Technology Trends of Embedded Multimedia System Design

C.-C. Jay Kuo
Department of Electrical Engineering
University of Southern California
Los Angeles, CA 90089-2564
Introduction (1)

Keywords

- Portability
  - Pervasive computing
  - Ubiquitous computing

- Communication capability
  - Personal communication system
  - 3G
  - 4G
  - WLAN, Bluetooth
  - Overlay networks

- Multimedia capability
  - A/V capturing
  - A/V display

- Marketing
  - Consumer electronics oriented
  - Rather than PC-oriented

Embedded processors
Introduction (2)

- Technical Challenges

- power management
- processor utilization
- security issue

![Battery](image)

![Graph](image)

- Blackfin 531
- DM642
- C6415
- TM1300

Speed (MHz)

0 200 400 600 800

001110100100
- Two-Chip Solution
  - Communication chip and Multimedia chip

- Embedded Media Processor Architecture
Example of Media Processors

- Trimedia: TM1300
  - Speech/Image/Video
  - Somehow, not very well received

- Equator media processor

- TI
  - DSC-25, DM-270, DM-320
  - OMAP for cellular phone
  - C64xx series

- Sunplus, Altek, etc.
New trend of embedded hardware design

Operation profiling and speed up for multimedia applications

Several design problems
- Power optimization
- Memory bank confliction resolution
- Loop unrolling
- Cryptographic operations
## Comparison of Embedded Processors and General Purpose (GP) Processors

<table>
<thead>
<tr>
<th></th>
<th>Embedded Media Processors</th>
<th>GP Processors</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Architecture</strong></td>
<td>VLIW</td>
<td>Superscalar</td>
</tr>
<tr>
<td><strong>Clock Speed</strong></td>
<td>100 – 600 MHz</td>
<td>2 – 4 GHz</td>
</tr>
<tr>
<td><strong>Cache Size</strong></td>
<td>16/16 KB (L1) Usu. no L2 cache</td>
<td>32 – 128KB(L1) L3 cache available</td>
</tr>
<tr>
<td><strong>Memory size</strong></td>
<td>8 – 256 MB</td>
<td>512 MB – 1GB</td>
</tr>
</tbody>
</table>
### Key Issues About Processor Design

<table>
<thead>
<tr>
<th>CPU</th>
<th>Memory &amp; I/O</th>
<th>OS overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instruction Set Architecture + Compiler</td>
<td>Size</td>
<td>Single-user or Multi-user</td>
</tr>
<tr>
<td>Micro-architecture implementation</td>
<td>Speed</td>
<td>Time-sharing</td>
</tr>
<tr>
<td>VLSI speed</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Embedded multimedia applications (EMAs) have stringent requirements

- Real-time performance
- Frequent and uniform memory access
- High computation complexity

Using multiple processors to increase performance and improve availability

- Single instruction stream, single data stream (SISD)
- Single instruction stream, multiple data streams (SIMD)
- Multiple instruction streams, single data stream (MISD)
- Multiple instruction streams, multiple data streams (MIMD)
Why is SIMD?
- Multimedia data’s low-precision
  - 8-bit pixels for image/video application
  - 16-bit samples for audio application
  ◆ Challenges: representation, storage and processing
- Multimedia algorithm’s inherit data parallelism
  - Add, subtract, and simple forms of multiplication and division are common operations

First developed by UIUC
- Used as imaging processing engine (CM series) in early days

Popular engine: Intel MMX, TI iMX
SIMD Example

DSPIADD R1, R1, R2

DSPUQUADADDUI R1, R1, R2

SISD Vs. SIMD

<table>
<thead>
<tr>
<th>SISD</th>
<th>SIMD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load A</td>
<td>Pack L1, A, C</td>
</tr>
<tr>
<td>Load B</td>
<td>Pack L2, B, D</td>
</tr>
<tr>
<td>Add A, B</td>
<td>SIMDADD L1, L2</td>
</tr>
<tr>
<td>Load C</td>
<td></td>
</tr>
<tr>
<td>Load D</td>
<td></td>
</tr>
<tr>
<td>Add C, D</td>
<td></td>
</tr>
</tbody>
</table>

Cycle count is reduced by 50%
Comparison of VLIW & Superscalar

VLIW

Superscalar

3-parallel executions per cycle

3-issue superscalar per cycle
Hyper-pipeline or super-pipeline

Super-pipeline with depth of 3
3 functional pipelines in parallel

3-issue superscalar super-pipeline
VLIW

- **Advantages**
  - Simpler hardware → cost is lower than superscalar
  - Allows multiple issues per clock cycle

- **Disadvantages**
  - Purely rely on compiler for scheduling → static scheduling only
  - Most of the compilers are not efficient

- **Challenges**
  - Exploit ILP efficiently: find out more instructions to be executed in parallel
  - Compatibility or flexibility: assembly codes are difficult to port
  - Code size
Content

- New trend of embedded hardware design
- Operation profiling and speed up for multimedia applications
- Several design problems
  - Power optimization
  - Memory bank confliction resolution
  - Loop unrolling
  - Cryptographic operations
Profiling in Video Coding Operations

Profiling of MPEG-4 Encoder

Profiling of MPEG-4 Decoder
Profiling in Video Coding Operations

Computational Profiling for MPEG-1 Decoder
Speeding up 8x8 DCT

- IFIR16

Performance Comparison of DCT

<table>
<thead>
<tr>
<th></th>
<th>Instruction Cycle</th>
<th>Issue Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fast DCT on pure C</td>
<td>472</td>
<td>3.24</td>
</tr>
<tr>
<td>Optimized DCT based on Trimedia ISA</td>
<td>160</td>
<td>4.42</td>
</tr>
</tbody>
</table>
Speeding up Motion Estimation

- Direct instruction to compute SAD
  - TM1300: UMEU88
  - TI DM642: SUBABS4

Performance Comparison of SAD Computation

<table>
<thead>
<tr>
<th></th>
<th>Instruction Cycle</th>
<th>Issue Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>8x8 MAD Computation on pure C</td>
<td>100</td>
<td>3.98</td>
</tr>
<tr>
<td>8x8 MAD Computation with UMEU88</td>
<td>21</td>
<td>3.43</td>
</tr>
</tbody>
</table>
Many media instructions can be used in interpolation (TM1300):

- QUADAVG, unsigned byte-wise quad average
- UFIR8UU, unsigned sum of products of four unsigned bytes
- UCLIPI, clip the operand into 0 to an unsigned number

<table>
<thead>
<tr>
<th>Instruction Type</th>
<th>Instruction Cycle</th>
<th>Issue Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interpolation on pure C</td>
<td>113,012</td>
<td>1.51</td>
</tr>
<tr>
<td>Interpolation with media</td>
<td>2,795</td>
<td>4.70</td>
</tr>
</tbody>
</table>
Content

- New trend of embedded hardware design
- Operation profiling and speed up for multimedia applications
- Several design problems
  - Power optimization
  - Memory bank confliction resolution
  - Loop unrolling
  - Cryptographic operations
Basic Principles

- **Activity Management**
  - Switch off inactive components when possible
  - Reduced bus switching
  - Reduce cycle counts

- **Memory Management**
  - Reduce memory size
    - Larger size or longer bit-width → long transfer path / complex address decoding...
    - More connectivity → More wiring → more power
  - Increase locality of memory access
    - Less data transfer from off-chip memories (using cache or buffer)
  - Reduce memory access frequency
Dynamic Power Management (1)

- Selective shut-off or slow-down of components
  - Effective way to reduce power dissipation
  - Much effort in the design, debug and validation

Approaches
- “Time-out” policy
  - Turn on a component when it is in use
  - Turn off a component when it is not used for some pre-specified length of time
    - The parameter can be selected based on the access pattern characteristics of the component
  - Limitations:
    - Cannot handle components with more than 2 states
    - Cannot handle complex system behaviors
    - No optimality guaranteed
Dynamic Power Management (2)

- Discrete-time Markov decision process
  - Four-component system model
  - Objective: expected performance (e.g. waiting time and no. of objects in the queue)
  - Constraint: expected power consumption

- Continuous-time Markov decision process
  - Event-driven PM (power manager)
  - Perform decision process only when necessary
Reduced Bus Transition/Switching (1)

- **Energy associated with bus transition**
  - More frequent transition $\rightarrow$ more power consumption
    
    $p_1 > p_2$

- **Bus encoding**
  - **Goal**
    - Reduce the number of bus transitions
  - **How**
    - Reduce the hamming distances between consecutive address and data transfers
  - **Techniques**
    - Gray, Pyramid, Working Zone, Bus Invert techniques, etc.
Data/Instruction organization

Reorganize data and/or instructions so that
- Consecutive memory references exhibit spatial locality
- Spatial locality leads to power efficiency
  - Smaller Hamming distance between closer addresses
  - Reduced bus switching activity
- Example of data organization
  - Storage schemes for an array: row-major, column-major, tile-based
  - Evaluate the their impact on bus switching, choose the optimal one
Memory and Data Optimization (1)
Memory and Data Optimization (2)

- Cache and scratch-pad memory management
  - Cache Management
    - Careful data layout to reduce cache conflict
    - Prefetching to increase cache hit ratio
    - Cache module assignment
      - Classify variables based on their spatial locality
      - Assign variables of different locality into different cache modules
  - Scratch-pad memory
    - Data cache + on-chip memory + off-chip memory
    - Data cache and on-chip memory
      - Both allow for fast access
      - Guaranteed single-cycle access by on-chip memory
      - Cache miss could occur in data cache
    - Key: Identify critical data to put in on-chip memory
DRAM optimization

- DRAM: different addressing from those of SRAM
  - Address = row address + column address
    - Row address: address of a page
    - Column address: offset within a page
  - Separate row address decoding and column address decoding
  - Each DRAM module has a page buffer
  - Each READ read a whole page into page buffer

- DRAM-oriented optimization
  - R-M-W optimization, hoisting, unrolling....

- DRAM multi-bank optimization
  - Each bank has its own page buffer
  - Simultaneous active memory pages is enabled
Memory packing and array-to-memory assignment

- Memory dimension determination
  - Pack all arrays into one single memory module
    - Energy waste caused by area waste
  - Pack each array into a separate memory module
    - Energy waste caused by connectivity and address decoding waste
  - Optimum lies between

- Algorithms
  - Combined horizontal and vertical array packing

Logical memories

Physical memories

Assignment

```
  a  b  c
```

Physical memories

Assignment

```
a
b
c
```
Memory and Data Optimization (5)

- Exhaustive search solution
  - Bit-width
  - Word count
  - Port number
- Recursive memory splitting
  - Start from a single port solution
  - Result in a distributed assignment

**In-place logical array mapping**
- Map different sections of logical arrays into the same physical memory, if their lifetimes are non-overlapping
- Goal: reduce physical memory size
Memory and Data Optimization (5)

- Register or multiport-memory allocation
  - Like register allocation as done in compiler
    - simultaneous access is limited by the port number
  - Advantages over separate registers:
    - Reduced interconnection cost
    - Selective connection of registers and ports

- Memory access scheduling
  - Based on DFG (data flow graph)
  - Goal: reduce no. of memory modules and memory ports for low power
Loop transformation

- Essential in power optimization
  - Large matrix data often manipulated in loops
  - Cache performance improvement
- Leads to reduced memory size and memory accesses
- Example

```c
for( i = 0; i < M; i++ )
    for( j = 0; j < N; j++ )
        for( k = 0; k < P; k++ )
            b[i][j] = a[i][j+k];

for( i = 0; i < M; i++ )
    for( j = 0; j < N; j++ )
    {  
        for( k = 0; k < L; k++ )  
            b[i][j] = a[i][j+k];  
    }
```
Instruction-level scheduling

- Power dissipation table (PDT) based scheduling
  - CDG (control&data dependency graph) construction
  - PDT construction
    - List the power consumption for each consecutive instruction pair
  - SCG (weighted strongly connected graph) construction
    - Based on the CDG
    - Each edge is weighted by the value in PDT
  - Find the Hamiltonian tour of the SCG

- An example of CDG and its search space is given in the next page
Power-aware Compiler Technology (3)

CDG

PDT & SCG

Start

1

2

3

4

5
Register allocation
- Minimization of the no. of registers in use after instruction scheduling
- The register count impacts
  - The area of the resulting design
  - The size of register storage
  - Thus, the power consumption
- Graph coloring approach
  - NP-complete problem
  - Many approximate algorithms proposed
Compiler-controlled power management

- Dynamically tradeoff power for performance
- Embedded systems follow specific power/energy profiles
- Compiler works with OS tightly via
  - Static analysis
  - Profile-driven data
  - Feedback-driven optimization
- An emerging research area
Content

- New trend of embedded hardware design
- Operation profiling and speed up for multimedia applications
- Several design problems
  - Power optimization
  - Memory bank confliction resolution
  - Loop unrolling
  - Cryptographic operations
Interleaved Memory and Bank Confliction

Indicates the memory bank index

Indicates which byte inside the bank

0 1 2 3
32+0 32+1 32+2 32+3

0x0 0x1 0x2 0x3

Bank 0

0 1 2 3
32N+0 32N+1 32N+2 32N+3

5 4 2 1 0

bank offset

Integer 0

Integer 8

Trimedia TM1300 Interleaved Memory Address Layout

Bank 0

Bank 1

Bank 7

32N+4 32N+5 32N+6 32N+7

32+4 32+5 32+6 32+7

32N+28 32N+29 32N+30 32N+31

32+28 32+29 32+30 32+31
Memory bank Conflict under Different Resolution and Stream Complexity

- As resolution increases from QCIF to CIF format, memory bank conflict increased significantly.
Cache Miss before and after Optimization (Three test sequences)

- As optimization goes, memory bank conflict gradually becomes the dominant data cache misses.
- Cache misses has important impact on the performance of EMA (30%)
Content

- New trend of embedded hardware design
- Operation profiling and speed up for multimedia applications
- Several design problems
  - Power optimization
  - Memory bank confliction resolution
  - Loop unrolling
  - Cryptographic operations
Loop unrolling: overlaps different iterations of the loop body, creates N copies of the loop.

```c
byte *p, *g, *m, *d;

for ( i=0; i<256; i++ ){ p[i] = g[i] - m[i]; d[i] = p[i] * s; }
p += stride; g += stride; m += stride; d += stride;
```

```
for ( j=0; j<256; j+=4 )
{
    p[j] = g[j] - m[j]; // (1)
    d[j] = p[j] * s; // (2)

    j1 = j+1;
    p[j1] = g[j1] - m[j1]; // (4)
    d[j1] = p[j1] * s; // (5)

    j2 = j+2;
    p[j2] = g[j2] - m[j2]; // (7)
    d[j2] = p[j2] * s; // (8)

    j3 = j+3;
    p[j3] = g[j3] - m[j3]; // (10)
    d[j3] = p[j3] * s; // (11)
}
```

Original Loop

Unrolled Loop
Memory Access Pattern Analysis

- Memory access is a major bottleneck on the performance of EMAs.
- It decides the loop unrolling strategy.

EMA memory access pattern can be classified in spatial domain as:
- Horizontal Access
- Vertical Access
- Multi-word Width Access
Horizontal Access

- A series of accesses to a string of consecutive memory locations
- Most common memory access pattern found in EMAs
- In DCT, motion search, intra-macroblock prediction, inter-macroblock prediction, and etc.

```c
for (j=0; j<4; j++)
{
    for (i=0; i<4; i++)
    {
        m7[i] = imgy_orig[i] - mpr[i];
    }
    imgy_orig += stride_imgy_orig;
    mpr += stride_mpr;
    m7  += 4;
}
```
Vertical Access

- A transpose access pattern relative to horizontal access
- In loop filter, intra-prediction, interpolation, chroma prediction function and etc.

```c
for (i=0; i<4; i++)
{
    for (j=0; j<4; j++)
    {
        mprr_vert[i+j*stride] = (byte)((&P_A)[i]);
    }
}
```
Multi-word Width Access

- memory access width that exceeds the common word boundary (32 bits)
- For example, Interpolation function
- Embedded media processor not as powerful as general processors
- In general, Instruction Level Parallelism (ILP) is no longer sufficient to meet the unique requirements of EMAs
- Compiler must have the ability to exploit Superword Level Parallelism (SLP) for EMAs
- Loop unrolling is the essential way and the first step to exploit SLP
- No SIMD-oriented loop unrolling algorithm available now
New trend of embedded hardware design
Operation profiling and speed up for multimedia applications
Several design problems
  - Power optimization
  - Memory bank confliction resolution
  - Loop unrolling
  - Cryptographic operations
Why Security Concern?

- Traditional security concern like integrity, privacy and authentication
  - Public transmission medium

- Software solution alone is not sufficient
  - Embedded system increasingly assembled from pre-designed components

- Side-channel attack techniques becomes menace
  - Fault analysis
  - Power analysis
  - Timing analysis
  - Template analysis

- Wireless security standard’s infancy
ECDSA Workload – Computation (1)

percentage in total instruction count (binary field)
ECDSA Workload – Computation (2)

percentage in total instruction count (prime field)

- Muldv2
- Residue
- Multiplication

192 Sig, 224 Sig, 256 Sig, 384 Sig, 521 Sig
Efficient Multiplication Algorithm over VLIW Media Processor

\[ x = x_0 + x_1 g + x_2 g^2 \oplus + x_n g^n \]
\[ y = y_0 + y_1 g + y_2 g^2 \oplus + y_n g^n \]

\[ z = x \cdot y \]
\[ z = z_0 + z_1 g + z_2 g^2 \oplus + z_2n g^{2n} \]

\[
\begin{array}{c|c|c|c|c|c}
\hline
y_n & y_{n-1} & \cdots & y_1 & y_0 \\
\hline
x & x & x & x & x \\
\hline
0 & 0 & \cdots & 0 & x_0 & x_1 & \cdots & x_{n-1} & x_n \\
\hline
\hline
0 & x_0 & x_1 & \cdots & x_{n-1} & x_n \\
\hline
\hline
x_0 & x_1 & \cdots & x_{n-1} & x_n & 0 & \cdots & 0 & 0 \\
\hline
\end{array}
\]

\[ \sum \]

\[ \text{FIR}(t, y) = \sum t_i y_{n-i} \]

\[ z_0 = x_0 \cdot y_0 \]
\[ z_{n-1} = x_0 \cdot y_{n-1} + x_1 \cdot y_{n-2} + \ldots + x_{n-1} \cdot y_0 \]
\[ z_n = x_0 \cdot y_n + x_1 \cdot y_{n-1} + \ldots + x_n \cdot y_0 \]
\[ z_{2n} = x_n \cdot y_n \]
Rectangular Fashion Implementation (Scheme 1)

For $i$ from 0 to $n-1$ do

$carry = 0$

For $j$ from 0 to $n-1$ do

$(u,v) = x_i \cdot y_j$

$(u,v) = (u,v) + z_{i+j}$

$(u,v) = (u,v) + carry$

$z_{i+j} = v$

$carry = u$

$z_{n+i} = carry$

return $z$

Scheme 1 rectangular fashion multiplication
Diagonal Fashion Implementation (Scheme 2)

\( r_2=r_1=r_0=0, \ t_n=\ldots=t_0=0 \)

For \( i \) from 0 to \( 2n-2 \) do
\[
\begin{align*}
\text{tn} &= \text{tn-1} \\
(u_n,v_n) &= t_n \cdot y_n \\
\vdots \\
\text{t}_{1} &= \text{t}_0 \\
(u_1,v_1) &= t_1 \cdot y_1 \\
\text{t}_0 &= x_i, \text{if } i<n, \text{0 others} \\
(u_0,v_0) &= x_0 \cdot y_0 \\
\text{r}_0 &= \text{r}_0+v_0+\ldots+v_n, \\
\text{r}_1 &= \text{add}_{\text{with_carry}}(r_1, u_0, \ldots, u_n) \\
\text{r}_2 &= \text{add}_{\text{with_carry}}(r_2,0) \\
\text{z}_i &= r_0, \text{r}_0=r_1, \text{r}_1=r_2, \text{r}_2=0 \\
z_{2n-1} &= r_0 \\
\text{return } z
\end{align*}
\]
FIR Fashion Multiplication (Scheme 3)

Illustration for the FIR instruction

Scheme 3 FIR based multiplication

t0=t1=…=tn=carry=0;
For i from 0 to 2n-1 do
    Set x = xi, if i<n
    0, if i>=n
    Left-shift 8 bits from x to tn, tn-1, …, t0
    Do FIR for all the pairs of tm and ym
    Set r0 as the sum of all the FIR results
    Compute r1 by the same progress as above (left-shift, FIR, sum)
    Compute r2 by the same progress
    Compute r3 by the same progress
    zi = r0+( r1<<8 ) + ( r2<<16 ) + ( r3<<24 ) + carry
    carry = add_with_carry( ( r2>>16 ), ( r3>>8 ) )
return z
Result of Optimization – Issue Rate and Results

- Metric for program efficiency measure
- VLIW structure can issue several operations simultaneously (5 for Trimedia TM1300)
- Enhance issue rate could speedup the performance

<table>
<thead>
<tr>
<th></th>
<th>Scheme 1</th>
<th>Scheme 2</th>
<th>Scheme 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cycle count</td>
<td>6465452</td>
<td>4161626</td>
<td>5344969</td>
</tr>
<tr>
<td>Issue rate</td>
<td>2.25</td>
<td>3.70</td>
<td>3.96</td>
</tr>
</tbody>
</table>
Result of Optimization

Instruction Count for Multiplication Module

- Scheme 1
- Scheme 2
Conclusion

- Important SIMD-related solutions
  - SIMD-controlled padding-based memory bank-conflict reduction
  - SIMD-oriented loop unrolling scheme
  - SIMD-based FIR solution for cryptographic algorithm implementation

- Compiler
  - Memory access mode
  - Register allocation
  - Loop unrolling

- Power management optimization