Special instructions		VREFP	CACHE HINT	SELECT	FEMMS	EMMS	EDGE
		VRSQRTFP	DEPOSIT		PFRCP	DIVPS	ARRAY
		SPLAT	EXTRACT		PFRSQRT	PREFETCH	PSIDT
		VSEL	SHR PAIR		PREFETCH	SFENCE	BLOCK TRANSFER

NA: Not Available

Multiply-Acc Precision (bits): acc(bits)+=a(bits) x b(bits)

4.D.1 AltiVec [Moto98, JJAN99]

Motorola's AltiVec features a 128-bit vector execution unit operating concurrently with the existing integer and floating-point units. There are totally 162 new instructions that can be classified into four major classes:

Intra-element arithmetic operations	addition, subtraction, multiply-add, average, minimum, maximum, conversion between 32-bit integer and floating-point
Intra-element non-arithmetic operations	compare, select, logical, shift, and rotate
Inter-element arithmetic operations	sum of elements within a single vector register to a separate accumulation register
Intra-element non-arithmetic operations	wide field shift, pack, unpack, merge/interleave, and permute

AltiVec shows significant amount of efforts to exploit maximum amount of parallelism. This results in a 32-entry 128-bit wide register file separating from existing integer and floating-point register files. This is different from other NSP architectures that often share NSP register file with the existing one. The purpose is to exploit additional parallelism through super-scalar dispatch of operations to multiple execution units; or through multithreaded execution unit pipelines. Each instruction can specify up to three source operands and a single destination operand. Each operand refers to a vector register. Target applications of AltiVec include multimedia applications as well as high bandwidth data communication, base station processing, IP telephony gateway, multi-

channel modem, network infrastructure such as Internet router and virtual private network server

4.D.2 MAX 2.0 [Lee96]

Multimedia Acceleration eXtension (MAX) 2.0 is an extension of HP Precision Architecture RISC ISA on PA8000 microprocessor with minimal increased die area concern. Both 8-bit and 32-bit subwords are not supported due to insufficient precision and insufficient parallelism compared to 32-bit single precision floating-point respectively. Although pixels may be input and output as 8 bits, using 16-bit subword in intermediate calculations is preferred. The additional hardware to support MAX2.0 is minimal, since integer pipeline already has two integer ALUs and Shift Merge Units (SMUs) while floating-point pipeline has two FMACs and two FDIV, FSQRT units. MAX special instructions are field manipulation instructions are

Cache Hint	For spatial locality
Extract	Selects any field in the source register and place it right-aligned in the target
Deposit	Selects a right-aligned field from source and place it anywhere in the target
Shift Pair	Concatenates and shift 64-bit or rightmost 32-bit contents of tow reg. into one result

4.D.3 MDMX [MIPS97]

Based on MIPS' experience of designing Geometry Engine, Reality Engine, Maximum Impact, Infinite Reality, Nintendo64, O2 and Magic Carpet, the goal of *MIPS Digital Media Extension* (MDMX) is devised to improve performance IEEE-compliant DCT accuracy. As a result. MDMX adds four-and eight-element SIMD capabilities for integer arithmetic through the definition of these two data types:

Octal Byte	8 unsigned 8-bit integer with 8 unsigned 24-bit accumulator
Quad Half	4 unsigned 16-bit integer with 4 unsigned 48-bit accumulator

Note that both Octal Byte and Quad Half data type share a 192-bit accumulator which permits accumulation of 2^N NxN multiples where N is either 8 or 16 bits according to Octal Byte and Quad Half respectively. MDMX's 32 64-bit wide registers and the 8-bit condition code coincide with the existing floating-point register file similar to the "Paired-single" precision floating-point data type. Data is moved between shared floating-point register file and memory with floating-point load/store double word and between floating-point/integer registers. In addition, MDMX has a unique feature with the vector arithmetic: It is able to operate on specific element of a subword as an operand or as constant immediate value. However, reduction instruction (sum across) and sum of absolute difference (SAD) are judiciously left out. In particular, SAD or L1 norm can be performed as L2 norm without loss of precision using the 192-bit accumulator.

4.D.4 MMX 3D Now! [AMD99, OFW99]

AMD 3D Now! is an Intel's MMX-like multimedia extension and first implemented in AMD K6-2 processor. Floating-point instructions are augmented to the integer-based MMX instruction set by introducing a new data type, single precision FP to support 2D and 3D graphics. Similar to the MMX, applications must determine if the processor supports MMX or not. In addition, 3DNow! is implemented with a separate flat register file in contrast to the stack based floating-point/MMX register file. Since no physical transfer of data between FP and multimedia unit register files is required, FEMMS (Faster entry/exit of the MMX or floating-point state) is included to replace MMX EMMS instruction and to enhance the performance. The register operations of all 3DNow! Either the register X or Y execution pipelines can execute floating-point instructions for a maximum issue and execution rate of two operations per cycle [AMD99]. There are no instruction-decode or operation-issue pairing restrictions. All operations have an execution latency of 2 cycles and are fully pipelined. As long as two operations do not fall into the same category, both operations will start execution without delay. The two categories of additional 21 instructions are

1. PFADD, PFSUB, PFSUBR, PFACC, PFCMPx, PFMIN, PFMAX, PI2FD, PFRCP, and PFRSQRT
2. PFMUL, PFRCPIT1, PFRSQIT1, and PFRCPIT2.

Normally, all instructions should be properly scheduled so as to avoid delay due to execution resource contention or structural hazard by taking dependencies and execution latencies into account.

FEMMS	is similar to MMX's EMMS but faster since 3D Now does not share MMX registers with those of floating-point.
PFRCP	scalar floating-point reciprocal approximation
PFRSQRT	scalar floating-point reciprocal square root approximation
PREFETCH	loads 32 or greater number of bytes either non-temporal or temporal in the specified cache level

4.D.5 MMX/SIMD [Intel99]

MMX (multimedia extension) is Intel's first native signal processing extension instruction set. Subsequently, additional instructions are augmented to the Streaming SIMD Extensions (SSE) [Intel99] in Pentium III class processors. SIMD supports 4-way parallelism of 32-bit single precision floating-point for 2D and 3D graphics or 32-bit integer for audio processing. These new data types are held in a new separate set of eight 128-bit SIMD registers. Unlike MMX execution, traditional floating-point instructions can be mixed with Streaming SIMD extensions without the need to execute a special instructions such as EMMS. In addition, SIMD features explicit SAD instruction and introduces a new operating-system visible state.

EMMS	must be used to empty the floating-point tag word at the end of an MMX

(Empty MMX State)	routine before calling other routines executing floating-point instructions.
DIVPS	divides four pairs of packed, single-precision, floating-point operands.
PREFETCH	loads 32 or greater number of bytes either non-temporal or temporal in the specified cache level
SFENCE (store fence)	ensures ordering between routines that produce weakly ordered results and routines that consume this data just like multiprocessor weak consistency. Non-temporal stores are implicitly weak ordered, no write-allocate, write combine / collapse so that cache pollution is minimized.

4.D.6 VIS [Sun97, TOVH96]

Sun's VIS (Visual Instruction Set) is the only NSP reviewed here that does not support parallelism of floating-point data type. However, the subword data share the floating-point register file with floating-point number as indicated in Table 10. Some special instructions in VIS are Array, Pdist, and Block transfer.

Array	Facilitates 3D texture mapping and volume rendering by computing a memory address for data look up based on fixed-point x, y, and z. Data are laid out in a block fashion so that points which are near one another have their data stored in nearby memory locations.
Edge	Computes a mask used for partial storage at an arbitrarily aligned start or stop address typically at boundary pixels.
Pdist	Computes the sum of absolute value of difference of eight pixel pairs.
Block transfer	Transfers 64 bytes of data between memory and registers.

5. Software Programming Tools for PDSP

5.A Software Development Tools for Programming PDSP

Since their introduction more than a decade ago, PDSPs have been incorporated in many high performance embedded systems such as modems and graphic acceleration cards. A unique requirement of these applications is that they all demand high quality (machine) code generation to achieve the highest performance while minimizing the size of the program to conserve premium on-chip memory space. Often the difference of one or two extra instructions implies either a real-time processing constraint may be violated, leaving the code generated useless, or additional memory module may be needed, causing significant cost to overrun.

High-level (programming) languages (HLLs) are attractive to PDSP programmers because they hide hardware dependent details, and simplify the task of programming. Unlike assembly codes, HLL programs are readable, maintainable and portable to other processors. In the case of an object-oriented HLL, such as C++, those programs are also more reliable and reusable. All these features contribute to reduce development time and cost.

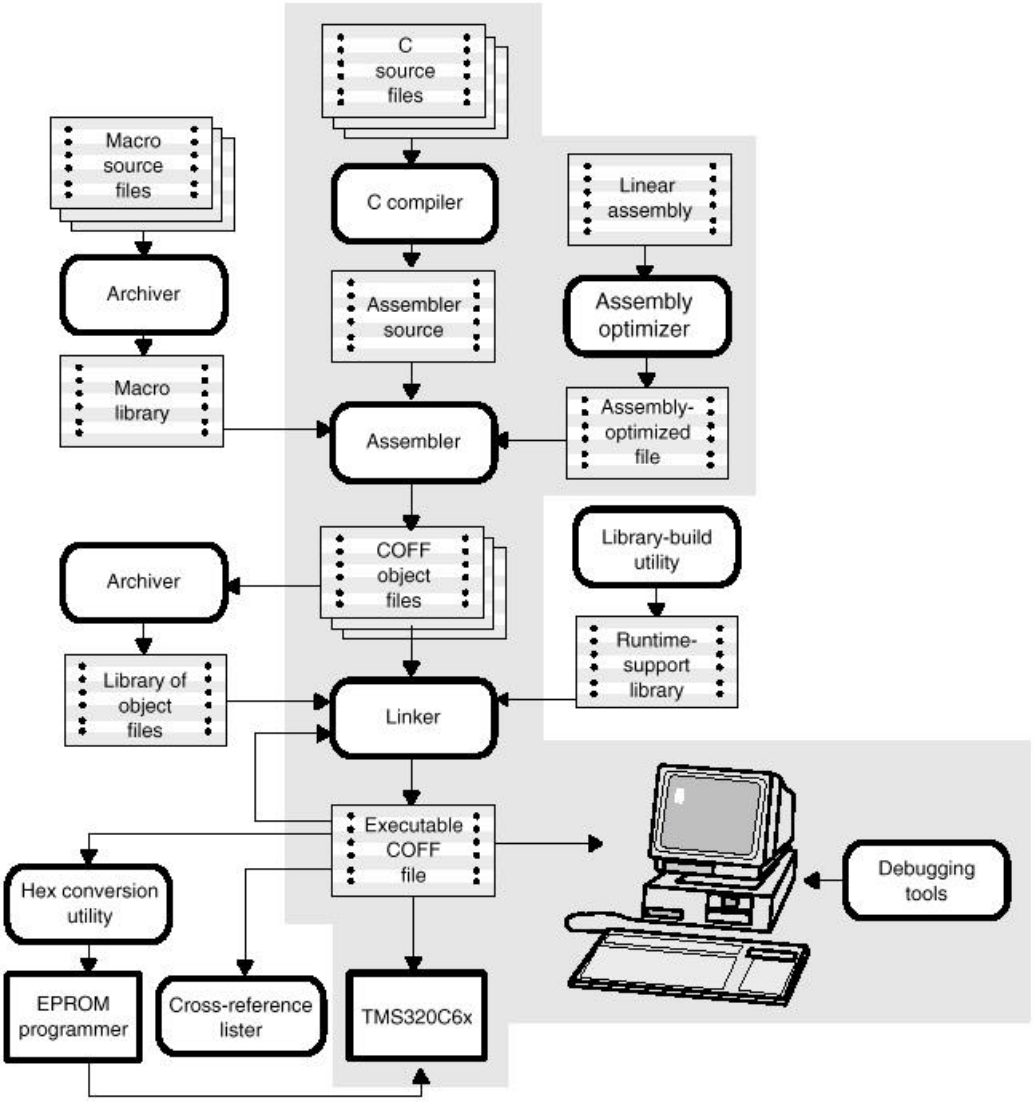

Figure 1. TMS320C6x Software Development Flow [Texas98a]

As an example of typical software development for PDSPs, depicted in Figure 1 is the TMS320C6x software development flow chart. There are three possible source programs: C source files, macro source files and linear assembler source files. The latter sources are both at assembly program level. The assembly optimizer assigns registers and uses loop optimization to turn linear assembly into highly parallel assembly that takes advantage of software pipelining. The assembler translates assembly language source files into machine language object files. The machine language is based on common object file format (COFF). Finally, the linker combines object files into a single executable object module. As it creates the executable module, it performs relocation and resolves external references. The linker also accepts relocatable COFF object files and object libraries as input.

To improve the quality of the code generated, C compilers are always equipped with extensive optimization options. Many of these compiler optimization strategies [Lucent99] are based on GNU C Compiler (GCC):

Table 10. Compiler optimization options in DSP16000 series [Lucen99]

	Optimization Performed	*Targeted Application*
-O0	Default operation, no optimization	C level debug to verify functional correctness
-O1	Optimize for space	Optimize space for control code
-O2	Optimize for space and speed	Optimize space and speed for control code
-O	Equivalent to –O2	Equivalent to –O2
-O3	-O2 plus loop cache support, some loop unrolling	Optimize speed for control and loop code
-O4	Aggressive optimization with software pipeline	Optimize speed and space for control and loop code
-Os	Optimize for space	Optimize space for control and loop code

Debugger can usually be both simulator and profiler like the C source debugger [Texas98b]. The C source debugger is an advanced graphic user interface (GUI) to develop, test, and refine 'C6x C programs and assembly language programs. In addition to that, the 'C6x debugger accepts executable COFF files as input. It features the following capabilities, that can also be found in other PDSP development environments:

- multilevel debugging, user can debug both C and assembly language code,
- fully configurable graphical user interface,
- comprehensive data displays, and
- dynamic profiling provides a method for collecting execution statistics and immediate feedback to identify performance bottlenecks within the code.

5.B On-chip Emulation

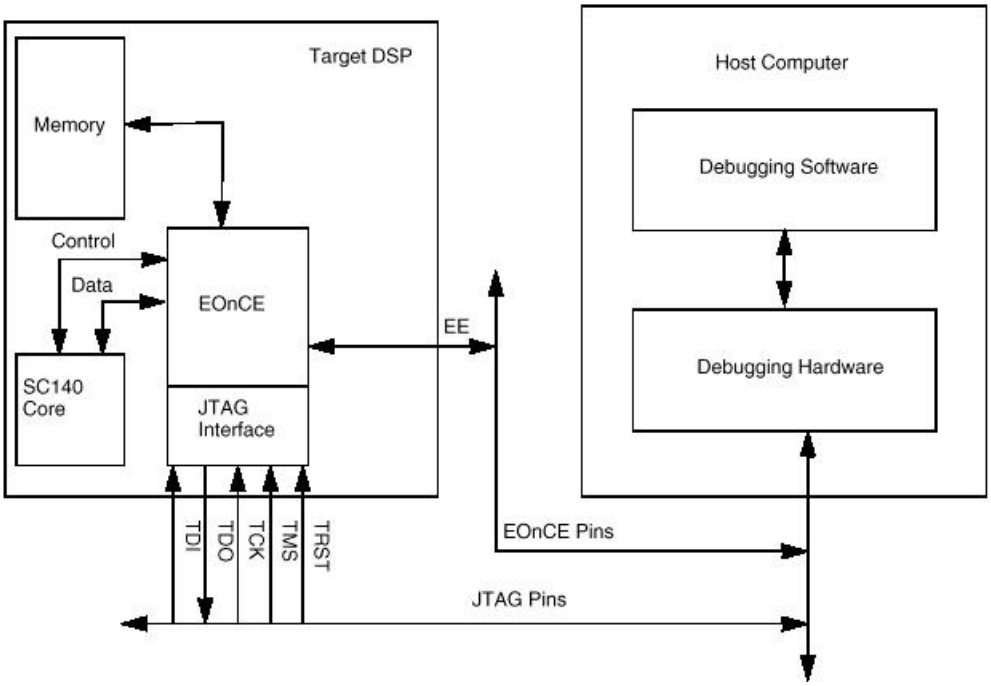

Figure 2. Typical Debugging System using EOnCE [Star99]

The presence of the Joint Test Action Group (JTAG) test access module or Enhanced on-chip emulation (EOnCE) module interface allows the user to insert the PDSP into a target system while retaining debug control. The EOnCE module is used in PDSP devices to debug application software in real-time. It is a separate on-chip block that allows non-intrusive interaction with the core. User can examine the contents of registers, memory or on-chip peripherals through the JTAG pins. Special circuits and dedicated pins on the core are defined, to avoid sacrificing user-accessible on-chip resources.

As applications grow in terms of both size and complexity, the EOnCE provides the user with many features, including:

- breakpoints on data bus values,
- detection of events, which can cause a number of different activities configured by the user,
- non-destructive access to the core and its peripherals,
- various means of profiling, and
- program tracing buffer

The EOnCE module provides system-level debugging for real-time systems, with the ability to keep a running log and trace of the execution of tasks and interrupts, and to debug the operation of real-time operating systems (RTOS).

5.C Optimizing Compiler and code generation for PDSP

The PDSP architecture evolves from ad-hoc heterogeneous towards homogeneous resource like general-purpose RISC microprocessor. One of the reasons is to make compiler optimization techniques less difficult. Classical PDSPs' architecture are characterized by

- a small number and nonuniform register sets in which certain registers and memory blocks are specialized for specific usage,
- highly irregular datapaths to improve code performance and reduce code size,
- very specialized functional units, and
- restricted connectivity and limited addressing to multipartitioned memory.

Several techniques have been proposed based on a simplified architecture of TMS320C2X/5X, e.g. instruction selection and instruction scheduling [YuHu94], register allocation and instruction scheduling [LDKTW95].

The success of RISC and its derivatives such as the superscalar architecture and the VLIW architecture have asserted significant impacts on the evolution of modern PDSP architecture. However, code density is a central concern in developing embedded DSP systems. This concern leads to the development of new strategies such as code compaction [Carmel99] and user-defined long instruction word [KLMW98]. With smaller code space in mind, code compaction using integer programming [LeMa95] was proposed for applications to PDSPs that offer instruction-level parallelism such as VLIW. Later on, an integer programming problem formulation for simultaneous instruction selection, compaction and register allocation were investigated by [Geboy97]. It can be seen that earlier optimization techniques were focus on the optimization of basic code blocks. Thus, they can be considered as a local optimization approach. Recently, the focus has shifted to global optimization issues such as loop unrolling and software pipelining. In [StLe99], results of implementing a software pipelining using modulo scheduling algorithm on C'6x VLIW DSP have been reported.

Artificial intelligence (AI) technique such as planning was employed to optimize instruction selection and scheduling [YuHu94]. With AI, concurrent instruction selection and scheduling yield code comparable to that of hand-written assembly codes by DSP experts. The instruction scheduler is a heuristic list based scheduler. Both instruction scheduling and selection involve node coverage by pattern matching and node evaluation by heuristic search using means-end-analysis and hierarchical planning. The efficiency is measured in terms of size and execution time of generated assembly code whose size is up to 3.8 times smaller than that of commercial compiler.

Simultaneous instruction scheduling and register allocation for minimum cost based on a branch-and-bound algorithm are reported [LDKTW95]. The framework can be generalized to accumulator-based machines to optimize accumulator spilling such as the TMS320C40. Their uses are likely intended to obtain more compact code.

Integer linear programming is shown to be effective in compiler optimization recently in [LeMa95] and [Geboy97]. The task of local code compaction in VLIW architecture is solved under a set of linear constraints such as a sequence of register transfers and maximum time budget. Since some DSP algorithms show more data flow and less control flow behavior [LeMa95], code compaction exploits parallel register transfers to be scheduled into a single

control step resulting in a lower cycle count to satisfy timing constraint. Of course, resource conflicts and dependencies as well as encoding restrictions and operations having side effect must be taken into consideration. Later effort has integrated instruction selection, compaction and register allocation together [Geboy97]. The targeted PDSP is defined using arc mappings and later into logical propositions to make it retargetable. These propositions are then translated into mathematical constraints to form the optimization model using integer linear programming. Code is generated and optimized for minimum code size, maximum performance in estimated energy dissipation.

In modern VLIW PDSP, the architecture features homogeneous functional and storage resources enabling global optimization by compiler. Software pipelining is known as one of the most popular code scheduling techniques. It exploits the available instruction level parallelism in different loop iterations. A software-pipelined loop consists of three components:

- A prolog: set up the loop initialization
- A kernel: execute pipelined loop body in steady state
- An epilog: drain the execution of the loop kernel

Modulo scheduling takes an innermost loop body and constructs a new schedule. The new schedule is equivalent to overlapping loop iterations. The algorithm utilizes data precedence graph (DPG) and reservation table to construct permissible schedule of the loop body under the available resource constraints. DPG is a directed graph (possibly cyclic) with nodes and edges representing operations and data flow dependencies of the original inner loop body. The resource requirements for an operation are modeled using reservation table. [StLe99] reports result of software pipelining on a set of 40 loop kernels based on C‘6x architecture. However, the architectural features that impact performance gain of software pipelining are moderately sized register file, constraints on code size, and multiple assignment code.

6. DSP System Design Methodologies

Designing modern DSP systems requires more than just programming PDSP or processing cores. Instead, the system's performance must be the utmost performance criterion. The DSP system design methodologies are developed at different levels of abstractions. At the system level, the design scope includes task and data partitioning, and software synthesis/simulation. At the architectural level, the focus is on architecture and compiler development. At the chip implementation level, hardware description languages such as VHDL and hardware/software co-design methodologies are quite important.

6.A Application development with existing hardware/processing core

A software engineering approach is incorporated to assist the development of an application using a DSP array processor at Raytheon System Corporation [KeOs98]. Three performance measurements are used to gauge the quality of the design:

- processor throughput rate,
- memory utilization, and
- I/O bandwidth utilization.

A sensitivity analysis of these performance metrics is performed to examine trade-offs of various design approaches. Factors that affect the processor throughput rate include: the quality of DSP algorithm formulation, the operation cost in processor cycles, the sustained throughput rate to peak throughput efficiency, and the expected speedup when it is upgraded to the next generation of PDSP. Regarding the memory utilization, it has been observed that the size of the data samples, and the dynamic nature of memory usage patterns are the most two important factors. The I/O bandwidth utilization, on the other hand, depends on the algorithm as well as the hardware design. Several design tools used to develop the entire system and their factors that may degrade the performance during the design process are listed in the table below:

Table 11. DSP System development tools and factors that may degrade performance

Tools	*Factors that may degrade performance*
Code generation	– Compiler efficiency – Quality of generated assembly code – Size of load image
Instruction level processor simulator	– Cycle counts for elementary operation
Cycle-accurate device level VHDL model	– External memory access time – Instruction caching effects – Resource contention between processor and DMA channels

Rate monotonic analysis (RMA) [LiLa73] is necessary to validate the schedulability of software architecture. In general, it has been reported that the following lessons have been learned through this design experience:

- prototype early in the development cycle,
- ignore processor marketing information – actual throughput is highly dependent on the application profile,
- analyze careful the most frequently executed function - task switching, and
- take inherent interfaces overhead such as interrupt handling, data packing and unpacking into account in estimating the throughput.

Another example of DSP system development is The Computer Assisted Dynamic Data Monitoring and Analysis System (CADDMAS) developed for the U.S. Airforce and NASA [SKB98]. Its details have been described earlier in section 2.c. An adaptive approach was necessary to allow the structure of the system to adapt to changing external requirement and sensor availability. This leads to application of a reconfigurable controller called *structurally adaptive signal processing*, for process control. Unlike parametric adaptation where topology of the graph is fixed and coefficients can change over time, a structurally adaptive signal processing system can change its computational structure on the fly. Therefore, its control functionality can be maintained even in the face of sensor failures, the performance will be gracefully degraded but correct control action is still present.

6.B Application-driven design - Fine tuning the processing core

Two reasons contributing to the poor performance of HLL PDSP commercial compilers are first, that compilers are developed after a target architecture has been established and second, the inability to exploit DSP-specific architectural features in DSP compiler [SCL94]. The following application-driven design methodologies are adopted.

- A DSP architecture and its compiler are developed in parallel,
- its dynamic statistics is assessed the impact tradeoffs on performance, and
- an iterative analysis to fine-tune the architecture and compiler.

The PDSP architecture is based on VLIW. As a result an optimizing C compiler is necessary to exploit static instruction-level parallelism as well as DSP-specific hardware features. Those hardware features are modulo addressing, low over-head looping, and dual data memory banks. Meanwhile, instruction set simulator is developed to gather statistics on the run-time behavior of DSP programs. A suite of DSP benchmarks in terms of kernel and application are chosen to evaluate the system. The performance success of the compiler is due to the flexibility of the model VLIW architecture. The statistics indicate the areas of improvement to be fed back for fine-tuning the architecture. However, its drawback is the high instruction-memory bandwidth requirements that can be too expensive and impractical to implement.

As another means of DSP architecture development, machine description language (MDL) has been proposed to achieve rapid prototyping at architectural level. Recently, LISA [PHZM99] is developed for the generation of bit and cycle accurate models of a PDSP. It includes instruction set architecture that enables automatic generation of simulators and assemblers. LISA is composed of resource and operation declarations. Resource declaration represents the storage objects of the hardware architecture (e.g. registers, memories, pipelines). Declaration description collects the description of different properties of the system, i.e. instruction set model, the behavioral model, the timing model, and necessary declarations. LISA supports cycle-accurate processor models, including constructs to specify pipelines and their mechanisms. It's targeting SIMD, VLIW, and superscalar architectures. Direct support for compiled simulation techniques and strong orientation on C programming language are contributed in LISA. the Texas Instruments TMS320C6201 DSP, Realized as a real world example, is modeled in cycle by cycle basis by only one designer and finished within two months.

6.C Reconfigurable Computing: Hw/Sw codesign for a given application

In the system design process, it has been traditional that decision is made on a subtask-by-subtask basis to be implemented in either custom hardware or software running on PDSP(s). On one hand, custom hardware or ASIC can be customized to particular subtask resulting in relatively fast and efficient implementation. ASIC is physically programmed by patterning devices (transistors) and metal interconnection prior to fabrication process. Higher throughput and lower latency can be achieved with more space dedicated to particular functional units. On the other hand, PDSP is programmed later by software resulting in flexible but relatively slow and inefficient realization. Temporal or sequential operations can be accomplished by a set of instructions to program a processor after its fabrication.

Between these two extremes, reconfigurable computing (RC) architecture can be programmed to perform any specific function by a set of configuration bits. In other words, RC combines temporal programmability with spatial computation in hardware after it is fabricated at low overhead. In other words, the hardware/software boundaries can been altered by RC paradigm [DeWa99].

Also known as 90/10 rule of thumb where 90% of runtime is spent on 10% of the program, hardware/software partitioning is inspired by the higher percentage of run time of specialized computation, the more improvement of cost/performance if it is implemented in hardware and the more specialized computation dominate the application the more closely the specialized processor should be coupled with host processor. This rule of thumb has been successfully applied to floating-point processing unit as well as RC.

In heterogeneous system approach, RC is combined with a general-purpose processing capability of traditional microprocessor. Interface between these two can be either closely or loosely coupled depending upon its applications. How frequent RC's functionality should be reconfigured dynamically must be determined based on the performance optimality of particular hardware/software architecture.

Existing RC architectures are mostly of the form 1D or 2D array of configurable cells interconnected by programmable links. The array communicates with outside world through peripheral I/O cells. The architecture can be characterized by its logic capability of each cell at different granularity.

- Fine-grained configurable cell performs simple logic functions with support of more complex function such as fast carry chain.
- Coarse-grained configurable cell performs word parallel arithmetic functions such as addition, multiplication, etc. with support of simple bit-level logic functions.

Either general-purpose computing or DSP application mainly drives the architecture of configurable cell either fine-grained or coarse-grained. Its configuration or *context* is stored locally like SRAM-based FPGA. Its implementation is either FPGA-like or custom design configurable cell based on current technology and scale of integration. Instead of being off-chip and loosely coupled to the host processor, RC is going towards an on-chip coprocessor as the technology advances to SOC.

Table 13 Some Reconfigurable Computing Architectures for DSP Applications

Fine grained	CHAMP [PaGu95], DRLE [Nishi99], Pleiades [AbRa96]
Coarse grained	MATRIX [MiDe96], MorphoSys [LSLB99], REMARC[MiOl98]

Here are some exiting RC architectures for DSP Applications.

CHAMP [PaGu95] is a system of 8 PEs interconnected in a 32-bit ring topology. Each PE consists of 2 Xilinx XC4013 FPGAs, dual-port memory, and 32-bit 38 ports reconfigurable crossbar switch between PEs and memory.

DRLE (Dynamically Reconfigurable Logic Engine) [FFMN99, Nishi99] is capable of real-time reconfiguration with several layers of configuration tables. An experimental chip is composed of 4x12 array of configurable cell. Each cell can realize two different logic operations

equivalent to 4-bit input to 1-bit output or 3-bit input to 2-bit output. Up to eight different configurations can be locally stored in each cell memory. With 0.25 micron CMOS technology, the chip contains 5.1 million transistors in 10x10 mm^2 die-area and consumes 500 mW @ 70 MHz.

MATRIX [MiDe96] is composed of 2D array of identical 8-bit configurable cells overlaid with a configurable network. Each cell consists of 256x8-bit memory, 8-bit ALU and multiplier, and reduction control logic. The interconnect network supports 3 ranges of interconnection: nearest neighbor, bypass of length four, and global line with pipeline register. The configurable cell area is approximated to 29 million λ^2. Its cycle time is 10 ns at 0.5 micron CMOS technology.

MorphoSys [LSLB99] is 2D mesh 8x8 reconfigurable cell array coprocessor. Each cell is similar to datapath found in conventional processors consisting of ALU/multiplier, shifter, and 4-entry register file. Moreover, bit-level application is also supported. Up to 32 contexts can be simultaneously resident in context memory. The whole array can be reconfigured in 8 cycles or 80 ns @ 100 MHz.

Pleiades [AbRa96] is proposed as heterogeneous system partitioned by control-flow computing on microprocessor and data flow computing on RC for future wireless embedded device. RC array is composed of satellite (configurable) processors and programmable interconnect to main microprocessor. Data-flow driven computing is implemented using global asynchronous and local synchronous clocking to reduce overhead. Therefore, operation starts only when all input data are ready.

7. Conclusion

In this chapter, we briefly surveyed the architecture, application and programming methodologies of modern programmable digital signal processors (PDSPs). With a bit of stretch of the definition, we included in this survey the hardware programmable FPGA realization of DSP algorithms and the special multimedia extension instructions incorporated in general-purpose microprocessors to facilitate native signal processing. We also offered an overview of the existing applications of PDSPs. Finally, we summarized current software design methodologies and briefly mentioned future trends and open research issues.

8. Bibliography

[Analog99] Analog Device Inc., ADSP-21160 SHARC DSP Hardware Reference, 1999

[AbRa96] A. Abnous and J. Rabaey, Ultra-low-power domain-specific multimedia processors, VLSI Signal Processing, IX, 1996, pp. 461-470

[AEDR91] Alter, J.J., Evins, J.B., Davis, J.L., and Rooney, D.L., A programmable radar signal processor architecture, Proceedings of the 1991 IEEE National Radar Conference, 1991, pp. 108-111.

[AMD99] 3DNow!™ Technology Manual, AMD Inc., Aug. 1999, http://www.amd.com/K6/k6docs/pdf/21928.pdf

[Andr98] Ray Andraka, A Survey of CORDIC Algorithms for FPGA based Computers, *Proceedings of the 1998 ACM/SIGDA sixth international symposium on Field programmable gate arrays*, 1998, pp. 191 - 200.

[Bier97] Bier, J. J., DSP16xxx Targets Communication Apps, Microprocessor Report, vol. 11, no. 12, Dec. 15, 1997.

[BJER98] R. Bhargava, L. K. John, B. L. Evans, R.Radhakrishnan, Evaluating MMX Technology Using DSP and Multimedia Applications, Proceedings of the IEEE Symposium on Microarchitecture (MICRO-31), Dallas, Texas, pp. 37-46, Dec. 1998.

[BRWT99] Budagavi, M., Rabiner, W., Webb, J., Talluri, R., Wireless MPEG-4 video on Texas Instruments DSP chips, Proc. 1999 IEEE International Conference on Acoustics, Speech, and Signal Processing, Vol. 4 , 1999, pp. 2223 -2226.

[BMMOP96] F. Bonomini, F. De Marco-Zompit, G.A. Mian, A. Odorico, D. Palumbo, Implementing an MPEG2 Video Decoder Based on the TMS320C80 MVP, September 1996.

[Bork99] Shekhar Borkar, Design challenges of technology scaling, IEEE Micro, July-Aug., 1999, pp. 23-29

[Carmel99] CARMEL™ Development Chip, Aug. 1999, http://www.carmeldsp.com/Pdf/CDEV2.2.pdf

[Chish94] Mansoor A. Chishtie, U.S. Digital Cellular Error-Correction Coding Algorithm Implementation on the TMS320C5x, Texas Instrument, SPRA137, October 1994.

[CJL97] Peter Chou, Hong Jiang, and Z.-P. Liang, Implementing Real-Time Cardiac Imaging Using theTMS320C3x DSP, July 1997.

[CKITT99] Ngai-Man Cheung, Kawashimea, T., Iwata, Y., Traunnumn, S., Tsay, J., Pawate, B.I., Software Mpeg-2 Video Decoder On A DSP Enhanced Memory Module, Multimedia Signal Processing, 1999 IEEE 3rd Workshop on , 1999, pp. 661 -666.

[DaVi98] Daffara, F., Vinson, P., Improved search algorithm for fast acquisition in a DSP-based GPS receiver, 1998 URSI International Symposium on Signals, Systems, and Electronics 1998, pp. 310 -314.

[DDNSZ95] H. Dullink, B. van Daalen, J. Nijhuis, L. Spaanenburg, H. Zuidhof, Implementing a DSP Kernel for Online.

[DeWa99] Andre Dehon and John Wawrzynek, Reconfigurable Computing: What, Why, and Implications for Design Automation, *Proceedings of the 36th ACM/IEEE conference on Design automation conference* , 1999, pp. 610 - 615.

[Dick96] Chris Dick, Computing the Discrete Fourier Transform on FPGA Base Systolic Arrays, *Proceedings of the 1996 ACM fourth international symposium on Field-programmable gate arrays*, 1996, pp. 129 - 135.

[EyBi98] J. Eyre and J. Bier, Camel Enables Customizable DSP, Microprocessor Report, vol. 12, no. 17, Dec. 28, 1998.

[FFMN99] Fujii, T.; Furuta, K.-i.; Motomura, M.; Nomura, M.; Mizuno, M.; Anjo, K.-i.; Wakabayashi, K.; Hirota, Y.; Nakazawa, Y.-e.; Ito, H.; Yamashina, M., A dynamically reconfigurable logic engine with a multi-context/multi-mode unified-cell architecture**,** Solid-State Circuits Conference, 1999. Digest of Technical Papers. ISSCC. 1999 IEEE International , 1999, pp. 364 -365

[GaTh99] S. Ganesh and Vivek Thakur, Print Screening With Advanced DSPs, Texas Instrument, SPIA004, 1999

[GeBoy97] Catherine H. Geboyts, An Efficient Model for DSP Code Generation: Performance, Code Size, Estimated Energy, *Proceedings of the tenth international symposium on System synthesis* , 1997, pp. 41 - 47.

[Golst96] Jeremiah Golston, Single-Chip H.324 Videoconferencing, IEEE Micro, Aug. 1996, pp. 21-33.

[Gosli95] Gregory Ray Goslin, A Guide to Using Field Programmable Gate Arrays (FPGAs) for Application-Specific Digital Signal Processing Performance, Xilinx, Inc., 1995.

[Gottl94] Albert M. Gottlieb, A DSP-Based Research Prototype Reverse Channel Transmitter/Receiver for ADSL, 1994 IEEE International Conference on Acoustics, Speech, and Signal Processing, Volume: iii, 1994 pp. III/253-III/256 vol.3.

[IGGL99] IKlaus Illgner, Hans-Georg Gruber, Pedro Gelabert, Jie Liang, Youngjun Yoo, Wissam Rabadi, and Raj Talluri, Programmable DSP Platform for Digital Still Cameras, Texas Instrument, 1999

[Intel99] Intel® Architecture Optimization Reference Manual, Intel Corp., http://developer.intel.com/design/pentiumii/manuals/245127.htm

[JJAN99] Tyler-J, Lent-J, Mather-A, Huy-Nguyen, AltiVec/sup TM/: bringing vector technology to the PowerPC/sup TM/ processor family, 1999 IEEE International Performance, Computing and Communications Conference, pp.437-44.

[Kala98] Paul Kalapathy, Hardware-Software Interactions on Mpact, IEEE Micro, Mar./Apr. 1997, pp. 20.

[KeOs98] David P. Kelly and Robert S. Oshana, Software Performance Engineering a Digital Signal Processing Application, *Proceedings of the first international workshop on Software and performance*, 1998, pp. 42 - 48.

[Knapp95] Steven K. Knapp, Using Programmable Logic to Accelerate DSP Functions, Xilinx, Inc., 1995, http://www.xilinx.com/appnotes/dspintro.pdf

[KLMW98] P. Kievits, E. Lambers, C. Moerman, and R. Woudsma, R.E.A.L. DSP Technology for Telecom Baseband, Processing, Philips Semiconductors, http://www-us.semiconductors.philips.com/acrobat/literature/other/dsp/icspat98_pks.pdf

[Mitch97] Joan L. Mitchell, et al., **MPEG** video compression **standard**, Chapman & Hall, New York, 1997

[KSJ96] Kibe, S.V., Shridhara, K.A., Jayalalitha, M.M., Software-based GIC/GNSS compatible GPS receiver architecture using TMS320C30 DSP processor**,** Satellite Systems for Mobile Communications and Navigation, 1996., Fifth International Conference on , 1996, pp. 36 -39.

[Lee96] Ruby B. Lee, Subword parallelism with MAX-2, IEEE Micro, Aug. 1996, pp. 51-59.

[LeMa95] Rainer Leupers and Peter Marwedel, Time-Constrained Code Compaction for DSPs, *Proceedings of the eighth international symposium on System synthesis* , 1995, pp. 54 - 59.

[LeTo98] Dan Ledger and John Tomarakos, *Using The Low Cost, High Performance ADSP-21065L Digital Signal Processor For Digital Audio Applications, Revision 1.0,* Analog Device, Inc., *12/4/98.*

[LiLa73] Liu, C.L. and Layland, J.W., Scheduling algorithms for multiprogramming in a hard-real-time environment, Journal of the Association for Computing Machinery.vol.20, no.1; Jan. 1973; p.46-61.

[LPM97] Chunho Lee , Miodrag Potkonjak and William H. Mangione-Smith, MediaBench: A Tool for Evaluating and Synthesizing Multimedia and Communications Systems, *Proceedings of the thirtieth annual IEEE/ACM international symposium on Microarchitecture*, 1997, pp. 330 - 335.

[Lucen99] DSP16000 C Compiler, Dec. 1999, Lucent, Inc., http://www.lucent.com/micro/dsp16000/pdf/AP99052.pdf

[LSLB99] Guangming-Lu; Singh-H; Ming-Hau-Lee; Bagherzadeh-N; Kurdahi-FJ; Filho-EMC; Castro-Alves-V, The MorphoSys dynamically reconfigurable system-on-chip, Proceedings of the First NASA/DoD Workshop on Evolvable Hardware 1999, pp.152-60

[MIC96] Meisl, P.G., Ito, M.R., Cumming, I.G., Parallel processors for synthetic aperture radar imaging**,** Proceedings of the 1996 International Conference on Parallel Processing, 1996. Vol.3. Software., , Volume: 2 , 1996, pp. 124 -131 vol.2.

[MIEB99] Morgan, P.N., Iannuzzelli, R.J., Epstein, F.H., Balaban, R.S., Real-time cardiac MRI using DSPs, IEEE Transactions on Medical Imaging, Volume: 18 Issue: 7 , July 1999, pp. 649 -653.

[MIPS97] MIPS Extension for Digital Media with 3D, MIPS Technologies, Inc. March, 1997, http://www.mips.com/Documentation/isa5_tech_brf.pdf

[MiDe96] E. Mirsky and A. DeHon, MATRIX: a reconfigurable computing architecture with configurable instruction distribution and deployable resources, Proc. IEEE Symposium on FPGAs for Custom Computing Machines, 1996, pp. 157 -166

[Moto98] AltiVec Technology Programming Environments Manual, Rev. 0.1, Nov. 1998, Motorola, Inc.

[MiOl98] T. Miyamori and U. Olukotun, A quantitative analysis of reconfigurable coprocessors for multimedia applications, Proc. IEEE Symposium on FPGAs for Custom Computing Machines 1998, 2-11

[NEC97] NEC Inc., NA853C Microcontroller Overview, 12085e40.pdf

[Nishi99] T. Nishitani, An Approach to a Multimedia Systeom on a Chip, IEEE Workshop on Signal Processing Systems 1999, pp. 13-21

[OFW99] Stuart Oberman, Greg Favor, and Fred Weber, AMD 3Dnow! Technology: Architecture and Implementations, IEEE Micro, Apr. 1999, pp. 37-48.

[OwPu97] Owen, R.E.; Purcell, S., An enhanced DSP architecture for the seven multimedia functions: the Mpact 2 media processor, IEEE Workshop on Signal Processing Systems - Design and Implementation 1997, Page(s): 76 -85

[PaGu95] R. Patriquin and I. Gurevich, An automated design process for the CHAMP module, Proceedings of the IEEE National Aerospace and Electronics Conference 1995, pp. 417-424

[PaRo96] B. I. (Raj) Pawate and Peter D. Robinson, Implementation of an HMM-Based, Speaker-Independent Speech Recognition System on the TMS320C2x and TMS320C5x, 1996.

[Philips99a] TM1300 Preliminary Data Book, Oct. 1999, http://www-us.semiconductors.philips.com/trimedia/products/#tm1300,

[PHZM99] Stefan Peesl, Andreas Hoffmannl, Vojin Zivojnovic2, Heinrich Meyrl, LISA - Machine Description Language for Cycle-Accurate Models of Programmable DSP Architectures, *Proceedings of the 36th ACM/IEEE conference on Design automation conference*, 1999, pp. 933 - 938.

[PWW97] Alex Peleg, Sam Wilkie, and Uri Weiser, Intel MMX for Multimedia PCs, Communications of the ACM, Jan. 1997, Vol. 40, No. 1, pp. 25-38.

[PeMi93] William B. Pennebaker, Joan L. Mitchell, JPEG still image data compression standard, Van Nostrand Reinhold, New York , 1993

[Purc98] Steve Purcell, The Impact of Mpact2, IEEE Signal Processing Magazine, Mar. 1998, pp. 102-107.

[Rena] Pascal Renard, Implementation of Adaptive Controllers on the Motorola DSP56000/DSP56001, Motorola, Inc., APR15.pdf

[Resw96] Etienne J. Resweber, A DSP GMSK Modem for Mobitex and Other Wireless Infrastructures, *Synetcom Digital Incorporated*, 1996.

[RoLu98] Alec Robinson, Chuck Lueck, and Jon Rowlands, Audio Decoding on the C54X, Texas Instrument.

[Sesh98] Nat Seshan, High VelociTi Processing, IEEE Signal Processing Magazine, Mar. 1998, pp. 86-101.

[ShLe2000] Ali M Shankiti, Miriam Leeser, Implementing a RAKE Receiver for Wireless Communications on an FPGA-based Computer System, *Proceedings of the ACM/SIGDA international symposium on Field programmable gate arrays* , 2000, pp. 145 - 151.

[SKB98] Janos Sztipanovits, Gabor Karsai, and Ted Bapty, Self-Adaptive Software for Signal Processing, Communications of the ACM, May 1998, vol. 41, no.5, pp. 66-73

[Star99] SC140 DSP Core Reference Manual, Rev. 0, 1999, http://www.mot.com/pub/SPS/DSP/LIBRARY/STARCORE/sc140dspcore_rm.pdf

[StLe99] Eric Stotzer, and Ernest Leiss , Modulo Scheduling for the TMS320C6x VLIW DSP, *Proceedings of the ACM SIGPLAN 1999 workshop on Languages, compilers, and tools for embedded systems* , 1999, pp. 28 - 34.

[StSo] Jay Stokes and Guy R. L. Sohie, Implementation of PID Controllers on the Motorola DSP56000/DSP56001, Motorola, Inc., PR5.pdf

[Sun97] VIS Instruction Set User's Manual, Jul. 1997, http://www.sun.com/microelectronics/manuals/805-1394.pdf

[SVMJ95] Brian Schoner, John Villasenor, Steve Molloy and Rajeev Jain, Techniques for FPGA Implementation of Video Compression Systems, *Proceedings of the third international ACM symposium on Field-programmable gate arrays*, 1995, pp. 154-159.

[Taipa98] Dana Taipale, Implementing Viterbi Decoders Using the VSL Instruction on DSP Families DSP56300 and DSP56600, APR40/D (Revision 0, May 1998).

[TexasE97] Caller ID on TMS320C2xx, Texas Instruments Europe, BPRA056, July 1997.

[Texas98a] TMS320C6x Optimizing C Compiler User's Guide, Texas Instruments, Literature Number: SPRU187C, February 1998.

[Texas98b] TMS320C6x C Source Debugger User's Guide, Literature Number: SPRU188D, January 1998.

[Texas98c] TMS320C62x/C67x, Programmer's Guide, Literature Number: SPRU198B, February 1998

[Texas99] TMS320VC5421, Advance Information, Literature Number: SPRS098, December, 1999

[Texas99a] Texas Instrument Inc., TMSC320C6203 Product Preview, 1999, sprs104.pdf

[Texas99b] Texas Instrument Inc., TMSC320C6701 Product Preview, 1999, sprs067c.pdf

[TMI99] H. Terada, S. Miyata, and M. Iwata, DDMP's: Self-Time Super-Pipelined Data Driven Multimedia Processors, Proceeding of the IEEE, vol. 87, no. 2, Feb. 1999, pp. 282-296.

[TOVH96] Marc Tremblay, J. Michael O'Connor, Venkatesh Narayan, and Liang He, VIS Speeds New Media Processing, IEEE Micro, Aug. 1996, pp. 10-20.

[TriCore99] TriCore Architecture Manual, Infineon, Inc. 1999, http://www.infineon.com/us/micro/tricore/arch/archman.pdf

[WoBi98] O. Wolf and J. Bier, StarCore Launches First Architecture, Microprocessor Report, vol. 12, no. 14, Oct. 26, 1998.

[YDG98] Susan Yim, Yinong Ding, and E.Bryan George, Implementing Real-Time MIDI Music Synthesis Algorithms, ABS/OLA, and SMS for the TMS320C32 DSP, Texas Instrument, January 1998.

[Yiu98] Henry Yiu, Implementing V.32bis Viterbi Decoding on the TMS320C62xx DSP, Texas Instruments Hong Kong Ltd, SPRA444, April 1998.

[YLZ98] Fengying Yao, Bizhou Li, Min Zhang, A fixed-point DSP implementation for a low bit rate vocoder, Proceedings. 1998 5th International Conference on Solid-State and Integrated Circuit Technology, 1998, pp. 365 -368.

[YuHu94] Kin H. Yu and Yu Hen Hu, Efficient Scheduling and Instruction Selection for Programmable Digital Signal Processors, IEEE Transactions on Signal Processing, vol. 42, no. 12, Nov. 1994, pp. 3549-52.