Software Defined Implementation of MPEG4 Decoder on Sandblaster SB3500 DSP

Vaidyanathan Ramadurai, Mayan Moudgill, Daniel Iancu, Gary Nacer, John Glossner

Optimum Semiconductor Technologies Inc.

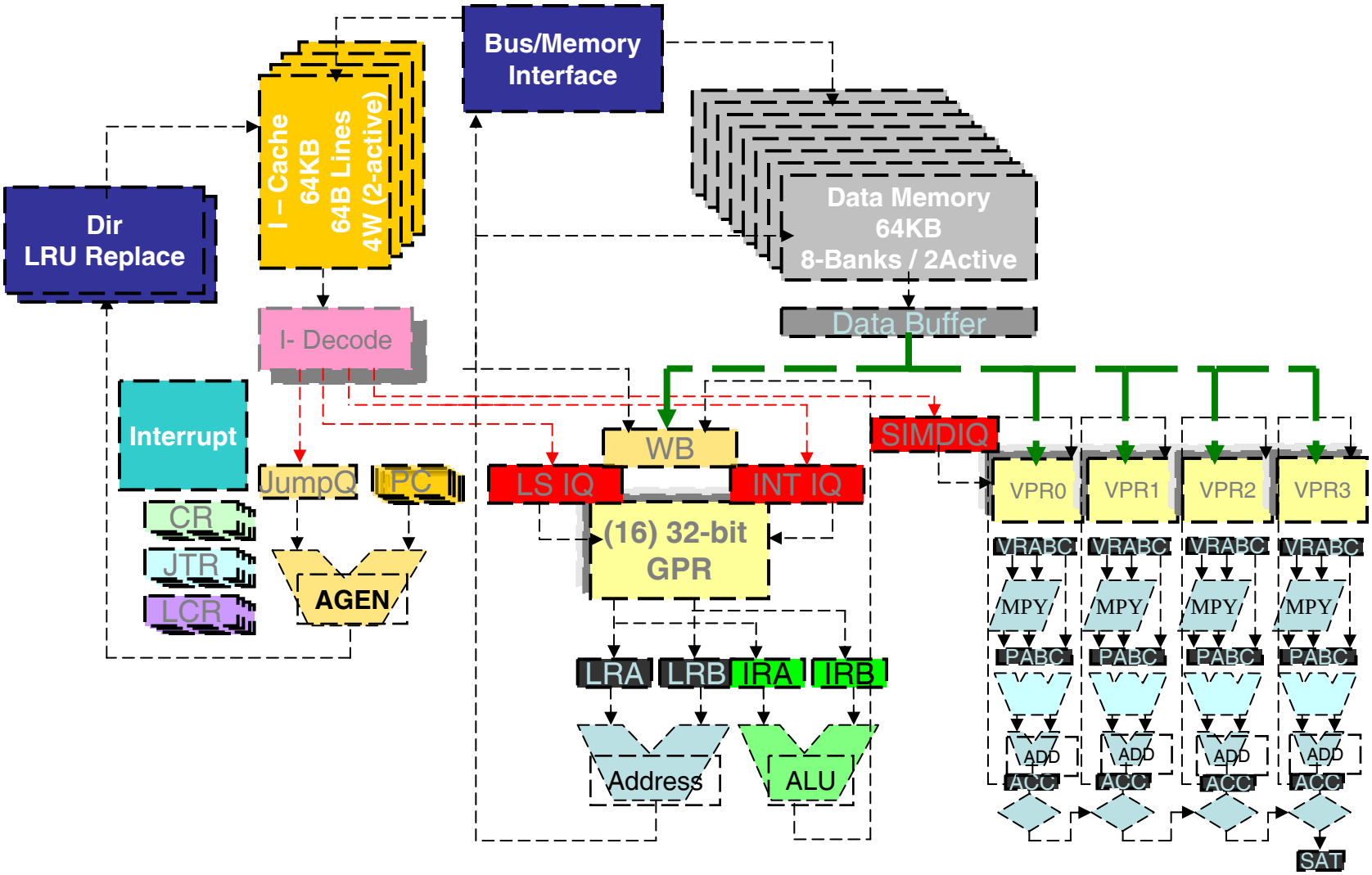
Agenda

- **Sandblaster SB3500 DSP**
  - Architecture
  - DSP Core
- **MPEG4 Decoder**
  - MPEG4 Decoder Blocks
  - Optimizing MPEG4 Decoder
  - Frame Buffer Management
- **Performance/Results**
- **Summary**

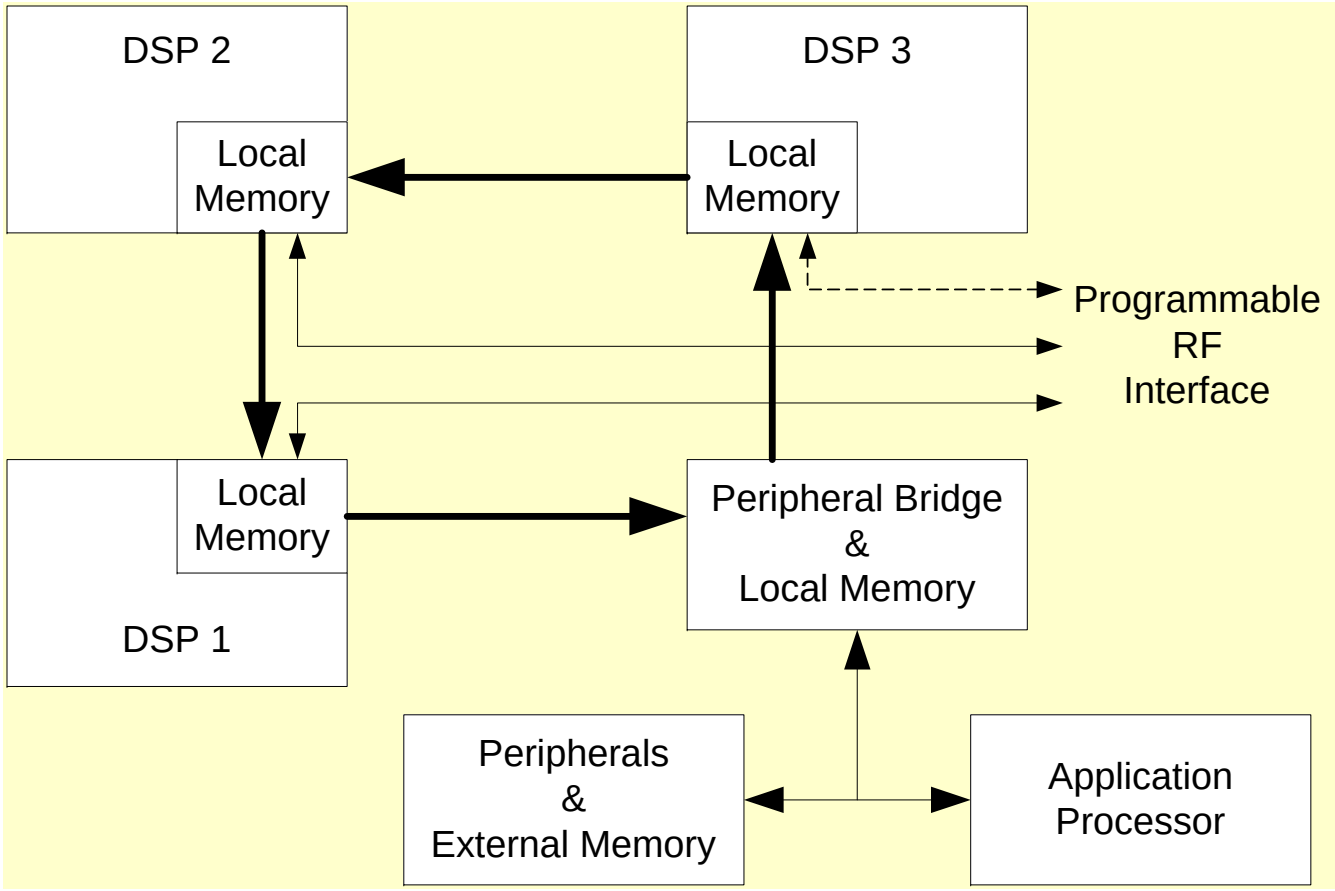
Background

- **Sandblaster SB3500 DSP**
  - Programmable DSP
  - Architectural features to support multimedia & wireless computation
- **Software Solution**
  - No hardware blocks
  - Implementation done in C
  - Provides flexibility and reusability
  - Low cost
  - Easy to maintain

Sandblaster Architecture

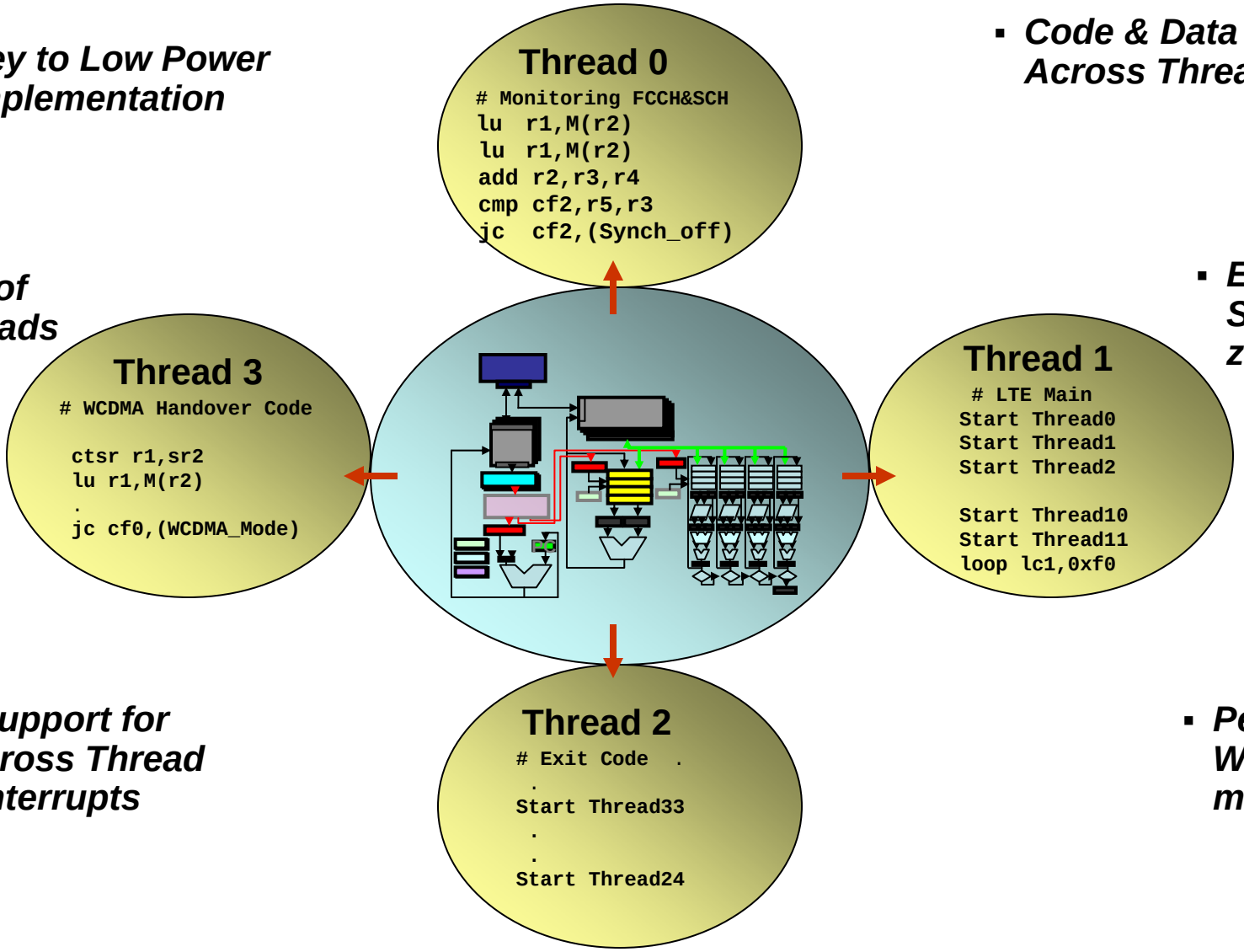


Sandblaster SB3500 DSP Core

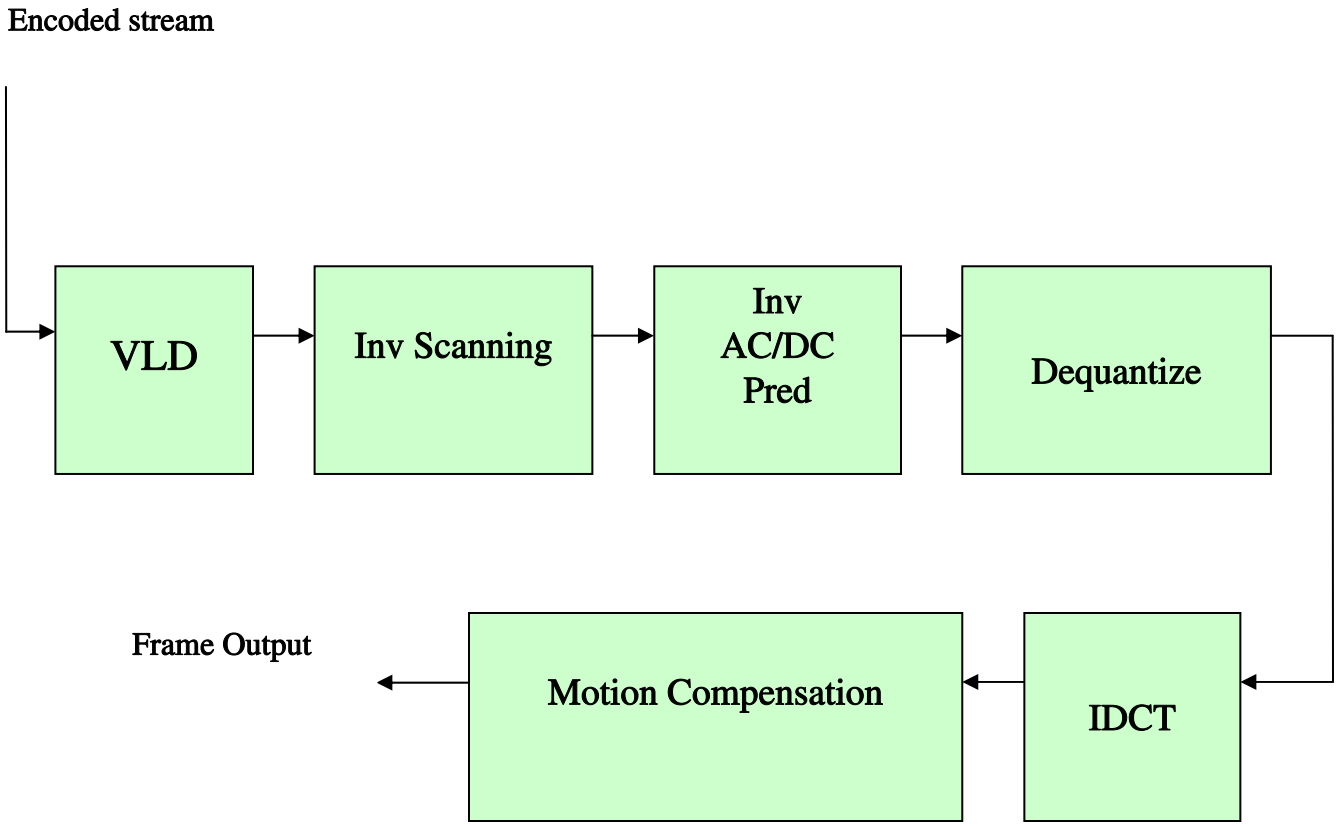


Thread Parallelism

- *Key to Low Power Implementation*
- *Code & Data Sharing Across Threads*
- *Sea of Threads*
- *Easy Synchronization*
- *Support for Cross Thread Interrupts*
- *Perfect for Wireless & multimedia*



MPEG4 Decoder Blocks



MPEG4 Decoder Complexity

Function	Computation (%)
Variable Length Decode	15%
Inverse Scan, Inv AC/DC	10%
Dequantize	15%
IDCT	20%
Motion Compensation	40%

Optimizing MPEG4 Decoder

8 Point 1-D DCT

Y0 = x0 + ... x7
Y1 = C1(X0-X7) + C3(x1-x6) + C5(x2-x5) + X7(x3-x4)
Y2 = C2(x0-x3-x4+x7) + C6(x1-x2-x5+x6)
Y3 = -C1(x2-x5) + C3(x0-x7) - C5(x3-x4) - C7(x1-x6)
Y4 = C4(x0-x1-x2+x3+x4-x5-x6+x7)
Y5 = C1(x6-x1) + C3(x3-x4) + C5(x0-x7) + C7(x2-x5)
Y6 = C2(x2-x1+x5-x6) + C6(x0-x3+x7-x4)
Y7 = C1(x4-x3) + C3(x2-x5) + C5(x6-x1) + C7(x0-x7)

1. Commonly used terms are computed first and reused using vector operations
2. Transpose of the resultant DCT matrix is taken using vector shuffles
3. Step 1 is repeated on the transposed result
4. Result of step 3 is rescaled
5. Transpose of the rescaled result is taken to retain row/column ordering

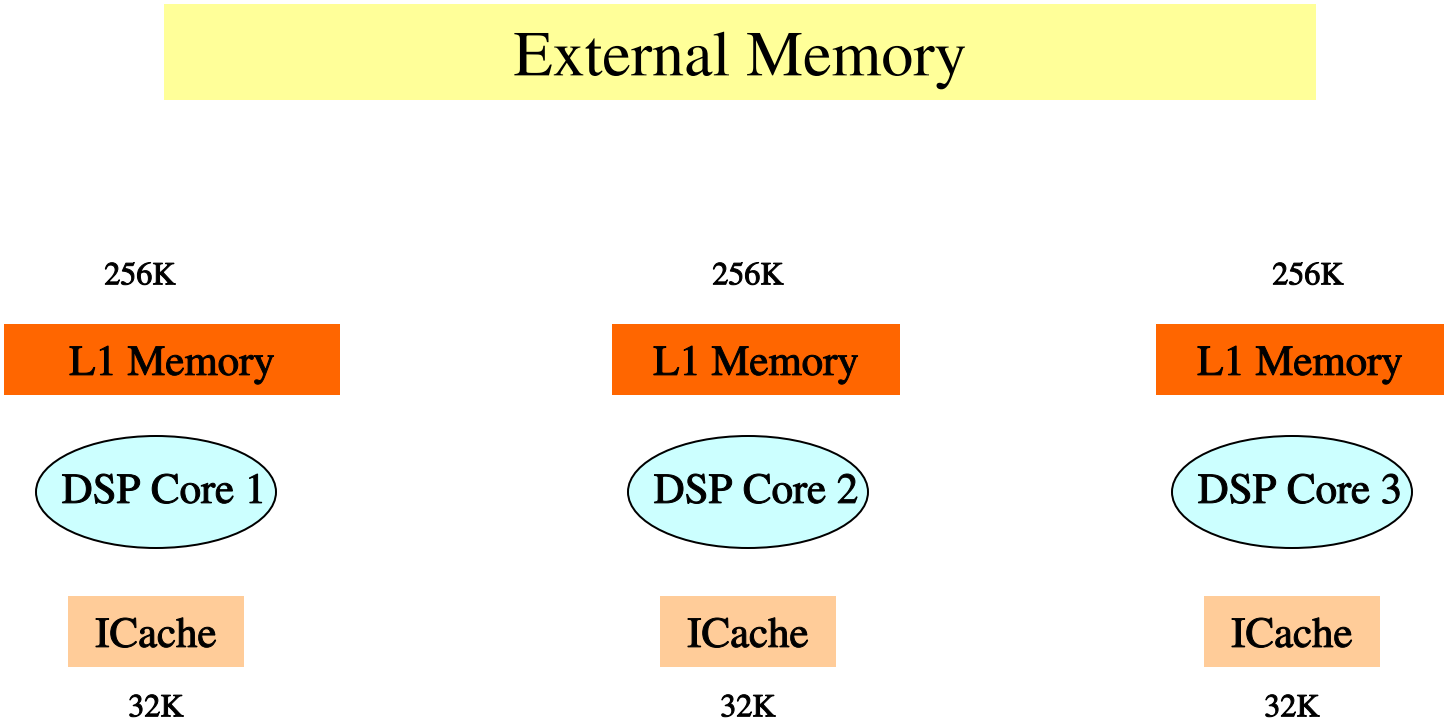
Optimizing MPEG4 Decoder

Motion Compensation

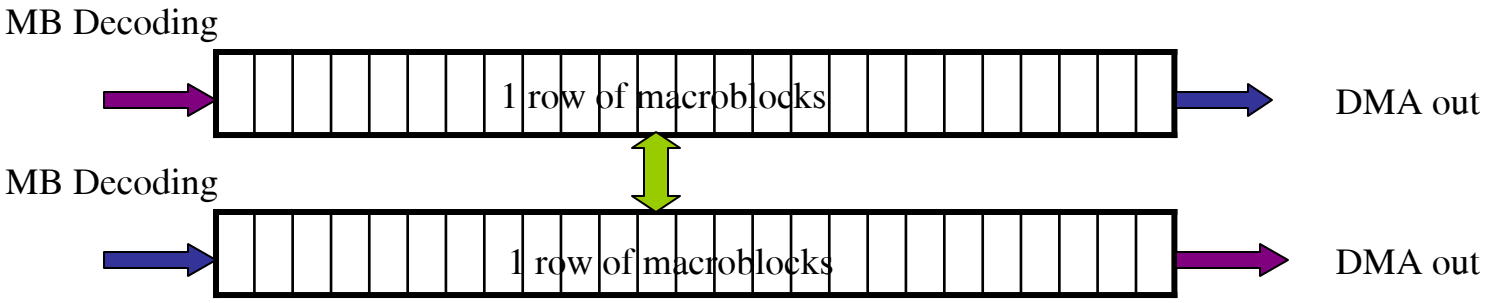
```
for (i = 0; i < BLOCK_SIZE; i++)
{
      for (j = 0; j < BLOCK_SIZE; j++)
      {
        WORD16 temp_res1;
        WORD16 temp_res2;
        temp_res1 =  ref9x9 [i*(BLOCK_SIZE+1)+j] +   ref9x9 [i*(BLOCK_SIZE+1)+j+1];
        temp_res2 =   temp_ref9x9 [i*(BLOCK_SIZE+1)+j] + temp_ref9x9 [i*(BLOCK_SIZE+1)+j+1];
        temp_res = temp_res1 + temp_res2;
        temp = (BlkIn [i*BLOCK_SIZE+j] + ((temp_res + 2 – VopRoundingType)>> 2));
        BlkOut [i*BLOCK_SIZE+j] = CLIP (temp, 0, 255);
      }
}
```

1. temp_res1 is computed using a single vector add instruction
2. Similarly, temp_res2 and temp_res are each computed using a single vector add instruction
3. (2-VopRoundingType), a constant, is broadcasted and stored in a 16 element vector register
4. Two vector adds and a vector shift thus computes temp
5. The final clipping is done using a vector minimum and a vector maximum instruction
6. Thus the inner BLOCK_SIZE loop is executed using only 7 SB3500 vector instructions

Sandblaster SB3500 Memory System



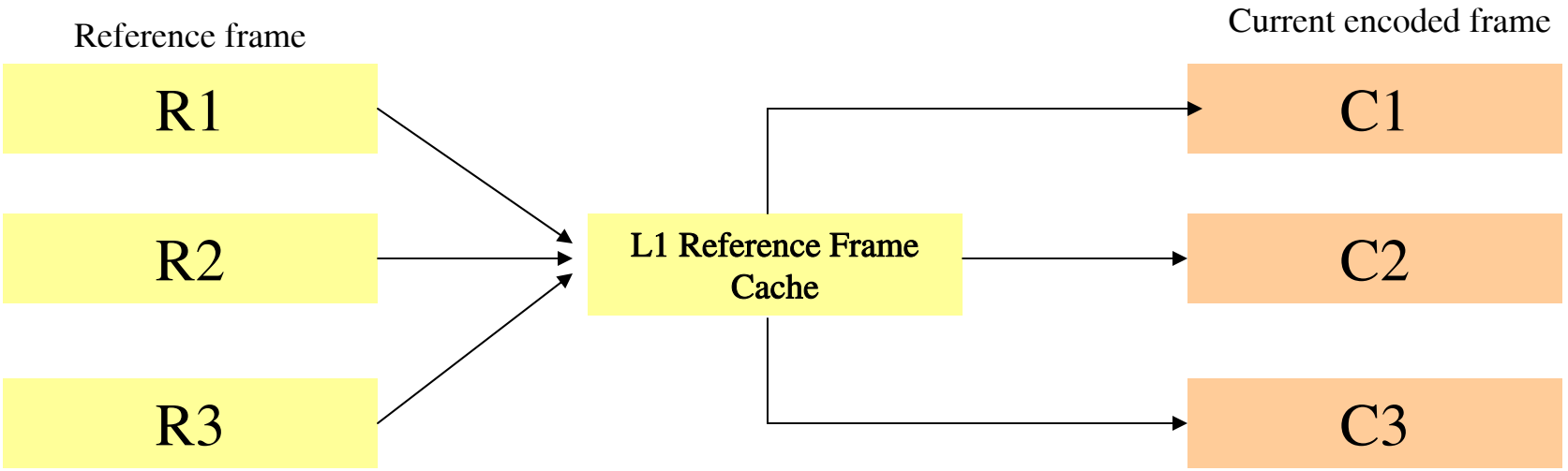
Frame Buffer Management



Output Buffer Management

- Checks if 1 row of macroblocks has been decoded
- e.g. in VGA(640x480) 1 row contains 40 macroblocks
- 2 rows of macroblocks stored in L1 memory
- Starts output DMA of 1 row of macroblocks from L1 memory to external memory
- Decoding of next row of macroblocks occurs in parallel with the output DMA of previous row

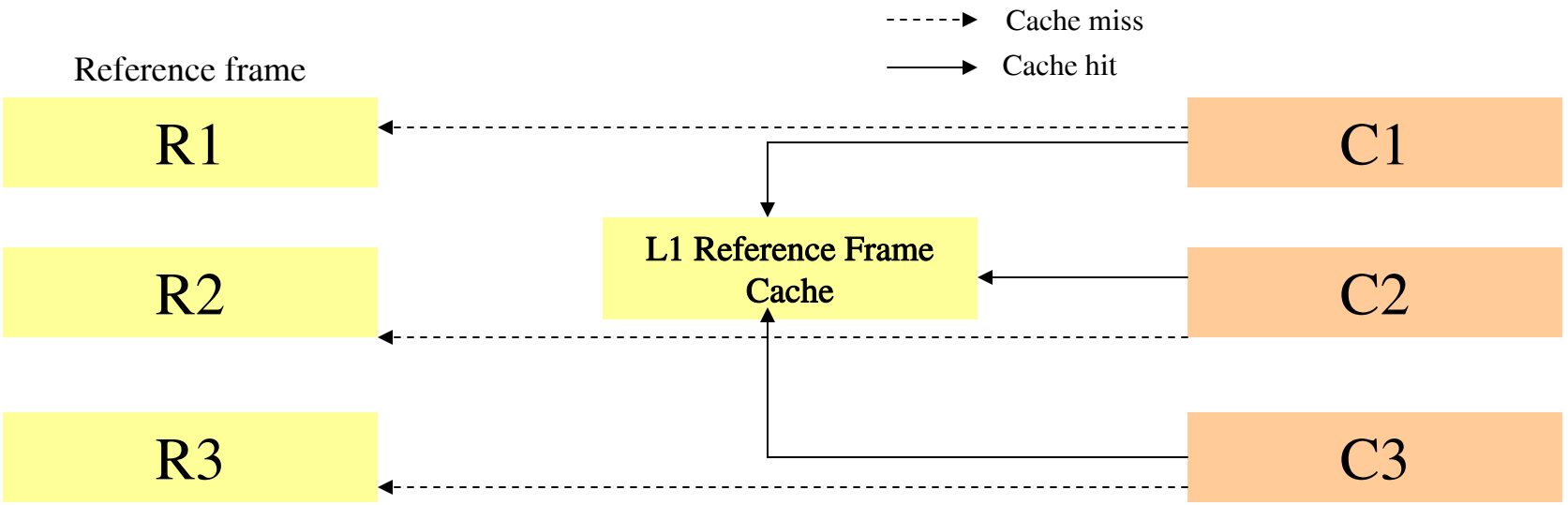
Frame Buffer Management



Input Buffer Management

- Reference frame divided into 3 pieces: R1, R2, R3
- Current frame divided into 3 pieces C1, C2 and C3
- R1 is DMA transferred from external memory to L1 memory once the decoding of first macroblock in C1 begins.
- R1 is transferred to L1 memory before motion compensation

Frame Buffer Management



- During motion compensation, a check is made if the reference block is in L1 memory.
- If the block is not in L1 memory, the block is taken from external memory, otherwise it is used from L1 memory.
- Similar method for R2 and R3.
- Implementation similar to a cache
- 80% hit ratio
- Save boundary pixels between {R1,R2} and {R2,R3} to improve hit ratio

SB3500 - SB3011 MPEG4 (VGA @ 30fps) Comparison

Device	MHz	Threads	Local memory utilization(Kbytes)
SB3011	900	12	640
SB3500	150	1	256

Summary

- Software Implementation of MPEG4 Decoder on Sandblaster 3500 DSP
- Software solution provides flexibility and reusability
- Optimized for real time VGA decoding @30fps
- Exploited SB3500 vector instructions for compute and DMAs for frame buffer management
- The SB3500 implementation utilizes less than 9% of the total DSP signal processing capacity
- SB3500 provides approximately 6x performance improvement with less than 50% less memory requirement over SB3011 device
- Scalable to higher resolutions like D1 and HD