Programmable Digital Signal Processor

Lan-Da Van (范倫達), Ph. D.
Department of Computer Science,
National Chiao Tung University,
Taiwan, R.O.C.
Spring, 2010

ldvan@cs.nctu.edu.tw
http://www.cs.nctu.tw/~ldvan/
Outlines

- **Introduction**
- Evolution and Features of Programmable DSP Processors
- DSP Processors for Multimedia and Communications
- Conclusions
Hierarchical View of Signal Processing

Applications
- Video Teleconferencing
  - Image Compress/Decompress
  - Image Enhancement
  - Video Resolution Conversion
- Audio
  - Audio Compress/Decompress
- Communications
  - Embedded Systems: PDA
  - Radar Detection
- Documents Image Processing
  - Optical Character Recognition
  - Image Compress/Decompress
  - Font Generation
- Computer Graphics
  - 2-D Graphics
  - 3-D Graphics
- Others
  - Seismic Processing
  - Disk Driver

Techniques
- Signal Processing
  - Filtering
  - Transform
  - Restoration
- Image Analysis/Synthesis
  - Segmentation
  - Classifying
  - Grouping/Labeling
  - Feature Extraction
  - Matching
  - Recognition
- Compression
  - Entropy Encoding
  - Difference Encoding
  - Quantization
  - Motion Estimation
- 2-d Graphics
  - Text
  - Font Generation
  - Curve/Line/Circle/Ellipse
- MIMO-OFDM
  - Transform
  - V-BLAST
  - DFE

Algorithms
- Convolution
- Correlation
- Morphology
- Edge Ops
- Warp
- Pyramids
- FFT
- DCT
- FIR Filters
- Fractal Transforms
- Median Filter
- Run Length Coding
- Histograms
- Projections
- Segmentation
- Adaptive Threshold
- Boolean Pixel Ops
- Font Compiling
- ADPCM
- Color Space Conversion
- Sub Band Coding
- Linear Predictive Coding
Design Parameters of DSP System

- **Data rate**
  - seismic
  - sonar
  - video
  - audio
  - radar
  - AM-FM radios, TV
  - telecommunication
  - microwave

- **Data format**
  - Digital audio < 20 bit
  - Speech < 12~14 bit
  - Image < 16 bits

1D Speech

2D Image

3D Video

User Interface

Display

Digital Signal Processor

Sensors Digital/Analog inputs

A/D

D/A

Actuators Digital/Analog Outputs
Role of Processor (1/2)

Programmable DSPs / ASSPs
General Purpose Processor
Reconfigurable Processor
ASIC

Energy Efficiency (MOPS/mW)

Flexibility (Coverage)

0.1
1
10
100
1000
### Role of Processor (2/2)

<table>
<thead>
<tr>
<th></th>
<th>Performance</th>
<th>Cost (for a System)</th>
<th>Power</th>
<th>Flexibility</th>
<th>Design Effort (NRE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASIC</td>
<td><strong>OO</strong></td>
<td><strong>XX</strong></td>
<td><strong>OO</strong></td>
<td><strong>XX</strong></td>
<td><strong>XX</strong></td>
</tr>
<tr>
<td>Reconfigurable Processor</td>
<td><strong>O</strong></td>
<td><strong>X</strong></td>
<td><strong>O</strong></td>
<td><strong>X</strong></td>
<td><strong>X</strong></td>
</tr>
<tr>
<td>Programmable DSP/ASSP</td>
<td><strong>X</strong></td>
<td><strong>O</strong></td>
<td><strong>X</strong></td>
<td><strong>O</strong></td>
<td><strong>O</strong></td>
</tr>
<tr>
<td>General Purpose Processor</td>
<td><strong>XX</strong></td>
<td><strong>OO</strong></td>
<td><strong>XX</strong></td>
<td><strong>OO</strong></td>
<td><strong>OO</strong></td>
</tr>
</tbody>
</table>

**OO:** Better, **O:** Good, **X:** Bad, **XX:** Worse
Outlines

- Introduction
- *Evolution and Features of Programmable DSP Processors*
- DSP Processors for Multimedia and Communications
- Conclusions
Processor Architecture

Von Neuman architecture

- A single set of address and data buses
- Both program instructions and data are stored in a single memory
- The processor can only access the memory once during each instruction cycle
- It is not suitable for DSP processors
DSP Processor Architecture

Harvard architecture

- The processor can simultaneously access 2 memory banks using 2 independent sets of buses
- One memory holds program instructions and the other holds data
- Multiported memories allow multiple concurrent memory accesses over 2 or more independent sets of buses within 1 instruction cycle
Conventional DSP Processor

Architecture

- Similar to the original DSP processors of the early 1980s
- A single multiplier or MAC unit and an ALU, but few additional execution units, if any.

Feature

- Low-cost and low-performance

Speed

- 20-50 MHz

Product

- AD ADSP-16xx
- TI TMS320C2xx
- Motorola DSP560xx
Enhanced-Conventional DSP Processors

Architecture

- Improved conventional DSP processors but not new!
- Two multiplier or MAC units => Increase parallelism in a single instruction. (this kind of instructions maybe complex!)

Feature

- High-cost and improved performance

Speed

- 40-100 MHz

Product

- AD ADSP-16xxx
Multi-Issue Architectures

Architecture
- VLIW or Superscalar
- Execute multiple instructions in parallel => Highest parallelism in multiple instructions. (this kind of instructions are simple!)

Feature
- Cost-effective and High-performance

Speed
- More than 100 MHz

Product
- TI TMS320C62xx
VLIW vs Superscalar

**Definition:** Execute multiple instructions in parallel

<table>
<thead>
<tr>
<th>Comparisons</th>
<th>VLIW</th>
<th>Superscalar</th>
</tr>
</thead>
<tbody>
<tr>
<td>Issue and execute</td>
<td>Four and eight instructions per clock cycle</td>
<td>Two and four instructions per clock cycle</td>
</tr>
<tr>
<td>Instruction Scheduling</td>
<td>Static</td>
<td>Dynamic</td>
</tr>
<tr>
<td>Compiler Loading</td>
<td>Need to solve hazard =&gt; Heavy</td>
<td>No need to solve hazard =&gt; Low</td>
</tr>
<tr>
<td>Need of Dedicated Hardware</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Approach Popular?</td>
<td>Popular</td>
<td>Not Common</td>
</tr>
</tbody>
</table>

*VLIW: Very Long Instruction Words*
## Evolution of DSP Processors

<table>
<thead>
<tr>
<th>Year</th>
<th>Generation</th>
<th>Feature</th>
<th>Example</th>
</tr>
</thead>
</table>
| 1982 | 1st        | Basic Harvard  
1 data bus, 1 program bus  
1 multiply-ALU unit       | TMS32010  
NEC 7720         |
| 1986 | 2nd        | “Modified” Harvard  
1 data/program bus, 1 data bus | TMS320C25  
AT&T DSP16A        |
| 1990 | 3rd        | Extra addressing modes  
Extra functions          | TMS320C5x  
AT&T DSP161x        |
| 1994 | 4th        | 1 data bus, 1 program bus  
Separate MAC, ALU       | TMS320C54          |
| 1995 | 5th        | 2 data buses, 1 program bus  
2 MACs, 1 ALU           | Lode            |
Data Path

- **Multiplier and accumulator (MAC)**
  - Multiply-accumulate (MAC) operation can be carried out in one instruction cycle

- **Arithmetic logic units (ALU)**
  - Single instruction cycle
  - Add/subtract, increment, negate
  - Logical AND, OR, NOT and XOR

- **Shifter**
  - Multiplication and accumulation
  - Scale the input by a power of 2
  - *Limited-capability* shifter: permits 1-bits shift at a time
  - *Barrel* shifter: support shifts by any number of bits in a single instruction cycle

- **Overflow and Saturation**
  - Overflow: the magnitude of the sum may exceed the max or min value which can be represented by the accumulator register ⇒
    - Scaling down: it is unable to maintain the signal fidelity
  - Saturation arithmetic: a special circuit detects overflow and the largest positive number or the smallest negative number is selected as the output
SIMD (Single-Instruction Multiple-Data)

- An architecture technique but not a class of DSP processor architecture.
- SIMD improves performance on some algorithms by allowing the processor to execute multiple instances of the same operation in parallel using different data.
- SIMD is only effective in algorithms that can process data in parallel; for algorithms that are inherently serial, SIMD is generally not of use.

**TigerSHARC SIMD**

```
SIMD MAC instruction

ALU  MAC  Shift
Four 16-bit x 16-bit multiplications

ALU  MAC  Shift
Four 16-bit x 16-bit multiplications
```
Zero-Overhead Looping

- Vast majority of processing time are spent in relatively small sections of software such as loops.
- A special loop or repeat instruction is provided which allows the programmer to implement a for-next loop without expending any clock cycles for updating and testing the loop counter or branching back to the top of the loop.
Fractional Operation and Overflow Detection (1/3)

\[
X = -x_{s-1} + \sum_{i=0}^{s-2} x_i 2^{-s+1+i} \\
Y = -y_{s-1} + \sum_{j=0}^{s-2} y_j 2^{-s+1+j}
\]

\[
P_{\text{Standard}} = -1 \times 2 + x_{s-1} y_{s-1} + \sum_{i=0}^{s-2} \sum_{j=0}^{s-2} x_i y_j 2^{i+j-2s+2} + 2^{-s+1} \left( \sum_{j=0}^{s-2} x_{s-1} y_j 2^j + 1 \right)
\]

\[
+ 2^{-s+1} \left( \sum_{i=0}^{s-2} y_{s-1} x_i 2^i + 1 \right)
\]

\[
(P_{2s-1}, P_{2s-2}) = (0, 0) , (0, 1) , \text{ and } (1, 1)
\]

\[
\hat{P}_{2s-1} = P_{2s-1}
\]

\[
\hat{P}_{2s-i} = MUX (1, \hat{P}_{2s-i-1}) \quad \text{for} \quad i = 2, 3, \ldots, s - 1
\]

\[
\hat{P}_s = \frac{P_{2s-1} P_{2s-2}}{P_{2s-1} P_{2s-2}}
\]

Lan-Da Van

VLSI-DSP-15-18
## Fractional Operation and Overflow Detection (2/3)

<table>
<thead>
<tr>
<th>Binary Point</th>
<th>Binary Point</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_{2s-1}$</td>
<td>$\hat{P}_{2s-1}$</td>
</tr>
<tr>
<td>$P_{2s-2}$</td>
<td>$\hat{P}_{2s-2}$</td>
</tr>
<tr>
<td>$\ddots$</td>
<td>$\ddots$</td>
</tr>
<tr>
<td>$P_{2s-3}$</td>
<td>$\hat{P}_{2s-3}$</td>
</tr>
<tr>
<td>$P_{2s-4}$</td>
<td>$\hat{P}_{2s-4}$</td>
</tr>
<tr>
<td>$\ddots$</td>
<td>$\ddots$</td>
</tr>
<tr>
<td>$P_s$</td>
<td>$\hat{P}_s$</td>
</tr>
</tbody>
</table>

| $P_{2s-3}$  | $P_{2s-4}$  |
| $\ddots$    | $\ddots$    |

| $\alpha$ | $\beta$ |

MUX

Output

### Diagram

- A diagram showing a multiplexer (MUX) with inputs $\alpha$ and $\beta$, and output labeled "Output".
Fractional Operation and Overflow Detection (3/3)

\[
P_{2s-1} P_{2s-2} P_{2s-3} P_{2s-4} \rightarrow \cdots \rightarrow P_{s+1} P_s
\]

\[
\hat{P}_{2s-1} \hat{P}_{2s-2} \hat{P}_{2s-3} \hat{P}_{2s-4} \rightarrow \cdots \rightarrow \hat{P}_{s+1} \hat{P}_s
\]
Efficient Memory Architecture and Access

- **Parallel memory banks**
  - A larger memory space is required to store the program, or more memory cycles are required to access each multiword instruction.
  - Instruction is stored in one memory bank, while data is stored in another. Thus, processor could fetch an instruction and a data operand in parallel in each cycle.
  - Register-indirect or circular addressing modes are used

- **Address generation units (AGUs)**
  - Provide one or more complex address calculation per instruction cycle without using the data path

- **Reduce the number of memory accesses**
  - Program cache, modulo addressing mode
  - Cache reduces the number of accesses of the bus

- **Specialized instructions**
  - Fully utilize the hardware resources and reduce the program size
Benchmark Scores: BDTImark2000™

- Agere Systems DSP164xx (240 MHz) - 1150 *
- Analog Devices ADSP-21xx (80 MHz) - 240
- Analog Devices ADSP-219x (160 MHz) - 420
- ADI Blackfin/Analog Devices ADSP-BF53x (600 MHz) - 3280
- Hitachi SH3-DSP (200 MHz) - 490
- Intel PXA2xx (XScale) (400 MHz) - 930
- LSI Logic ZSP400/LSI Logic LSI40x (200 MHz) - 940
- Motorola DSP563xx (240 MHz) - 710
- Motorola DSP568xx (80 MHz) - 110
- StarCore SC140/Motorola MSC8101 (300 MHz) - 3430
- Texas Instruments TMS320C54xx (160 MHz) - 500
- Texas Instruments TMS320C55xx (200 MHz) - 970
- Texas Instruments TMS320C62xx (300 MHz) - 1920
- Texas Instruments TMS320C64xx (600 MHz) - 5320

* for one of two on-chip cores

- Analog Devices ADSP-2106x (66 MHz) - 250
- Analog Devices ADSP-2116x (100 MHz) - 510
- Hitachi SH-4 (240 MHz) - 750
- Intel Pentium III (1,400 MHz) - 3130
- Texas Instruments TMS320C67xx (225 MHz) - 1100
Development Tools

Diagram showing the development tools process, including:
- Assembly Libraries
- Object Code Libraries
- Assembly Code
- Assembler
- Linker
- Binary Executable
- Simulator
- Final Product
- DSP Development Board
- In-Circuit Emulator
- Debugger
Programmable Digital Signal Processor Features (1/2)

- Low power, low cost, high signal processing capabilities, high flexibility, low time-to-market

- Data path
  - Support repetitive, numerically intensive tasks
  - Fast multiply-accumulate unit
  - Pipelining (structural level parallelism)
    - Break a sequence of instruction stream into several pipelining stages and allow multiple instructions to be executed in parallel
    - Speed up the computation and increase the performance, but makes programming complicated and difficult

- Memory
  - Multiple-access memory architecture => Move data to and from memory quickly

- Special instruction sets
Programmable Digital Signal Processor Features (2/2)

- Most DSP processors use a fixed-point numeric data type instead of the floating-point format.
  - Fixed point: the binary point is located at a fixed location in the data word.
  - Floating point: the binary point is **not** located at a fixed location in the data word.

- Most fixed-point DSP processors use 16-bit data words, because that data word width is sufficient for many DSP applications.

- High-fidelity audio processing requires more than 16-bit DSP processor to obtain better accuracy such as 20, 24, 32-bit.

- Accumulator registers are typically wider than other registers.
Outlines

- Introduction
- Evolution and Features of Programmable DSP Processors
- *DSP Processors for Multimedia and Communications*
- Conclusions
Multimedia Signal Processing

 Algorithmic characteristics
  ■ Frequent use of small integer operands
  ■ High regular computation-intensive operations
  ■ Intensive input/output (I/O) or memory access,
  ■ High data reusability
  ■ Strong data locality

 Media processors
  ■ High data rate, real-time processing capability
  ■ They can be used as stand-alone CPUs or as multimedia accelerators in PCs

 General purpose microprocessors
  ■ Use new instructions and resources by *Multimedia extensions* to improve the performance
Multiprocessing

- SIMD (single instruction multiple data)
  - The same instruction is executed by multiple processors on different data streams
  - Each processor has its own data memory, but there is only one instruction memory and control processor
  - It is suitable for algorithm with **high data parallelism and little data-dependent control flow**

- MIMD (multiple instruction multiple data)
  - Each processor fetches its own instructions and operates on its own data
  - The centralized shared-memory, or the individual memory
  - **Flexible** ⇒ To run task by single or multiprogrammed machines
  - Control hardware is complicated and expensive ⇒ High instruction memory bandwidth and synchronization of data paths

**Diagram:**

- **SIMD**
  - Program
  - Processor

- **Hybrid**
  - Program
  - Processor

- **MIMD**
  - Program
  - Processor

*Enhancements for Multimedia Signal Processing (1/6)*

Lan-Da Van

VLSI-DSP-15-28
Enhancements for Multimedia Signal Processing (2/6)

Subword Parallelism: Data Level Parallelism

- The low-precision data (< 16-bit) in multimedia signal processing cannot exploit full utilization of general processors’ (32- or 64-bit) computation power

- Split Processors
  - Functional units can be split into submodules ⇒ It increases the effective memory bandwidth
  - Small-scale SIMD parallelism
LIW/VLIW (Very Long Instruction Word): Instruction-Level Parallelism

- (V)LIW processors
  - They can be viewed as the small-scale MIMD processors with centralized memory and multiple functional units which enable simultaneous execution of all operations within a single instruction word.

- The VLIW operations rely on the compiler to perform static scheduling (at software level) in contrast to superscalar architectures.
  - VLIW architectures shift complexity from hardware to software.
  - The efficiency depends on the degree of instruction level parallelism in a particular algorithm.

```
vy1← mem2
v1← u0
v1=y0+tmp2
vy1=cosθ*y1
vy1=y0-tmp2
```

Control Unit

Function Unit

Function Unit

Function Unit

Function Unit

Function Unit

Lan-Da Van

VLSI-DSP-15-30
Enhancements for Multimedia Signal Processing (4/6)

Vector Processors

- Computation-intensive tasks (loops) can be efficiently coded and executed
- i.e. Matrix-vector multiplication

\[
\begin{bmatrix}
A0 \\
A1
\end{bmatrix} = \begin{bmatrix}
M00 & M01 & M02 & M03 \\
M10 & M11 & M12 & M13
\end{bmatrix} \times \begin{bmatrix}
V0 \\
V1 \\
V2 \\
V3
\end{bmatrix}
\]

\[
\begin{array}{c}
V0 \\
V1 \\
V0 \\
V1 \\
M00 \\
M01 \\
M10 \\
M11 \\
V2 \\
V3 \\
V2 \\
V3 \\
M02 \\
M03 \\
M12 \\
M13
\end{array}
\]

\[
\begin{array}{c}
V0 \cdot M00 + V1 \cdot M01 \\
V0 \cdot M10 + V1 \cdot M11 \\
V2 \cdot M02 + V3 \cdot M03 \\
V2 \cdot M12 + V3 \cdot M13
\end{array}
\]

\[
\begin{array}{c}
A0 \\
A1
\end{array}
\]
Enhancements for Multimedia Signal Processing (5/6)

**Accelerator Model**
- Easy to integrate
- No extra instruction set
- Direct Memory Access (DMA)
  - Meet High memory bandwidth requirement
- Reconfigurable Accelerator
  - RCA
  - Configuration controller and memory
  - Register File

**Coprocessor**

RCA: Reconfigurable Computation Array
Enhancements for Multimedia Signal Processing (6/6)

Special Instruction Sets
- Used for frequently recurring operations with higher complexity
- Instruction count ↓
- The speed of program execution ↑

Two goals in mind
- Minimize the amount of memory space required to store DSP program
- Maximize use of the processor’s underlying hardware

How to achieve?
- Specify several parallel operations in a single instruction for the first goal.
- Instructions are kept short and simple for the second goal.
## Media Processors (1/2)

<table>
<thead>
<tr>
<th>Processors</th>
<th>TI C8x</th>
<th>Chromatic MPact</th>
<th>Philips Tri-Media</th>
<th>TI C6x</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture</td>
<td>4*64b DSPs</td>
<td>VLIW/SIMD</td>
<td>VLIW</td>
<td>VLIW (VeliciTI)</td>
</tr>
<tr>
<td></td>
<td>+32b RISC</td>
<td>4 ALUs</td>
<td>+27 Exe. Units</td>
<td>8 instr./clk</td>
</tr>
<tr>
<td></td>
<td>+cross-bar</td>
<td>+ME engine</td>
<td>+Video/Audio</td>
<td>2 MACs/clk</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+792b bus</td>
<td>+VLD units</td>
<td></td>
</tr>
<tr>
<td>Technology</td>
<td>-</td>
<td>0.5μ</td>
<td>-</td>
<td>0.25μ</td>
</tr>
<tr>
<td># Trans.</td>
<td>4M</td>
<td>1.5M</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Chip Size (mm²)</td>
<td>340</td>
<td>100</td>
<td>-</td>
<td>196</td>
</tr>
<tr>
<td>Clock (MHz)</td>
<td>40</td>
<td>62</td>
<td>100</td>
<td>200</td>
</tr>
<tr>
<td>Perk Rate</td>
<td>1.2 GOPs/sec</td>
<td>2 GOPs/sec</td>
<td>4 GOPs/sec</td>
<td>1.6 GOPs/sec</td>
</tr>
<tr>
<td>Power (watts)</td>
<td>8</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Memory</td>
<td>DRAM 400 MB/s</td>
<td>RAMBUS 500 MB/s</td>
<td>SDRAM 400 MB/s</td>
<td>SDRAM 400 MB/s</td>
</tr>
<tr>
<td>Programming</td>
<td>compiler +assembler</td>
<td>in-house</td>
<td>VLIW compiler</td>
<td>VLIW compiler</td>
</tr>
</tbody>
</table>
### Media Processors (2/2)

<table>
<thead>
<tr>
<th>Processors</th>
<th>TI C8x</th>
<th>Chromatic MPact</th>
<th>Philips Tri-Media</th>
<th>TI C6x</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Support Application</strong></td>
<td>Desktop videoconferencing, videophones, 3D virtual reality, real-time MPEG-1 decompression, high-quality audio processing, Document image processing….</td>
<td>real-time MPEG-1 encoding, MPEG-1 &amp; -2 decoding, 2D/3D graphics, high-quality audio processing, fax/modem, video conferencing….</td>
<td>Multimedia Application…</td>
<td>Cable modems, central office switches, digital imaging, advanced multifunction wireless PDAs, xDSL systems..</td>
</tr>
</tbody>
</table>
Vector DSP (VDSP) Media Processors

- Standards
  - CCITT H.261/H.263/H.264/AVC
  - MPEG (Moving Picture Experts Group)
  - JPEG (Joint Photographic Experts Group)

- Key Kernels
  - Motion vector detection (MVD)
  - Motion compensation (MC)
  - Discrete cosine transform (DCT)
  - Variable length coding/decoding (VLC/VLD)

- Need Vector-pipeline (VP) architecture for DSP Processor design

VDSP Architecture (1/2)
VDSP Architecture (2/2)

- **Arithmetic components**
  - A 16-b enhanced ALU
  - A 16-b x 16-b Booth multiplier with a Wallace tree
  - A 24-b accumulator
  - A DCT core
  - A Spatical filter (loop filter) core

- **Storage components**
  - A 32-b x 1024-word instruction RAM (IRAM)
  - A 32-b x 512-word instruction ROM (IROM)
  - Three 16-b x 1024-word data RAM (BM1, BM2, DM)
  - Five space address generators (SAG)
    - Capable of accessing a block of data in form of vector data
  - Two external memory ports
    - 24-b address port (A1, A2) and 16-bit data port (D1, D2)
# Instruction Set (1/2)

## (a) Data Transfer Operation

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SET ,</td>
<td>Register ← Immediate data</td>
</tr>
<tr>
<td>LD</td>
<td>Register ← Internal data memory</td>
</tr>
<tr>
<td>STR</td>
<td>Internal data memory ← Register</td>
</tr>
<tr>
<td>MVD</td>
<td>Register ← Register</td>
</tr>
<tr>
<td>LDEX</td>
<td>Internal data memory ← External data memory</td>
</tr>
<tr>
<td>STREX</td>
<td>External data memory ← Internal data memory</td>
</tr>
<tr>
<td>TRS</td>
<td>DAM start</td>
</tr>
</tbody>
</table>

## (b) Arithmetic and Logic Operation

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD</td>
<td>FREG ← S1 + S2</td>
</tr>
<tr>
<td>ADDC</td>
<td>FREG ← S1 + S2 + S flag</td>
</tr>
<tr>
<td>ADDX</td>
<td>FREG ← S1 + S2 + carry</td>
</tr>
<tr>
<td>SUB</td>
<td>FREG ← S1 - S2</td>
</tr>
<tr>
<td>SUBX</td>
<td>FREG ← S1 - S2 - borrow</td>
</tr>
<tr>
<td>ASUB</td>
<td>FREG ←</td>
</tr>
<tr>
<td>CMP</td>
<td>S1 - S2</td>
</tr>
<tr>
<td>NEG</td>
<td>FREG ← 0 - S1</td>
</tr>
<tr>
<td>MAX</td>
<td>FREG ← max (S1, S2)</td>
</tr>
<tr>
<td>MIN</td>
<td>FREG ← min (S1, S2)</td>
</tr>
<tr>
<td>AND</td>
<td>FREG ← and (S1, S2)</td>
</tr>
<tr>
<td>OR</td>
<td>FREG ← or (S1, S2)</td>
</tr>
<tr>
<td>EOR</td>
<td>FREG ← exor (S1, S2)</td>
</tr>
<tr>
<td>INV</td>
<td>FREG ← not (S1)</td>
</tr>
<tr>
<td>SIGN</td>
<td>FREG ← sign (S1)</td>
</tr>
<tr>
<td>CLIP</td>
<td>FREG ← clip (S1, S2, S3)</td>
</tr>
<tr>
<td>CLIP0</td>
<td>FREG ← clip0 (S1, S2)</td>
</tr>
<tr>
<td>MPYLY</td>
<td>MREG ← S1 × S2 (lower 16 b)</td>
</tr>
<tr>
<td>MPYLU</td>
<td>MREG ← S1 × S2 (upper 16 b + round)</td>
</tr>
</tbody>
</table>
### Instruction Set (2/2)

#### (c) Vector Pipeline Operation

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QAA</td>
<td>LD(L-SFT)-(ALU)-Acc</td>
</tr>
<tr>
<td>QMA</td>
<td>LD(L-SFT)-(MPY)-Acc</td>
</tr>
<tr>
<td>QAMAA</td>
<td>LD(L-SFT)-(ALU)-(MPY)-Acc</td>
</tr>
<tr>
<td>P(ALU)</td>
<td>LD(L-SFT)-(ALU)-STR(R-SFT)</td>
</tr>
<tr>
<td>P(MPY)</td>
<td>LD(L-SFT)-(MPY)-STR(R-SFT)</td>
</tr>
<tr>
<td>PMVD</td>
<td>LD(L-SFT)-STR(R-SFT)</td>
</tr>
<tr>
<td>PZZ</td>
<td>LD(BM1)-LD(IP)-STR</td>
</tr>
<tr>
<td>FDCTX</td>
<td>LD(L-SFT)-DCT(X)-STR(R-SFT)</td>
</tr>
<tr>
<td>FDCTY</td>
<td>LD(L-SFT)-DCT(Y)-STR(R-SFT)</td>
</tr>
<tr>
<td>IDCTX</td>
<td>LD(L-SFT)-IDCT(X)-STR(R-SFT)</td>
</tr>
<tr>
<td>IDCTY</td>
<td>LD(L-SFT)-IDCT(Y)-STR(R-SFT)</td>
</tr>
<tr>
<td>FLTXY</td>
<td>LD(L-SFT)-FLT(X)-STR(R-SFT)</td>
</tr>
<tr>
<td>FLTY</td>
<td>LD(L-SFT)-FLT(Y)-STR(R-SFT)</td>
</tr>
</tbody>
</table>

#### (d) Other Control Operation

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SI</td>
<td>Serial in start</td>
</tr>
<tr>
<td>SO</td>
<td>Serial out start</td>
</tr>
<tr>
<td>JMP</td>
<td>Unconditional jump</td>
</tr>
<tr>
<td>Jcc</td>
<td>Conditional jump (cc: condition code)</td>
</tr>
<tr>
<td>CALL</td>
<td>Subroutine call</td>
</tr>
<tr>
<td>RTS</td>
<td>Subroutine return</td>
</tr>
<tr>
<td>RTL</td>
<td>Loop return</td>
</tr>
<tr>
<td>WAIT</td>
<td>I/O synchronization</td>
</tr>
<tr>
<td>INIT</td>
<td>Initialization of SAG/CNT</td>
</tr>
<tr>
<td>HOLD</td>
<td>State transition to hold/slave mode</td>
</tr>
<tr>
<td>STOP</td>
<td>State transition to hold/slave mode</td>
</tr>
</tbody>
</table>
Instruction Configuration

**P-Instruction**

**Q-Instruction**

\[ Y_i = A_i \# B_i \quad (i = 1, 2, 3, ..., n) \]

\[ Y_i = \sum_{i=0}^{n} A_i \# B_i \quad (i = 1, 2, 3, ..., n) \]
Vector Pipeline Controller

program controller

Branch Controller

PC

IRAM

IREG

DEC

Vector-pipeline controller

REG

Processing units

s-SAG
d-SAG

start signal
end signal
start signal

Vector-pipeline control signal
Timing Chart of the Vector-Pipeline Instruction

- Instruction
- Vector-pipeline instruction cycle
- Pipeline-delay cycles
- Start signal to s-SAG
- End signal from s-SAG
- Vector read
- Operation 1
- Operation 2
- Vector write or accumulation
- Start signal to d-SAG
- Vector data size
- N: vector-pipeline instruction address

Diagram details:
- Stop PC to restart PC
- Timing cycles for vector operations
- pipeline-delay cycles
DCT Core Design Using DA (a) DCT Core (b) ROM/Accumulator (RAC)
Performance Evaluation of P-Instruction

- The address calculation of data with a subscan
- The address calculation of data of the new subscan start address
- The examination of the end of the scan

\[ Y_i = A_i \times B_i \]
Performance Evaluation of Q-Instruction

VLSI Digital Signal Processing Systems

Performance Evaluation of Q-Instruction

- Two-dimensional address operations: 3 op. × 2 address
- Read from internal memories: 1 op. × 2 data
- Shifting two source data: 1 op. × 2 data
- ALU operation: 1 op.
- Multiplier operation: 1 op.
- Accumulator operation: 1 op.

Total: 13 operations

\[ Y = \sum_{i=1}^{n} (A_i \times B_i)^2 \]
Performance Evaluation of DCT-Instruction

34 operations

two-dimensional address operations 3 op. × 1 address
read from internal memory 1 op. × 1 data
shifting one source data 1 op. × 1 data
DCT operation 3 op. × 8 conv.
shifting destination data 1 op. × 1 data
two-dimensional address operation 3 op. × 1 address
write to an internal memory 1 op. × 1 data

TOTAL 34 operations
Features of VDSP

34*60 MHz = 2.04 GOPS

Data Format
Instruction Cycle
Internal Memory

External Data Area
Processing Units

Address Generator
I/O Type
Power Supply Voltage
Power Consumption
Peak Performance
Package
Integrated Element Number
Fabrication Technology

16-b fixed point
16.7 ns (60 MHz)
Data: 1K Words × 16 b × 3
Program: 1.5K Words × 32 b
16M Words × 16 b × 2
Enhanced ALU, MPY,
Accumulator, Shifter × 4
DCT/IDCT, Loop Filter
2D-SAG × 5
TTL I/F
5.0 V
2.4 W (at 60 MHz)
2.04 GOPS (at 60 MHz)

Note: The critical path was determined to be the multiplier with the operation of 16.5 ns.
Benchmark Test: H.261

<table>
<thead>
<tr>
<th>Operation</th>
<th>Time (μs/MB)</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data in (Y) and out (Y, U, V)</td>
<td>36.00</td>
<td>in/out parallel</td>
</tr>
<tr>
<td>MVD</td>
<td>126.69</td>
<td>3 step search</td>
</tr>
<tr>
<td>MC on/off</td>
<td>1.52</td>
<td></td>
</tr>
<tr>
<td>Loop Filter</td>
<td>15.49</td>
<td></td>
</tr>
<tr>
<td>Inter/Intra</td>
<td>8.45</td>
<td></td>
</tr>
<tr>
<td>Subtract</td>
<td>1.18</td>
<td></td>
</tr>
<tr>
<td>Data in (U, V)</td>
<td>5.12</td>
<td></td>
</tr>
<tr>
<td>DCT</td>
<td>19.46</td>
<td></td>
</tr>
<tr>
<td>Q/IQ</td>
<td>76.87</td>
<td></td>
</tr>
<tr>
<td>VLC</td>
<td>5.37</td>
<td>typical image</td>
</tr>
<tr>
<td>IDCT</td>
<td>19.75</td>
<td></td>
</tr>
<tr>
<td>Add</td>
<td>1.18</td>
<td></td>
</tr>
<tr>
<td>TOTAL</td>
<td>317.08</td>
<td></td>
</tr>
</tbody>
</table>

**Operation Time per MB of the Decoder of CCITT H.261**

<table>
<thead>
<tr>
<th>Operation</th>
<th>Time (μs/MB)</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>VLD + IQ</td>
<td>13.67</td>
<td>typical image</td>
</tr>
<tr>
<td>IDCT</td>
<td>9.79</td>
<td></td>
</tr>
<tr>
<td>Data in (MC)</td>
<td>12.87</td>
<td></td>
</tr>
<tr>
<td>Loop Filter</td>
<td>12.70</td>
<td></td>
</tr>
<tr>
<td>Add</td>
<td>2.63</td>
<td></td>
</tr>
<tr>
<td>Data out</td>
<td>12.80</td>
<td></td>
</tr>
<tr>
<td>TOTAL</td>
<td>64.40</td>
<td></td>
</tr>
</tbody>
</table>
Mobile and Wireless Communications Application

- Desired factors
  - Low area
  - Low power

- VLIW & SIMD can improve the processor performance, but they are expensive in area and power

- An application-specific concept keeping in mind is used for the targeted application
  - Square distance and accumulate for vector quantization
  - Add-compare-select for Viterbi algorithm
  - Galois field operations for forward error-control coding
Conclusions

Requirements for DSP Processors

- Flexibility
  - The applications and their standards changing faster than their system counterparts
- High throughput
  - On the order of tens to hundreds of GOPS
- Low power and low cost
  - The need for desktop and portable solutions with reasonably silicon costing

Multimedia Signal Processing

- Some forces to increase speed, decrease energy consumption, decrease memory usage, and decrease cost.
- Need for architectures that facilitate development of more efficient compilers, allowing DSP applications to be written primarily in high-level languages.
- Real time video processing capability
  - Two VDSP’s for encoder
  - One VDSP for decoder

Mobile and Wireless Communications
References


