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

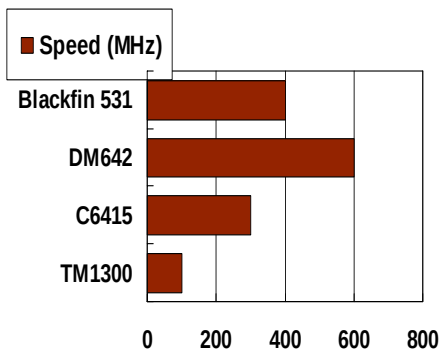
Embedded processors

Introduction (2)

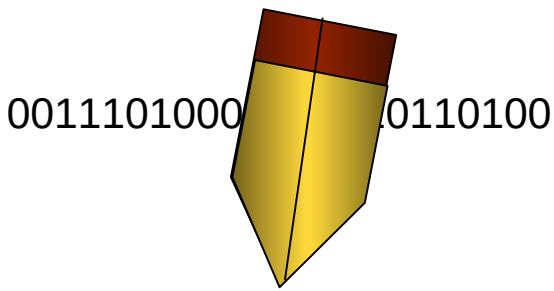
- Technical Challenges

power management

processor utilization

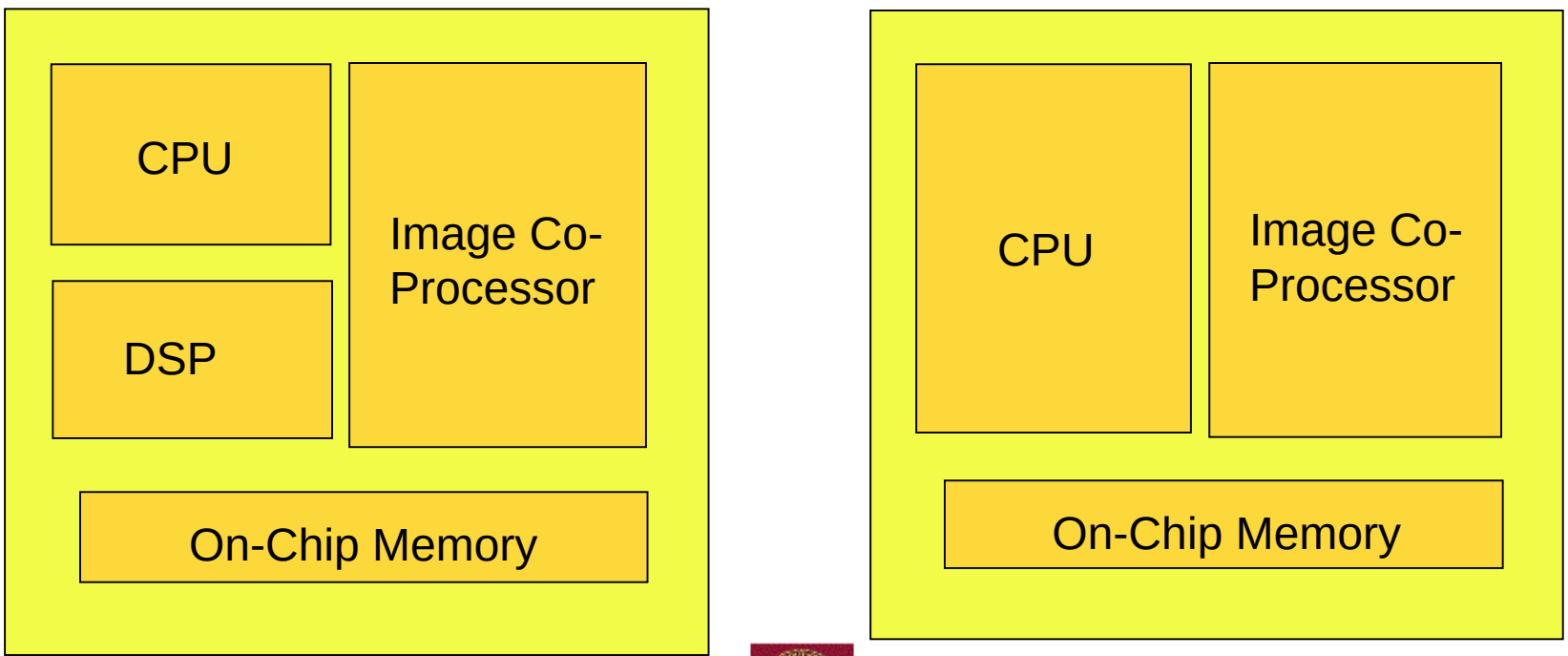


security issue



Introduction (3)

- Two-Chip Solution
  - Communication chip and Multimedia chip
- Embedded Media Processor Architecture



Introduction (4)

- 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.

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

Comparison of Embedded Processors and General Purpose (GP) Processors

	Embedded Media Processors	GP Processors
Architecture	VLIW	Superscalar
Clock Speed	100 – 600 MHz	2 – 4 GHz
Cache Size	16/16 KB (L1) Usu. no L2 cache	32 – 128KB(L1) 512 – 1MB (L2) L3 cache available
Memory size	8 – 256 MB	512 MB – 1GB

Key Issues About Processor Design

CPU	Memory & I/O	OS overhead
Instruction Set Architecture + Compiler	Size	Single-user or Multi-user
Micro-architecture implementation	Speed	Time-sharing
VLSI speed		

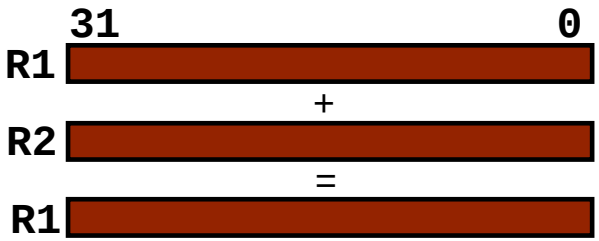
Embedded Multimedia Application (EMA)

- 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)

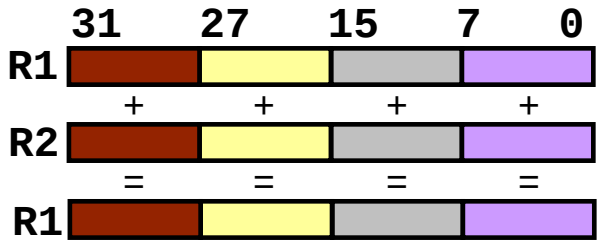
Single Instruction Multiple Data (SIMD)

- 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

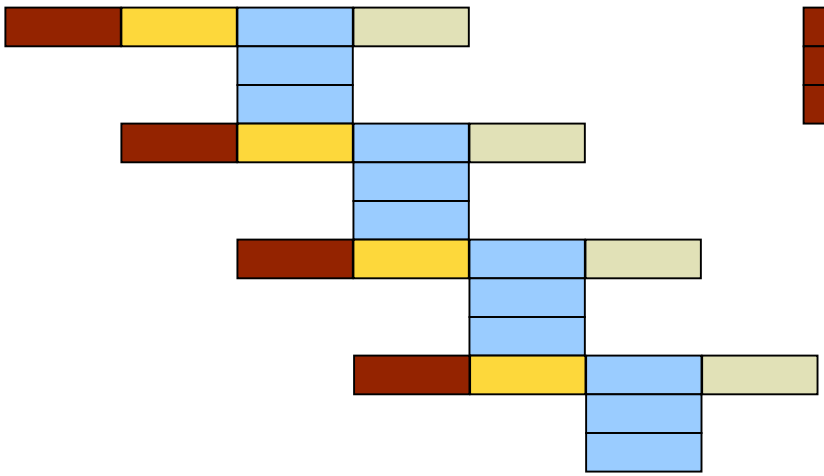
Load A
Load B
Add A, B
Load C
Load D
Add C, D

Pack L1, A, C
Pack L2, B, D
SIMDADD L1, L2

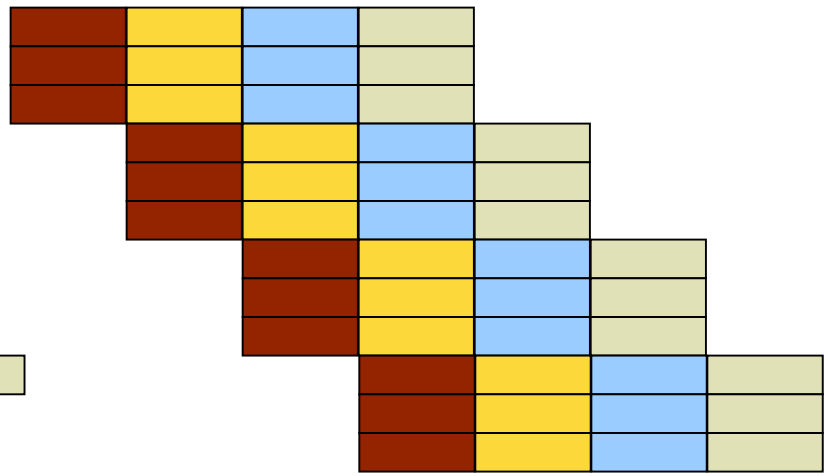
Cycle count is reduced by 50%

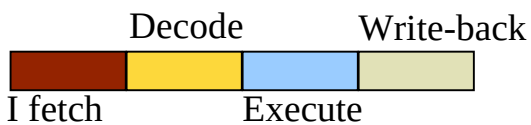
Comparison of VLIW & Superscalar

VLIW



Superscalar



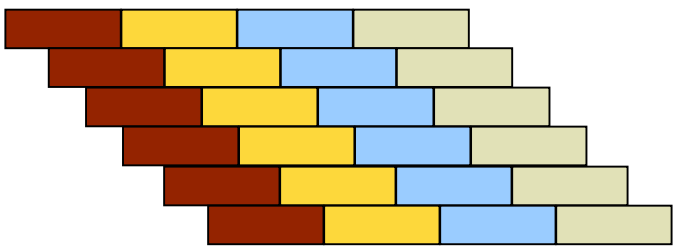


3-parallel executions per cycle

3-issue superscalar per cycle

Hyper-pipeline and Superscalar

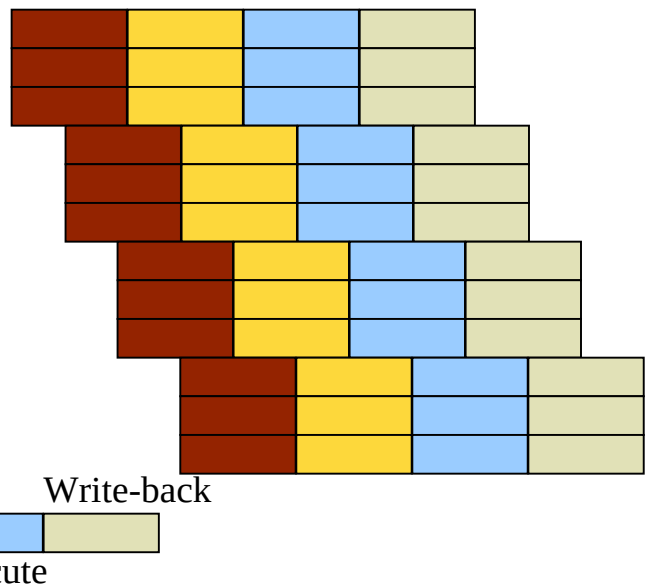
Hyper-pipeline
or super-pipeline



Super-pipeline with depth of 3

3 functional pipelines in parallel

Superscalar super-
pipeline



3-issue superscalar super-pipeline

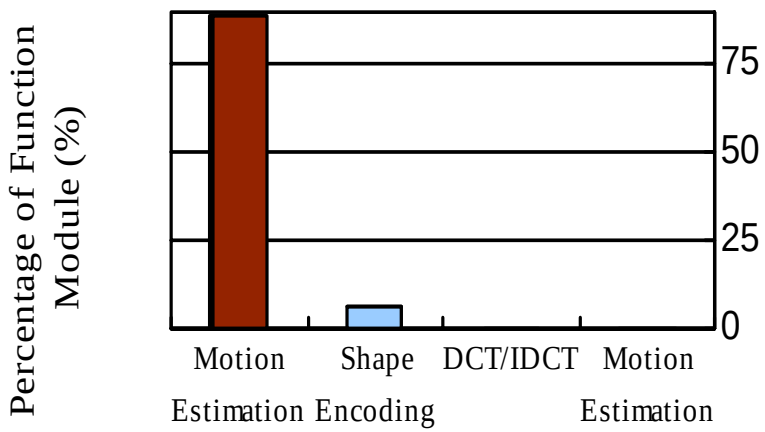
VLIW

- 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

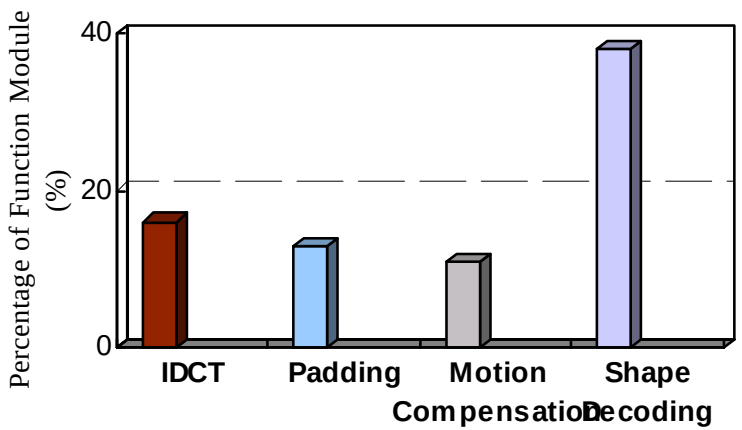
Content

- 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

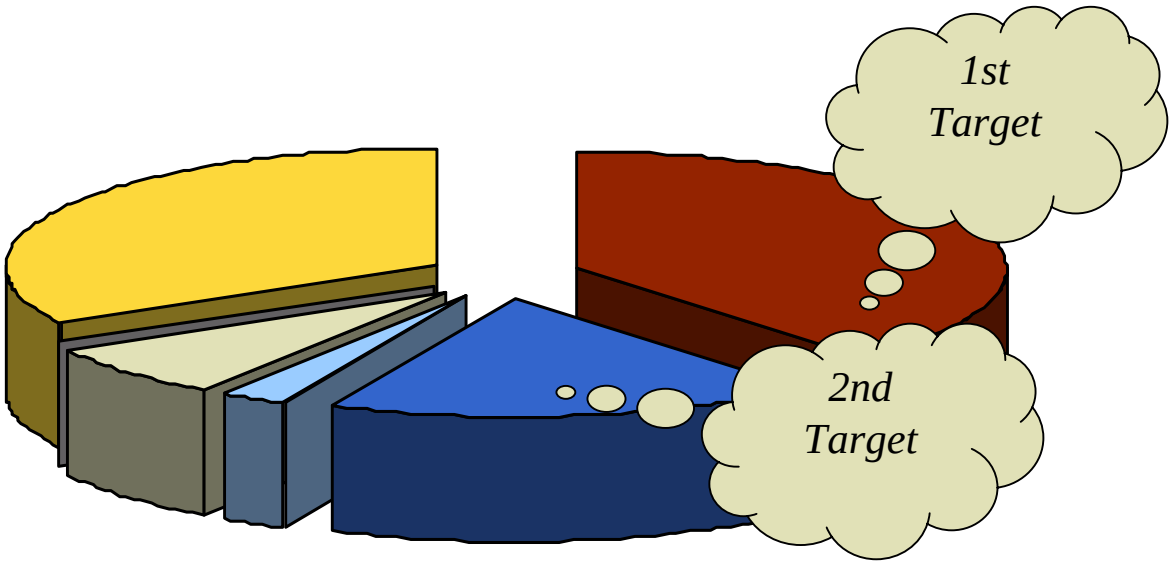


Profiling of MPEG-4 Encoder



Profiling of MPEG-4 Decoder

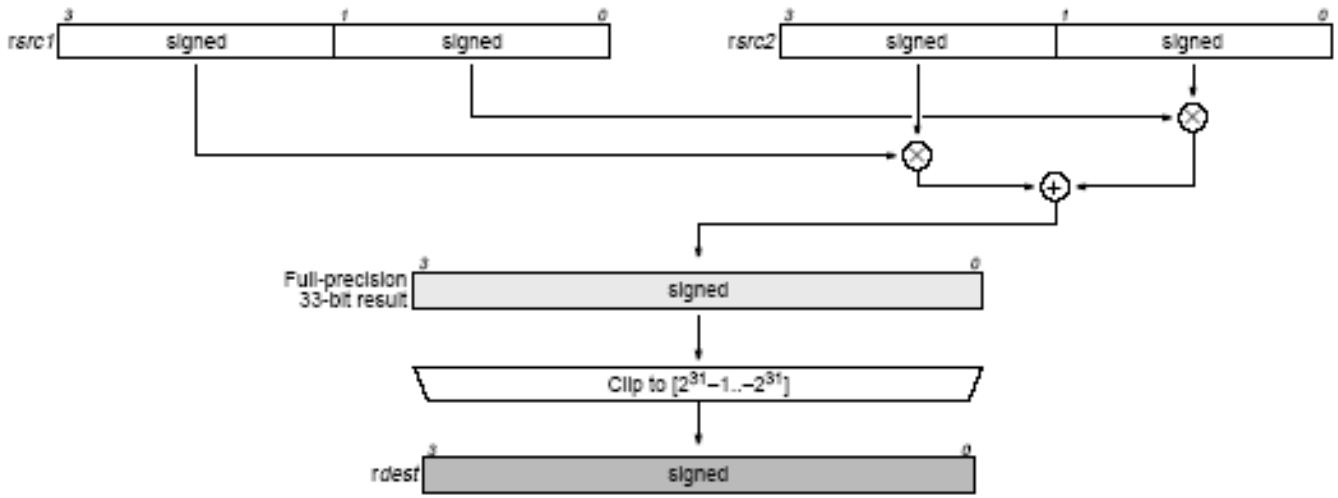
Profiling in Video Coding Operations



Computational Profiling for MPEG-1 Decoder

Speeding up 8x8 DCT

- IFIR16

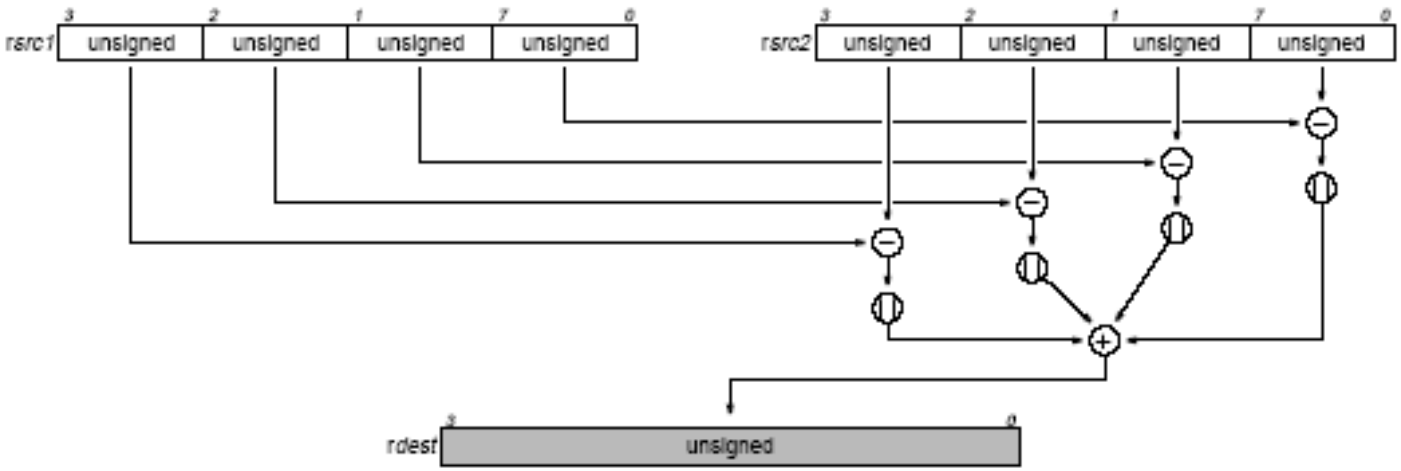


Performance Comparison of DCT

	Instruction Cycle	Issue Rate
Fast DCT on pure C	472	3.24
Optimized DCT based on Trimedia ISA	160	4.42

Speeding up Motion Estimation

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

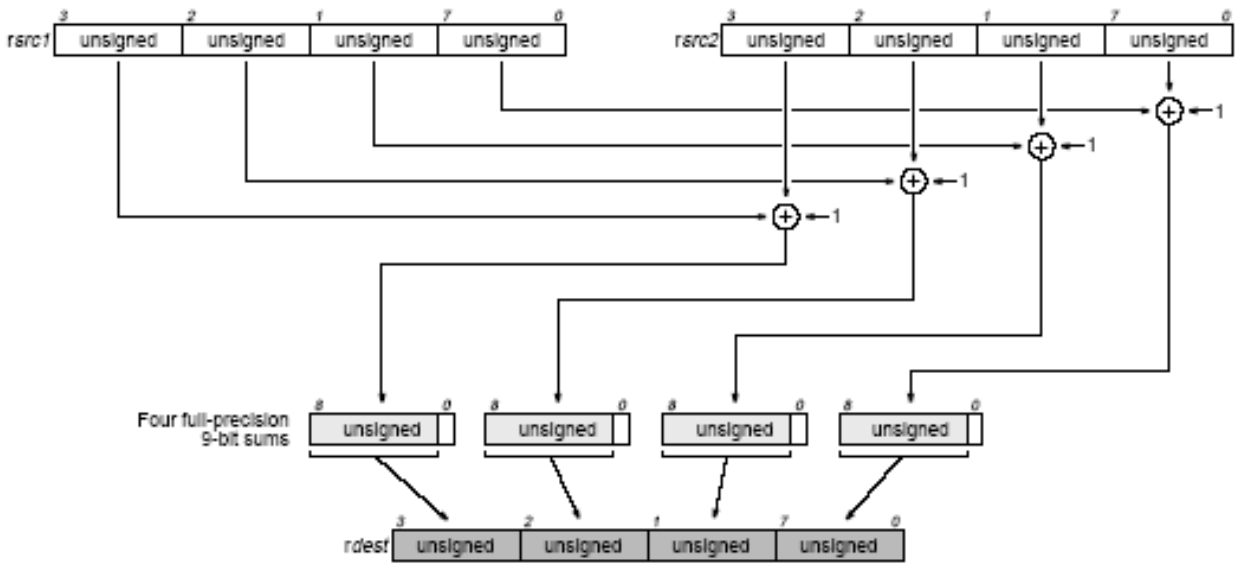


Performance Comparison of SAD Computation

	Instruction Cycle	Issue Rate
8x8 MAD Computation on pure C	100	3.98
8x8 MAD Computation with UMEU88	21	3.43

Speeding up Interpolation

- **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



	Instruction Cycle	Issue Rate
interpolation on pure C	113,012	1.51
Interpolation with media instructions	2,795	4.70

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

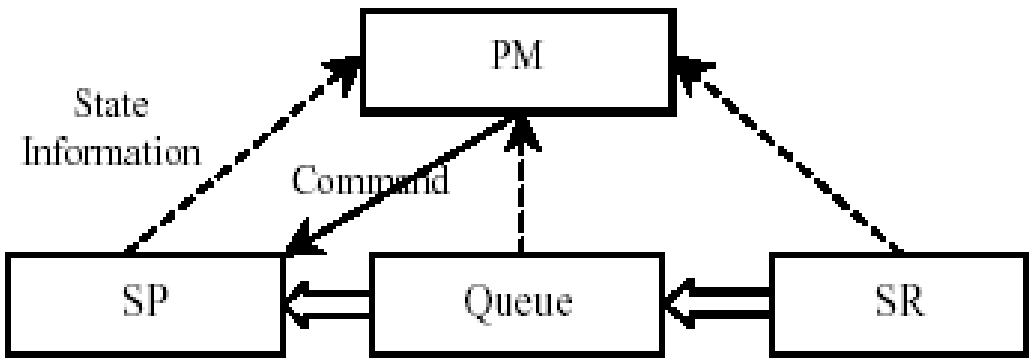
- Activity Management
  - Switch off inactive components when possible
  - Reduced bus switching
  - Reduce cycle counts
- Memory Management
  - Reduce memory size
    - Larger size or longer bit-width → long transfer path / complex address decoding…
    - More connectivity → More wiring → more power
  - 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 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)

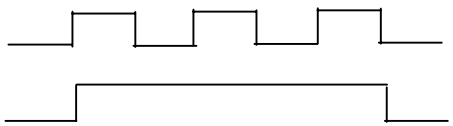
- Discrete-time Markov decision process
  - Four-component system model
  - Objective: expected performance (e.g. waiting time and no. of objects in the queue)
  - Constraint: expected power consumption



- Continuous-time Marchov 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 →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.

Reduced Bus Transition/Switching (2)

- Data/Instruction organization

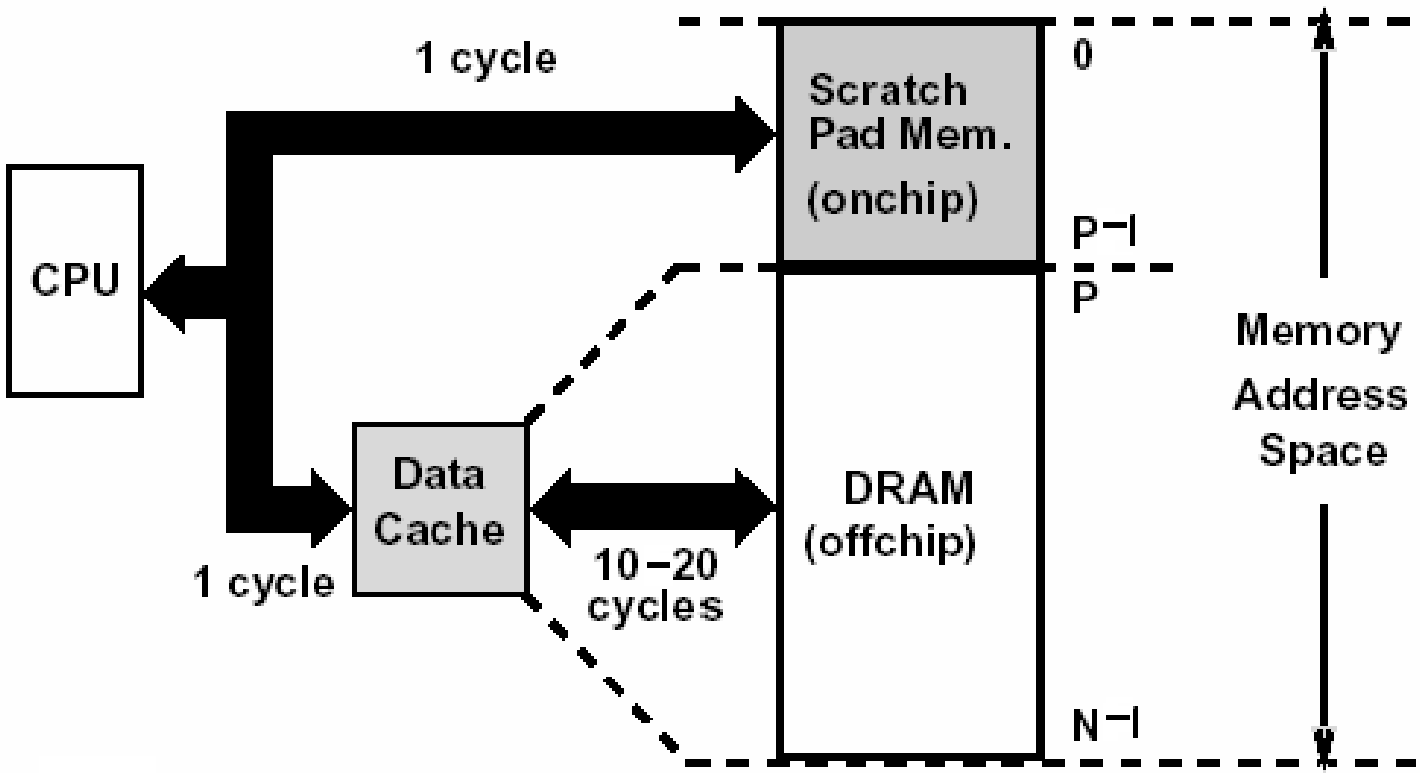
  Reorganize data and/or instructions so that

  – 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)

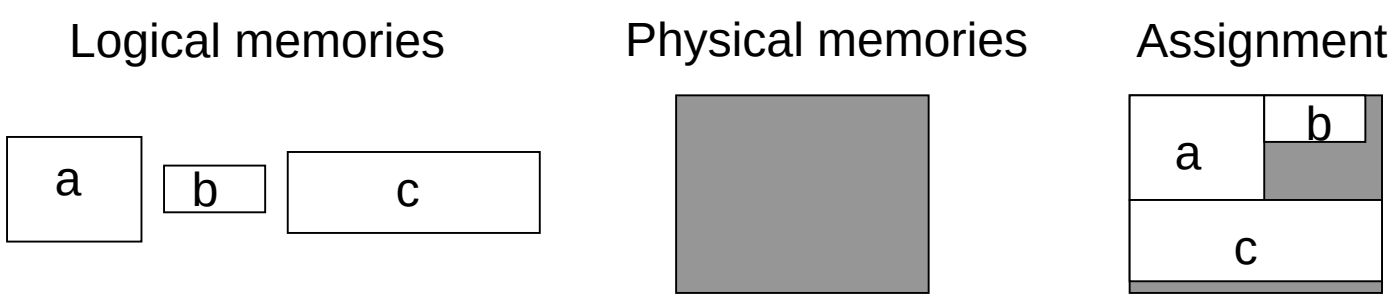
- Cache and scratch-pad memory management
  - Cache Management
    - Careful data layout to reduce cache conflict
    - Prefetching to increase cache hit ratio
    - Cache module assignment
      - Classify variables based on their spatial locality
      - Assign variables of different locality into 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

Memory and Data Optimization (3)

- 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 and Data Optimization (4)

- 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



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

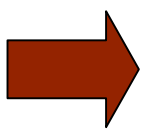
Memory and Data Optimization (5)

- 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

Power-aware Compiler Technology (1)

- 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

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



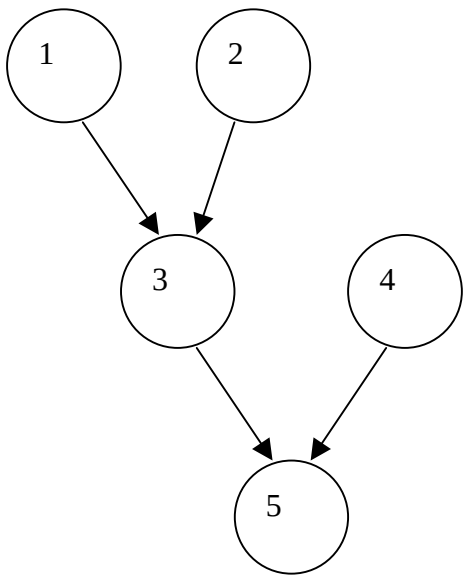
```
for( i = 0; i < M; i++ )
    for( j = 0; j < N; j++ )
    {
        b[i][j] = 0;
        for( k = 0; k < P; k++ )
            b[i][j] = a[i][j+K];
    }
```

Power-aware Compiler Technology (2)

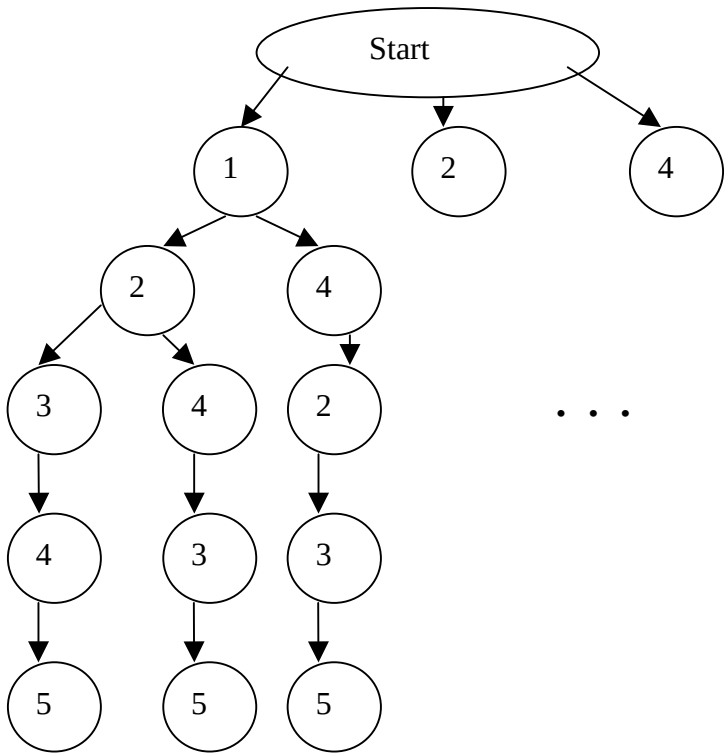
- Instruction-level scheduling
  - Power dissipation table (PDT) based scheduling
    - CDG (control&data dependency graph) construction
    - PDT construction
      - List the power consumption for each consecutive instruction pair
    - SCG (weighted strongly connected graph) construction
      - Based 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



Power-aware Compiler Technology (4)

- 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

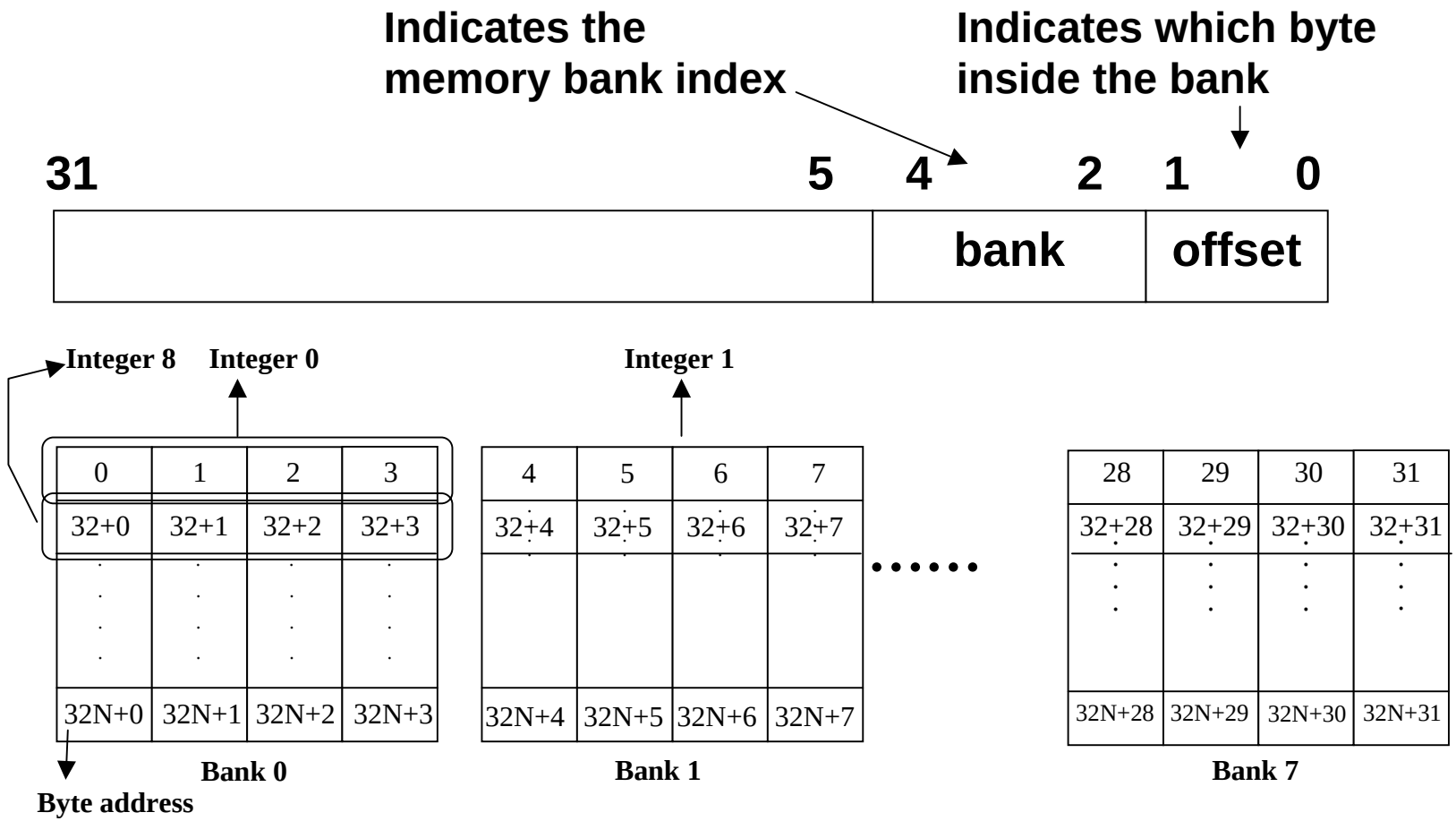
Power-aware Compiler Technology (5)

- 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

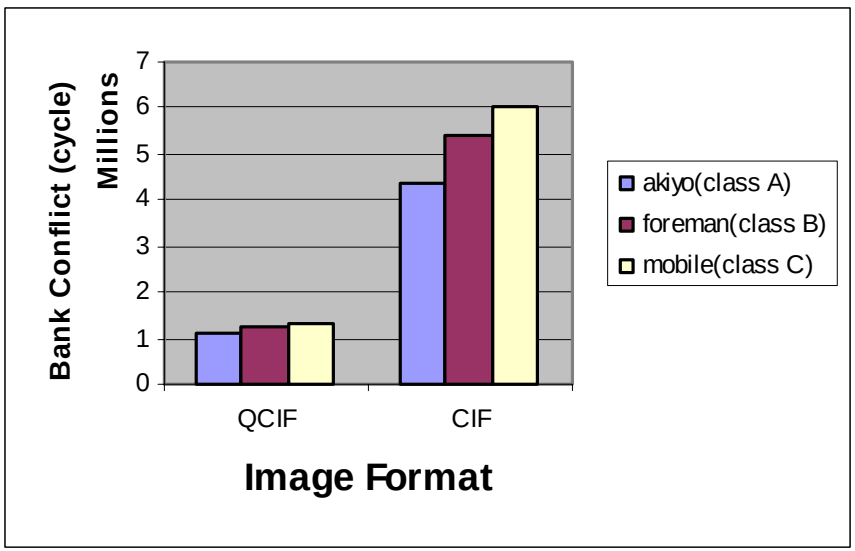
Interleaved Memory and Bank Confliction



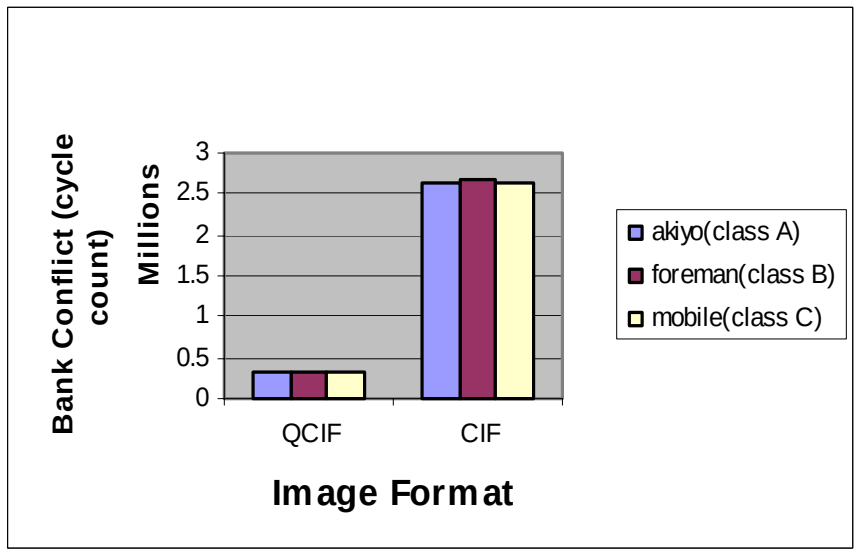
Trimedia TM1300 Interleaved Memory Address Layout

Memory bank Conflict under Different Resolution and Stream Complexity

- As resolution increases from QCIF to CIF format, memory bank conflict increased significantly



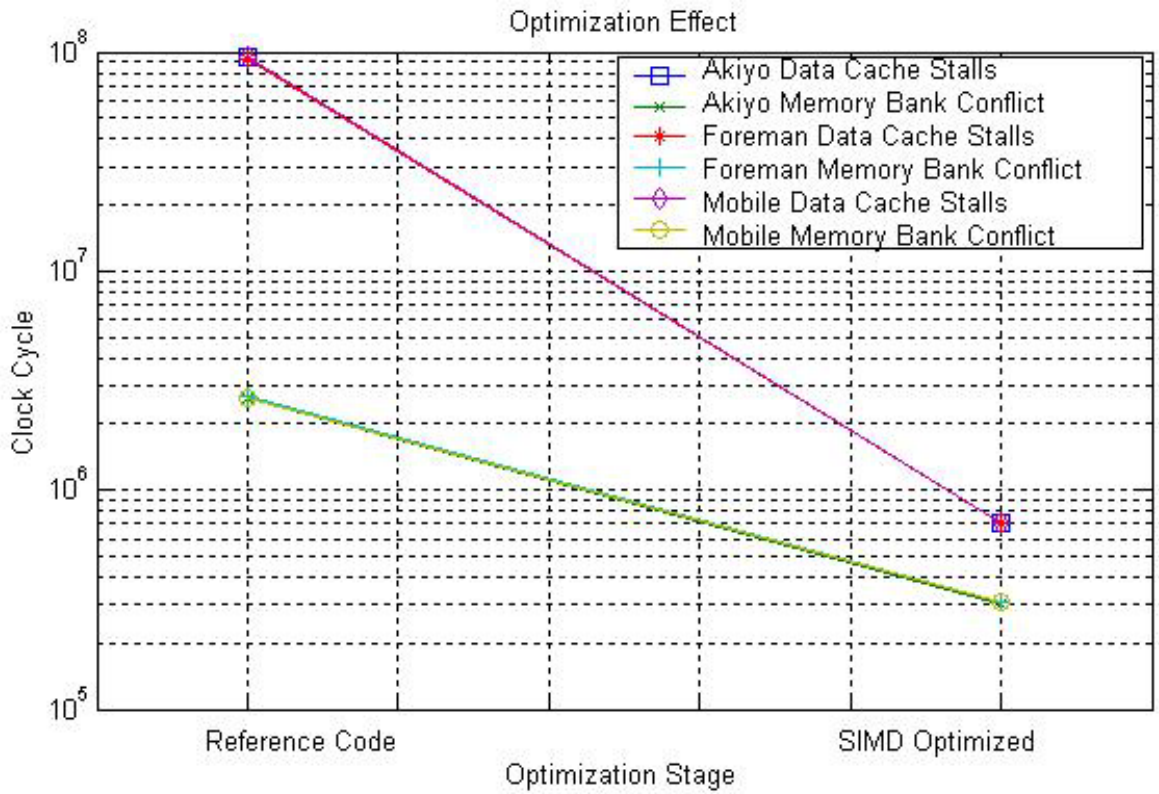
getRefCost



Interpolation

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

- Loop unrolling: overlaps different iterations of the loop body, creates N copies of the loop.

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

p += stride;
g += stride;
m += stride;
d += stride;
```

Original Loop

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

  j1 = j+1;                (3)
  p[j1] = g[j1] - m[j1];   (4)
  d[j1] = p[j1] * s;       (5)

  j2 = j+2;                (6)
  p[j2] = g[j2] - m[j2];   (7)
  d[j2] = p[j2] * s;       (8)

  j3 = j+3;                (9)
  p[j3] = g[j3] - m[j3];   (10)
  d[j3] = p[j3] * s;       (11)
}
```

Unrolled Loop

Memory Access Pattern Analysis

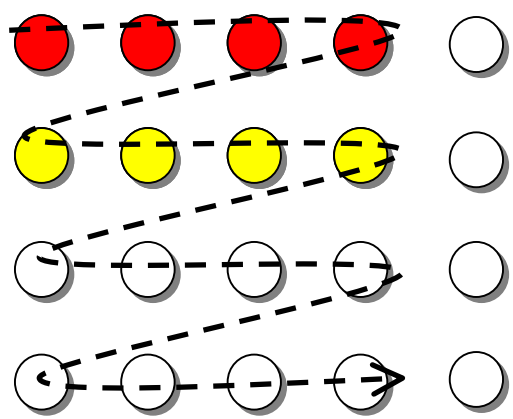
- Memory access is a major bottleneck on the performance of EMAs
- It decides the loop unrolling strategy.

- 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.

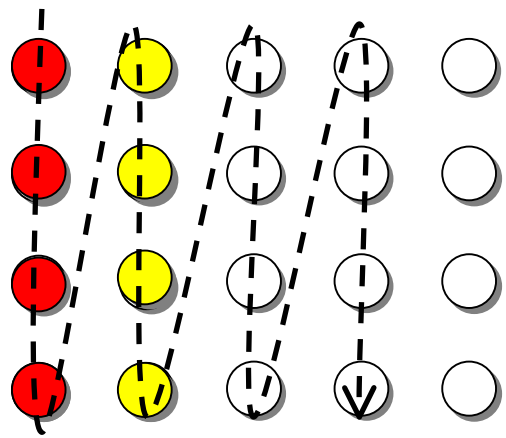
```
for (j=0; j<4; j++)
{
    for (i=0; i<4; i++)
    {
        m7[i] = imgy_orig[i] - mpr[i];
    }
    imgy_orig += stride_imgy_orig;
    mpr += stride_mpr;
    m7 += 4;
}
```



Vertical Access

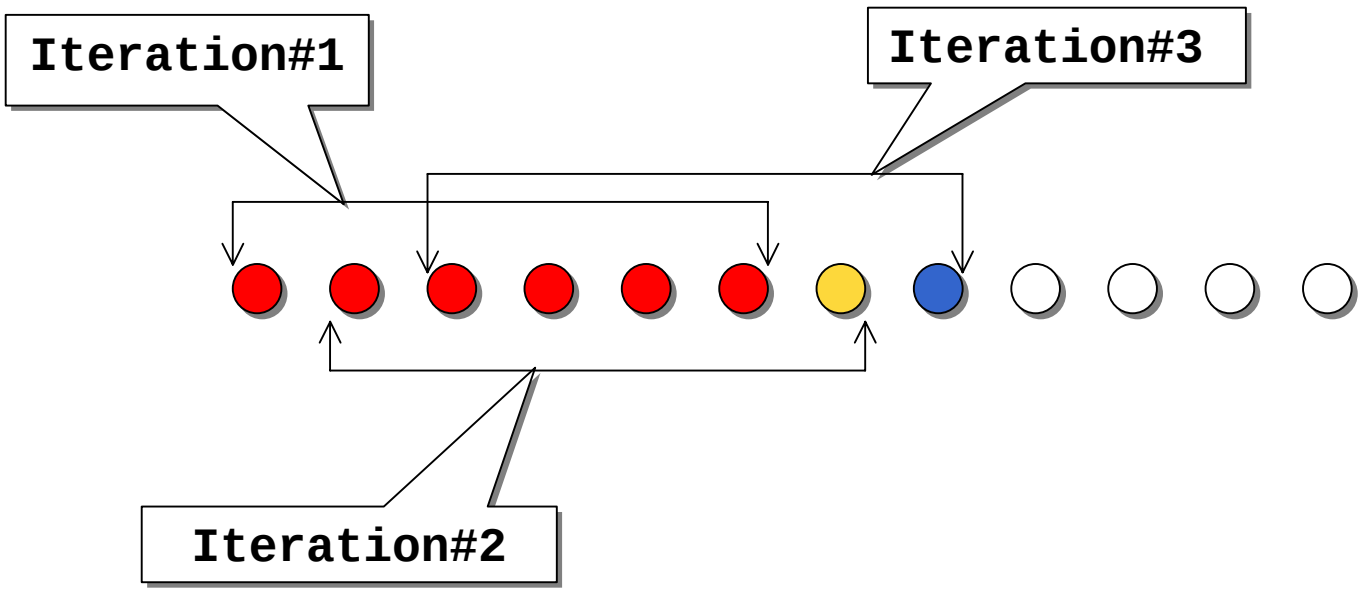
- A transpose access pattern relative to horizontal access
- In loop filter, intra-prediction, interpolation, chroma prediction function and etc.

```
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



SIMD-aware Loop Unrolling

- 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

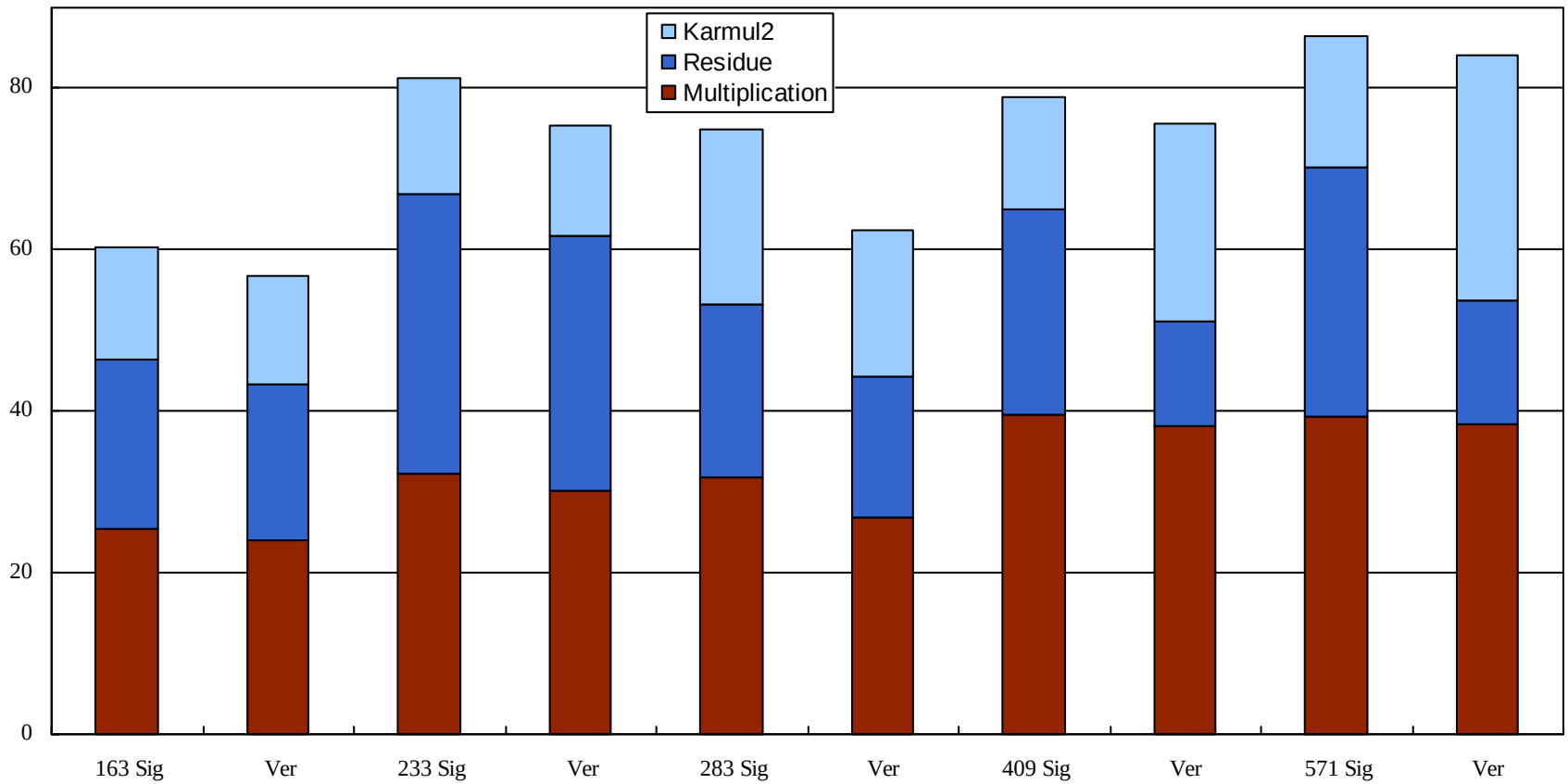
Content

- 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

Why Security Concern?

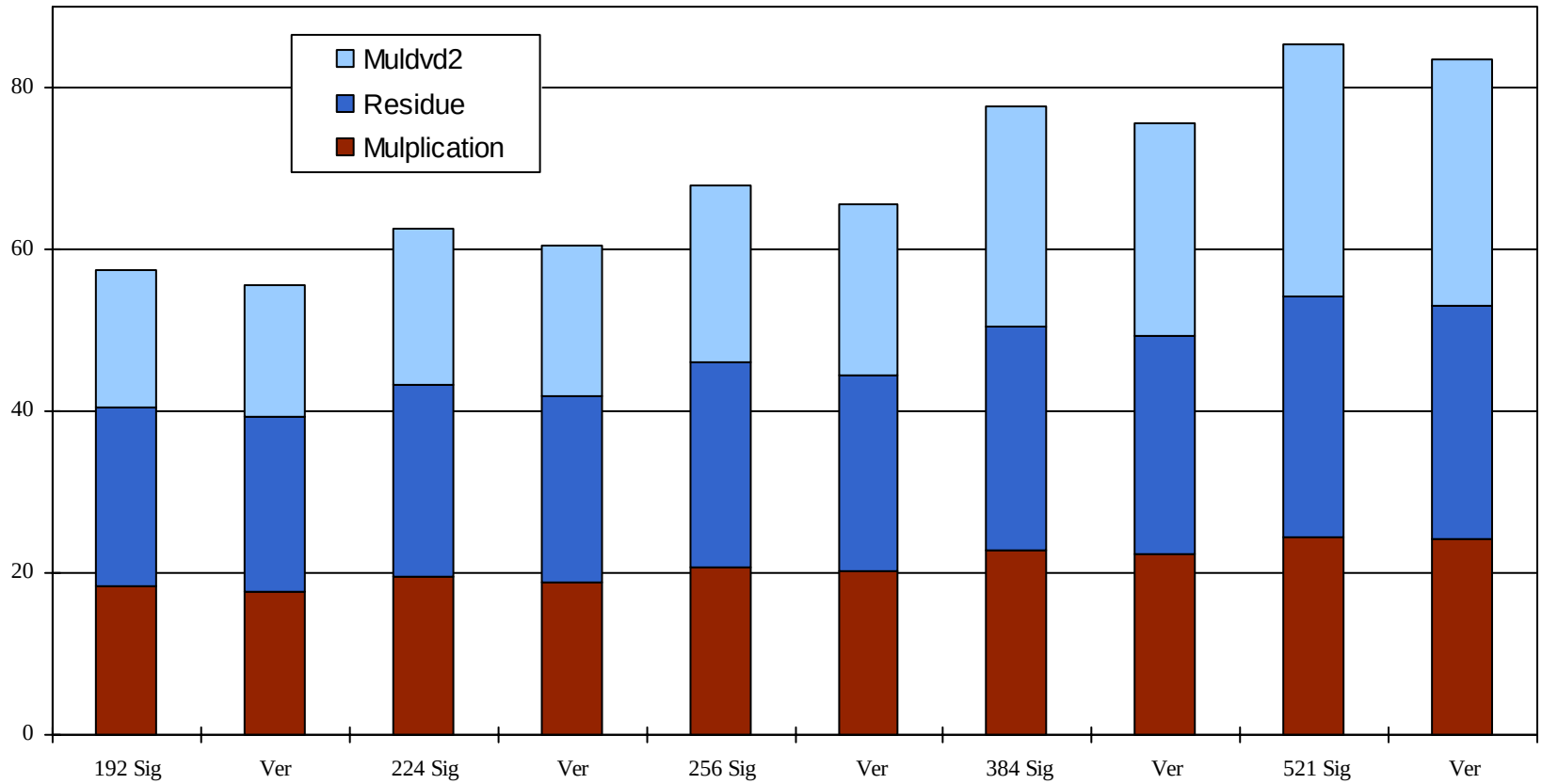
- Traditional security concern like integrity, privacy and authentication
  - Public transmission medium
- Software solution alone is not sufficient
  - Embedded system increasingly assembled from pre-designed components
- Side-channel attack techniques becomes menace
  - Fault analysis
  - Power analysis
  - 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)

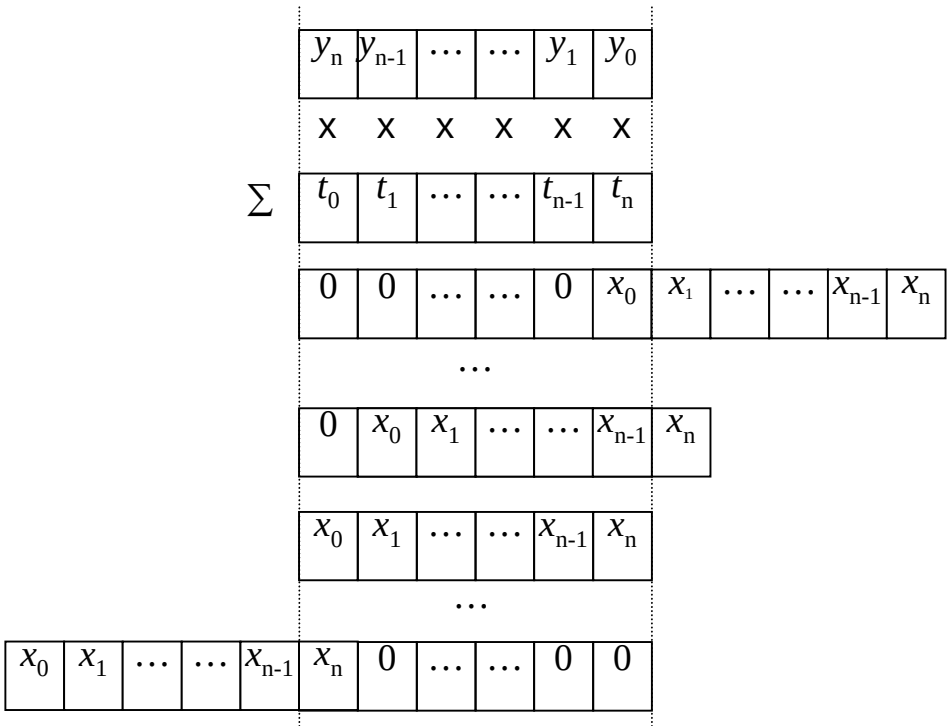
Efficient Multiplication Algorithm over VLIW Media Processor

$$x = x_0 + x_1 g + x_2 g^2 ☹ + x_n g^n$$

$$y = y_0 + y_1 g + y_2 g^2 ☹ + y_n g^n$$

$$z = x \bullet y$$

$$z = z_0 + z_1 g + z_2 g^2 ☹ + z_{2n} g^{2n}$$



$$\mathrm{FIR}(t, y) = \sum t_i y_{n-i}$$

$$z_0 = x_0 \cdot y_0$$

$$z_{n-1} = x_0 \cdot y_{n-1} + x_1 \cdot y_{n-2} + \ldots + x_{n-1} \cdot y_0$$

$$z_n = x_0 \cdot y_n + x_1 \cdot y_{n-1} + \ldots + x_n \cdot y_0$$

$$z_{2n} = x_n \cdot y_n$$

Rectangular Fashion Implementation (Scheme 1)

For i from 0 to n-1 do

$carry = 0$

For j from 0 to n-1 do

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

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

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

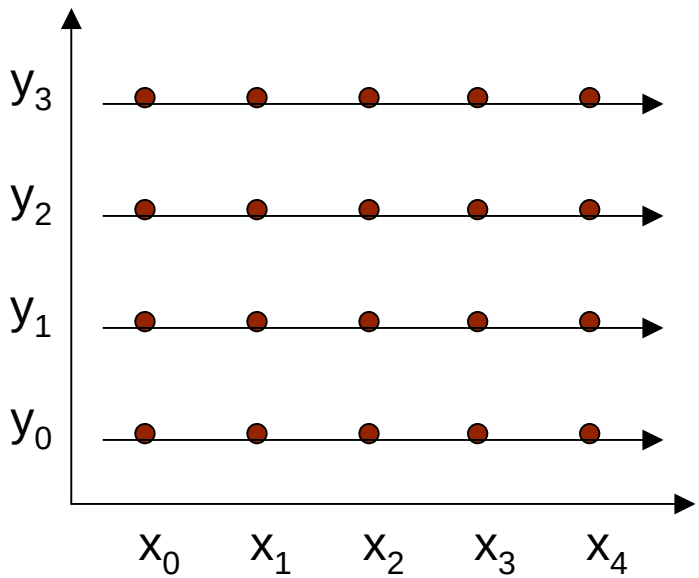
$z_{i+j} = v$

$carry = u$

$z_{n+i} = carry$

return z

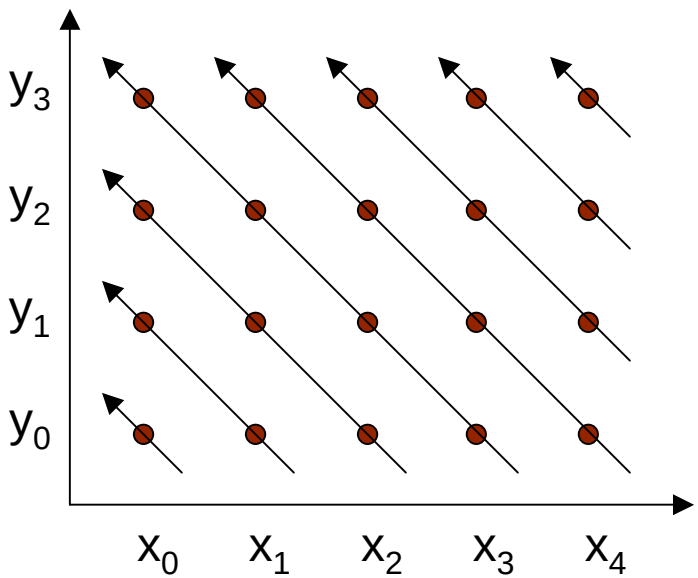
Scheme 1 rectangular fashion multiplication



Diagonal Fashion Implementation (Scheme 2)

r2=r1=r0=0, tn=…=t0=0
For i from 0 to 2n-2 do
    tn= tn-1
    (un,vn) = tn• y n
    …
    t1= t0
    (u1,v1) = t1• y 1
    t0= xi, if i<n, 0 others
    (u0,v0) = x0• y 0
    r0= r0+v0+…+vn,
    r1= add_with_carry(r1, u0,…, un)
    r2= add_with_carry(r2,0)
    zi = r0, r0= r1, r1= r2, r2= 0
z2n-1= r0
return z

Scheme 2 unrolled diagonal fashion multiplication



FIR Fashion Multiplication (Scheme 3)

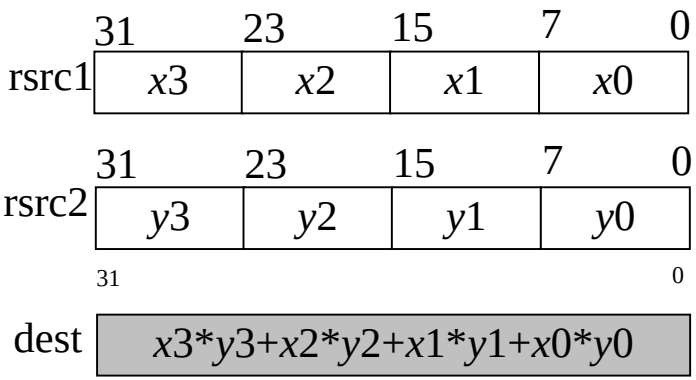


Illustration for the FIR instruction

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

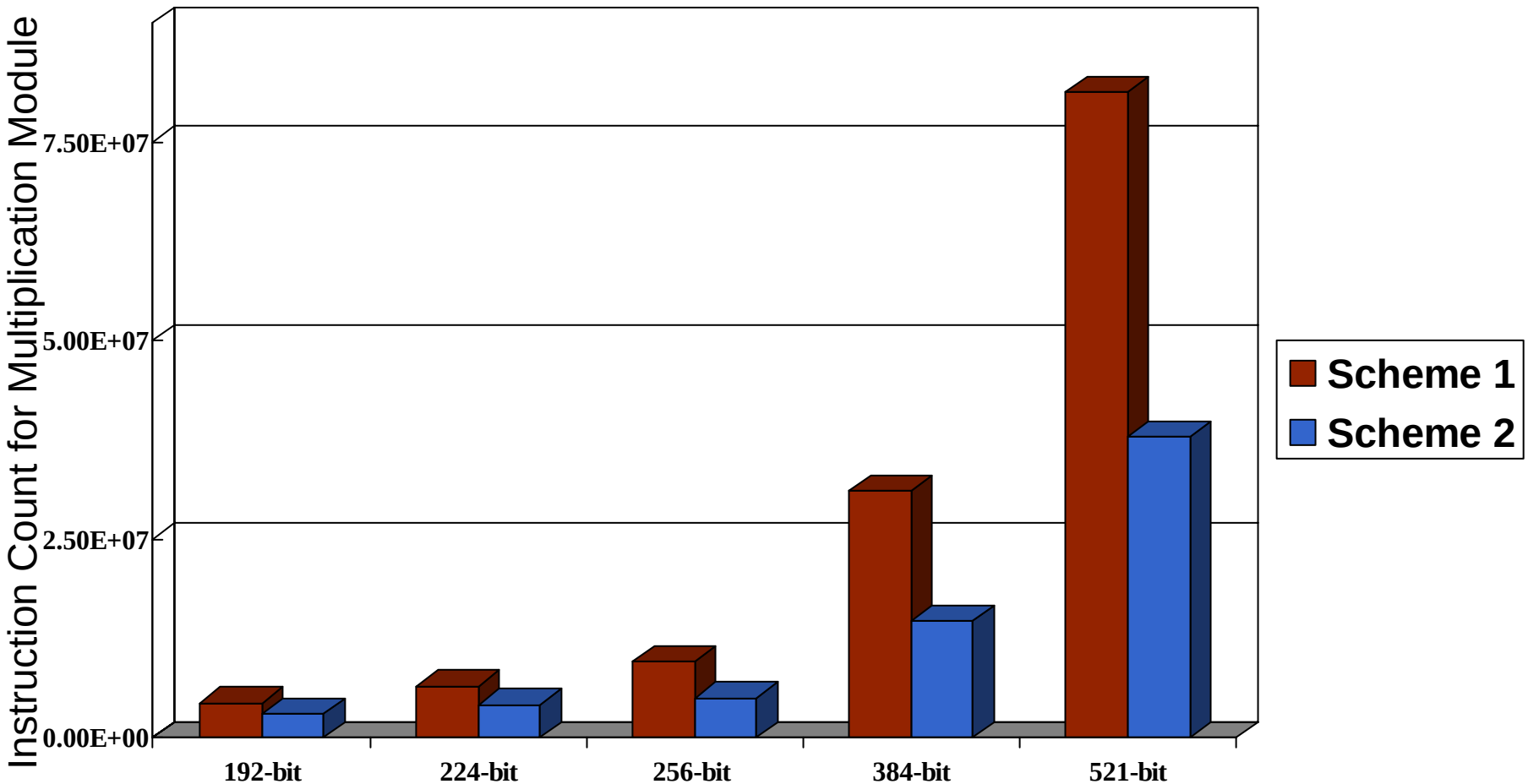
Scheme 3 FIR based multiplication

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

	Scheme 1	Scheme 2	Scheme 3
Cycle count	6465452	4161626	5344969
Issue rate	2.25	3.70	3.96

Result of Optimization



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