A Preliminary Study On the Vectorization of Multimedia Applications for Multimedia Extensions

Gang Ren
University of Illinois

Peng Wu
IBM T.J. Watson Research

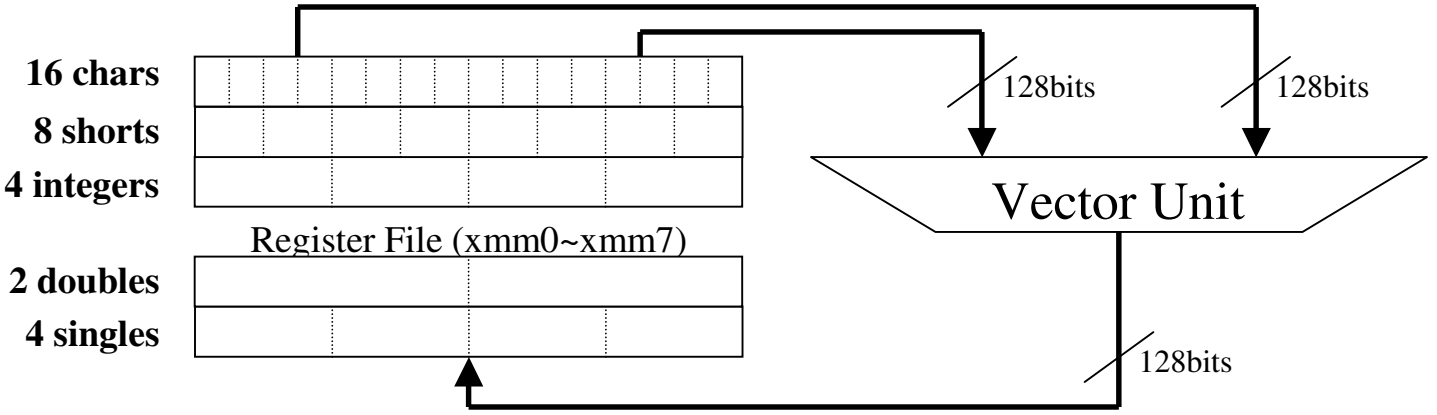
David Padua
University of Illinois

Presented by Gang Ren

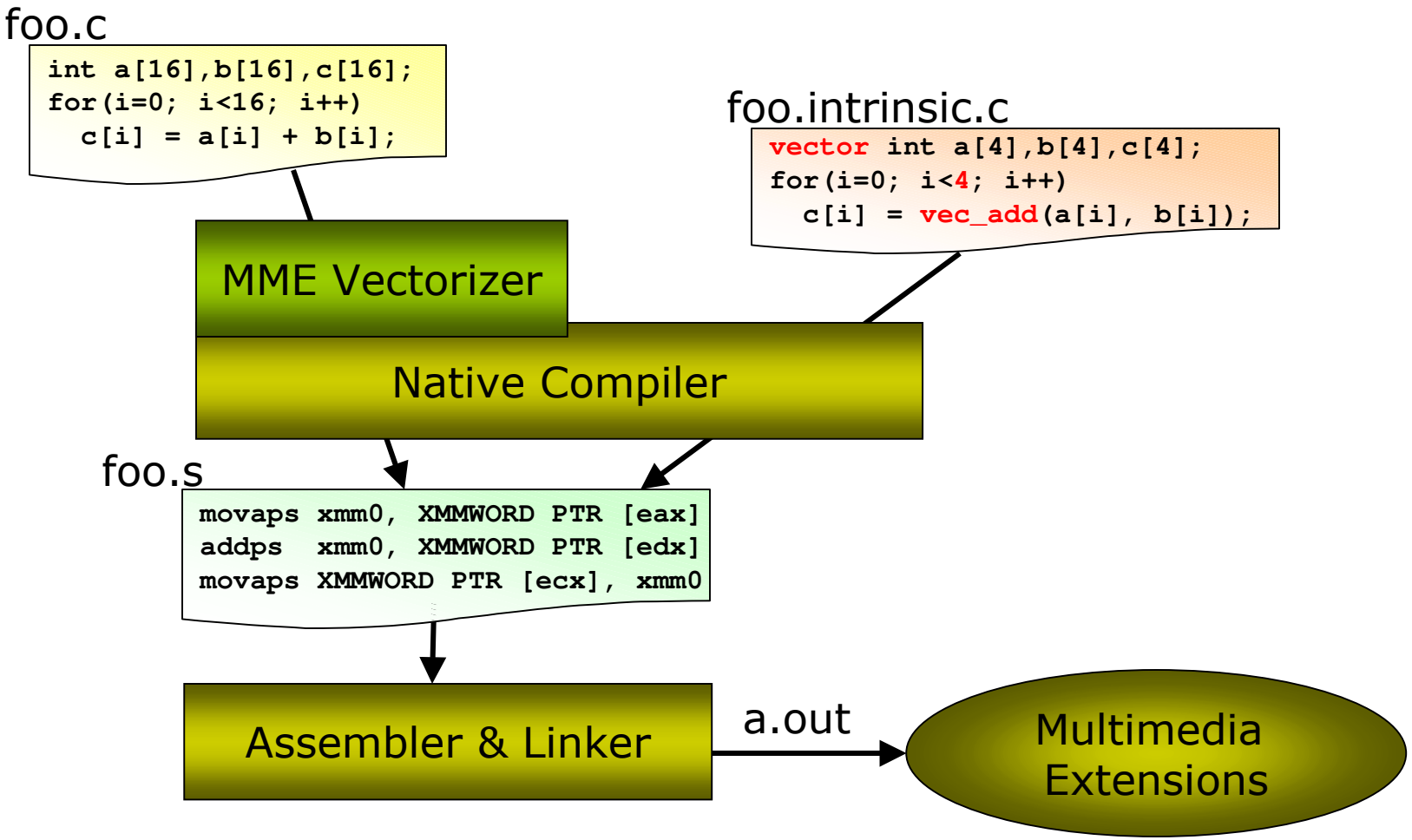
Multimedia Extensions (MME)

- Additions to accelerate multimedia applications
  - For general-purpose processors:
    - MAX(HP), VIS(Sun), AltiVec(Motorola/IBM/Apple), SSE(Intel)
  - For special-purpose processors:
    - PS2(SONY), Graphics Processing Unit(NVIDIA)
- Use SIMD architecture

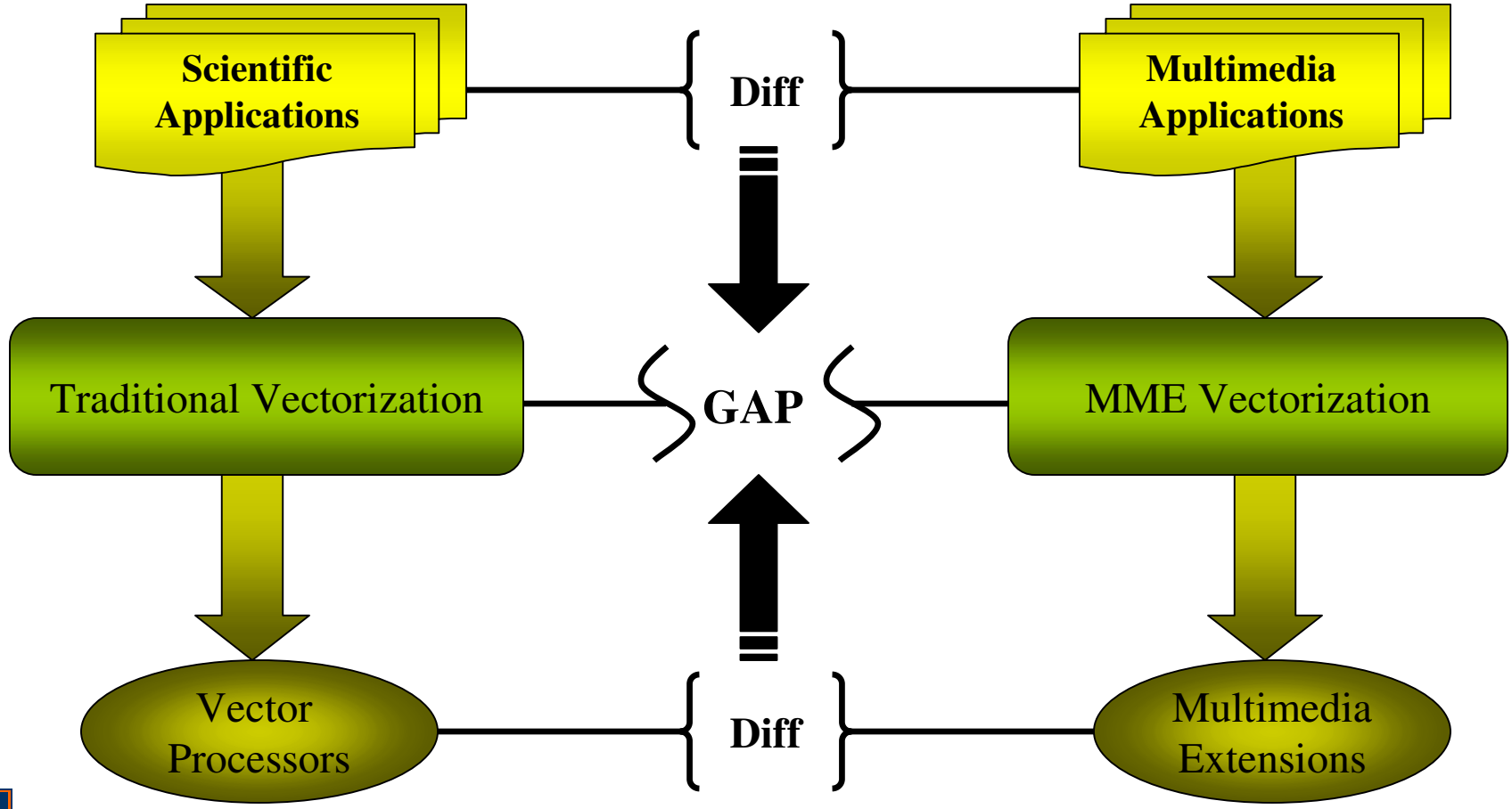
Intel SSE2



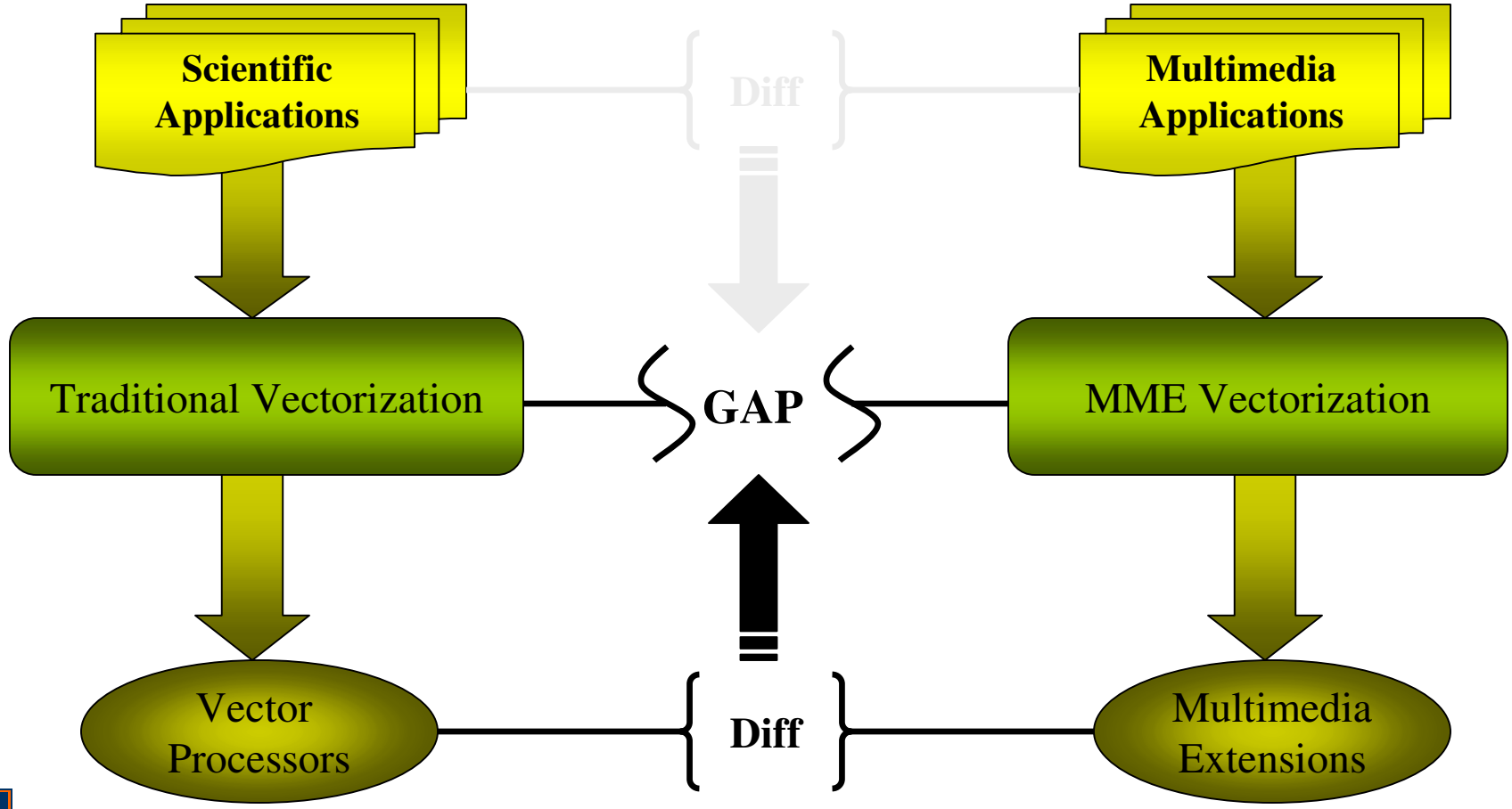
Programming Multimedia Extensions



Motivation


Scientific Applications
Diff
Multimedia Applications
Traditional Vectorization
GAP
MME Vectorization
Vector Processors
Diff
Multimedia Extensions

Gaps From Architecture



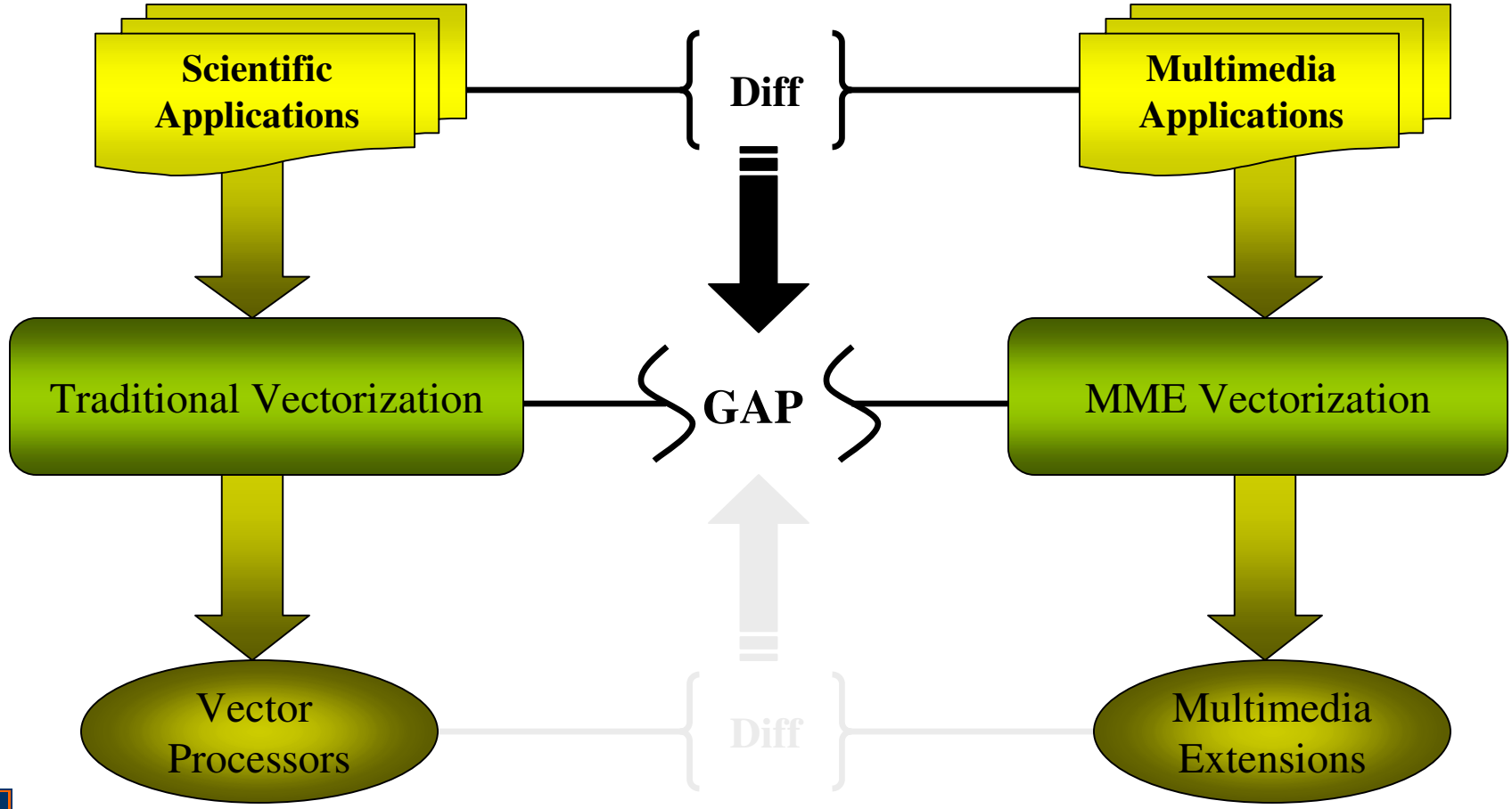
MME vs. Vector Processor

- Differences in memory unit
  - MME: No scatter/gather memory operations

```
for(i=0; i<8; i++)
  for(j=0; j<8; j++)
     s += a[i*8+j] * b[j*8+i];
```

  - MME: Only support aligned memory access
- Differences in ISA
  - MME: Special instructions for media processing
    - Example: Saturated Operations
  - MME: Non-uniform support for different element types
    - SSE2: Max/min operations on 16-bit short integers

Gaps From Applications



Berkeley Multimedia Workload

- Evolves from MediaBench
- 12 applications written in C/C++
  - Audio compression: ADPCM, GSM, LAME, mpg123
  - Image/video compression: DVJU, JPEG, MPEG2
  - Graphics: POVray, Mesa, Doom
  - Others: Rsynth, Timidity
- Where are the example codes from?
  - Important loops in core procedures (>10% total ex. time)

Where Are Gaps From?

- Different programming styles
  - Pointer access
  - Manually unrolled loops
- Mismatches between application and language
  - Integer promotion
  - Saturated operation
- Different code patterns
  - Bit-wise operations
  - Lookup tables

C Language Issues: Integer Promotion

- Integer promotion
  - Forced by ANSI C semantics (ISO/IEC 9899:1999)
    - All *char* or *short* types are automatically promoted to *integer* type before conducting any arithmetic operations.
  - Fit traditional scalar architecture well
- MME supports sub-word level parallelism
  - Integer promotion will waste computation bandwidth
- How to eliminate unnecessary integer promotion?
  - Some analyses needed

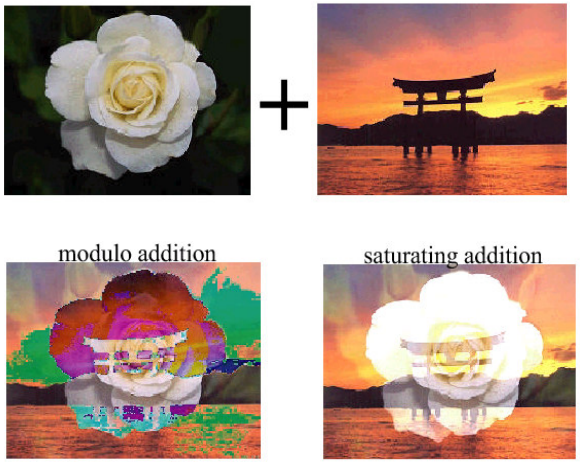
to ensure the same result

```
for(i=0; i<1024; i++)
 for(j=0; j<1024; j++)
  dst[i,j]=src1[i,j]+src2[i,j];
```

C Language Issues: Saturated Operations

```
for(i=0; i<1024; i++)
    for(j=0; j<1024; j++) {
     dst[i,j]=src1[i,j]+src2[i,j];
     if(dst[i,j] > 255)
       dst[i,j] = 255;
     if(dst[i,j] < 0 )
       dst[i,j] = 0;

    }
```




```
/* From BMW/GSM */
ltmp=a+b;
if(( (unsigned)ltmp - MIN_WORD ) >
          ( MAX_WORD - MIN_WORD ))
  if(ltmp>0)
    ltmp = MAX_WORD;
  else
    ltmp = MIN_WORD;
```

Code Pattern: Lookup Tables

- To implement saturated operations
- To replace expensive math function calls

```
/* From BMW/Lame */

if (init==0)
 for (i=0;i<LUTABSIZE;i++)
  lutab[i]=pow(...);
...
for (i=0;i<l_end;i++) {
  temp=...;
  if (temp<1000.0) {
    ix[i]=lutab[(temp*10)];
  }
}
```

Some Related Work

- Compilation based on traditional vectorization
  - Cheong and Lam's optimizer for VIS (Sun)
  - Krall and Lelait's traditional vectorizer for VIS
  - Sreraman and Govindarajan's vectorizer for MMX(Intel)
  - Aart's intra-register vectorization for the Intel architecture
- Other compilation techniques
  - Krall and Lelait's "*Vectorization by loop unrolling*"
  - Larsen and Amarasinghe's "*Superword level parallelism*"
  - Fisher and Dietz's "*SIMD-within-a-register*"
- Product compilers
  - VAST/AltiVec, CodePlay/VectorC, Intel compiler,…

Conclusions

- Gaps exist between traditional vectorization and compilation for multimedia extensions
  - From differences between two architectures
  - From different programming styles, mismatch with language semantics, different code patterns

- Additional compiler techniques need to be developed or extended to bridge these gaps

Future Work

- Our first step to unleash the power of MMEs
  - Manual vectorization to see how far we can go
  - Implement our vectorizer on SUIF
- Propose new techniques to bridge the gaps
- Extend application domain
  - Traditional applications: SPECfp, SPECint
  - Applications for embedded systems

Thank You