DataWave
A Data-Driven Video Signal Array Processor

Ulrich Schmidt
ITT Intermetall, Freiburg, Germany

Contents
- Hardwired vs. Programmable Solutions
- Architecture of the Data-Driven Array Processor
- Application Examples
- A Look into the Future
Hardwired vs. Programmable Solutions

Arguments for "structured hardware"
- Transistors/chip: 1 million today (1990), 100 million tomorrow (2000)
- Adding dedicated functions adds only marginal value
- Dedicated functions waste chip area if not needed
- Do not dedicate at design time, dedicate at run time!
- Program changes are done faster than hardware changes
- Regular structures are easier to design and test than irregular structures

Application Areas
- 3 – 144 MHz video signals
- Today: NTSC / PAL / SECAM / D2-MAC
- Tomorrow: EDTV / IDTV / HDTV
- Real-time motion-compensated scan rate conversion
- Image compression/decompression (DCT, sub-band)
- Graphics + Video
- Image analysis (robot vision)

> Required computational throughput > 10 GOPS
> Required I/O throughput > 100 MByte/s
Architecture of the Data-Driven Array Processor

Architecture
- 16 mesh-connected cells
- Multiple instructions, multiple data
- 12-bit word length

Technology
- 0.8 μm CMOS
- 125 MHz on-chip (8 ns)
- 62.5 MHz off-chip (16 ns)
- 124 pins

Performance
- 4000 MOPS
- 750 MByte/s chip I/O
- 1125 MByte/s cell I/O

One-dimensional Cascading of Array Processors

- 12-bit transfer (logical)
- 12-bit transfer (physical)
- 24-bit transfer (logical)
- 24-bit transfer (physical)
- 36-bit transfer (logical)
- 36-bit transfer (physical)
- 48-bit transfer (logical)
- 48-bit transfer (physical)
Two-dimensional Cascading of Array Processors

12-bit transfer

24-bit transfer (logical)

24-bit transfer (physical)

Data Flow Communication

Architecture
- Sequencing instead of timing
- Data-driven instead of clock-driven
- Cells "freeze" if neighbours are not ready
- Local changes do not cause global effects
- Executes statically scheduled data flow programs

Implementation
- Common 125 MHz clock
- Self-timed asynchronous handshake protocol
- "Freeze" by automatic insertion of wait cycles
- Completely transparent: no polling, no interrupts
- 1 cycle / intercell transfer
- 2 cycles / interchip transfer
Cell Architecture

Superscalar RISC Architecture

- Wallace-tree multiplier
- Multiply-Accumulate in every cycle
- 3 ring buses, 3 core buses
- 64 instructions in static RAM, 48 bits wide
- Four-port register file, 16 registers
- Deep pipeline with short feed-back loops
- 12-bit ALU, 1 cycle delay
- n 1-bit shifts/rotates in n cycles

Intercell Communication

- Up to 6 cell-to-cell transfers every cycle
- Self-timed handshake with FIFOs
- Dual-port circular buffers (8 words)
- Automatic pipeline freeze when waiting

Instruction Format

<table>
<thead>
<tr>
<th>47</th>
<th>40</th>
<th>35</th>
<th>27</th>
<th>22</th>
<th>17</th>
<th>12</th>
<th>10</th>
<th>8</th>
<th>6</th>
<th>4</th>
<th>2</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpCode</td>
<td>Condition</td>
<td>Branch Address</td>
<td>A</td>
<td>B</td>
<td>Reg</td>
<td>E</td>
<td>N</td>
<td>W</td>
<td>S</td>
<td>CM</td>
<td>RM</td>
<td></td>
</tr>
</tbody>
</table>

Constant

2 Sources 5 Destinations 2 Mux

Large Instruction Word:
- Arithmetic + Intercell Communication + Register Transfers + Branches
- Horizontally coded
- No special load/store instructions: communication ports are treated like registers
- Branches are delayed by 3 cycles

Example:

\[ n = s = r13 = acc + w \times r12, \quad e = w, \quad \text{bra Label} \]
### FIR Filter

![FIR Filter Diagram](image)

Loop
\[
\text{acc} = r1 \times w, \quad r11 = w \\
\text{acc} = \text{acc} + r2 \times r11, \quad r12 = r11, \quad \text{bra Loop} \\
\text{acc} = \text{acc} + r3 \times r12, \quad r13 = r12 \\
\text{acc} = \text{acc} + r4 \times r13, \quad r14 = r13 \\
e = \text{acc} + r5 \times r14
\]

Note: 12-bit I/O, 29-bit accumulation
One FIR tap per cycle
Loop diameter = clock rate / sample rate
Sustained performance: 250 MOPS

### IIR Filters

![IIR Filters Diagram](image)

Loop
\[
\text{mid} = w \\
e = r1 = \text{acc} + h1 \times r1 \\
\text{nop} \\
\text{bra Loop} \\
\text{nop} \quad ; \text{delay slot 1} \\
\text{nop} \quad ; \text{delay slot 2} \\
\text{nop} \quad ; \text{delay slot 3}
\]

First-order recursive filter

Loop
\[
\text{mid} = w \\
\text{acc} = \text{acc} + h2 \times r2 \\
e = r1 = \text{acc} + h1 \times r1 \\
r2 = r1, \text{bra Loop} \\
\text{nop} \\
\text{nop} \\
\text{nop}
\]

Second-order recursive filter
**2D (8 × 8) Discrete Cosine Transform**

**Formula:**

\[ Y = \text{DCT} \times (\text{DCT} \times X)^T \]

where

- \( X \): (8 × 8) frame buffer region
- \( \text{DCT} \): (8 × 8) coefficient matrix

**Assumptions:**
- Sample rate: 13.5 MHz
- Clock rate: \( \geq 108 \) MHz
- 8 cycles/sample

**Transposition:**

- Store in column order
- Read in row order

---

**Simulation and Debugging Environment**
**DataWave – Technical Data**

- 0.8 micron double-metal CMOS
- 125 MHz clock
- 1.2 million transistors
- 150 mm$^2$ chip area
- 124 pins
- 8 W maximum power dissipation

---

**What we did ...**

- Software for hardware
- Cellular array topology
- Process: 0.8 micron CMOS
- Density: 16 cells/chip
- Clock rate: 125 MHz
- Performance: 4 GOPS

---

**... what we are going to do**

- Software unchanged
- Array topology unchanged
- Process: 0.4 micron CMOS
- Density: 64 cells/chip
- Clock rate: 250 MHz
- Performance: 32 GOPS