

# Hardware Architecture of Motion Estimation (ME)

Shao-Yi Chien

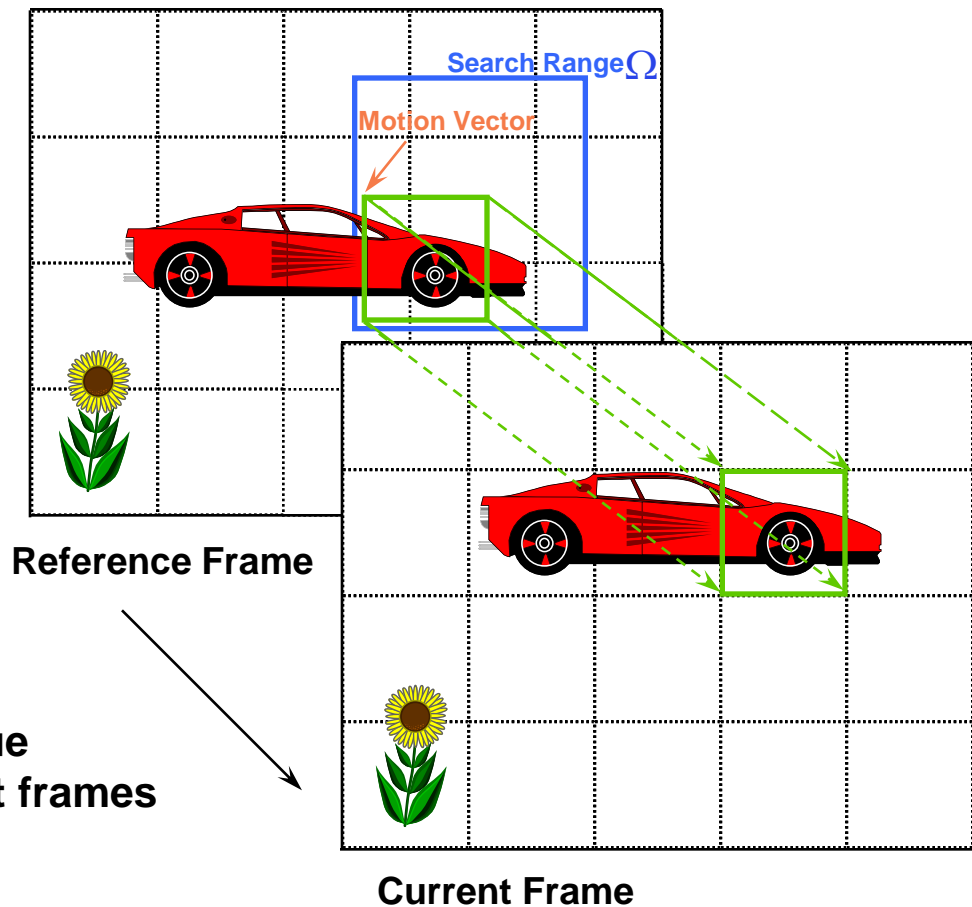
Most slides are prepared by Y.W.Huang,  
DSP/IC Design Lab.

# Block Matching Algorithm (BMA)

## (1/3)

- Track block regions from frame to frame
- Compromise between efficiently removing temporal redundancy and computation cost
- Usually, motion estimation is only performed at the **encoder side** to avoid the huge computation at the decoder side, so the motion data have to be transmitted from the encoder to the decoder as side information.
- Motion estimation takes more than **90%** of the total computation in modern video encoders
- Translational model is often adopted.

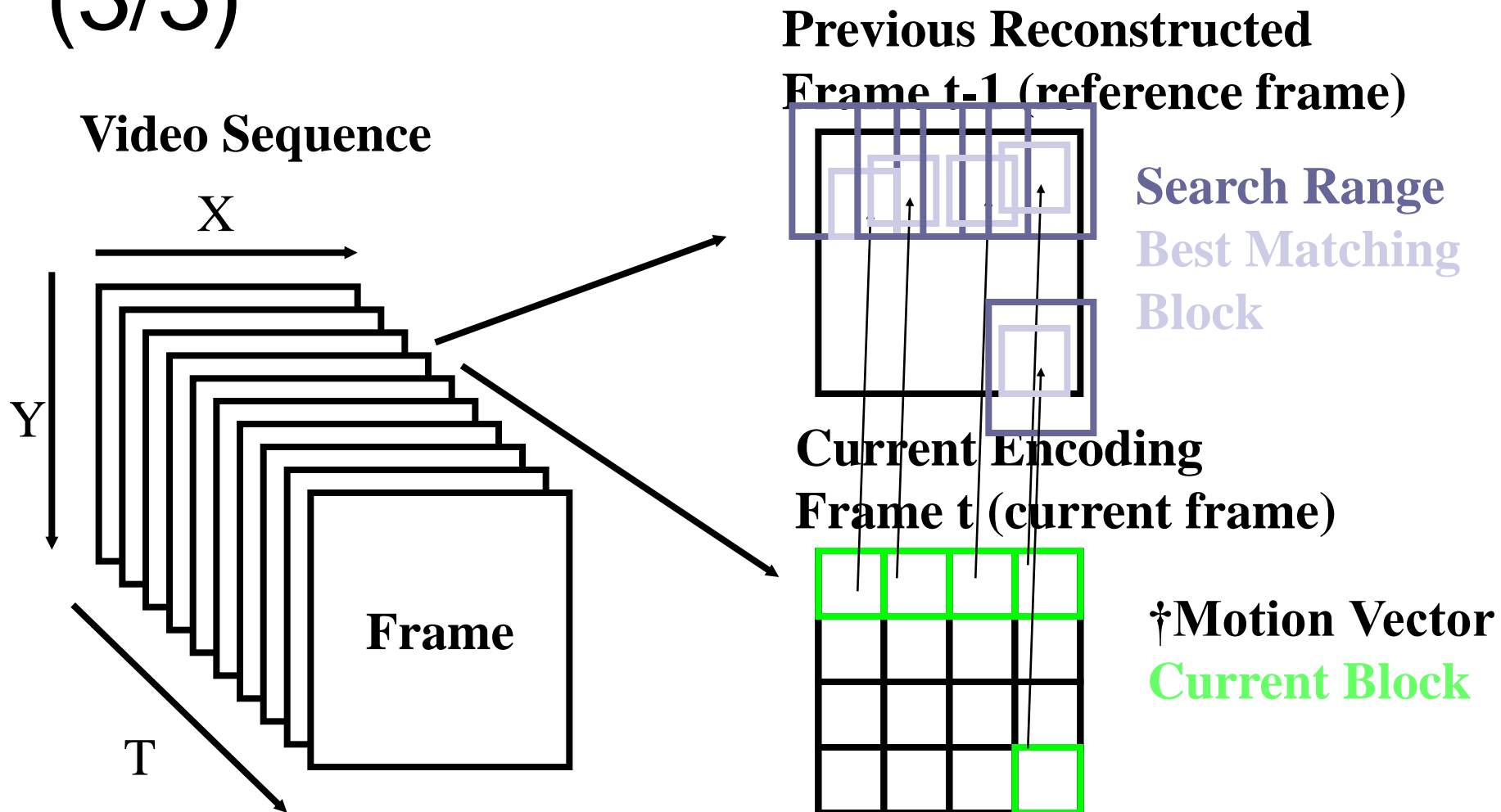
# Block Matching Algorithm (BMA) (2/3)



**Motion Vector**  
 $V_t(p,q) = (Vec_i, Vec_j)$   
the location in the search range  $\Omega$   
that has the maximum correlation value  
between blocks in temporally adjacent frames

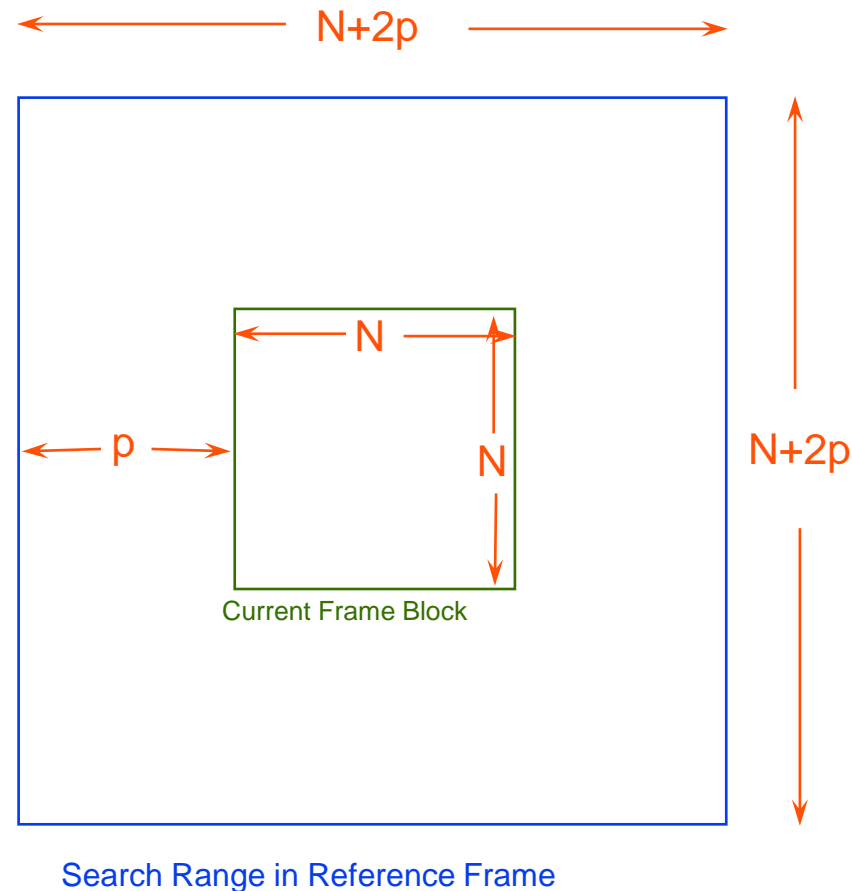
# Block Matching Algorithm (BMA)

## (3/3)

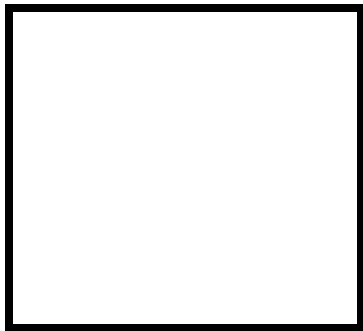


# Factors of Affecting BMA

- Search algorithm
- Matching criterion
  - SSD (sum of squared pixel difference, mostly used in software)
  - SAD (sum of absolute pixel difference, mostly used in hardware)
- Search range  $[-p, +p]$



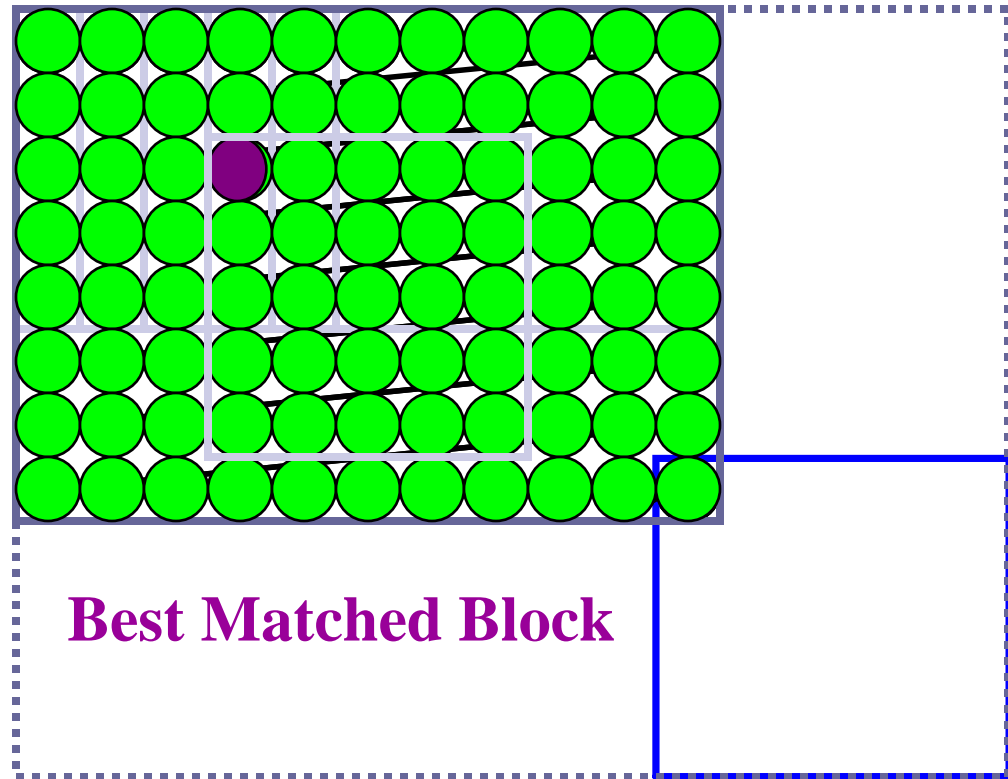
# Full-Search Block Matching Algorithm



**Current Block**  
Search Range

Reference Block  
(Candidate Block)

Candidate Search  
Position  
(Search Location)



$$SAD(i, j) = \sum_{k=1}^N \sum_{l=1}^N |x_t(k, l) - x_{t-1}(k+i, l+j)|$$



# Computation Complexity

```
Loop 1: For m= 0 to (width/blocksize)-1
Loop 2:   For n= 0 to (height/blocksize)-1
Loop 3:     For i = -d to d-1
Loop 4:       For j = -d to d-1
Loop 5:         For k = 0 to N-1
Loop 6:           For l = 0 to N-1
                  MAD(i,j) = MAD(i,j) + |X(k,l) - Y(k+i,l+j)|
                End (Loop 6)
              End (Loop 5)
            End (Loop 4)
          End (Loop 3)
        End (Loop 2)
      End (Loop 1)
```

} For each macroblock

} For each candidate search position

} Calculate the distortion, and chose the smallest one



# Inter-Level Parallelism (1/2)

■ **Loop 1:** For  $m = 0$  to  $N-1$

**Loop 2:** For  $n = 0$  to  $N-1$

**Loop 3:** For  $k = -p$  to  $p-1$

**Loop 4:** For  $l = -p$  to  $p-1$

$$\text{SAD}(k,l) = \text{SAD}(k,l) + |X(m,n) - Y(m+k,n+l)|$$

End (Loop 4)

End (Loop 3)

End (Loop 2)

End (Loop 1)

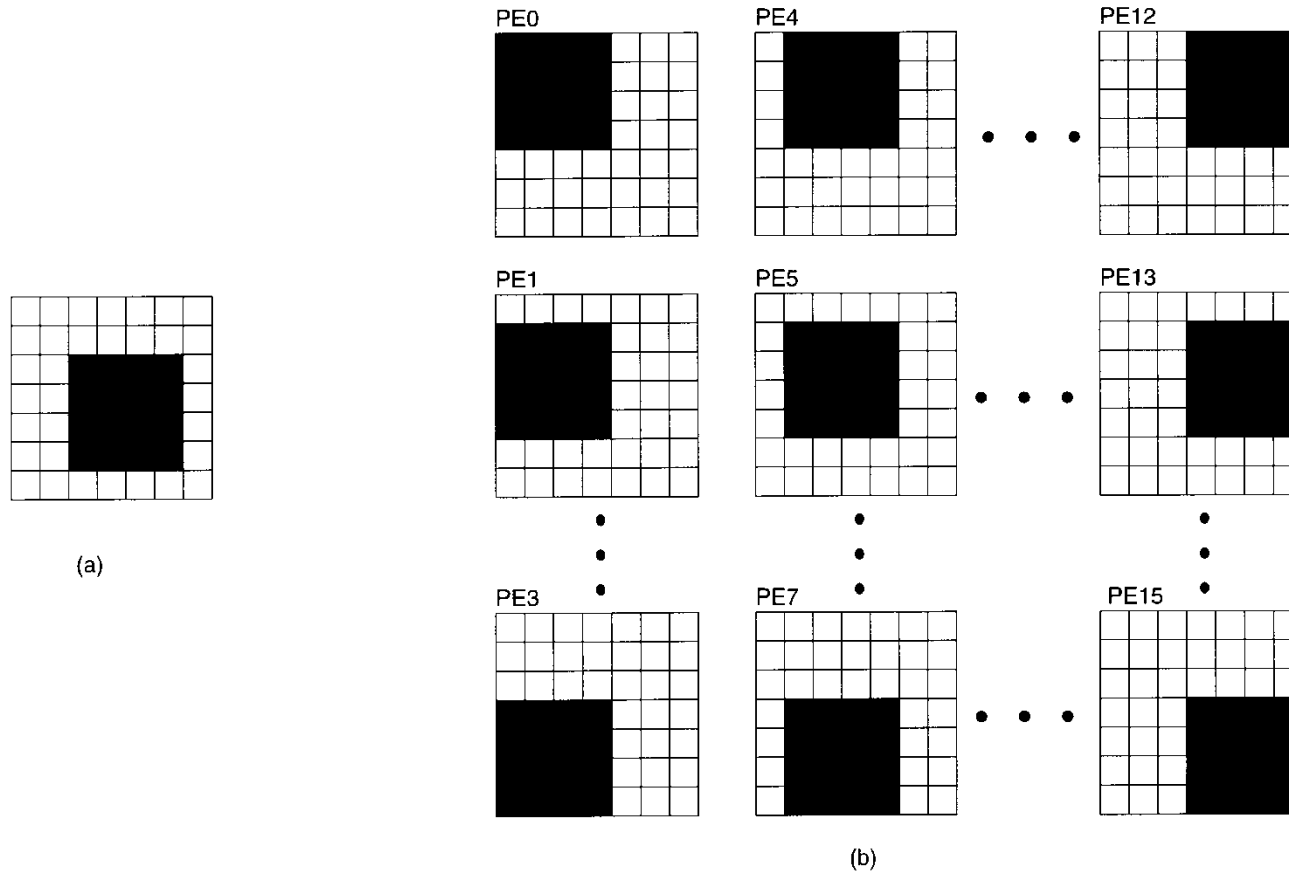
Each PE is responsible for the SAD of all pixels in a candidate.

For 2-D arrays ( $2p \times 2p$  PE's), at least  $(N \times N)$  cycles are required.

For 1-D arrays ( $2p \times 1$  PE's), at least  $(2p \times N \times N)$  cycles are required.



# Inter-Level Parallelism (2/2)



current block

candidate blocks



# Intra-Level Parallelism (1/2)

■ **Loop 1:** For  $k = -p$  to  $p-1$

**Loop 2:** For  $l = -p$  to  $p-1$

**Loop 3:** For  $m = 0$  to  $N-1$

**Loop 4:** For  $n = 0$  to  $N-1$

$$\text{SAD}(k,l) = \text{SAD}(k,l) + |X(m,n) - Y(m+k,n+l)|$$

End (Loop 4)

End (Loop 3)

End (Loop 2)

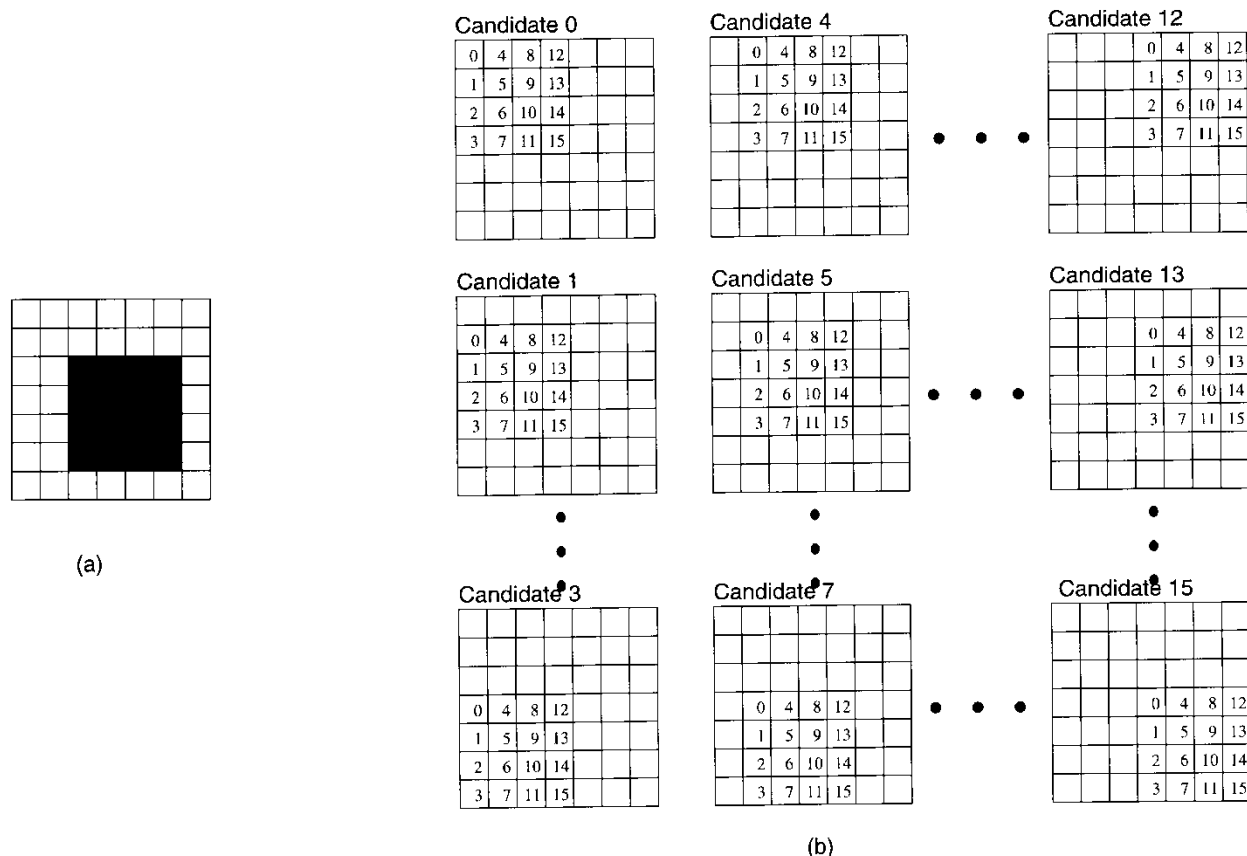
End (Loop 1)

Each PE is responsible for the SAD of one pixel in all candidates.

For 2-D arrays ( $N \times N$  PE's), at least  $(2p \times 2p)$  cycles are required.

For 1-D arrays ( $N \times 1$  PE's), at least  $(N \times 2p \times 2p)$  cycles are required.

# Intra-Level Parallelism (2/2)



current block

candidate blocks

# 1-D Systolic Array

K.-M. Yang, M.-T. Sun, L. Wu, “A family of VLSI design for the motion compensation block matching algorithm,” *IEEE Transactions on Circuits and Systems*, vol. 36, no. 10, October 1989

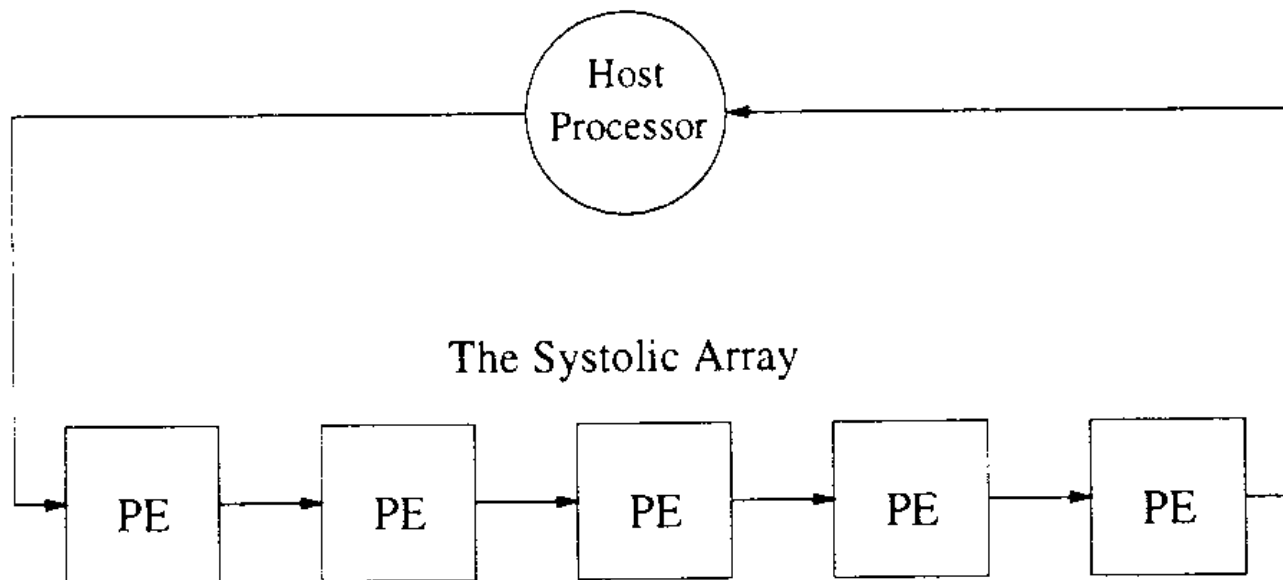


# Systolic Array (1/3)

- Systolic architecture (systolic array)
  - A network of processing elements (PEs) that rhythmically compute and pass data through the system
  - Modularity and regularity
  - All the PEs in the systolic array are uniform and fully pipelined
  - Contains only local interconnection

# Systolic Array (2/3)

- Typical systolic array





# Systolic Array (3/3)

## ■ Some relaxations

- Not only local but also neighbor interconnections
- Use of data broadcast operations
- Use of different PEs in the system, especially at the boundaries
- Also called as “semi-systolic array”



# 1-D Linear PE Array ME

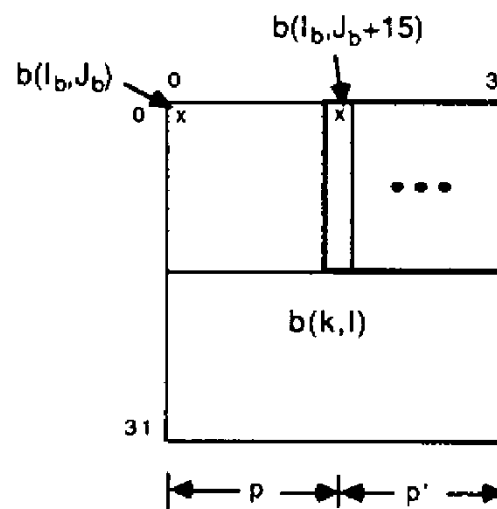
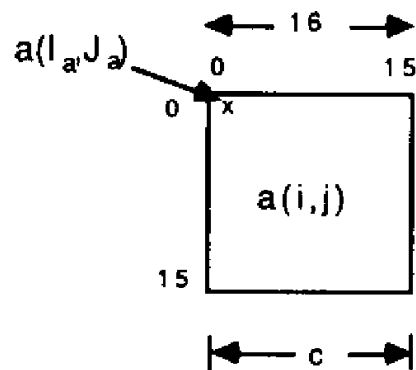
- The first chip design for block matching motion estimation in the world
- Two kinds of dataflow
  - Broadcasting reference frame, move current
  - Broadcasting current frame, move reference
- PE number = 1-D search range
- Each PE computes the SAD of a candidate macroblock
- Flexible block size (simply change data flow)
- Cascaded chip to enlarge the search range



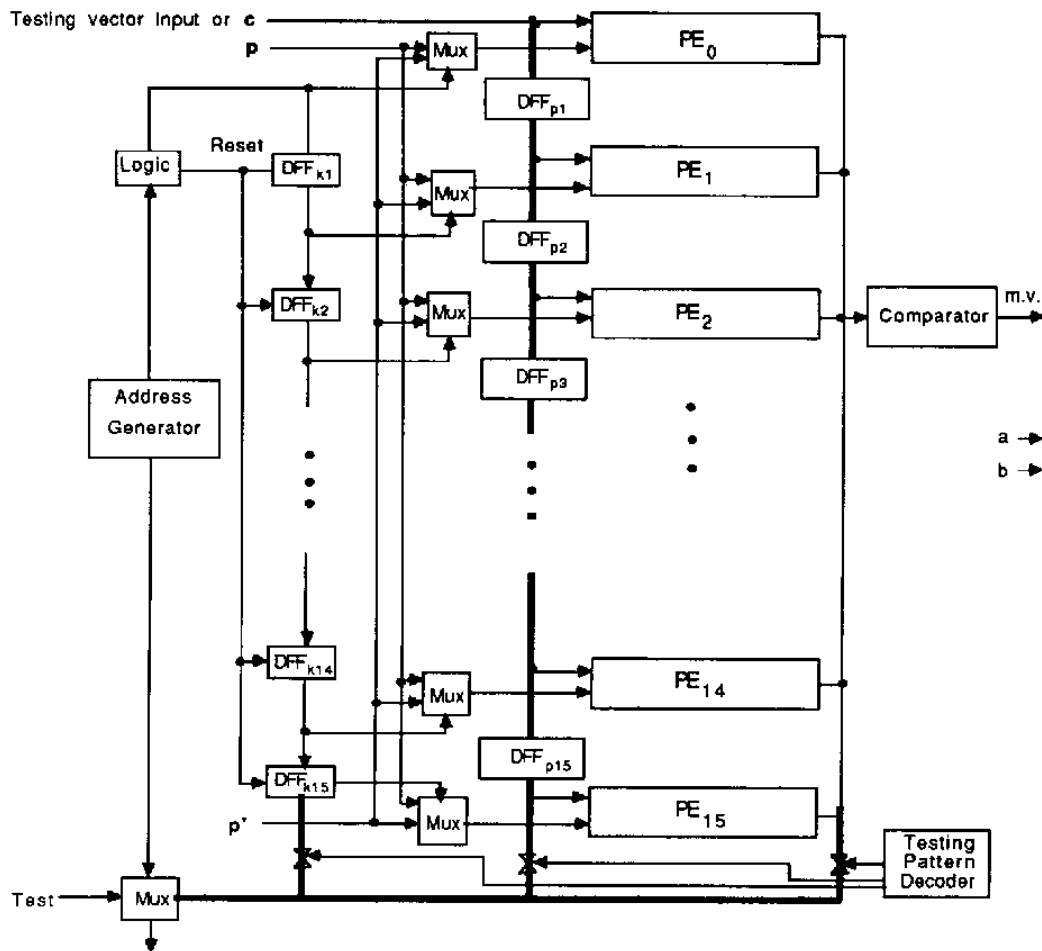
# Block Matching Algorithm

■ 
$$S(m_j) = \sum \sum |a(I_a + i, J_a + j) - b(I_b + k, J_b + l + m_j)|,$$
  
for  $m_j = 0, 1, \dots, 15$ .

- a: current frame
- b: reference frame

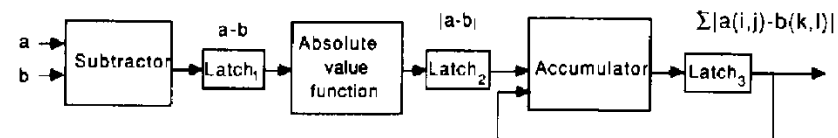


# Broadcasting Reference Frame



search range  $[-8, +7]$

block size  $16 \times 16$



## Processing Element

Accumulate the SAD of  
a candidate

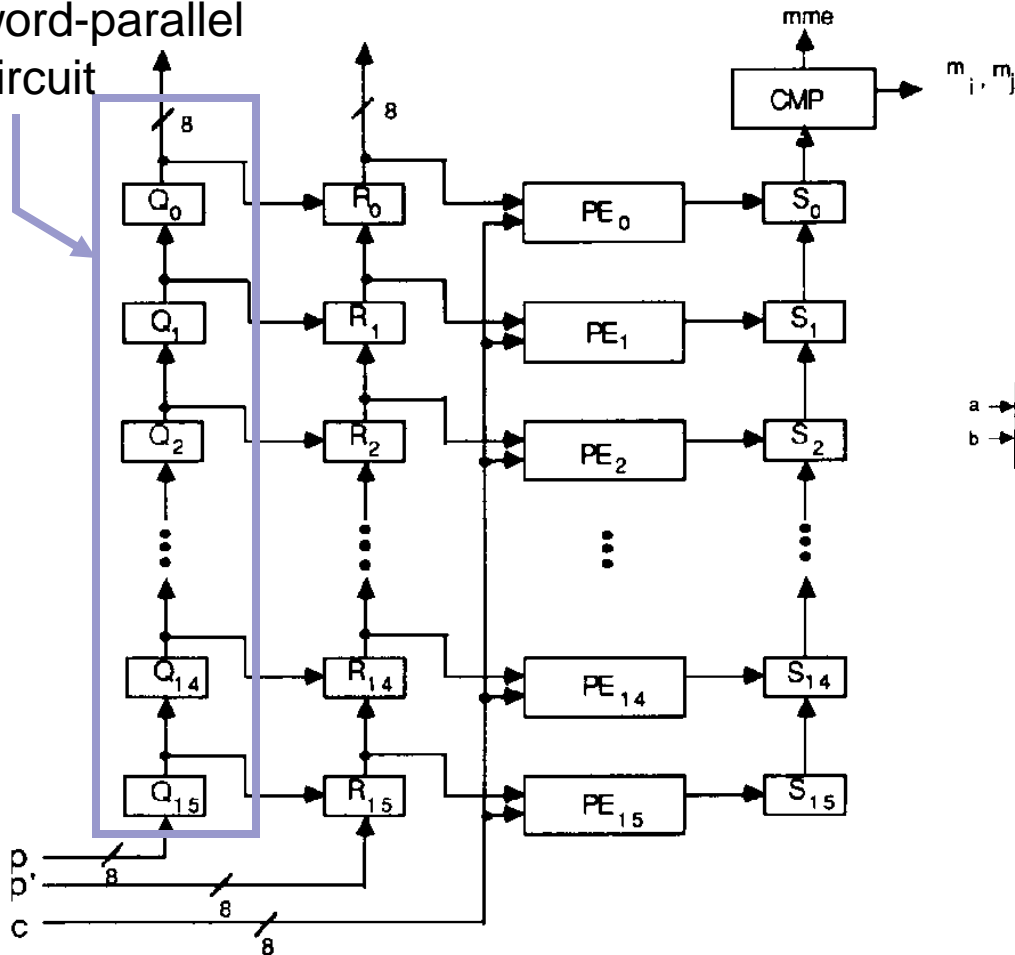


# Basic Data Flow

| Cycle time | Data sequences               | PE <sub>0</sub>               | PE <sub>1</sub>                 | PE <sub>2</sub>                 | PE <sub>14</sub>                 | PE <sub>15</sub>                 |
|------------|------------------------------|-------------------------------|---------------------------------|---------------------------------|----------------------------------|----------------------------------|
| t          | c p p'                       | $\sum \sum  a(i,j) - b(k,0) $ | $\sum \sum  a(i,j) - b(k,j+1) $ | $\sum \sum  a(i,j) - b(k,j+2) $ | $\sum \sum  a(i,j) - b(k,j+14) $ | $\sum \sum  a(i,j) - b(k,j+15) $ |
| 0          | a(0,0), b(0,0)               | a(0,0)-b(0,0)                 |                                 |                                 |                                  |                                  |
| 1          | a(0,1), b(0,1)               | a(0,1)-b(0,1)                 | a(0,0)-b(0,1)                   |                                 |                                  |                                  |
| 2          | a(0,2), b(0,2)               | a(0,2)-b(0,2)                 | a(0,1)-b(0,2)                   | a(0,0)-b(0,2)                   |                                  |                                  |
| 14         | a(0,14), b(0,14)             | a(0,14)-b(0,14)               | a(0,13)-b(0,14)                 |                                 | a(0,0)-b(0,14)                   |                                  |
| 15         | a(0,15), b(0,15)             | a(0,15)-b(0,15)               | a(0,14)-b(0,15)                 | a(0,13)-b(0,15)                 | a(0,1)-b(0,15)                   | a(0,0)-b(0,15)                   |
| 16+0       | a(1,0), b(1,0), b(0,16)      | a(1,0)-b(1,0)                 | a(0,15)-b(0,16)                 | a(0,14)-b(0,16)                 |                                  | a(0,1)-b(0,16)                   |
| 16+1       | a(1,1), b(1,1), b(0,17)      | a(1,1)-b(1,1)                 | a(1,0)-b(1,1)                   | a(0,15)-b(0,17)                 |                                  | a(0,15)-b(0,17)                  |
| 16+15      | a(1,15), b(1,15), b(0,31)    | a(1,15)-b(1,15)               | a(1,14)-b(1,15)                 | a(1,0)-b(1,2)                   |                                  | a(0,15)-b(0,30)                  |
| 2x16 + 0   | a(2,0), b(2,0), b(1,16)      | a(2,0)-b(2,0)                 | a(1,15)-b(1,16)                 | a(1,14)-b(1,16)                 |                                  | a(1,1)-b(1,16)                   |
| 2x16 + 1   | a(2,1), b(2,1), b(1,17)      | a(2,1)-b(2,1)                 | a(2,0)-b(2,1)                   | a(1,15)-b(1,17)                 |                                  |                                  |
| ⋮          | ⋮                            | ⋮                             | ⋮                               | a(2,0)-b(2,2)                   |                                  |                                  |
| 15x16 + 0  | a(15,0), b(15,0), b(14,16)   | a(15,0)-b(15,0)               | a(14,15)-b(14,16)               |                                 |                                  |                                  |
| 15x16 + 16 | a(15,15), b(15,15), b(14,31) | a(15,15)-b(15,15)             | a(15,14)-b(15,15)               | a(15,13)-b(15,15)               |                                  | a(14,15)-b(14,30)                |
| 16x16 + 0  |                              |                               | a(15,15)-b(15,16)               | a(15,14)-b(15,16)               |                                  |                                  |
| 16x16 + 1  |                              |                               |                                 | a(15,15)-b(15,17)               |                                  |                                  |
| 16x16 + 14 |                              |                               |                                 |                                 | a(15,15)-b(15,29)                | a(15,14)-b(15,29)                |
| 16x16 + 15 |                              |                               |                                 |                                 |                                  | a(15,15)-b(15,30)                |

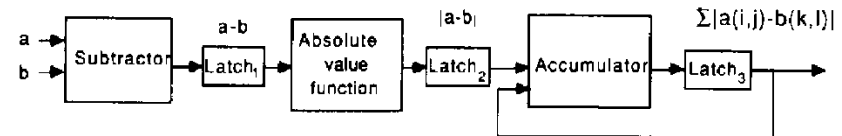
# Broadcasting Current Frame

Word-serial to word-parallel circuit



search range  $[-8, +7]$

block size  $16 \times 16$



## Processing Element

Accumulate the SAD of a candidate



# Basic Data Flow

b(0,0)  
b(0,1)

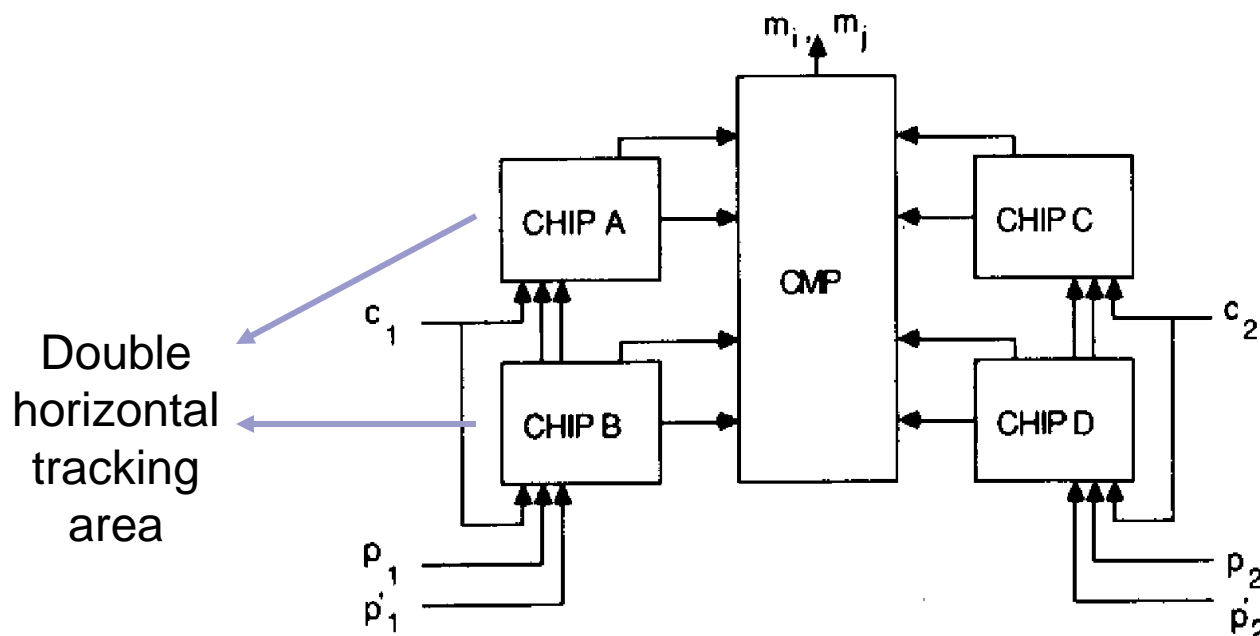
| Cycle time         | Data seque      |         |         | PE <sub>0</sub>                    | PE <sub>1</sub>                    | ... | PE <sub>14</sub>                   | PE <sub>15</sub>                   |
|--------------------|-----------------|---------|---------|------------------------------------|------------------------------------|-----|------------------------------------|------------------------------------|
| l                  | c               | p'      | p       | $\Sigma \Sigma  a(i,j)-b(k,l) $    | $\Sigma \Sigma  a(i,j)-b(k,l+1) $  | ... | $\Sigma \Sigma  a(i,j)-b(k,l+14) $ | $\Sigma \Sigma  a(i,j)-b(k,l+15) $ |
| 0x16+0<br>0x16+1   | a(0,0) b(0,16)  | b(1,0)  | b(1,1)  | a(0,0)-b(0,0)<br>a(0,1)-b(0,1)     | a(0,0)-b(0,1)<br>a(0,1)-b(0,2)     | ... | a(0,0)-b(0,14)<br>a(0,1)-b(0,15)   | a(0,0)-b(0,15)<br>a(0,1)-b(0,16)   |
| 0x16+14<br>0x16+15 | a(0,14) b(0,30) | b(1,14) | b(1,15) | a(0,14)-b(0,14)<br>a(0,15)-b(0,15) | a(0,14)-b(0,15)<br>a(0,15)-b(0,16) | ... | a(0,14)-b(0,28)<br>a(0,15)-b(0,29) | a(0,14)-b(0,29)<br>a(0,15)-b(0,30) |
| 1x16+0<br>1x16+1   | a(1,0) b(1,16)  | b(2,0)  | b(2,1)  | a(1,0)-b(1,0)<br>a(1,1)-b(1,1)     | a(1,0)-b(1,1)<br>a(1,1)-b(1,2)     | ... | a(1,0)-b(1,14)<br>a(1,1)-b(1,15)   | a(1,0)-b(1,15)<br>a(1,1)-b(1,16)   |
| 1x16+14<br>1x16+15 | a(1,14) b(1,30) | b(2,14) | b(2,15) | a(1,14)-b(1,14)<br>a(1,15)-b(1,15) | a(1,14)-b(1,15)<br>a(1,15)-b(1,16) | ... | a(1,14)-b(1,28)<br>a(1,15)-b(1,29) | a(1,14)-b(1,29)<br>a(1,15)-b(1,30) |

Load reference frame data from Q<sub>0</sub>-Q<sub>15</sub> to R<sub>0</sub>-R<sub>15</sub> in parallel for every 16 cycles

|                      |                   |          |          |  |  |     |  |  |
|----------------------|-------------------|----------|----------|--|--|-----|--|--|
| 14x16+0<br>14x16+1   | a(14,0) b(14,16)  | b(15,0)  | b(15,1)  | a(14,0)-b(14,0)<br>a(14,1)-b(14,1)     | a(14,0)-b(14,1)<br>a(14,1)-b(14,2)     | ... | a(14,0)-b(14,14)<br>a(14,1)-b(14,15)   | a(14,0)-b(14,15)<br>a(14,1)-b(14,16)   |
| 14x16+14<br>14x16+15 | a(14,14) b(14,30) | b(15,14) | b(15,15) | a(14,14)-b(14,14)<br>a(14,15)-b(14,15) | a(14,14)-b(14,15)<br>a(14,15)-b(14,16) | ... | a(14,14)-b(14,28)<br>a(14,15)-b(14,29) | a(14,14)-b(14,29)<br>a(14,15)-b(14,30) |
| 15x16+0<br>15x16+1   | a(15,0) b(15,16)  | b(0,0)   | b(0,1)   | a(15,0)-b(15,0)<br>a(15,1)-b(15,1)     | a(15,0)-b(15,1)<br>a(15,1)-b(15,2)     | ... | a(15,0)-b(15,14)<br>a(15,1)-b(15,15)   | a(15,0)-b(15,15)<br>a(15,1)-b(15,16)   |
| 15x16+14<br>15x16+15 | a(15,14) b(15,30) | b(0,14)  | b(0,15)  | a(15,14)-b(15,14)<br>a(15,15)-b(15,15) | a(15,14)-b(15,15)<br>a(15,15)-b(15,16) | ... | a(15,14)-b(15,28)<br>a(15,15)-b(15,29) | a(15,14)-b(15,29)<br>a(15,15)-b(15,30) |

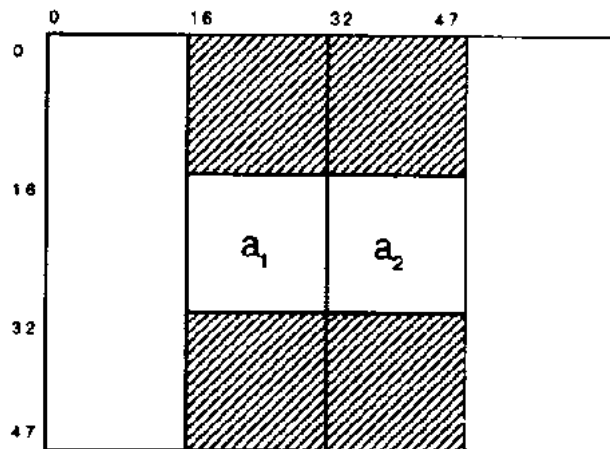
# Cascaded Chip Design

- For example, search range is extended from  $[-8, +7]$  to  $[-16, +15]$ , and motion estimation of 2 macroblocks are processed simultaneously

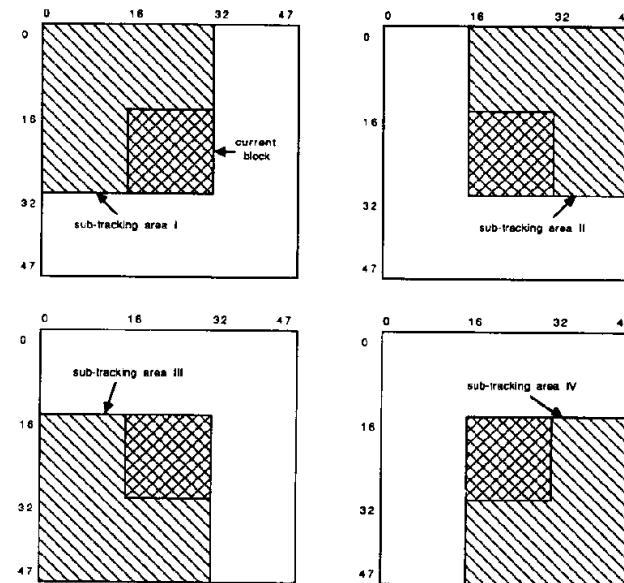


# Overlapped Search Area

- Overlapped search area can be broadcasted to each chip to save the bandwidth



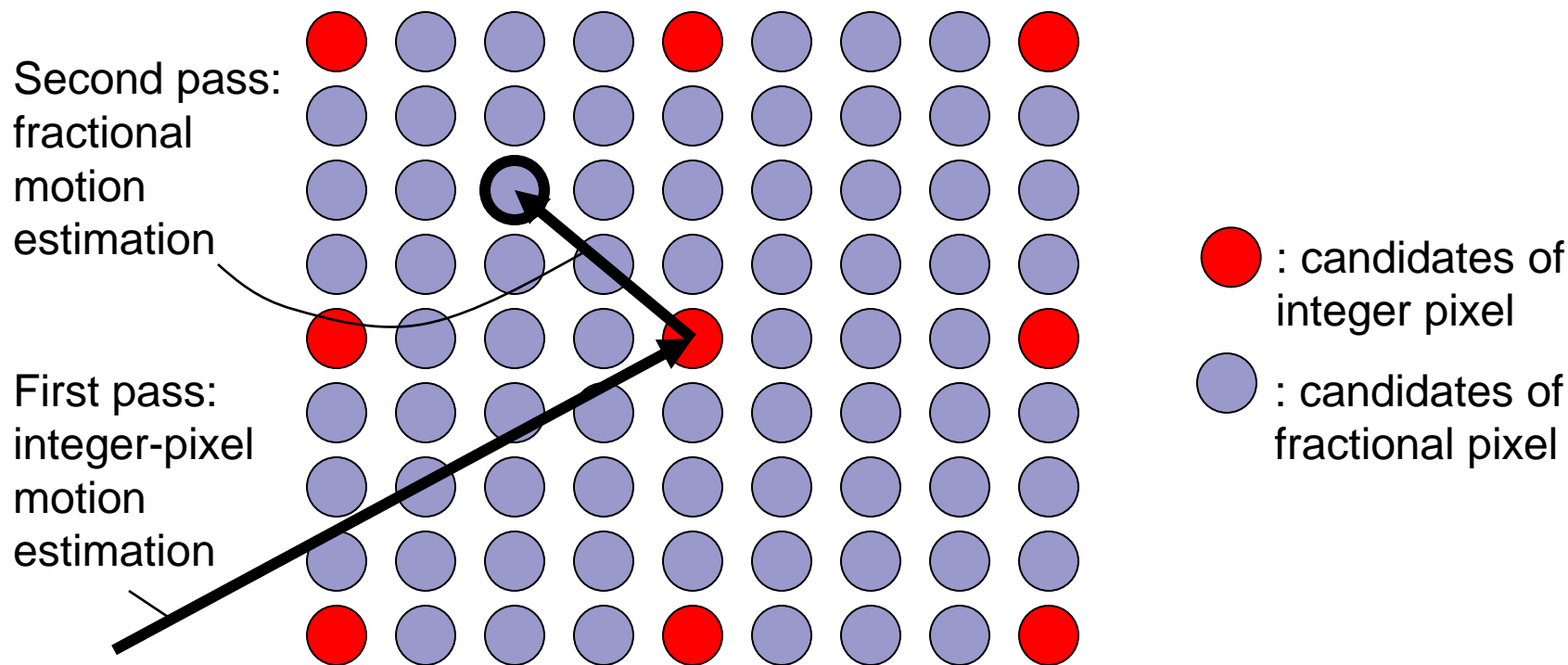
Overlapped tracking area of two adjacent blocks



Overlapped sub-tracking area

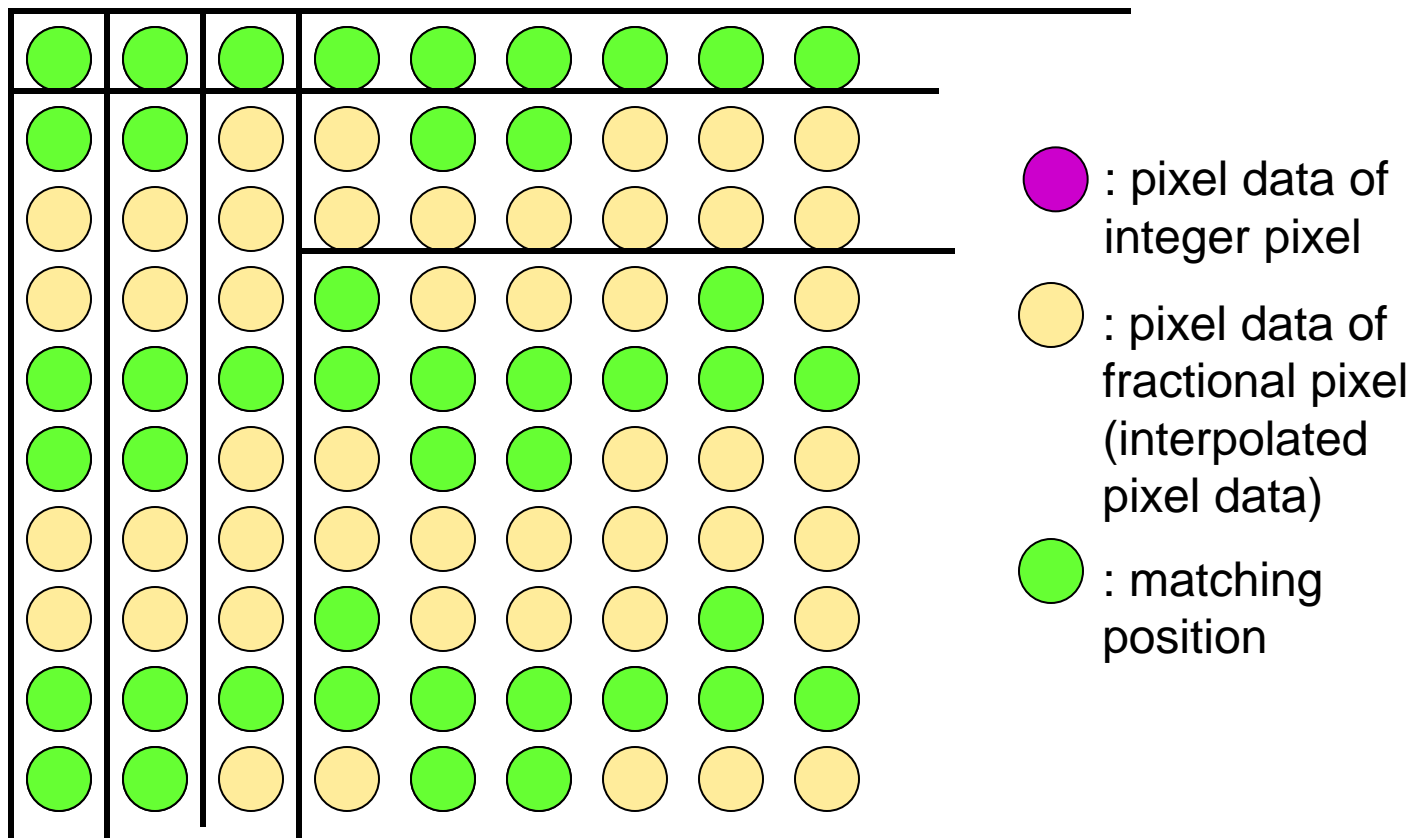
# Motion Estimation with Fractional Precision (1/2)

## ■ Quarter-pel precision



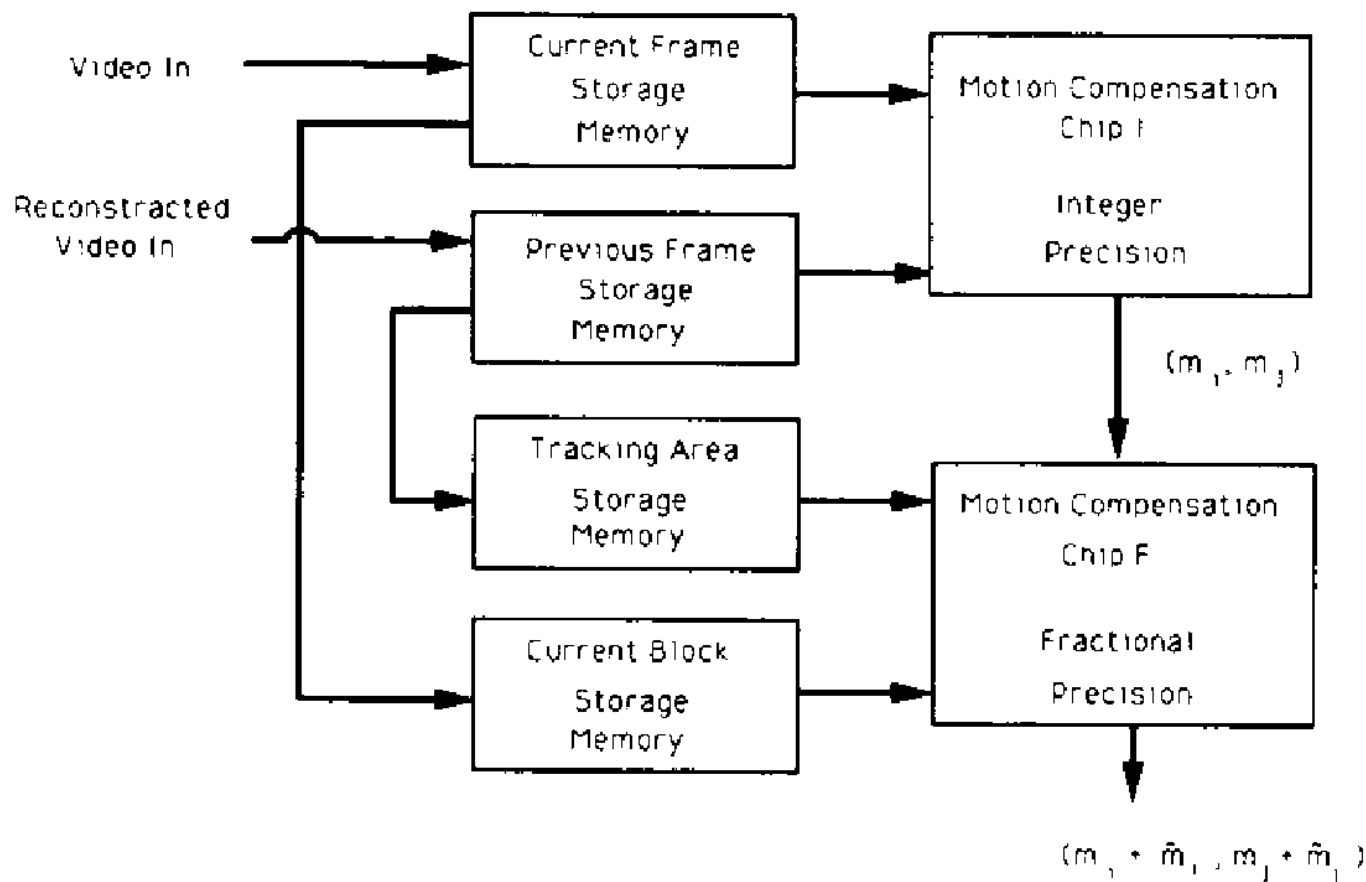


# Motion Estimation with Fractional Precision (2/2)



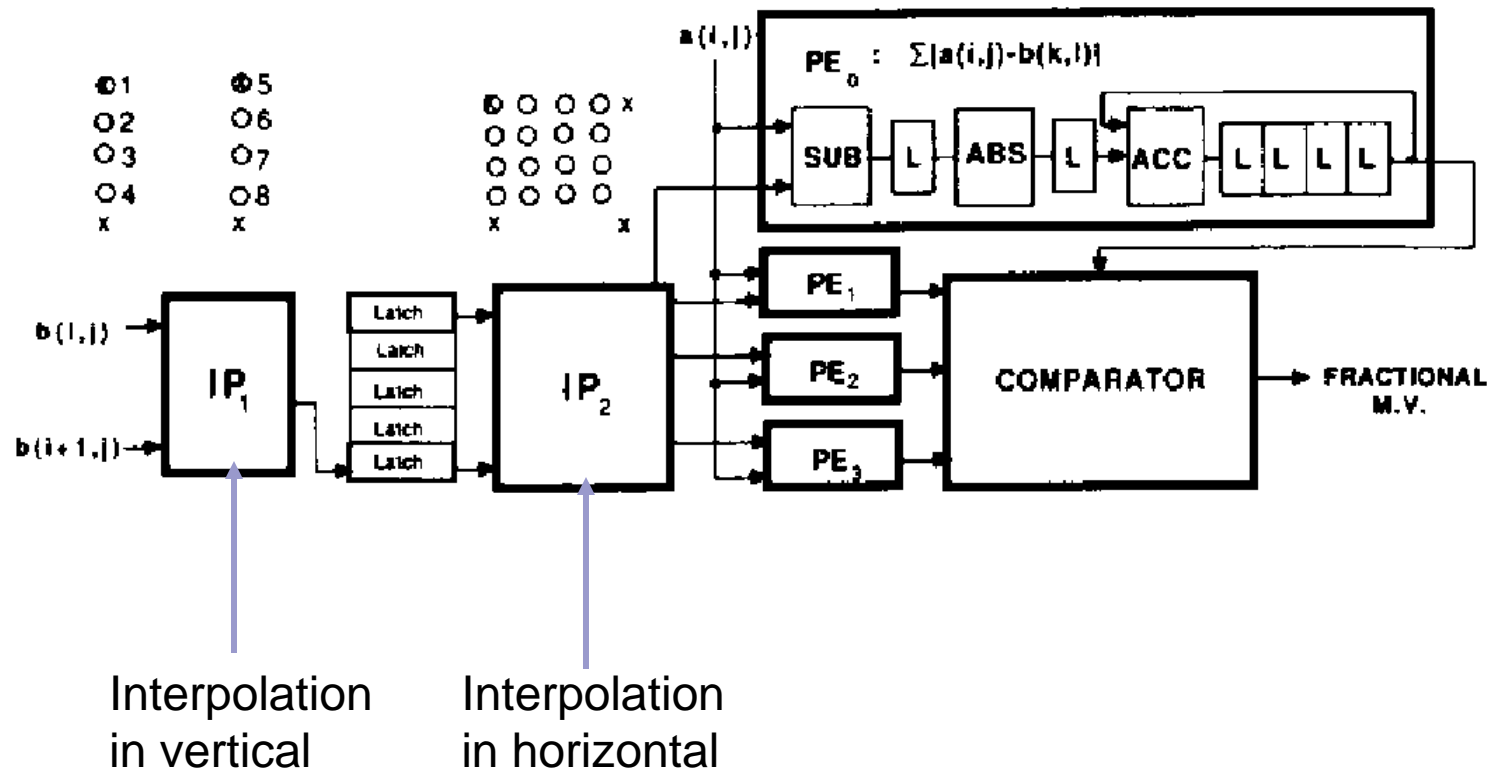


# Fractional Motion Compensation Chip-Pair Design





# Block Diagram of a Fractional Motion Estimation Chip



# 2-D Systolic Array

H. Yo and Y. H. Hu, “A novel modular systolic array architecture for full-search block matching motion estimation,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 5, no. 5, October 1995



# 6-D Array to 3-D Array

```

do v = 0 to Nv - 1
  do h = 0 to Nh - 1
    MV(h, v) = (0, 0)
    Dmin(h, v) = ∞
    do m = -p to p - 1
      do n = -p to p - 1
        MAD(m, n) = 0
        do i = 0 to N - 1
          do j = 0 to N - 1
            MAD(m, n) = MAD(m, n) + |x(i, j) - y(i + m, j + n)|
          enddo j
        enddo i
        if Dmin(h, v) > MAD(m, n)
          Dmin(h, v) = MAD(m, n)
          MV(h, v) = (m, n)
        endif
      enddo n
    enddo m
  enddo h
enddo v

```

$$\begin{cases}
 b = vN_h + h & 0 \leq b < N_{hv} = N_h N_v \\
 h = b \bmod N_h \\
 v = \lfloor \frac{b}{N_h} \rfloor \\
 l = 2p(m + p) + n + p & 0 \leq l < l_p = (2p)^2 \\
 m = \lfloor \frac{l}{2p} \rfloor - p \\
 n = l \bmod 2p - p \\
 k = iN + j & 0 \leq k < N^2 \\
 i = \lfloor \frac{k}{N} \rfloor \\
 j = k \bmod N
 \end{cases}$$

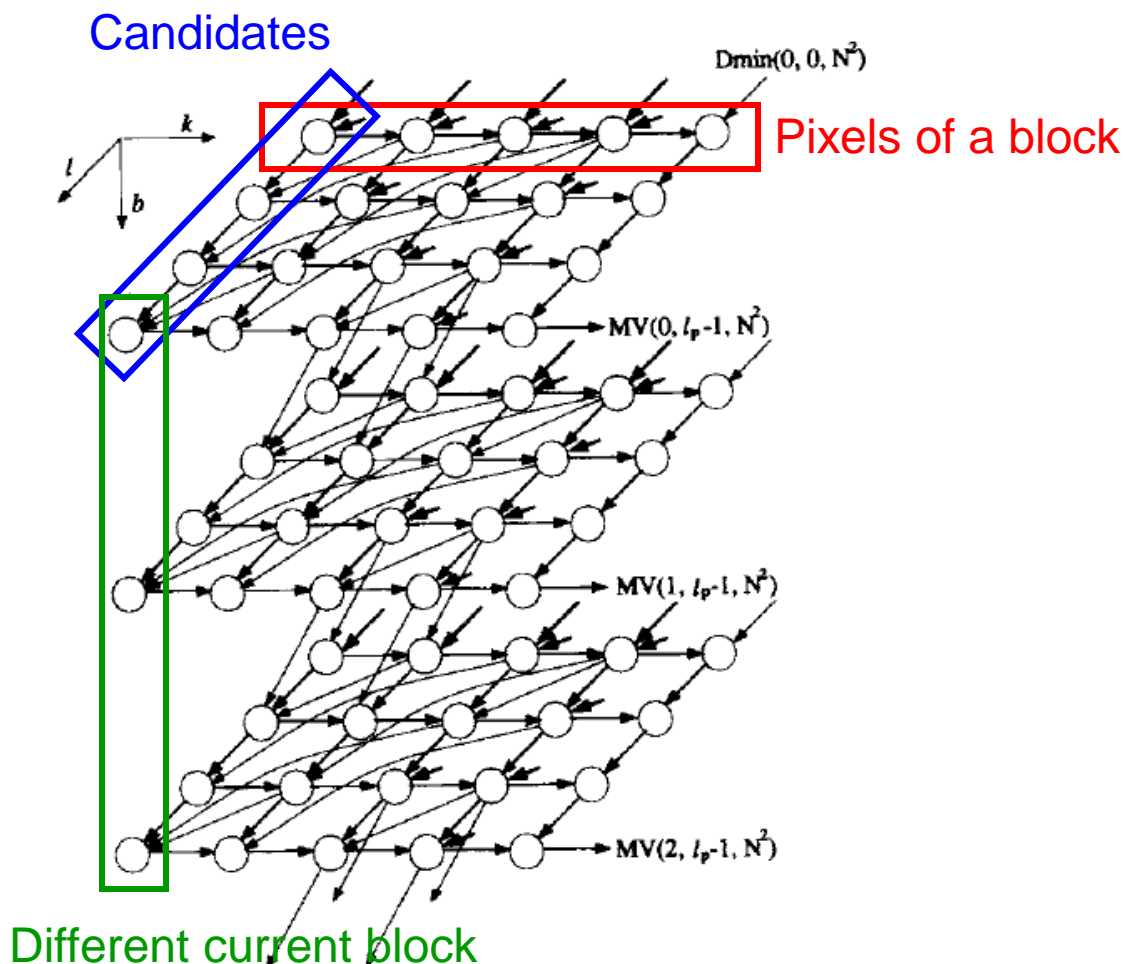
```

do b = 0 to Nhv - 1
  MV(b) = 0
  Dmin(b) = ∞
  do l = 0 to lp - 1
    MAD(l) = 0
    do k = 0 to N2 - 1
      MAD(l) = MAD(l) + |xs(k) - ys(k + l)|
    enddo k
    if Dmin(b) > MAD(l)
      Dmin(b) = MAD(l)
      MV(b) = l
    endif
  enddo l
enddo b

```

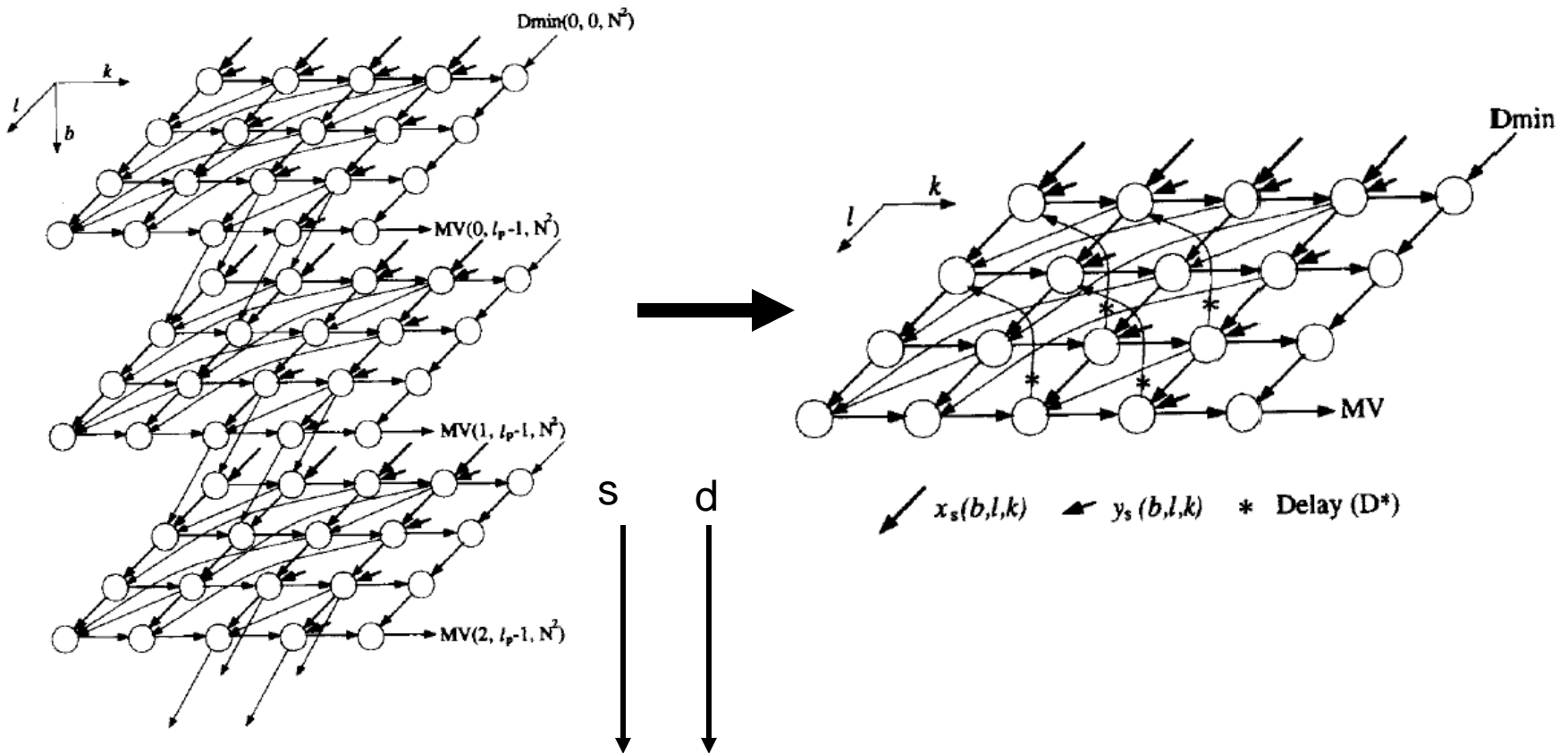
b: block index  
 l: search candidate index  
 k: pixel index

# 3-D DG of Motion Estimation

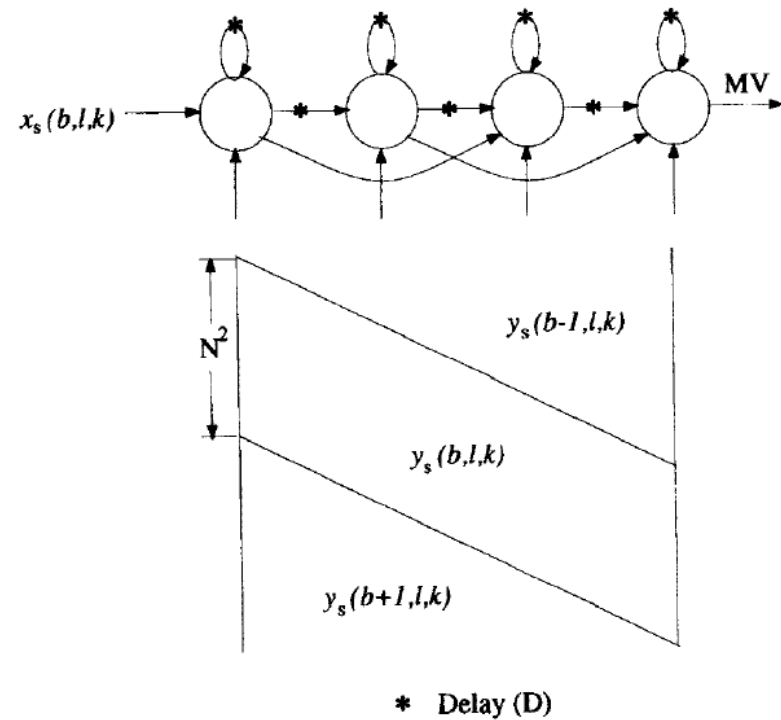
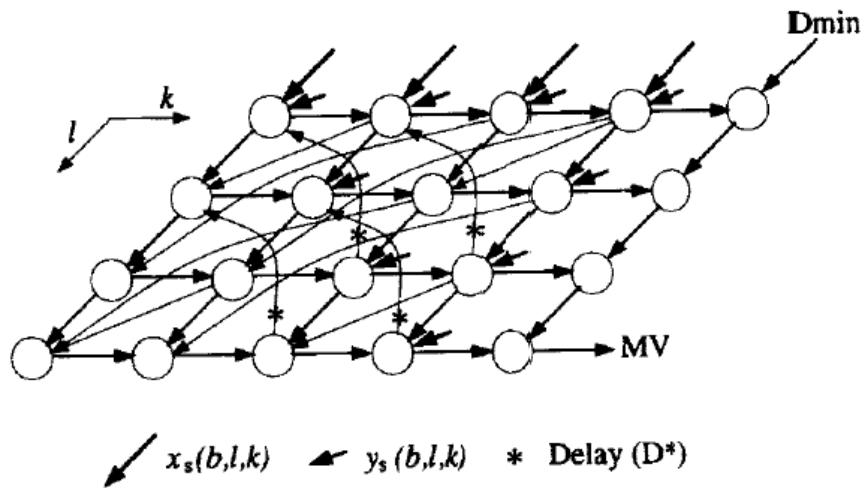


- Three axis
  - b
  - l
  - k
- Data reuse of reference frame is also considered

# 3-D DG to 2-D SFG



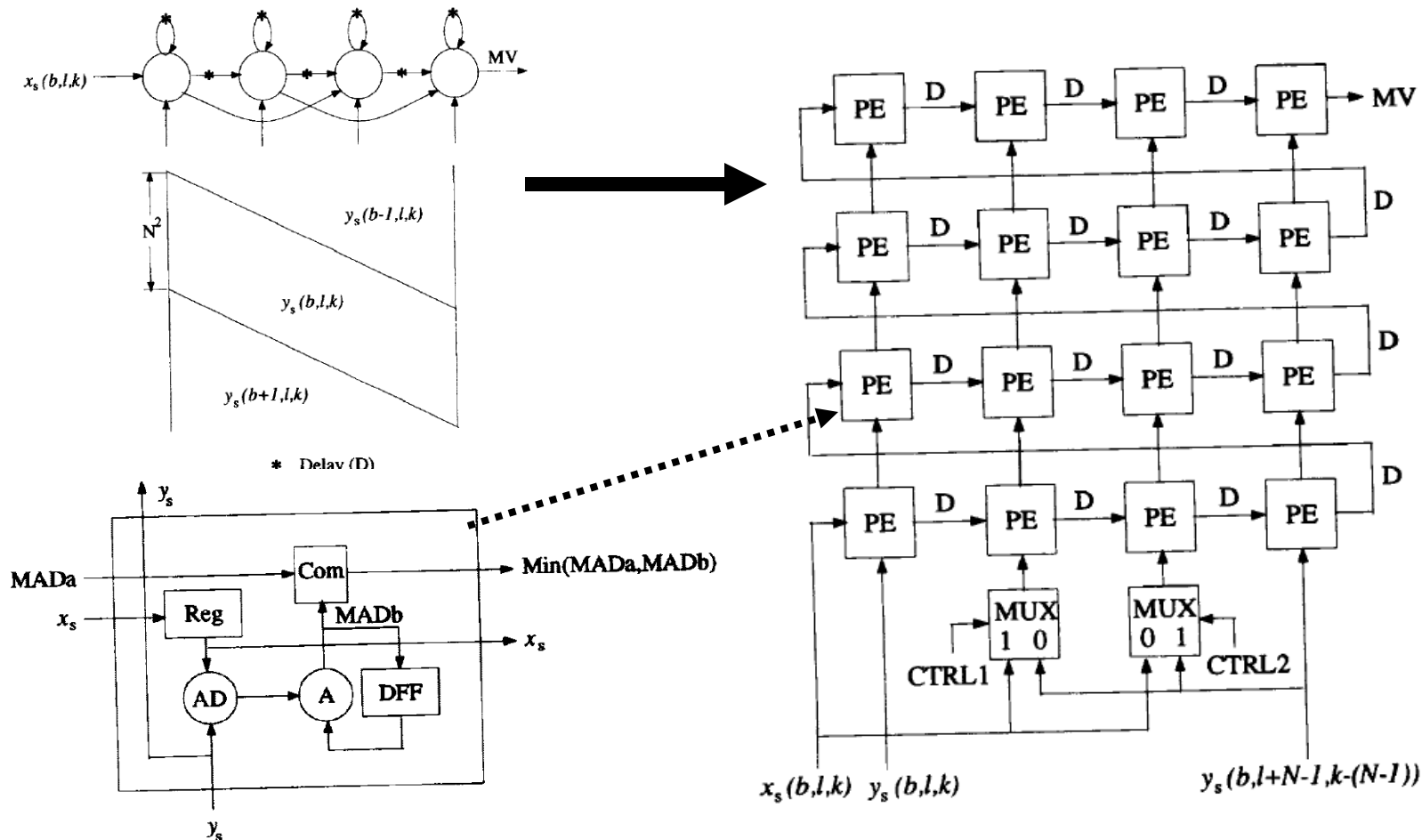
# 2-D SFG to 1-D SFG



$$D^* = N^2 D$$



# 1-D SFG to 2-D Mesh





# Tree-Based Architecture

Y.-S. Jehng, L.-G. Chen, and T.-D. Chiueh, “An efficient and simple VLSI tree architecture for motion estimation algorithms,” *IEEE Transactions on Signal Processing*, vol. 41, no. 2, February 1993

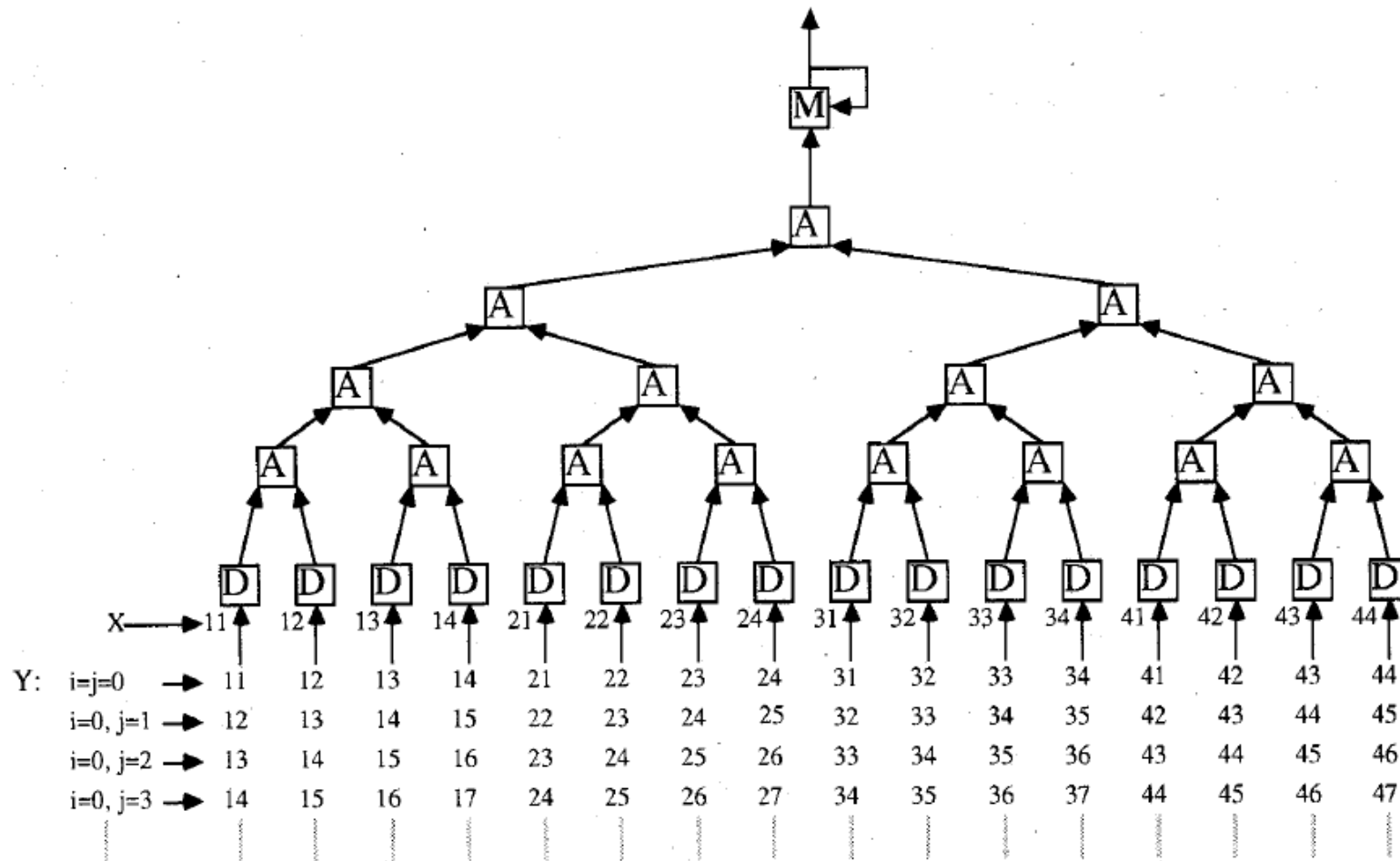


# Features

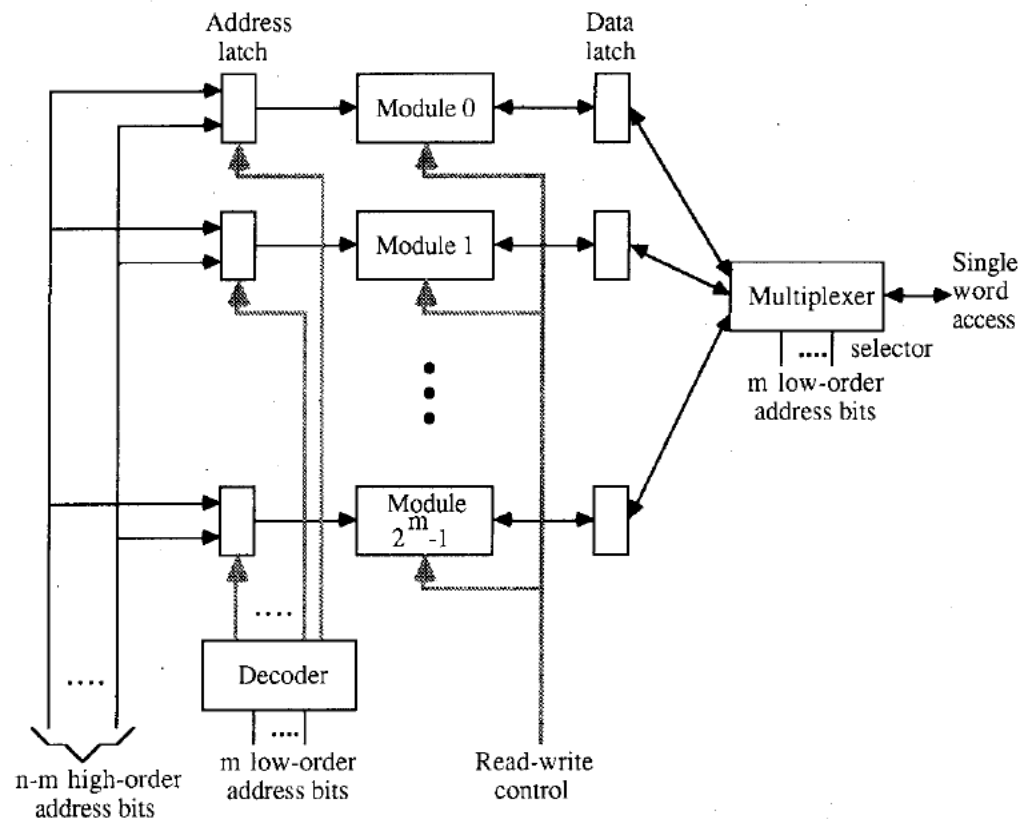
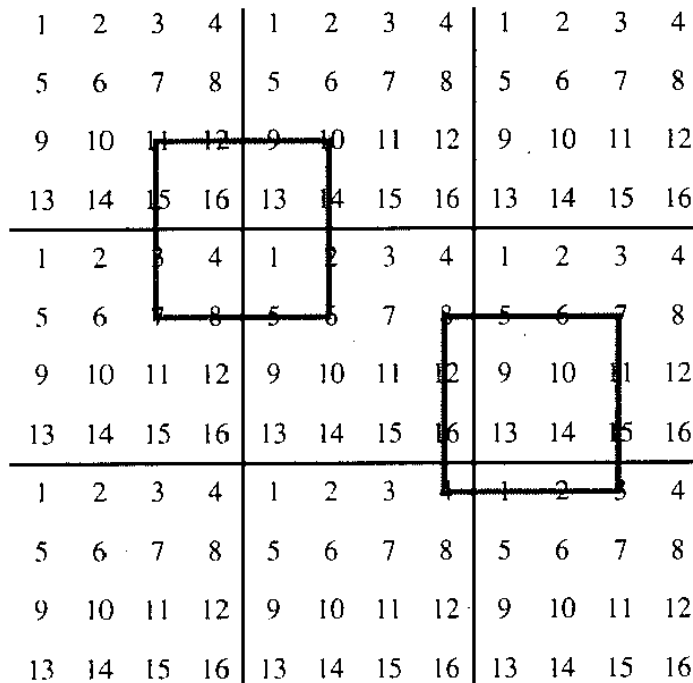
- High throughput
- Parallel computing
- Short data path length, low-latency delay
- Independent data flow computation that benefits the **irregular block matching** especially for the realization of **three-step hierarchy search algorithm**



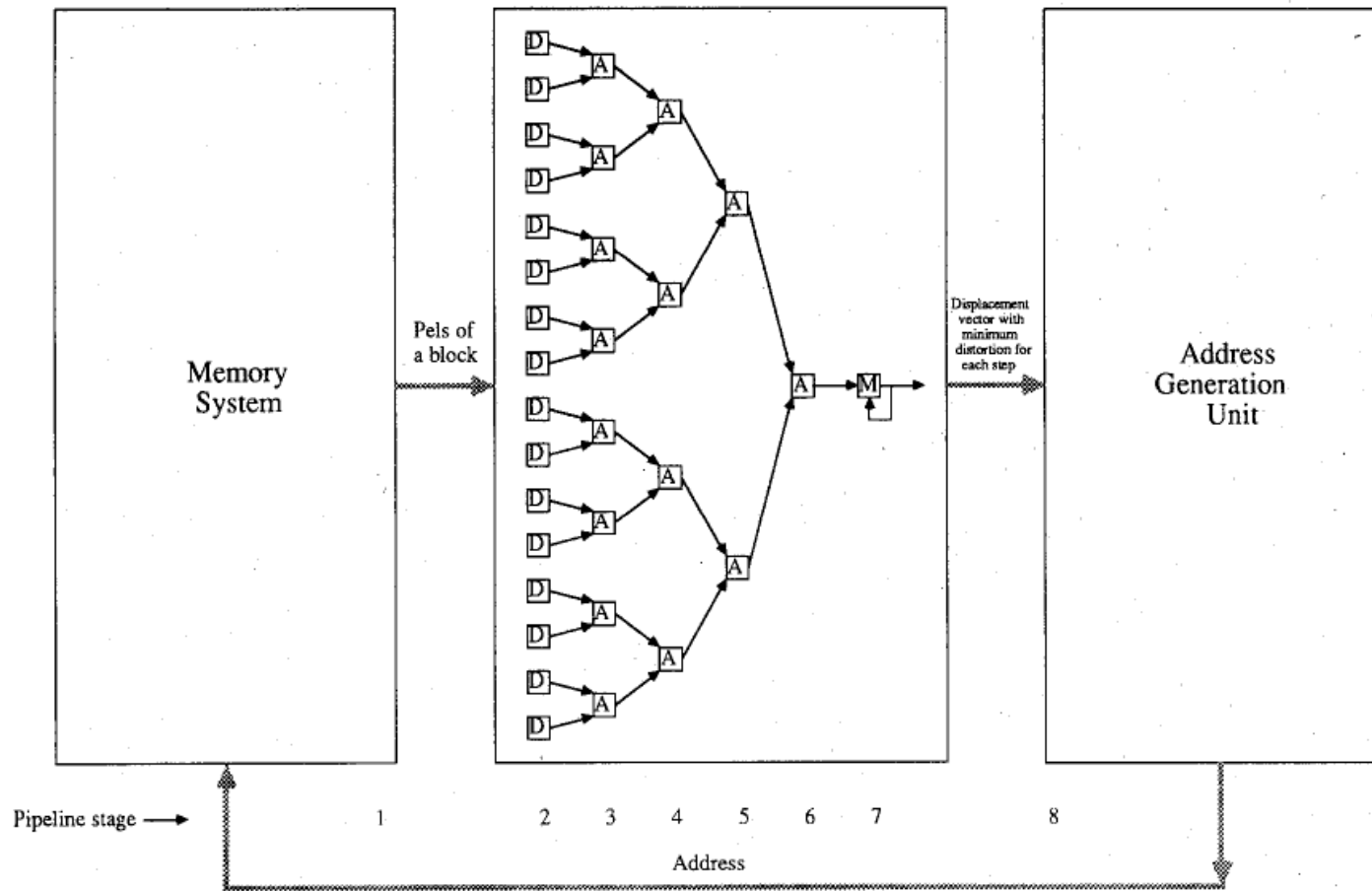
# Tree-Based Architecture



# Memory Interleaving

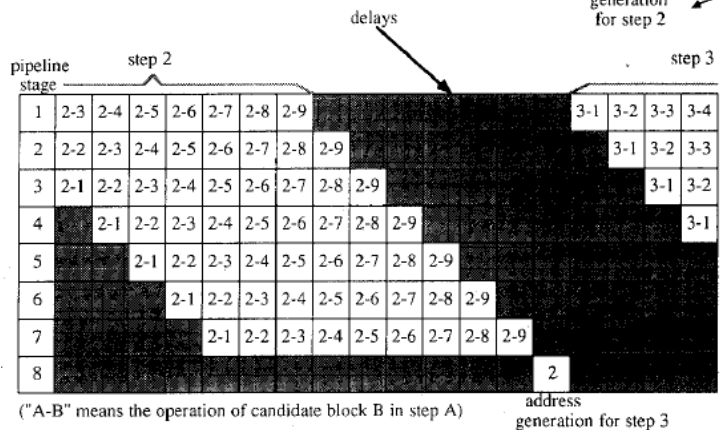
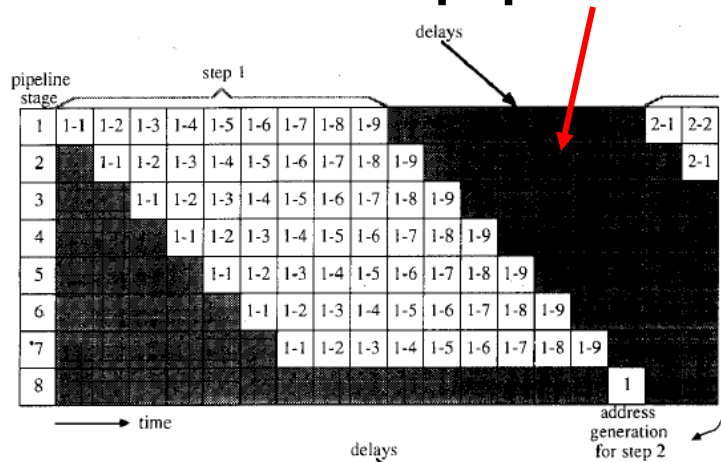


# Whole Architecture

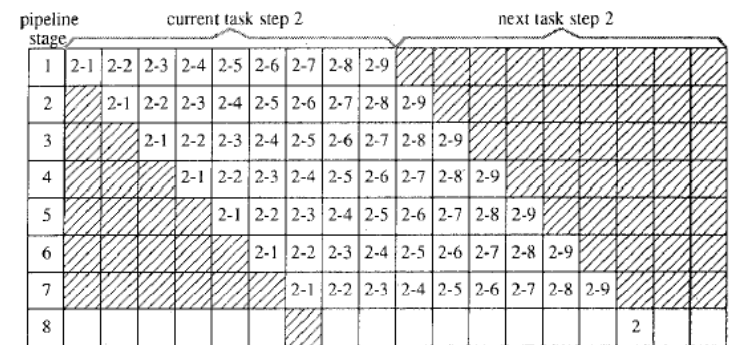
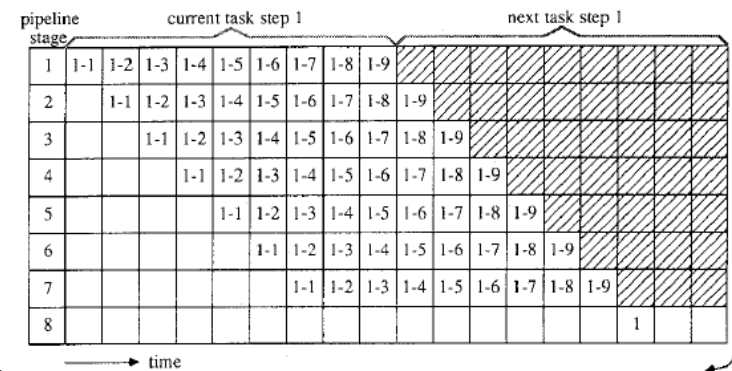


# Pipeline Interleaving

- Remove pipeline bubbles (step hazards)



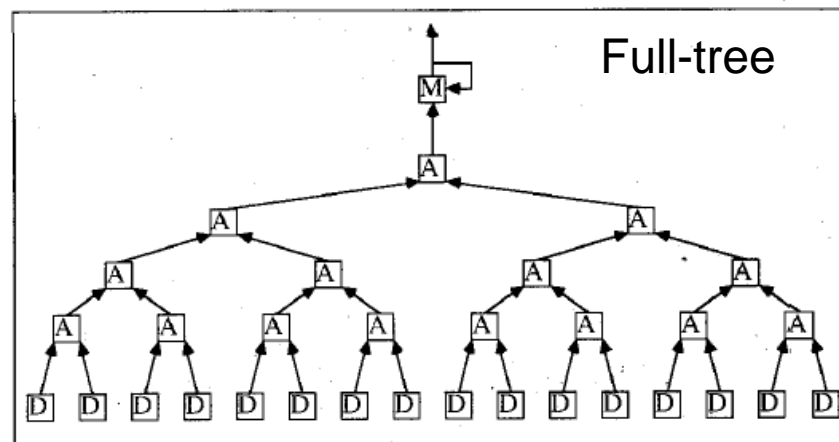
Interleave with adjacent block





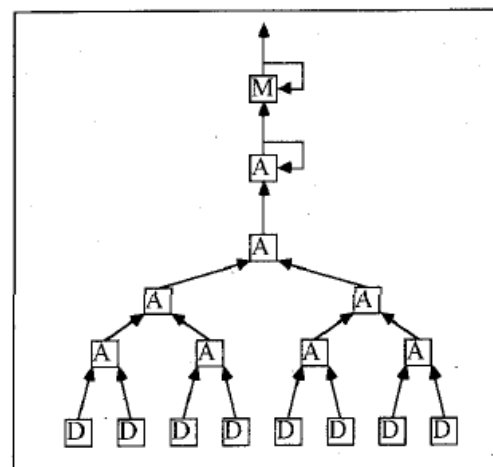
# Tree-Cut Technique

- Applying folding to
  - Processing elements
  - Memory

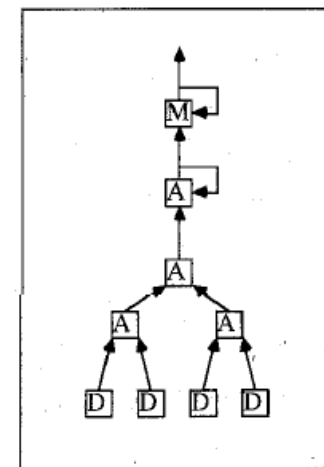


1/2-cut subtree (a)

1/4-cut subtree



(b)



(c)



# Comparison

TABLE I  
TREE-CUT TECHNIQUE FOR HARDWARE REDUCTION

| Configuration     | #ADDER         |        | Time Instances<br>Required (FBMA) | N = 8              | N = 16 |     |
|-------------------|----------------|--------|-----------------------------------|--------------------|--------|-----|
|                   | N = 8          | N = 16 |                                   | d = 3              | d = 7  |     |
| Systolic mesh     | $2N^2 + N + 1$ | 137    | 529                               | $(2d + 1)(N + 2d)$ | 98     | 450 |
| Full tree         | $2N^2$         | 128    | 512                               | $(2d + 1)^2$       | 49     | 225 |
| $\frac{1}{2}$ cut | $2N^2/2 + 1$   | 65     | 257                               | $2(2d + 1)^2$      | 98     | 450 |
| $\frac{1}{4}$ cut | $2N^2/4 + 1$   | 33     | 129                               | $4(2d + 1)^2$      | 196    | 900 |

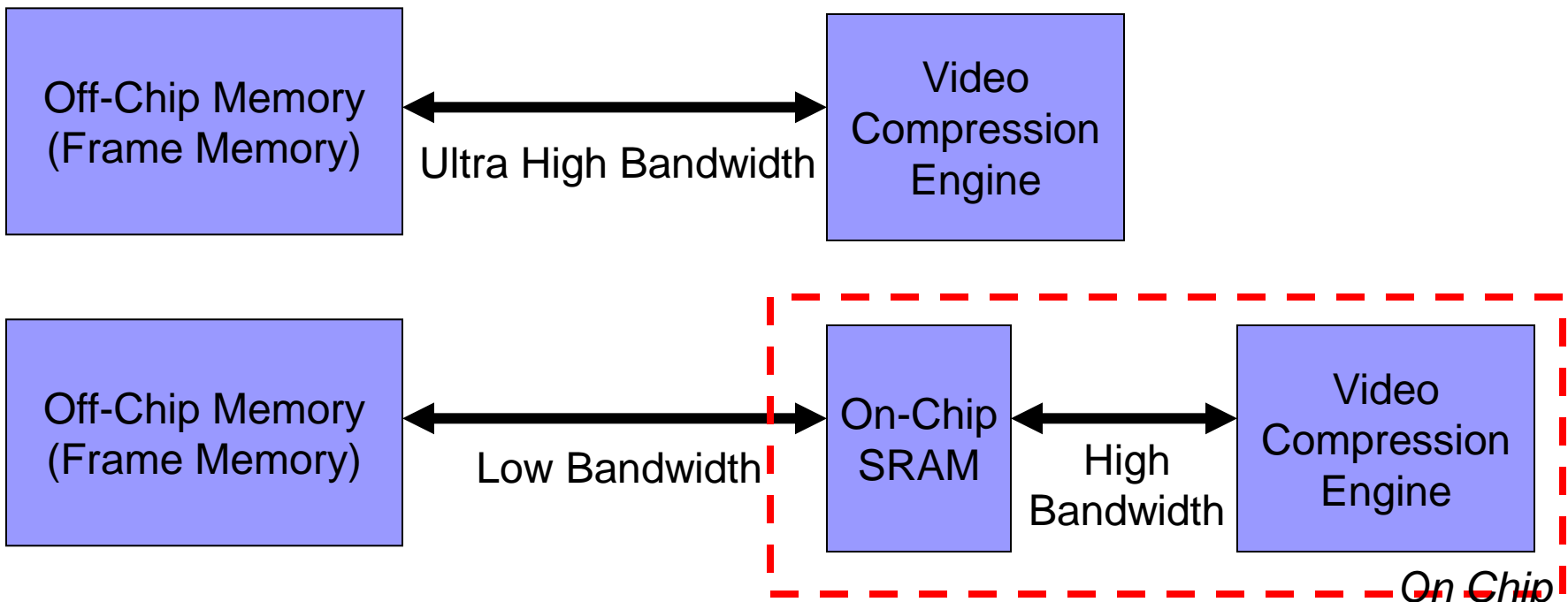


# On-Chip RAM Issues

Ref: Jen-Chieh Tuan, Tian-Sheuan Chang, and Chein-Wei Jen, "On the data reuse and memory bandwidth analysis for full-search block-matching VLSI architecture," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 1, pp. 61-72, Jan. 2002.

# On-Chip SRAM

- The off-chip memory bandwidth can be dramatically reduced with on-chip memory





# On-Chip SRAM

- If we can buffer current block pixels and search area pixels on the on-chip SRAM, we can **significantly decrease the required bandwidth on system bus** (external RAM)
  - **Data reuse** of search area pixels can further reduce the bandwidth of system bus
- Act **like cache** memory in CPU
- This is a **trade-off between area and bandwidth**
- In the following discussions, we assume block size is  $N \times N$ , and search range is  $[-P, +P-1]$

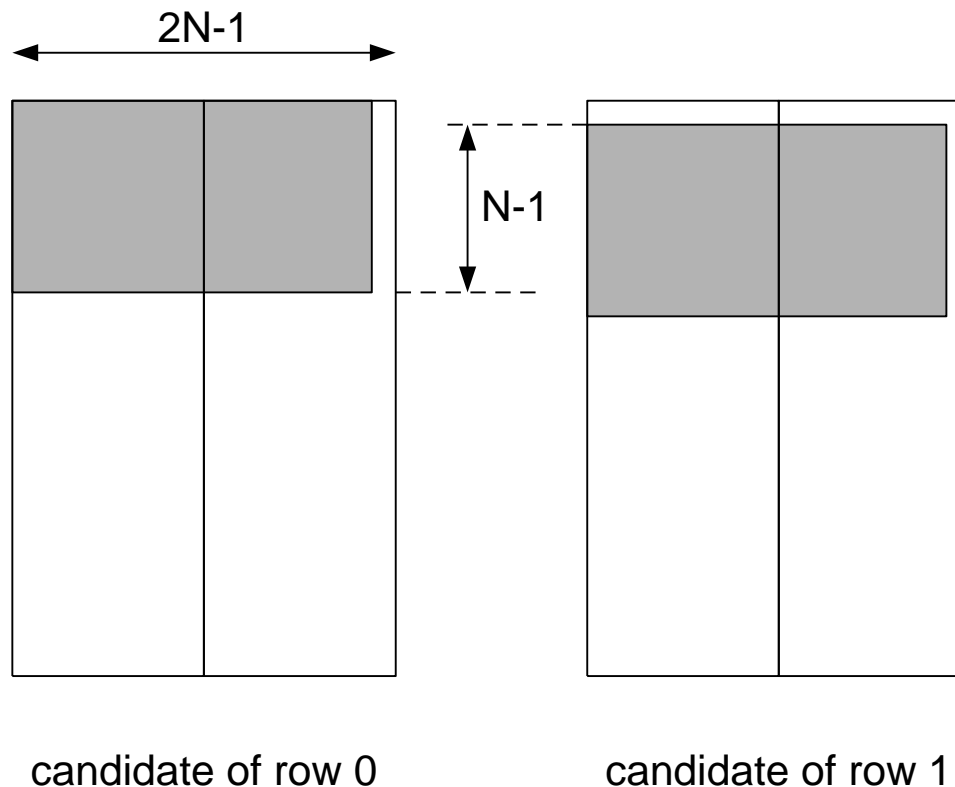


# Different Schemes of Data Reuse for Search Area Pixels

- Data reuse between different rows of candidates in one column of a block (scheme A)
- Data reuse between adjacent columns of candidates in a block (scheme B)
- Data reuse between adjacent blocks in one row of block (scheme C)
- Data reuse between different rows of block (scheme D)
- In today's technology, scheme C is mostly used.

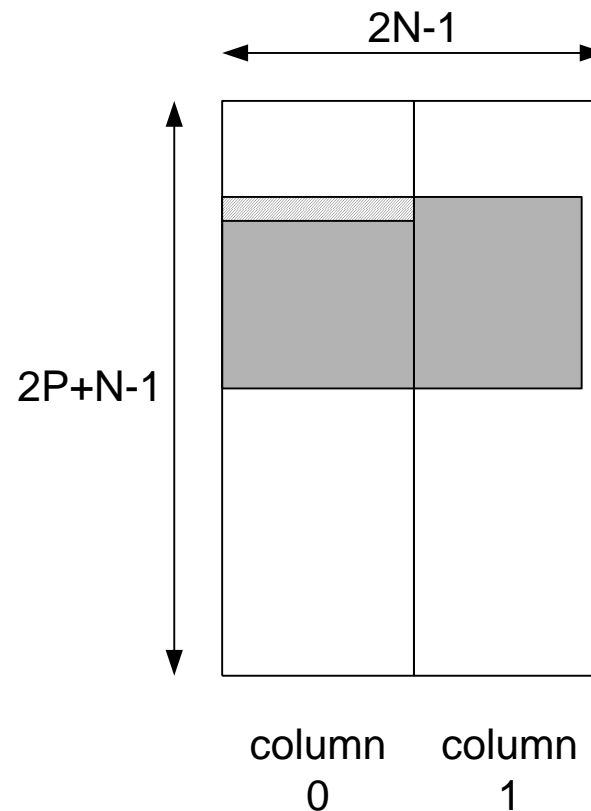
# Illustration of Scheme A

- Data reuse between different rows of candidates in one column of a block



# Illustration of Scheme B

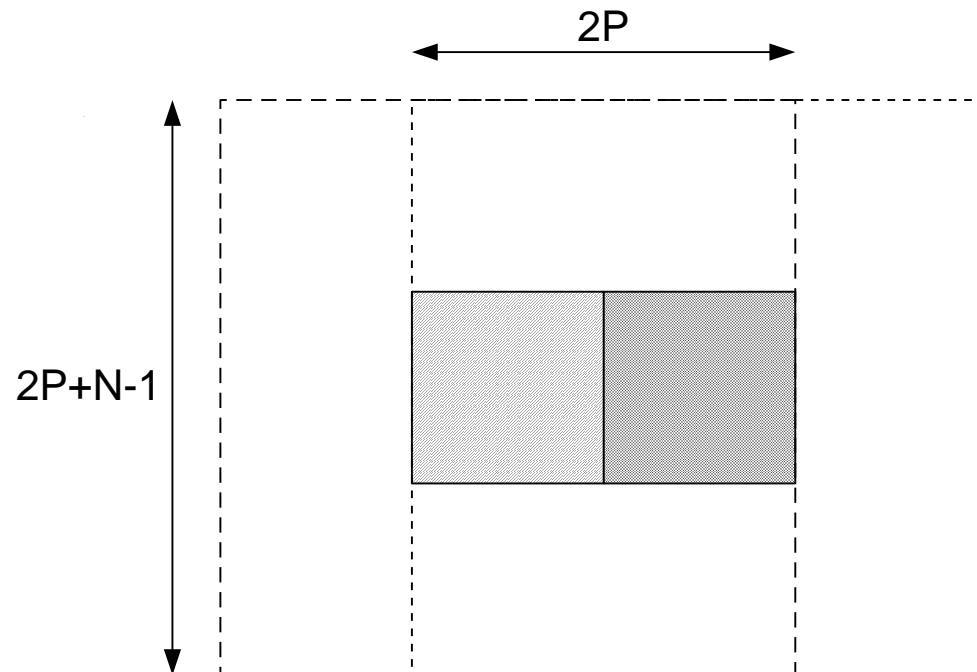
- Data reuse between adjacent columns of candidates in a block





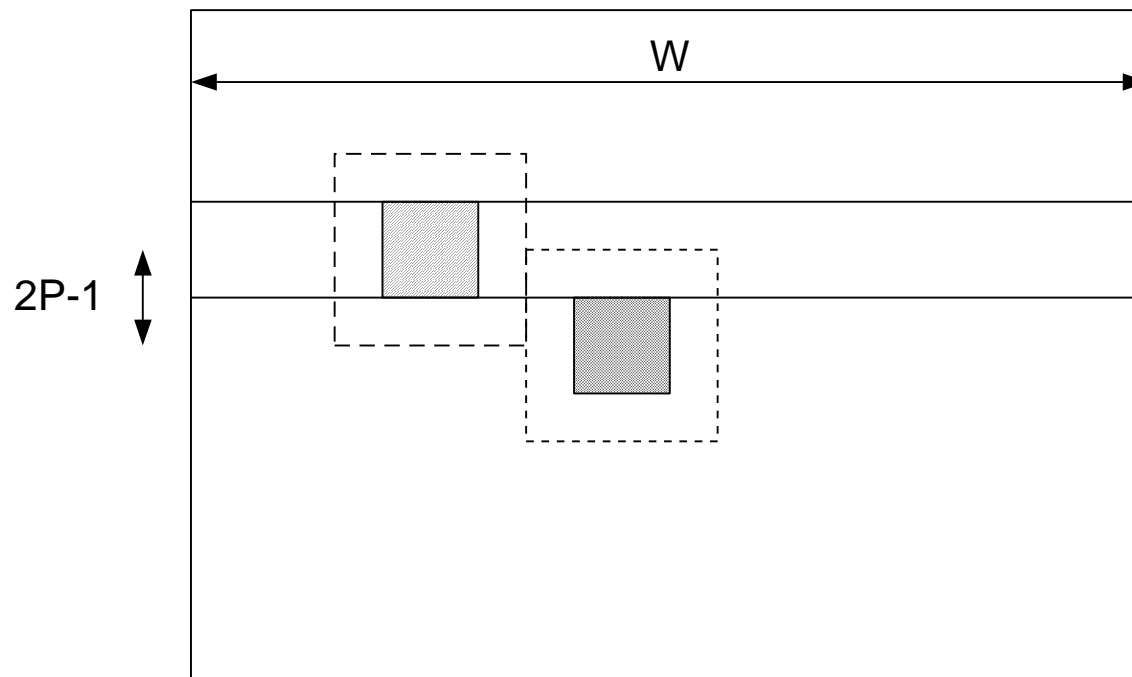
# Illustration of Scheme C

- Data reuse between adjacent blocks in one row of block



# Illustration of Scheme D

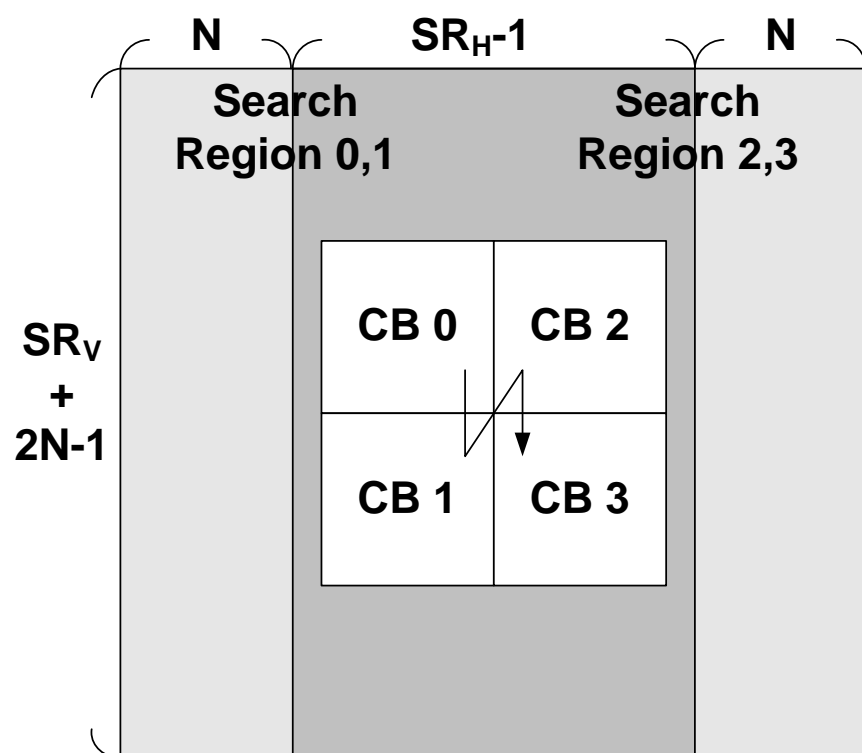
- Data reuse between different rows of block



# Comparison of Different Schemes of Search Area Data Reuse

|                                   | Scheme A                   | Scheme B                             | Scheme C                               | Scheme D                        |
|-----------------------------------|----------------------------|--------------------------------------|--|---------------------------------|
| On-chip buffer size (bytes)       | $(2N-1) \times (N-1)$      | $N \times (2P+N-1) + N \times (N-1)$ | $\text{Max}\{2N, 2P\} \times (2P+N-1)$ | $W \times (2P-1) + 2P \times N$ |
| Off-chip to on-chip (times/pixel) | $(2P/N+1)^2 \times (2P/N)$ | $(2P/N+1) \times (2P/N)$             | $2P/N+1$                               | 1                               |
| On-chip to core (times/pixel)     | $2NP / (2P+N-1)$           | $2NP / (2P+N-1) \times 2$            | $2NP / (2P+N-1) \times (2P/N+1)$       | $2P \times (2P/N+1)$            |

# Level C+ Data Reuse



- Conventional data reuse schemes are based on raster scan
- By use of stripe scan
  - Stitch  $n$  successive vertical MBs ( $n$ -stitched)
  - Load their searching ranges
  - Partially reuse vertical data

Ref: Ching-Yeh Chen, Chao-Tsung Huang, Yi-Hau Chen, and Liang-Gee Chen, "Level C+ data reuse scheme for motion estimation with corresponding coding orders," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, no. 4, pp. 553--558, April 2006.



# Comparison

| Data Reuse Scheme | Bandwidth (Ea)        | SRB                             |
|-------------------|-----------------------|---------------------------------|
| Level C scheme    | $1 + \frac{SR_V}{N}$  | $(SR_H + N - 1)(SR_V + N - 1)$  |
| Level C+ scheme   | $1 + \frac{SR_V}{nN}$ | $(SR_H + N - 1)(SR_V + nN - 1)$ |
| Level D scheme    | 1                     | $(SR_H + W - 1)(SR_V - 1)$      |

- System memory bandwidth (equivalent access factor)

$$Ea_{ME} = \frac{\text{Total memory bandwidth for reference frame}}{\text{processed current pixels}}$$

- On-chip memory size (SRB)



# Fast-Algorithm- Based Architecture

Y.-W. Huang, S.-Y. Chien, B.-Y. Hsieh, and L.-G. Chen, "An efficient and low power architecture design for motion estimation using global elimination algorithm," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2002

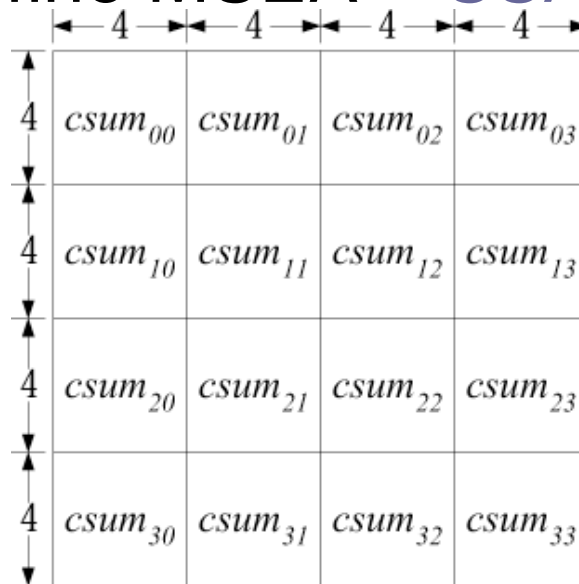


# Multi-Level SEA (MSEA)

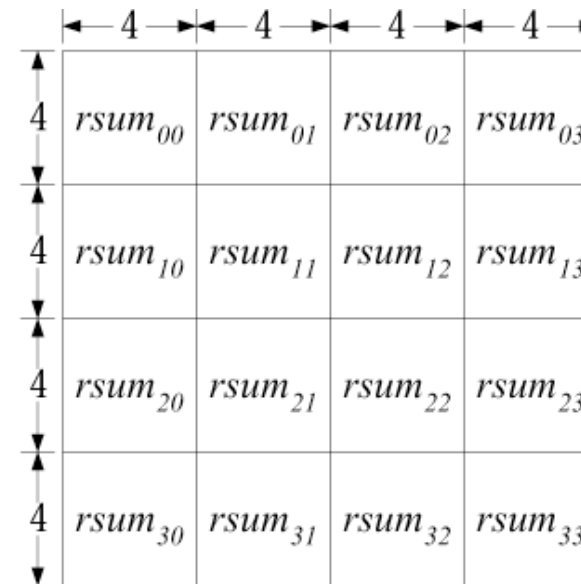
- Convert the sea value to **msea** value.
- Split the 16x16 macro-block into sub-blocks.
- **SAD**  $\geq$  **msea**  $\geq$  **sea**, which means MSEA can skip more unnecessary SAD calculations than SEA under the same scan order.
- However, the computation of msea value is heavier than that of sea value.

# Example of MSEA at Level = 3

- MSEA is the sub-sampled version of SAD !!!
- Define MSEA = SSAD



16 x 16 current block



16 x 16 candidate block

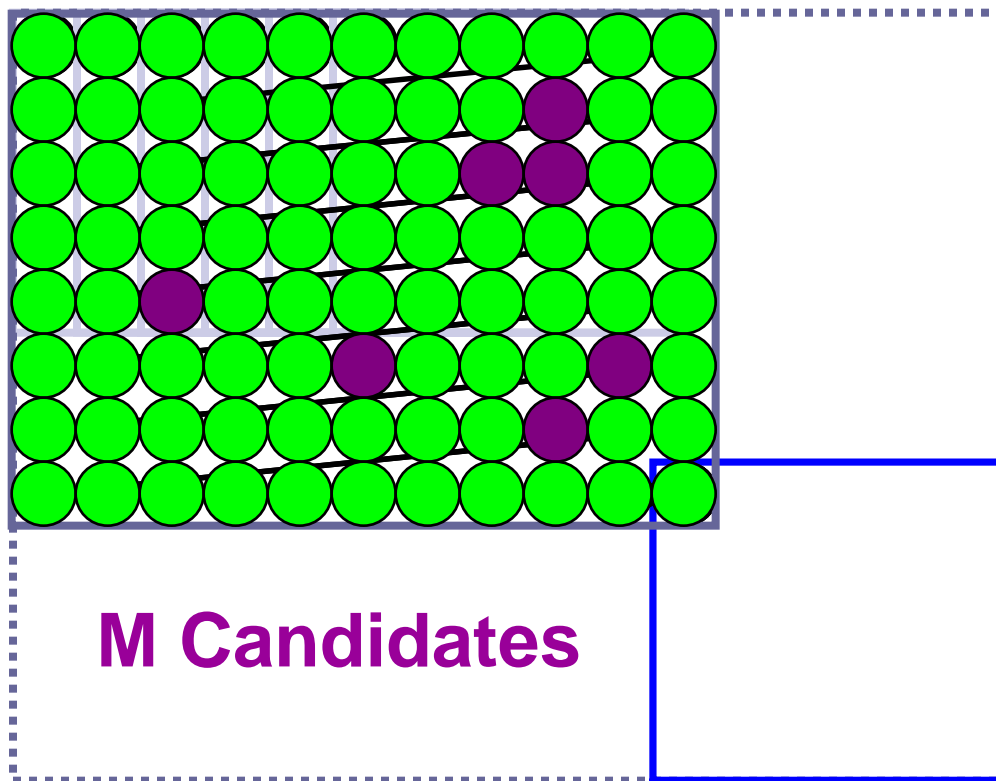
$csum_{ij}$  : sum of sub-block  $ij$  in current block

$rsum_{ij}$  : sum of sub-block  $ij$  in candidate block of search position  $(m,n)$

$$msea(m,n) = |csum_{00} - rsum_{00}| + |csum_{01} - rsum_{01}| + \dots + |csum_{33} - rsum_{33}|$$



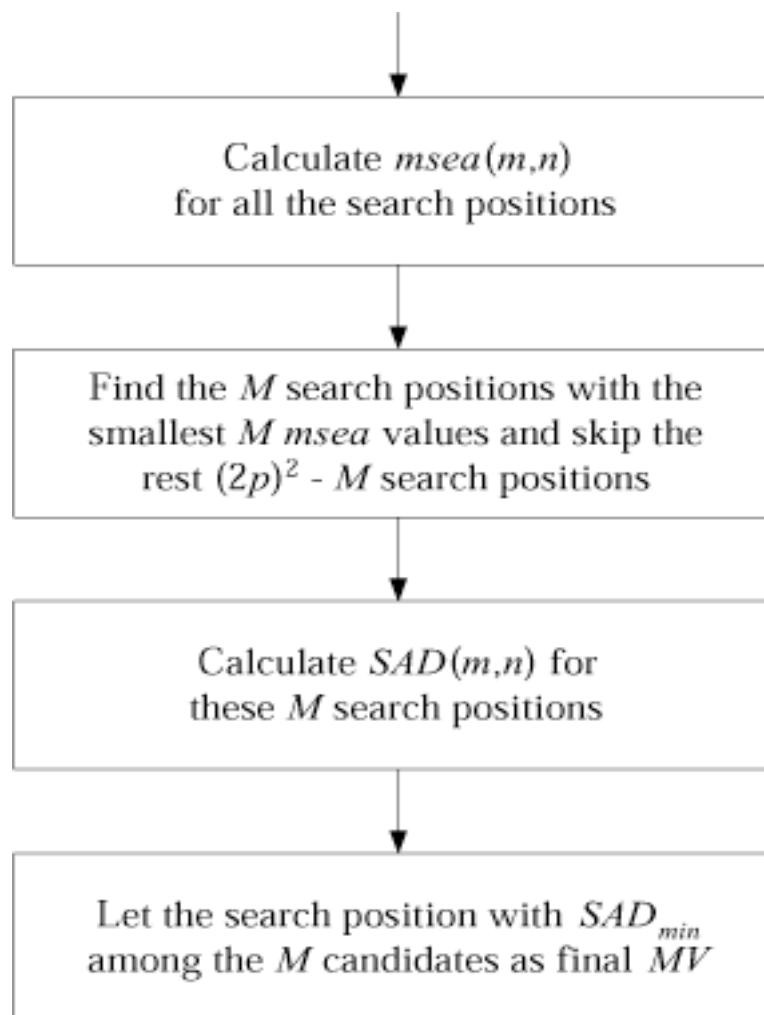
# Global Elimination Algorithm



1. Compute **SSAD** by raster scan
2. Keep the **M** search positions with the smallest **SSAD**
3. Compute **SAD** of the **M** positions and skip the rest



# Flowchart of GEA





# Reduced Operations

- Full-search block matching algorithm
  - Compute SAD at each search position
    - About 256 subtractions, 256 absolute value calculations, and 256 additions per search position
- Global elimination algorithm
  - Compute SSAD at each search position
    - SSAD is much easier to compute than SAD
    - About 16 subtractions, 16 absolute value calculations, and 16 additions per search position

# Motion Compensated Subjective

## View



Previous Frame



Current Frame

FSBMA

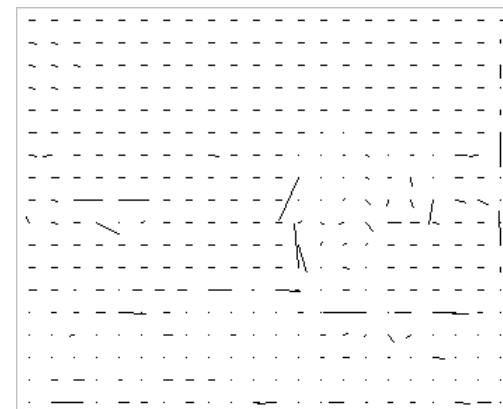
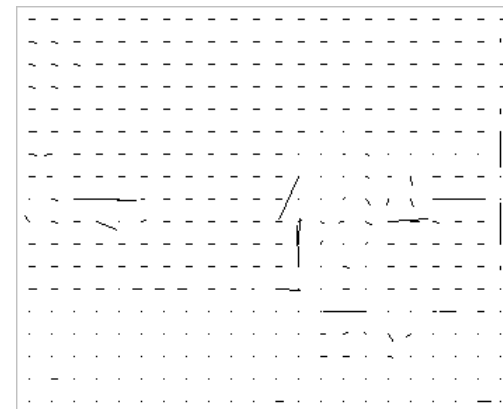
Compensated Frame



GEA



MV Plot

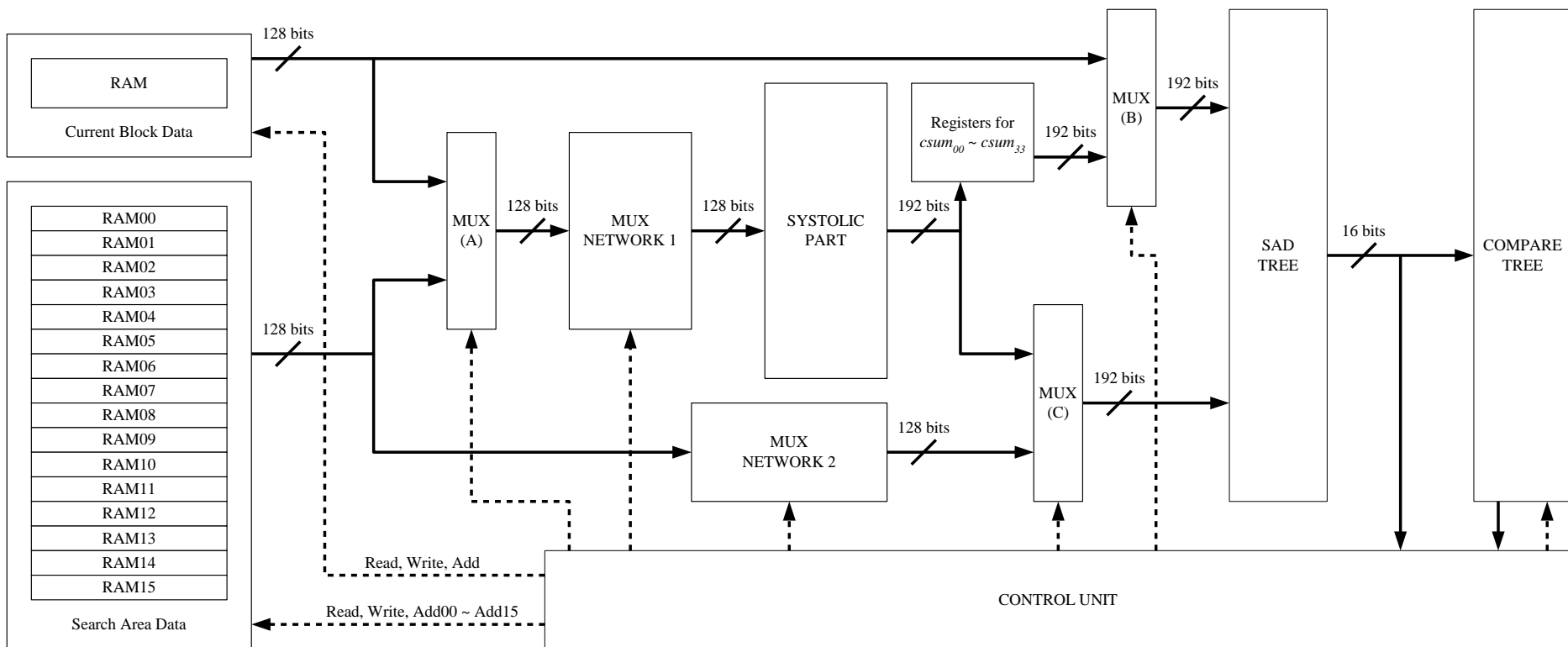


# Motion Compensated PSNR (dB)

| Level=3, M=7    | QCIF [-16,+15] |       | CIF [-32,+31] |       |
|-----------------|----------------|-------|---------------|-------|
| Video Sequence  | GEA            | FSBMA | GEA           | FSBMA |
| Coastguard      | 32.93          | 32.93 | 31.55         | 31.59 |
| Container       | 43.11          | 43.11 | 38.53         | 38.53 |
| Foreman         | 32.22          | 32.21 | 32.82         | 32.85 |
| Hall Monitor    | 32.97          | 32.98 | 34.82         | 34.90 |
| Mobile Calendar | 26.15          | 26.15 | 25.16         | 25.20 |
| Silent          | 35.16          | 35.14 | 36.11         | 36.12 |
| Stefan          | 24.67          | 24.71 | 25.71         | 25.73 |
| Table Tennis    | 32.11          | 32.10 | 32.96         | 33.03 |
| Weather         | 38.42          | 38.42 | 37.45         | 37.45 |



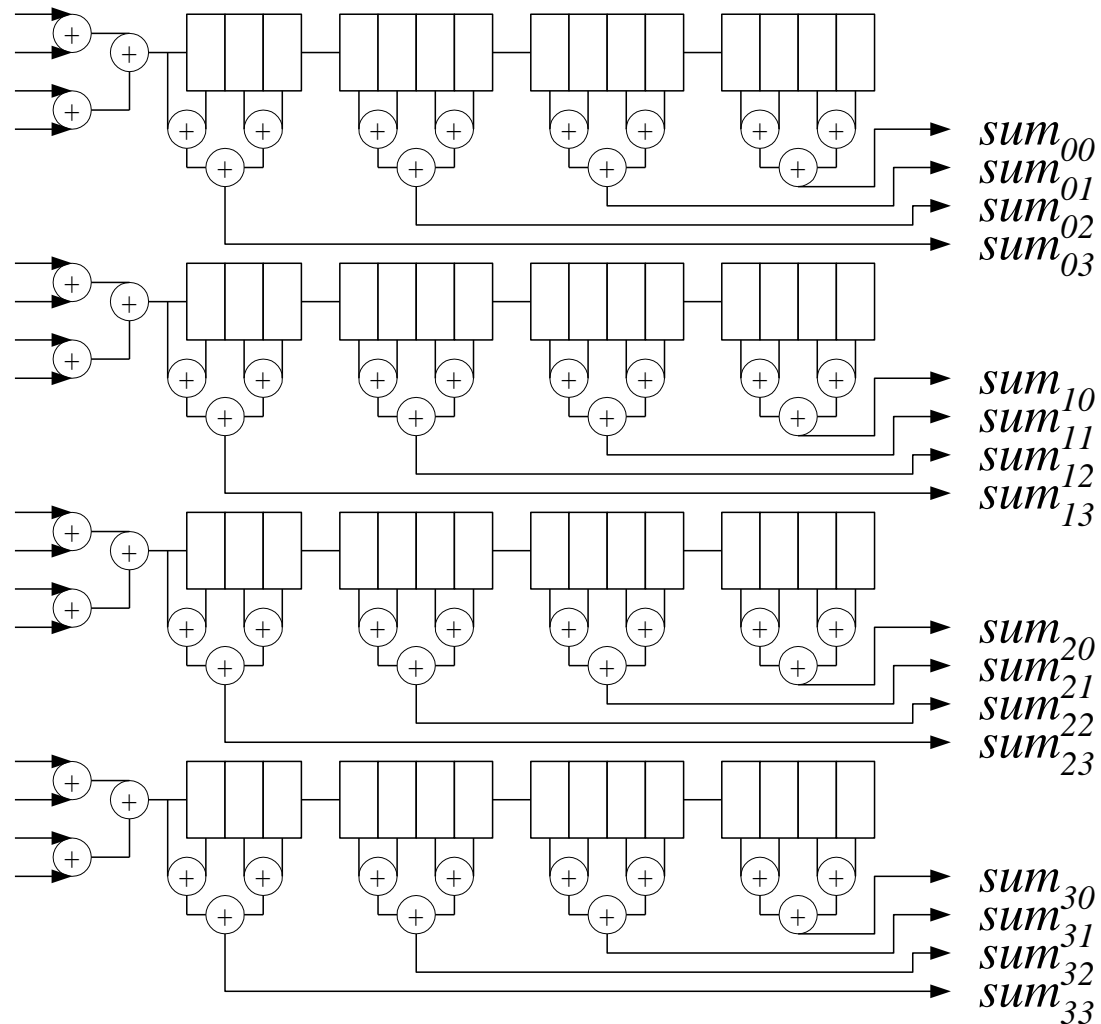
# VLSI Architecture Design for GEA



# Systolic Part

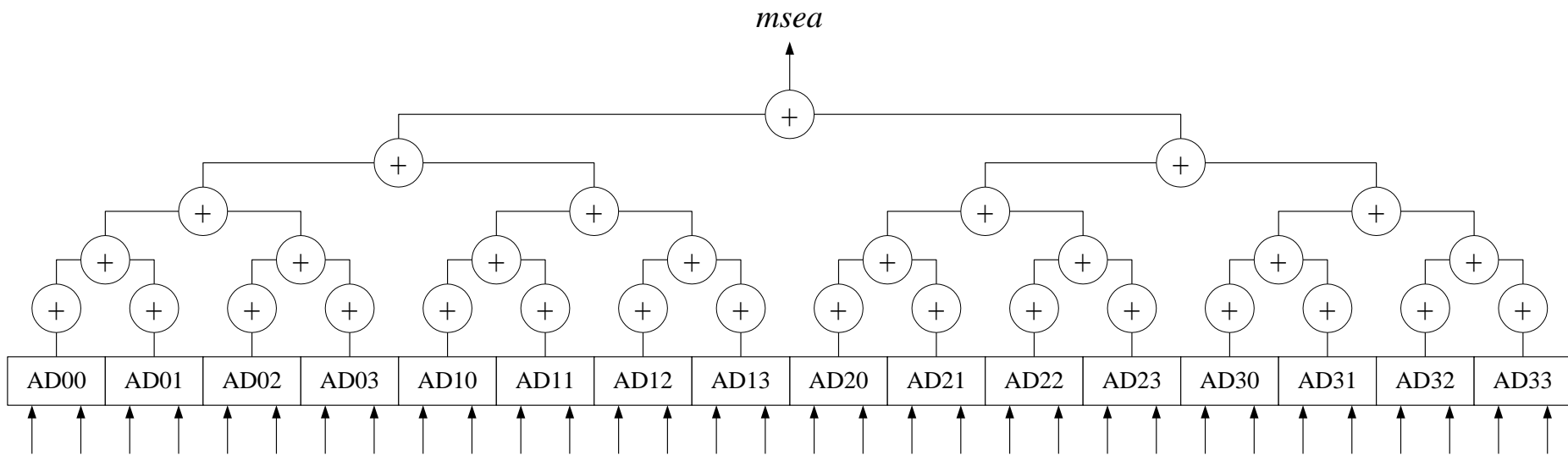
Current block data and search range data are inputted column by column from left to right, top to down.

Each PE computes the sum of a 4x4 sub-block, and the sixteen sums are processed in parallel.





# Parallel Adder Tree

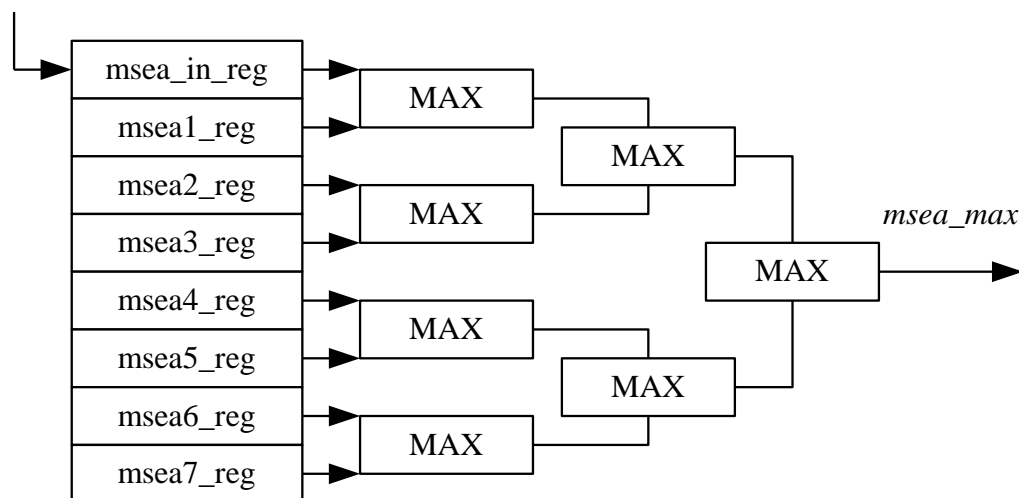


1. “AD” unit stands for “Absolute Difference.”
2. The sixteen sub-block sums of current block (stored in register right after they were previously calculated by the systolic part) and the sixteen sub-block sums of a candidate block (directly inputted from the systolic part) are inputted in parallel.
3. The msea value (at level 3) of a candidate block can be computed in one cycle.
4. This adder tree can be reused to compute SAD.



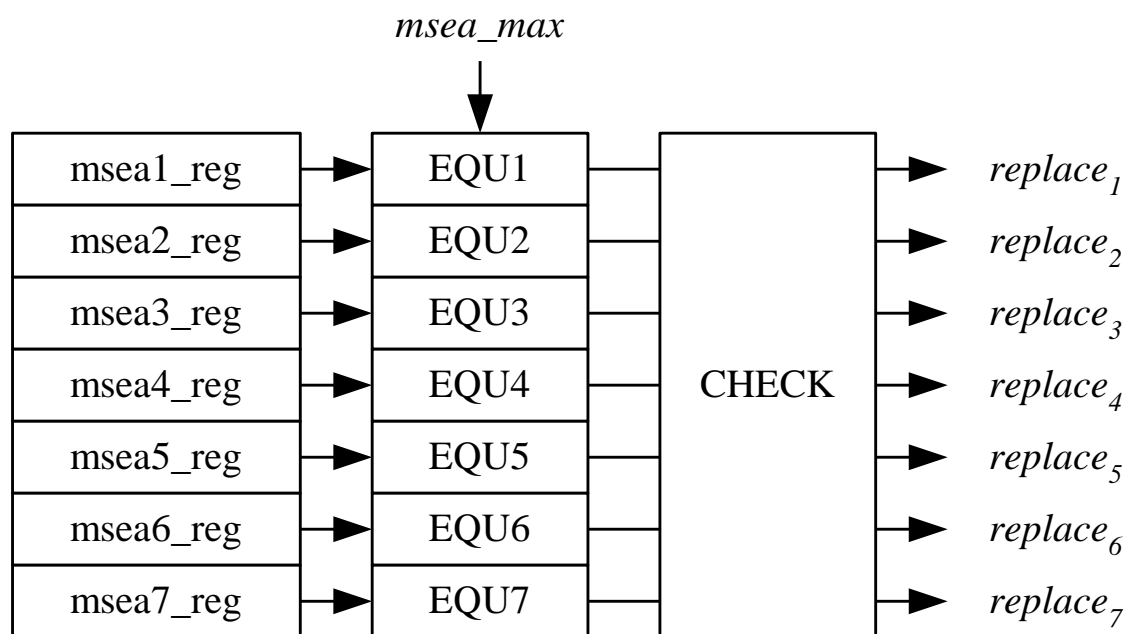
# Parallel Comparator Tree (1/3)

*msea* from the  
parallel adder tree



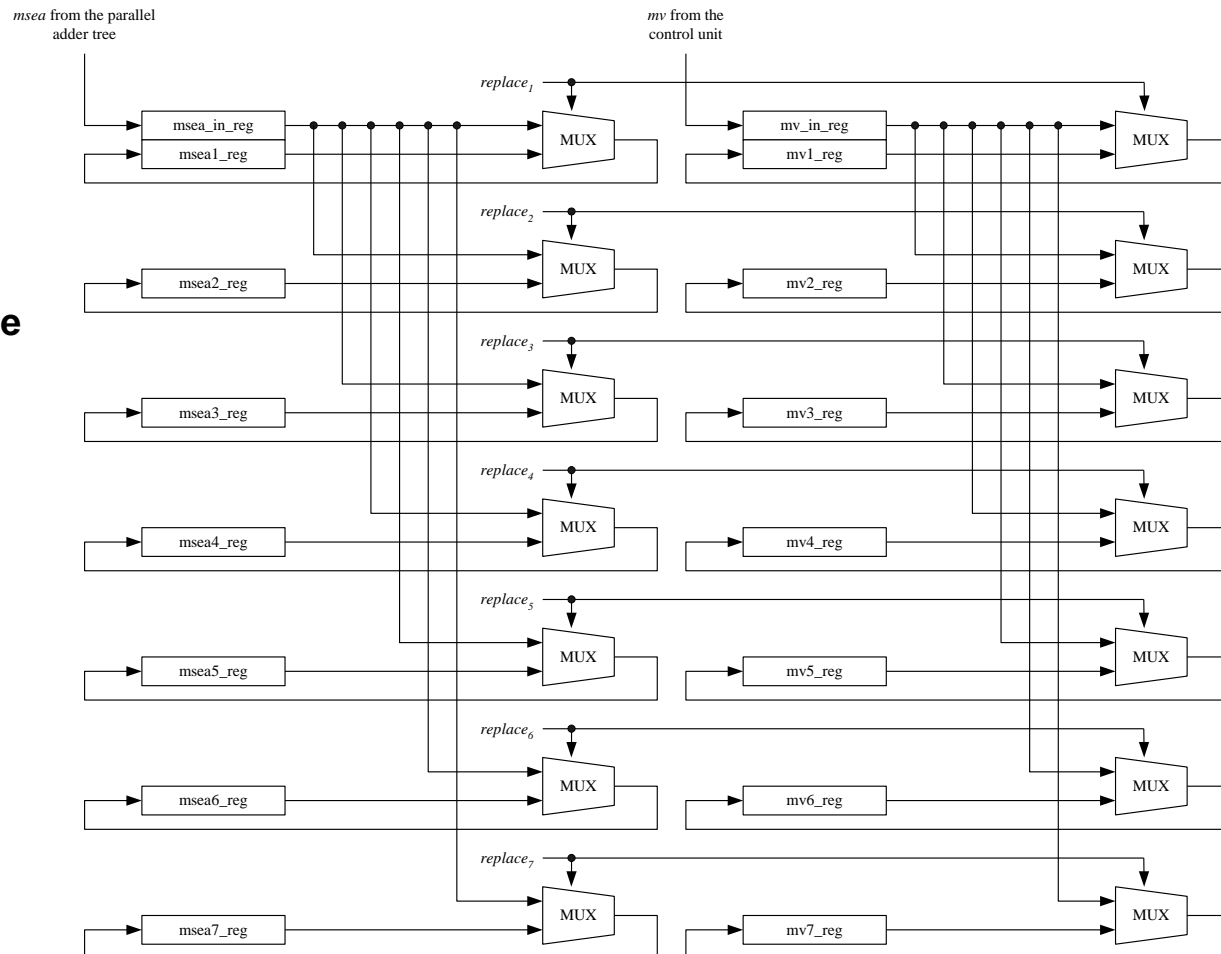
1. The goal of the parallel comparator tree is to find the smallest 7 *msea* values among all search positions in order to calculate the SADs of these 7 candidates in the later stage.
2. The “*mseax\_reg*” units ( $x=1\sim 7$ ) contain the up-to-date smallest 7 *msea* values and their corresponding search positions.
3. The new *msea* value of the next search position is inputted from the parallel adder tree.
4. The first part of the parallel comparator tree is to find the maximum among the stored values and the new coming value.

# Parallel Comparator Tree (2/3)



1. The second part of the parallel comparator tree is to find whether a stored msea value is equal to the maximum value.
2. If more than one msea registers are equal to the *msea\_max*, only one of them will be selected. This is done by the “CHECK” unit.

# Parallel Comparator Tree (3/3)



1. The third part of the parallel comparator tree is to replace the maximum msea value in the msea register by the new coming value if necessary.
2. The msea registers always contain the smallest 7 msea values and their corresponding search positions.



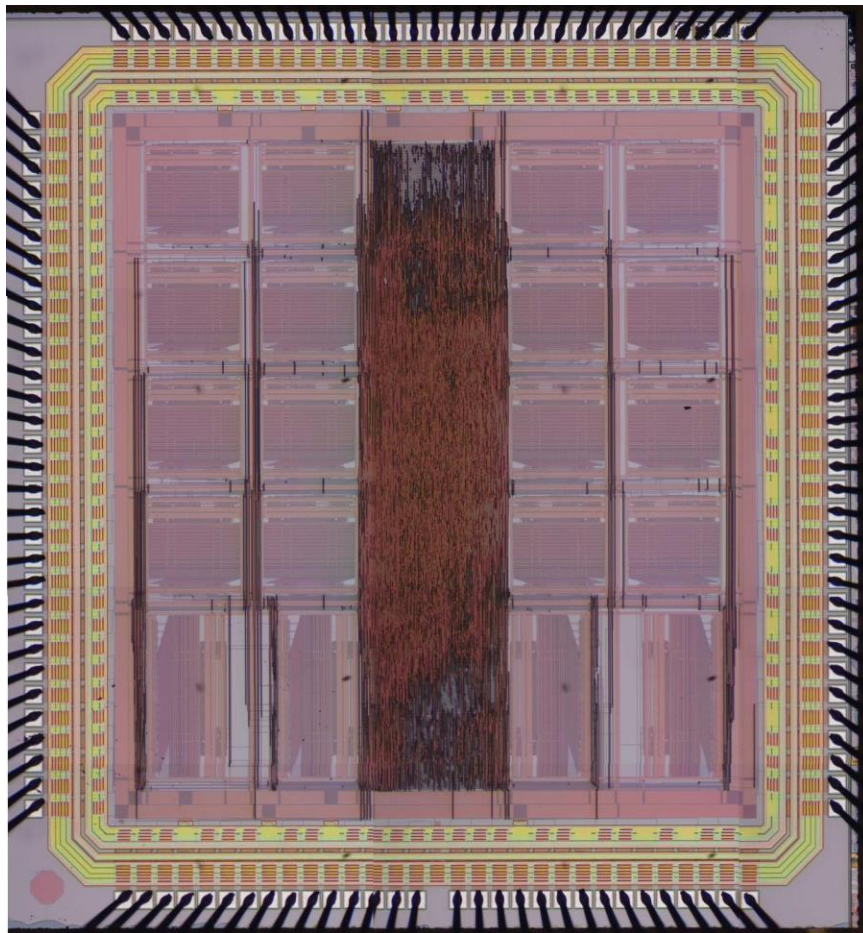
# Comparison with FSBMA [-16,+15]

| Architecture     | Description       | Required Freq.    | Gate Count     |
|------------------|-------------------|-------------------|----------------|
| <b>Yang [1]</b>  | 1-D semi-systolic | <b>97.32 MHz</b>  | <b>44.7 K</b>  |
| <b>AB1 [3]</b>   | 1-D systolic      | <b>285.88 MHz</b> | <b>14.4 K</b>  |
| <b>AB2 [5]</b>   | 2-D systolic      | <b>17.87 MHz</b>  | <b>98.2 K</b>  |
| <b>Hsieh [9]</b> | 2-D systolic      | <b>26.24 MHz</b>  | <b>100.1 K</b> |
| <b>Tree [10]</b> | Tree structure    | <b>12.17 MHz</b>  | <b>58.7 K</b>  |
| <b>Yeo [12]</b>  | 2-D semi-systolic | <b>3.04 MHz</b>   | <b>436.6 K</b> |
| <b>Lai [14]</b>  | 1-D semi-systolic | <b>3.04 MHz</b>   | <b>384.8 K</b> |
| <b>SA [15]</b>   | 2-D systolic      | <b>12.17 MHz</b>  | <b>127.0 K</b> |
| <b>SSA [16]</b>  | 2-D semi-systolic | <b>12.17 MHz</b>  | <b>110.6 K</b> |
| <b>Ours [18]</b> | Based on GEA      | <b>19.42 MHz</b>  | <b>23.1 K</b>  |

# Comparison with FSBMA [-32,+31]

| Architecture     | Description       | Required Freq.    | Gate Count      |
|------------------|-------------------|-------------------|-----------------|
| <b>Yang [1]</b>  | 1-D semi-systolic | <b>194.64 MHz</b> | <b>107.6 K</b>  |
| <b>AB1 [3]</b>   | 1-D systolic      | <b>961.04 MHz</b> | <b>16.5 K</b>   |
| <b>AB2 [5]</b>   | 2-D systolic      | <b>60.07 MHz</b>  | <b>107.9 K</b>  |
| <b>Hsieh [9]</b> | 2-D systolic      | <b>74.14 MHz</b>  | <b>100.0 K</b>  |
| <b>Tree [10]</b> | Tree structure    | <b>48.66 MHz</b>  | <b>58.4 K</b>   |
| <b>Yeo [12]</b>  | 2-D semi-systolic | <b>3.04 MHz</b>   | <b>1746.4 K</b> |
| <b>Lai [14]</b>  | 1-D semi-systolic | <b>3.04 MHz</b>   | <b>1539.3 K</b> |
| <b>SA [15]</b>   | 2-D systolic      | <b>48.66 MHz</b>  | <b>127.0 K</b>  |
| <b>SSA [16]</b>  | 2-D semi-systolic | <b>48.66 MHz</b>  | <b>110.6 K</b>  |
| <b>Ours [18]</b> | Based on GEA      | <b>61.62 MHz</b>  | <b>33.2 K</b>   |

# Chip Photo & Spec.



|                   |  |
|-------------------|--|
| Process           | TSMC 1P4M 0.35 um                                  |
| Package           | 128 CQFP   |
| Die Size          | 3.679 x 4.001 mm <sup>2</sup>                      |
| Core Size         | 2.591 x 2.879 mm <sup>2</sup>                      |
| Max. Frequency    | 27.8 MHz   |
| Logic Gate Count  | 25,997   |
| On-Chip SRAM      | 20,480 bits  |
| Transistor Count  | 357,551  |
| Search Range      | [-16,+15]  |
| Processing Speed  | 152 QCIF (176 x 144) fps<br>38 CIF (352 x 288) fps |
| Power Consumption | 272 mW @ 25 MHz                                    |



# Motion Estimation in H.264



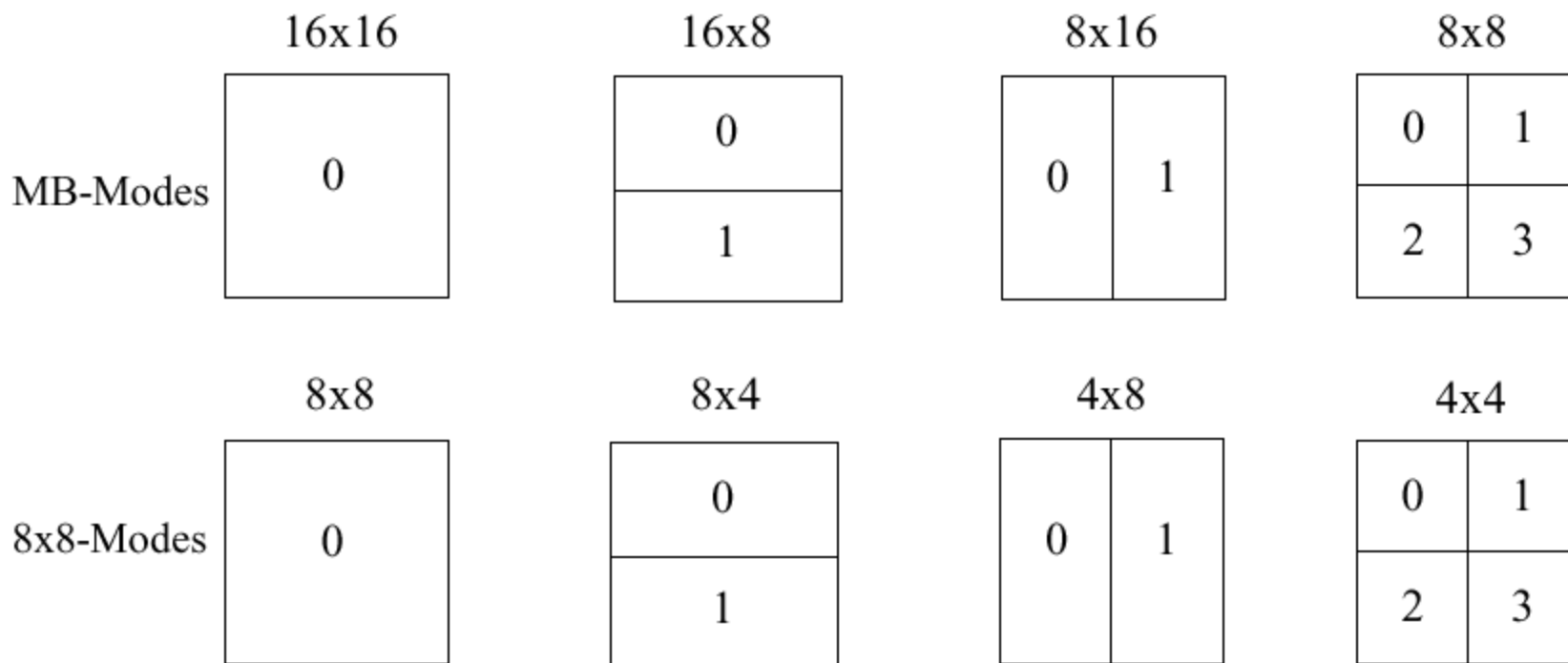
# Motion Compensation

- Seven kinds of block sizes (16x16, 16x8, 8x16, 8x8, 8x4, 4x8, and 4x4)
- 1/4 sample accuracy
  - 6-tap filtering for 1/2-pixel
  - Simplified filtering for 1/4-pixel
- Multiple reference pictures

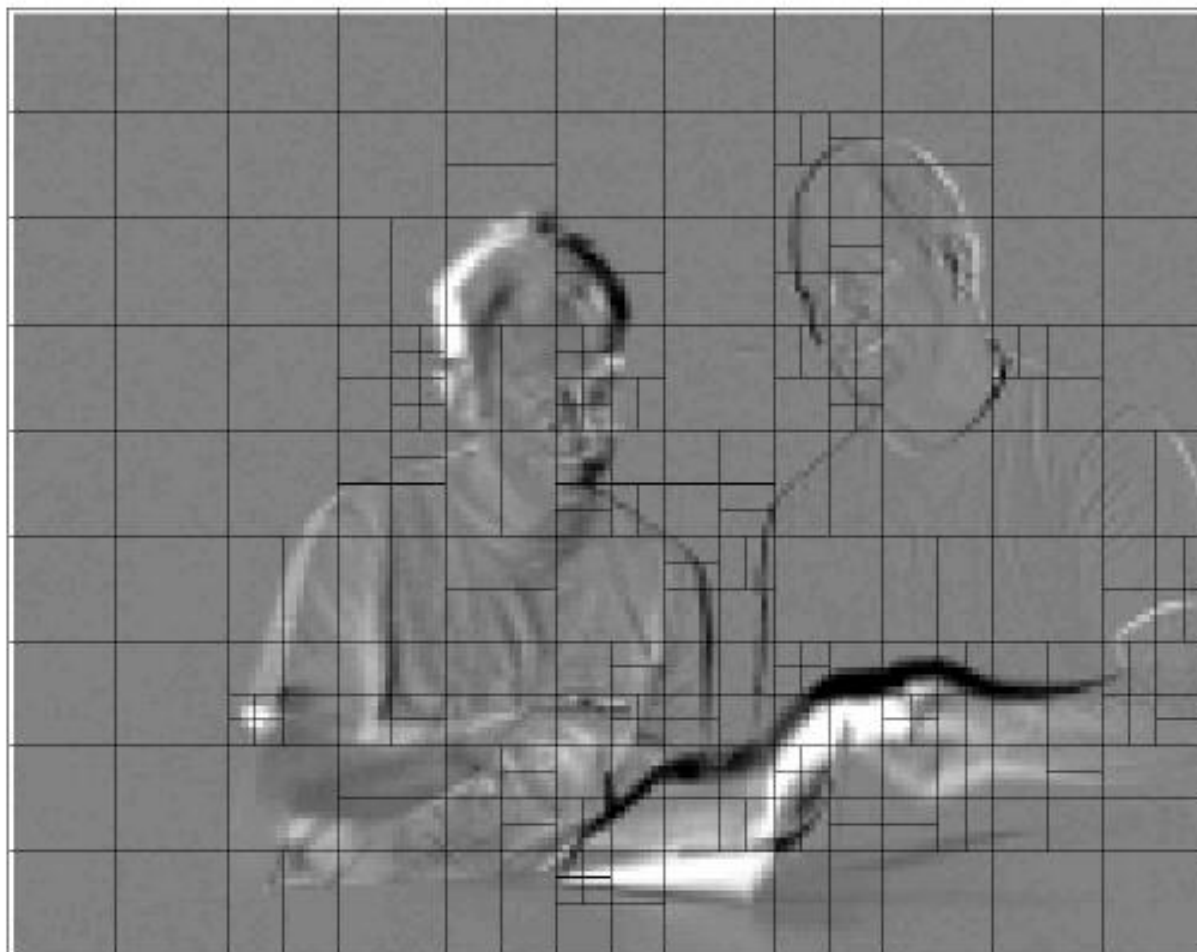




# Variable Block Sizes

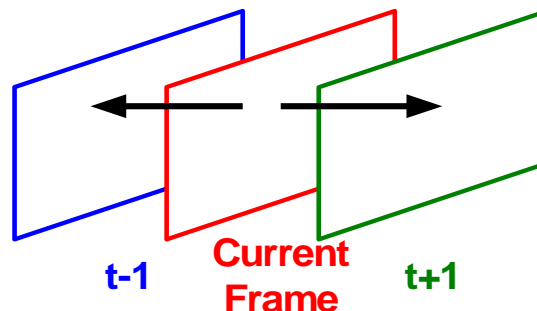


# Example of Variable Block Sizes

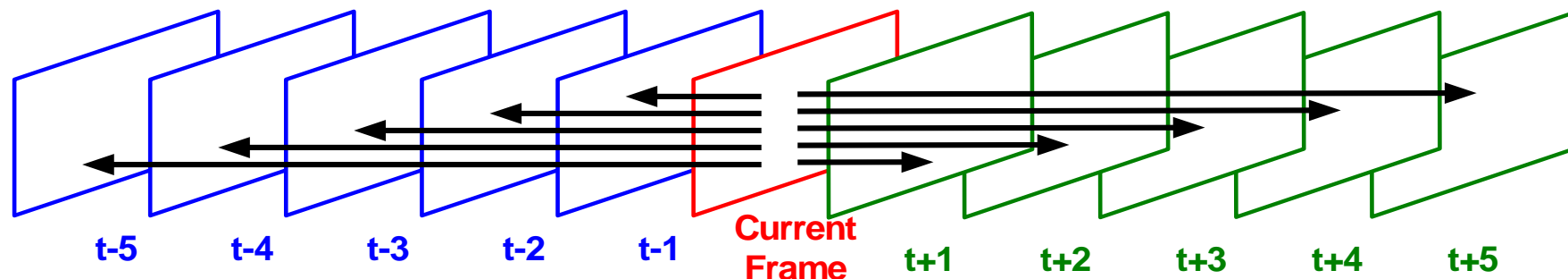


# Multiple Reference Frames

MPEG-1, MPEG-2, MPEG-4



H.264/JVT/AVC



Note: the whole 8x8 sub-partition must be predicted by the same reference frame.



# Rate-Distortion Optimized Mode Decision

- Lagrangian method to minimize  $J = D + \lambda R$
- $J$  is the cost function
- $D$  means distortion (SAD, SATD, SSD)
- $R$  stands for a function of bit-rate.
- $\lambda$  is called Lagrangian multiplier.
  - Theoretically, assume  $D$  is a function of  $R$ , denoted as  $D(R)$ , then  $\lambda$  should be obtained by differentiating  $J$  with respect to  $R$ .
  - Setting the first derivative to zero and solve  $\lambda$ .
  - In the reference software,  $\lambda$  is a function of QP.



# Macroblock Mode Decision

## MinCost0

## MinCost1

|                  |                  |
|------------------|------------------|
| 1. Inter4x4      | 1. Inter4x4      |
| 2. Inter4x8      | 2. Inter4x8      |
| 3. Inter8x4      | 3. Inter8x4      |
| 4. Inter8x8      | 4. Inter8x8      |
| (5 ref, 1/4-pel) | (5 ref, 1/4-pel) |
| 1. Inter4x4      | 1. Inter4x4      |
| 2. Inter4x8      | 2. Inter4x8      |
| 3. Inter8x4      | 3. Inter8x4      |
| 4. Inter8x8      | 4. Inter8x8      |
| (5 ref, 1/4-pel) | (5 ref, 1/4-pel) |

Inter16x16  
(5 ref, 1/4-pel)  
**MinCost**  
**Cost16x16 = MinCost**

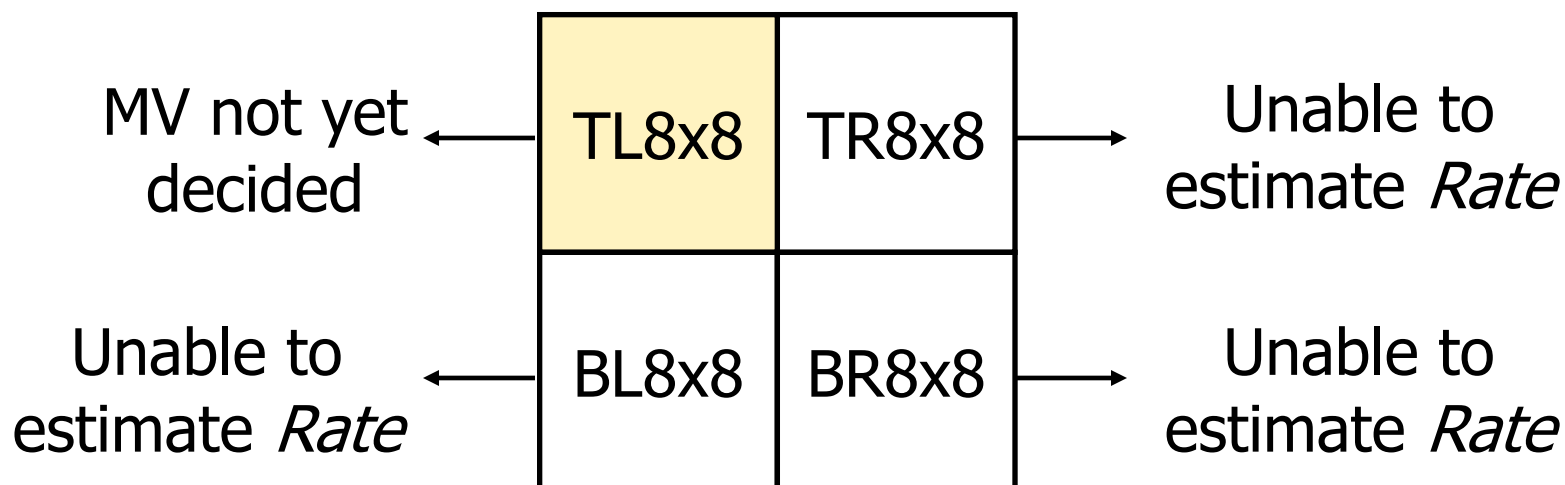
## MinCost2

## MinCost3

$$\text{Cost8x8} = \text{MinCost0} + \text{MinCost1} + \text{MinCost2} + \text{MinCost3}$$

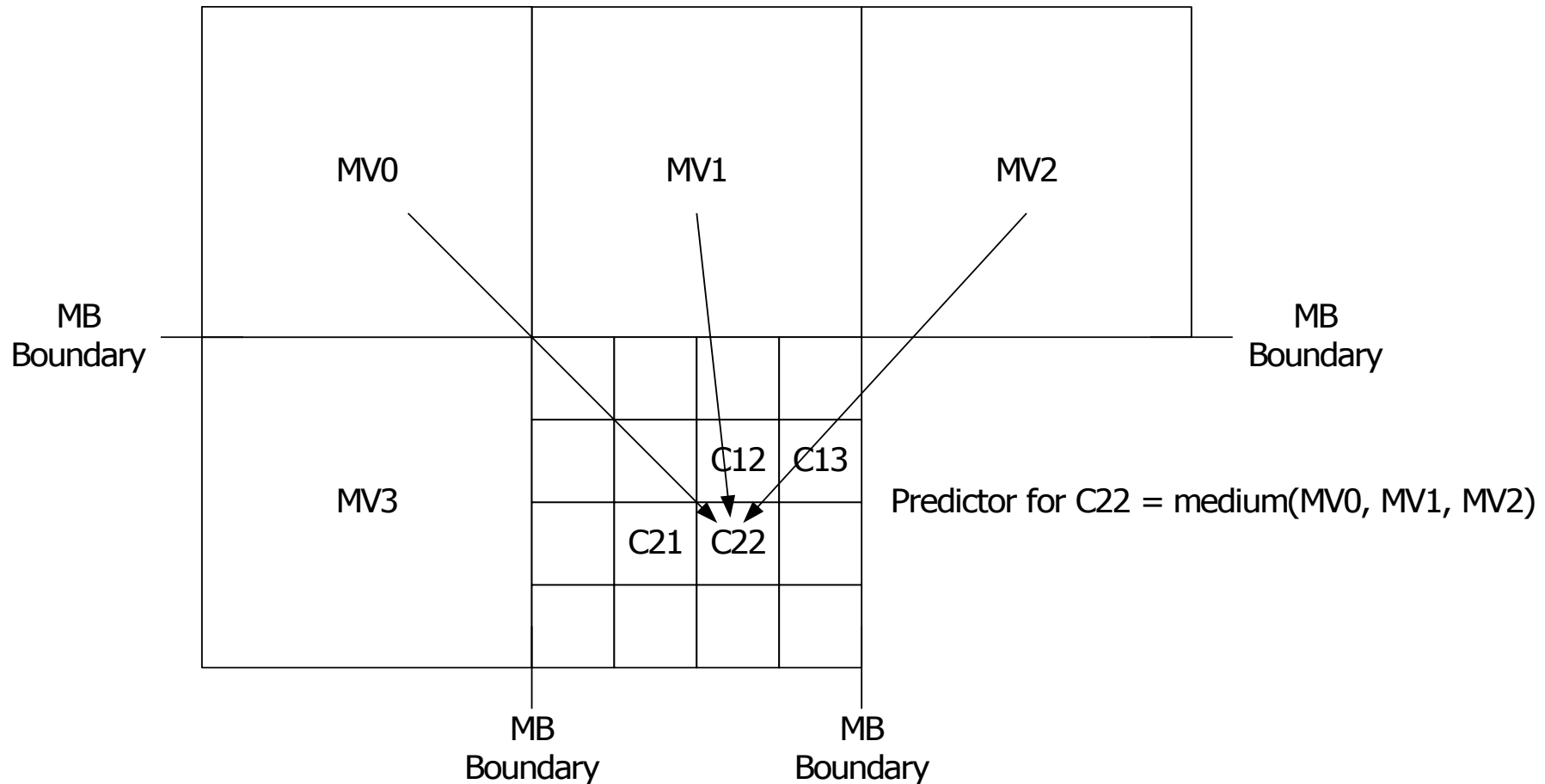
# Encountered Problem

- The exact estimation of *Rate* in the Lagrangian cost function makes parallel processing for different partitions impossible.



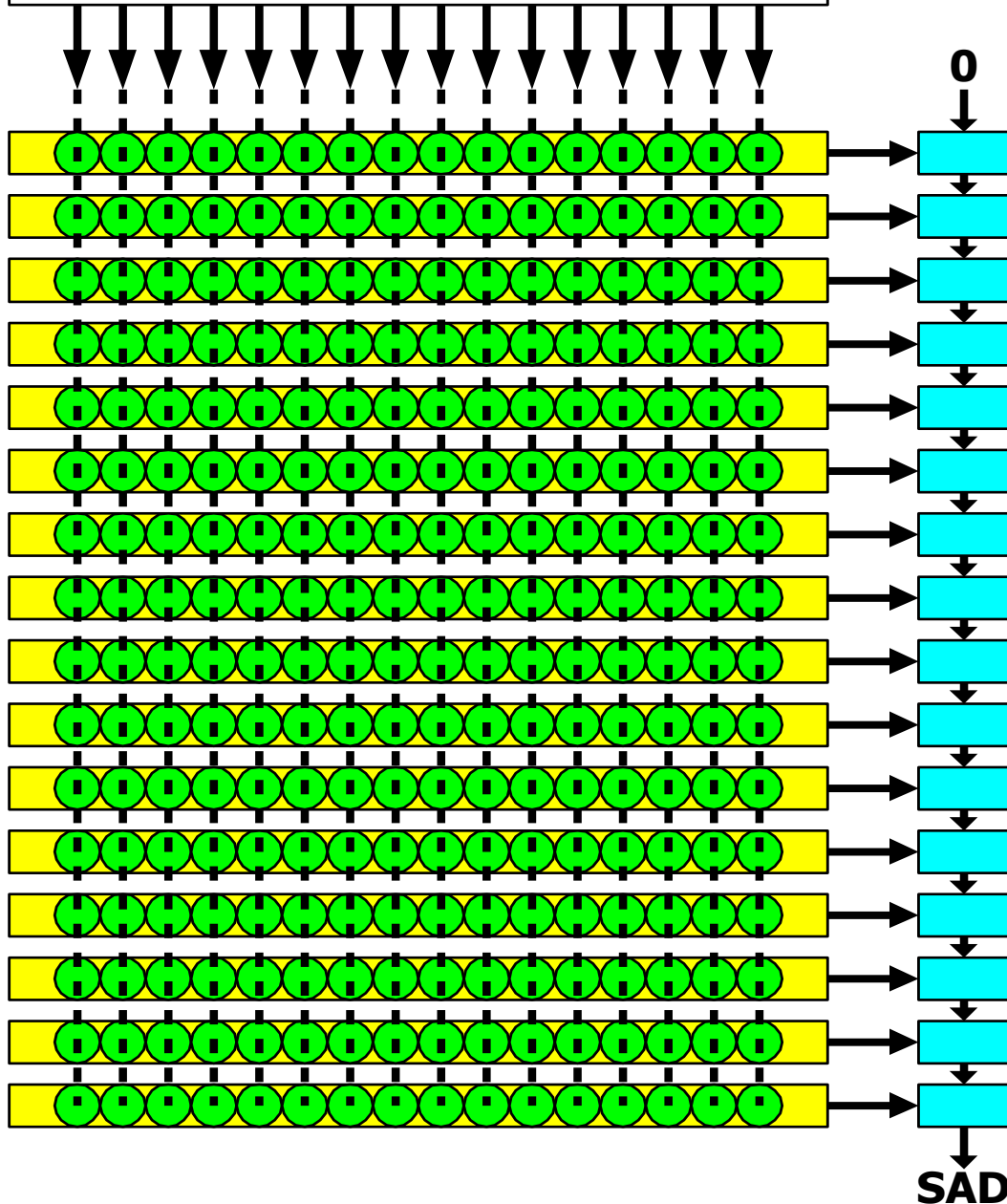


# Modified Macroblock Mode Decision





**16x1 Search Area Pixels**  
**Broadcast Each Pixel to 1x16 PE's**

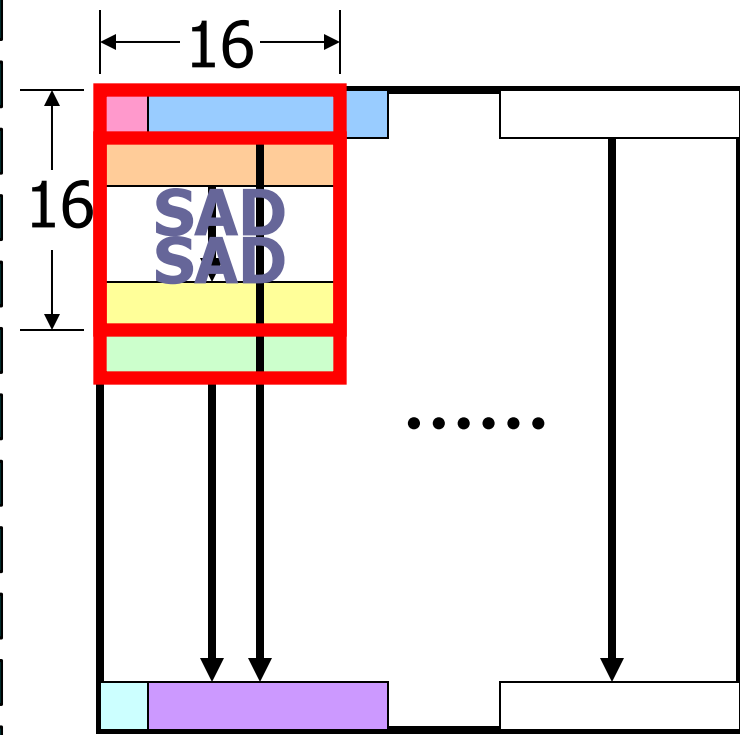
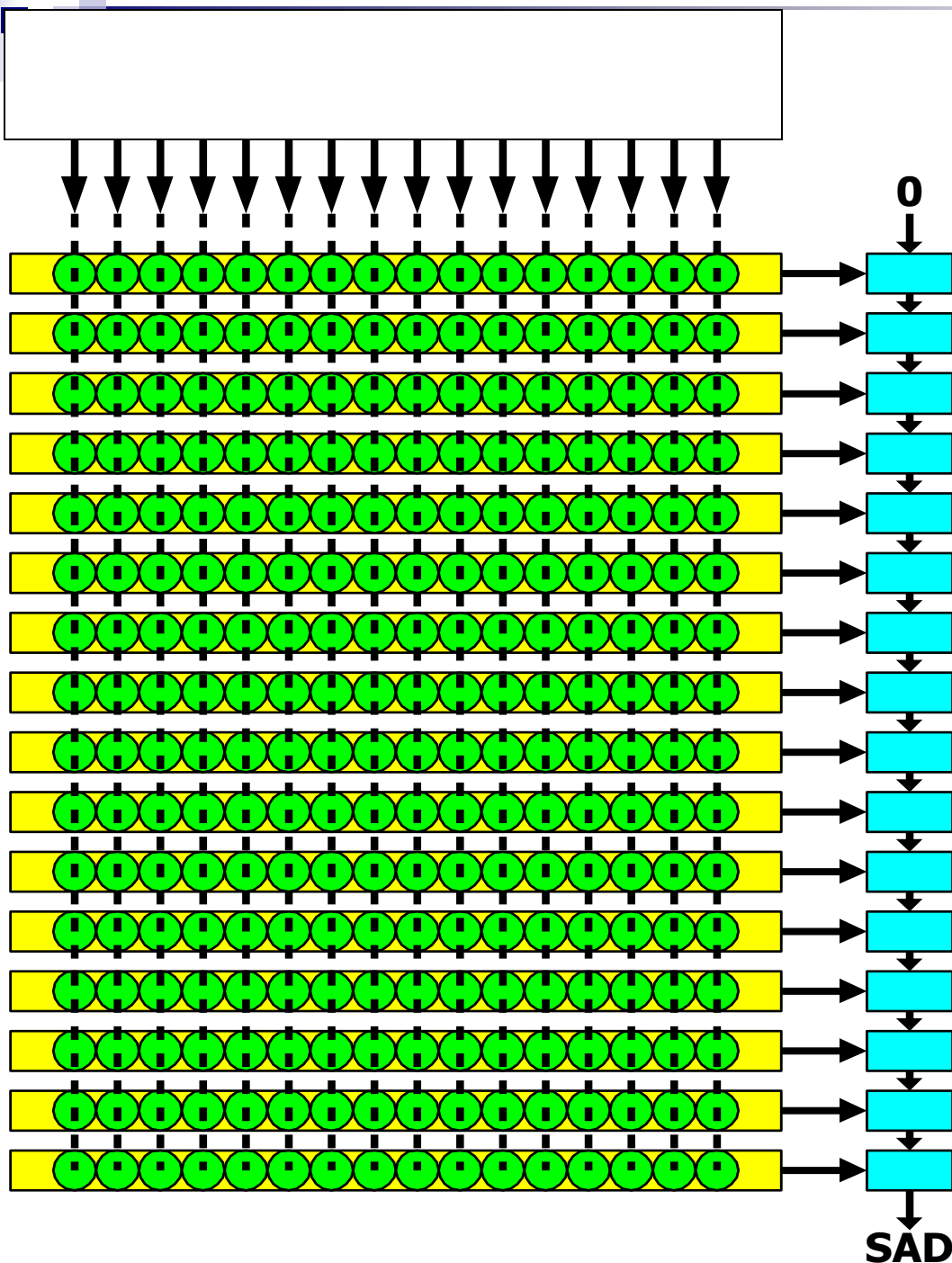


# Basic Architecture

- 256 PE's
- Current block stays
- Broadcast search area pixels
- Accumulate and propagate partial SAD values
- Do not require 256 8-bit registers to buffer the pixels for a candidate block



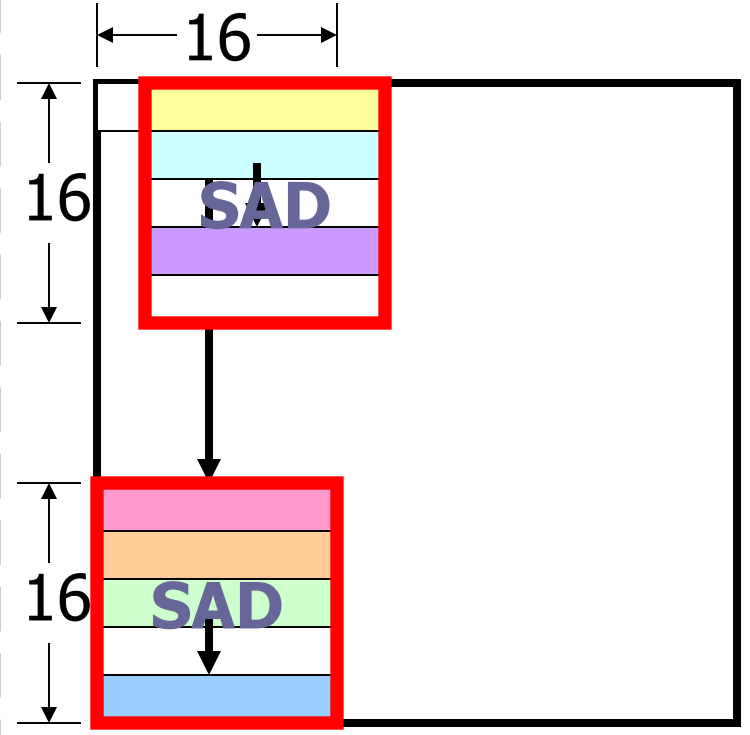
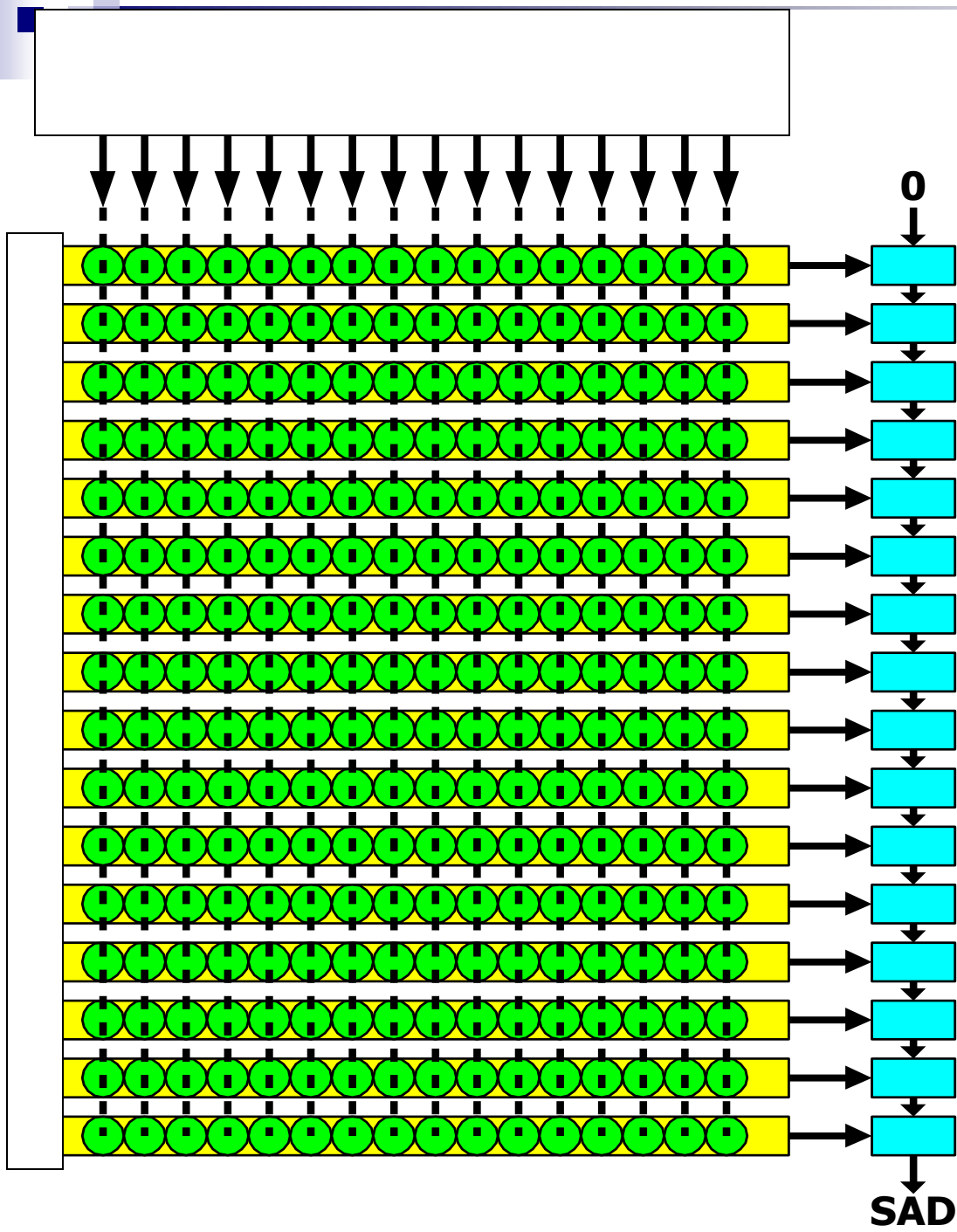
# Basic Data Flow



Search Area



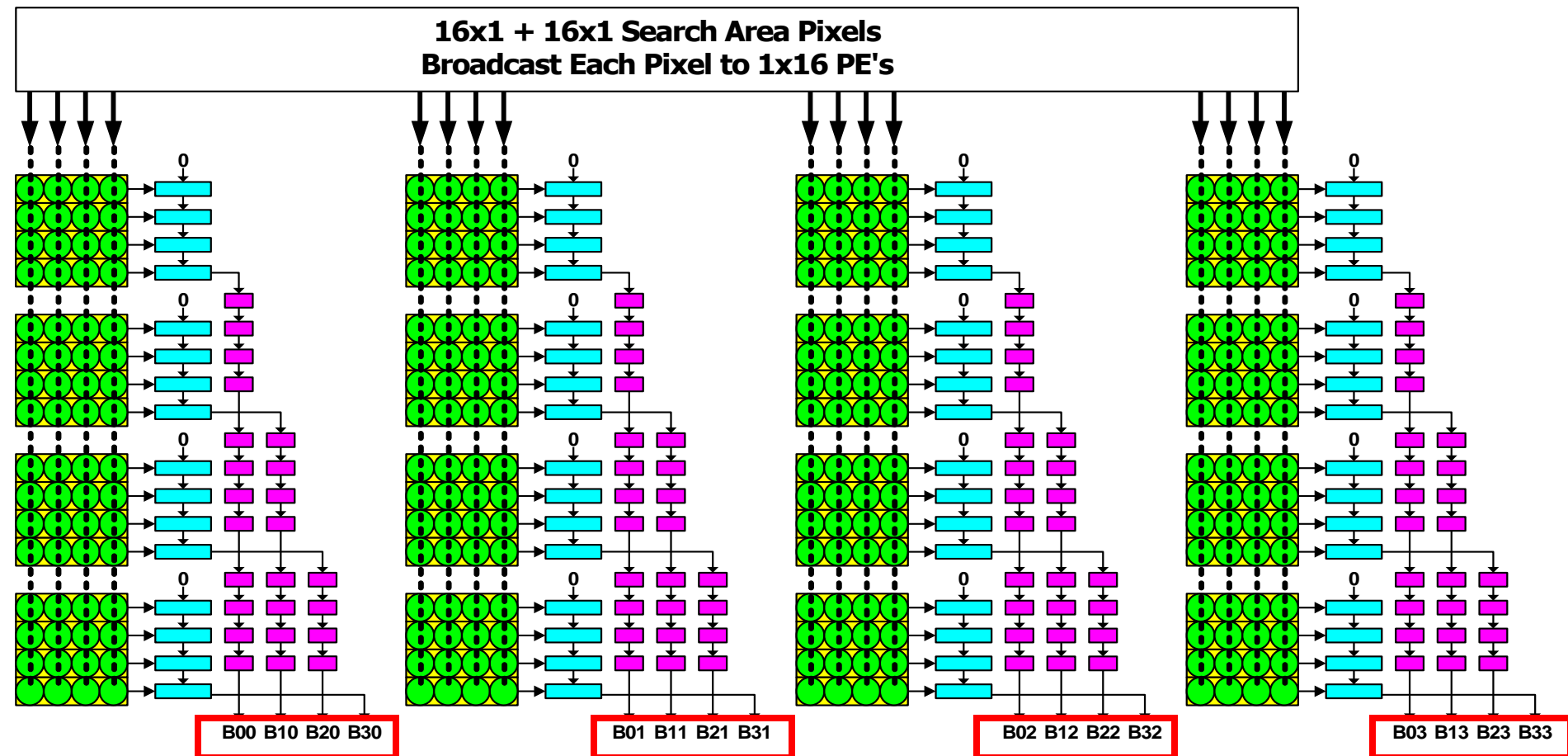
# Advanced Data Flow



Search Area

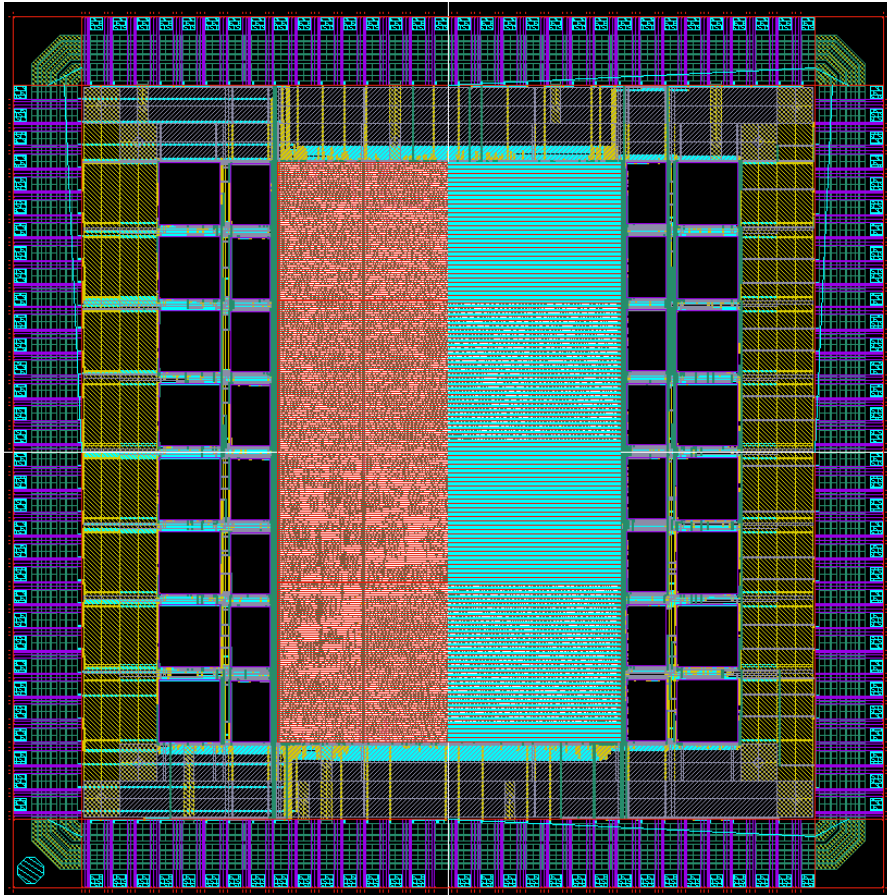
# Variable Block Size Architecture

16x1 + 16x1 Search Area Pixels  
Broadcast Each Pixel to 1x16 PE's



Reuse the SAD's of 4x4-blocks to form the SAD's of larger blocks

# Implementation



|              |  |
|--------------|--|
| Process      | TSMC 1P4M 0.35um   |
| Chip area    | 5.056 x 5.056 mm <sup>2</sup>                                    |
| Package      | 128 CQFP   |
| On-chip SRAM | 24,576 bits  |
| Gate count   | 105,575  |
| Max. freq.   | 66.67 MHz  |
| Search range | H [-24, +23], V [-16, +15]                                       |
| Capability   | 50 GOPS<br>D1 (720x480) 30fps 1Ref.<br>SIF (352x240) 30fps 4Ref. |



# Variable-Block-Size Motion Estimation

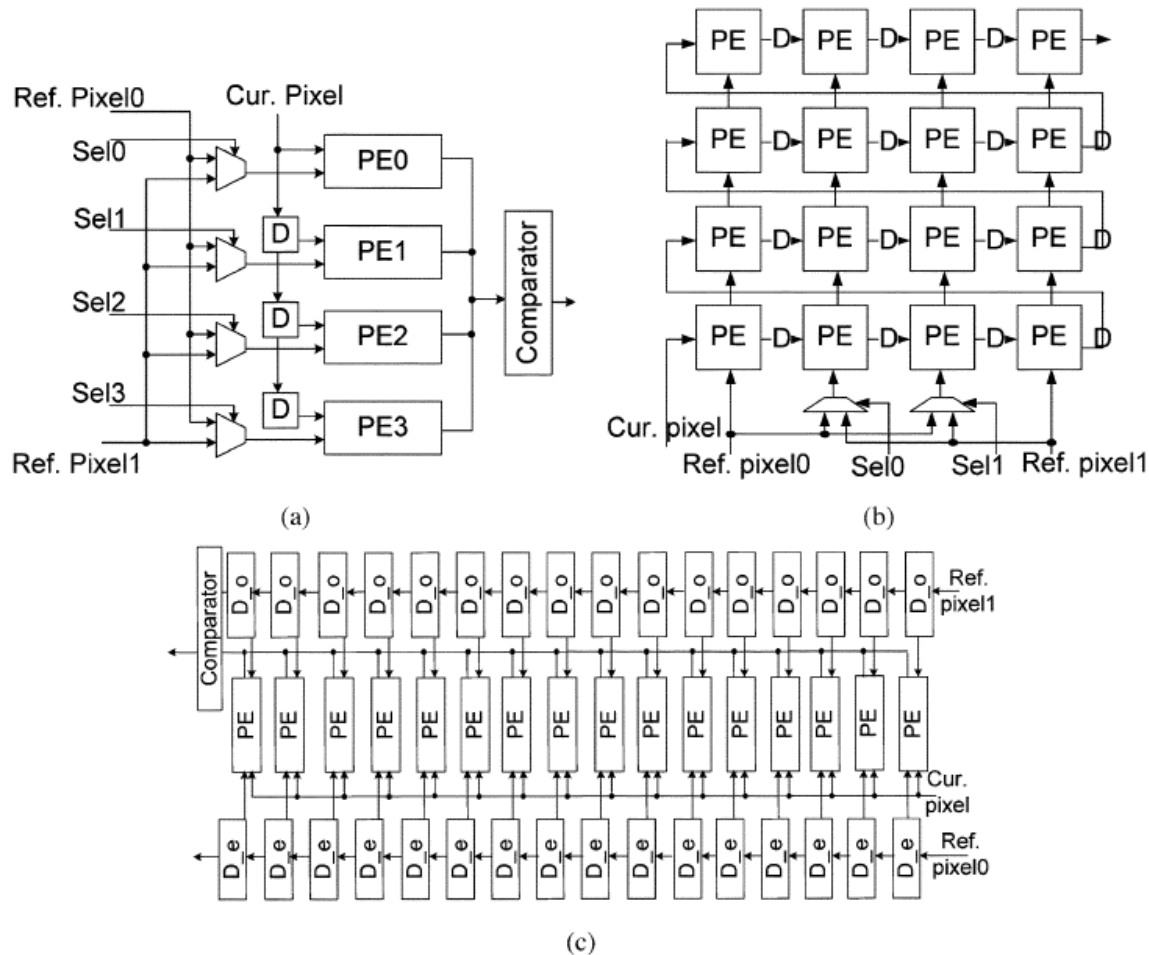
- Ching-Yeh Chen, Shao-Yi Chien, Yu-Wen Huang, Tung-Chien Chen, Tu-Chih Wang, and Liang-Gee Chen, "Analysis and Architecture Design of Variable Block-Size Motion Estimation for H.264/AVC," *IEEE Transactions on Circuits and Systems I*, vol 53. no. 2, pp. 578--593, Feb. 2006.
- Tung-Chien Chen, Shao-Yi Chien, Yu-Wen Huang, Chen-Han Tsai, Ching-Yeh Chen, To-Wei Chen, and Liang-Gee Chen, "Analysis and Architecture Design of an HDTV720p 30 Frames/s H.264/AVC Encoder," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, no. 6, pp. 673--688, June 2006.



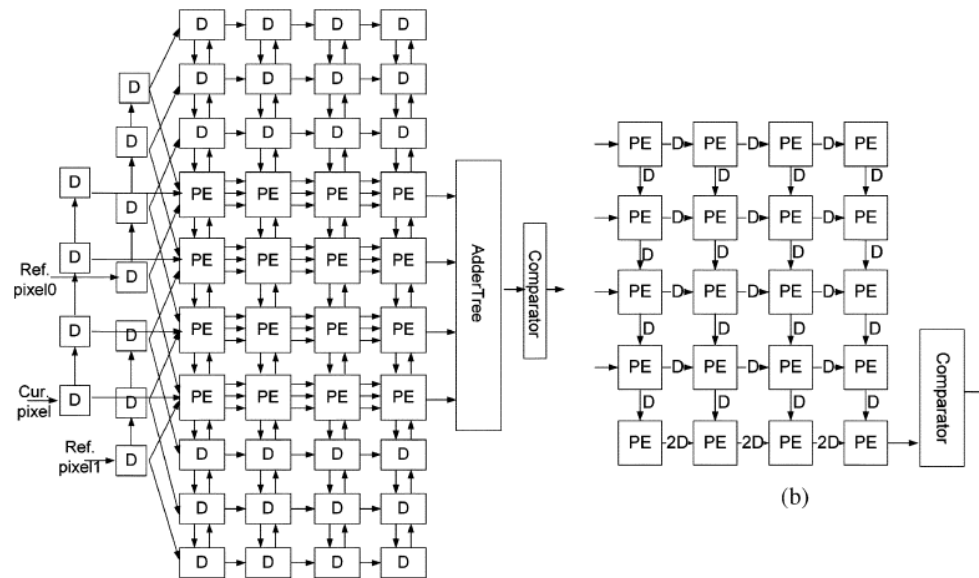
# Six Major Reference Architecture

- **1DInterYSW**: K. M. Yang, M. T. Sun, and L. Wu, “A family of VLSI designs for the motion compensation block-matching algorithm,” *IEEE Trans. Circuits Syst.*, vol. 36, no. 10, pp. 1317–1325, Oct. 1989.
- **2DInterYH**: H. Yeo and Y. H. Hu, “A novel modular systolic array architecture for full-search block matching motion estimation,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 5, no. 5, pp. 407–416, Oct. 1995.
- **2DInterLC**: Y. K. Lai and L. G. Chen, “A data-interlacing architecture with twodimensional data-reuse for full-search block-matching algorithm,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, no. 2, pp. 124–127, Apr. 1998.
- **2DIntraVS**: T. Komarek and P. Pirsch, “Array architectures for block matching algorithms,” *IEEE Trans. Circuits Syst.*, vol. 36, no. 10, pp. 1301–1308, Oct. 1989.
- **2DIntraKP**: L. De Vos and M. Stegherr, “Parameterizable VLSI architectures for the full-search block-matching algorithm,” *IEEE Trans. Circuits Syst.*, vol. 36, no. 10, pp. 1309–1316, Oct. 1989.
- **2DIntraHL**: C. H. Hsieh and T. P. Lin, “VLSI architecture for block-matching motion estimation algorithm,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 2, no. 2, pp. 169–175, Jun. 1992.

# Inter-Parallel Architecture

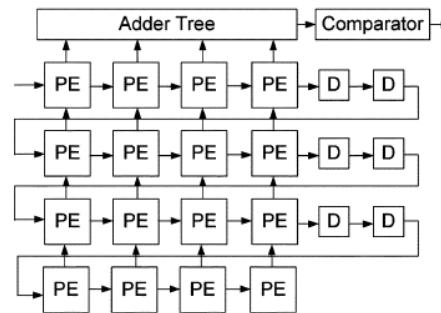


# Intra-Parallel Architecture



(a)

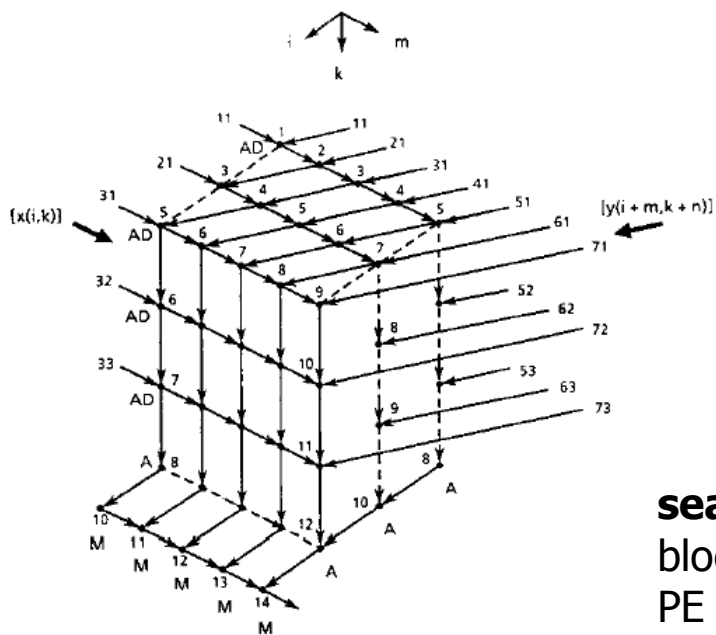
(b)



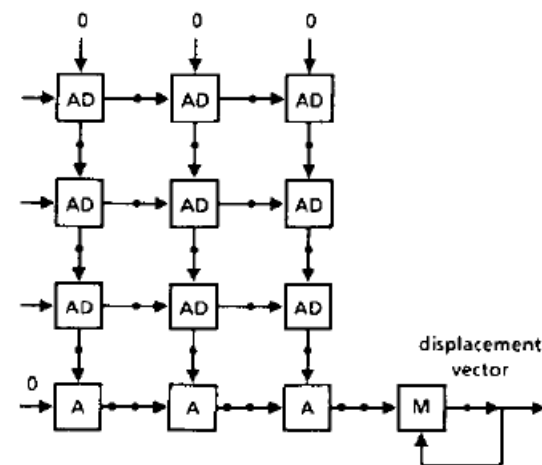
(c)



# AB2 Architecture



**search range  $[-p, +p]$**   
 block size  $N \times N$   
 PE no. =  $N^2 = 2\text{-D block size}$

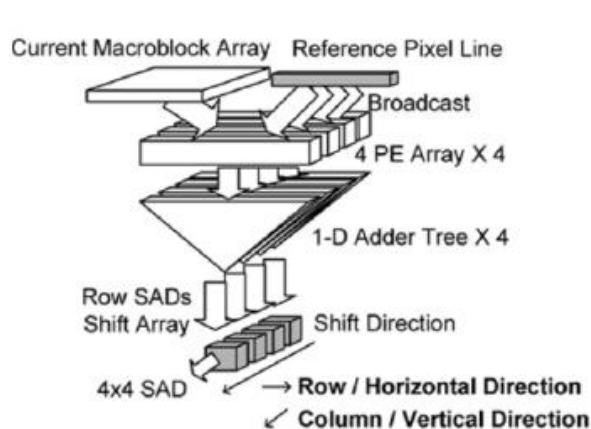


Dependence Graph  
 projection on  $i, k$  plane

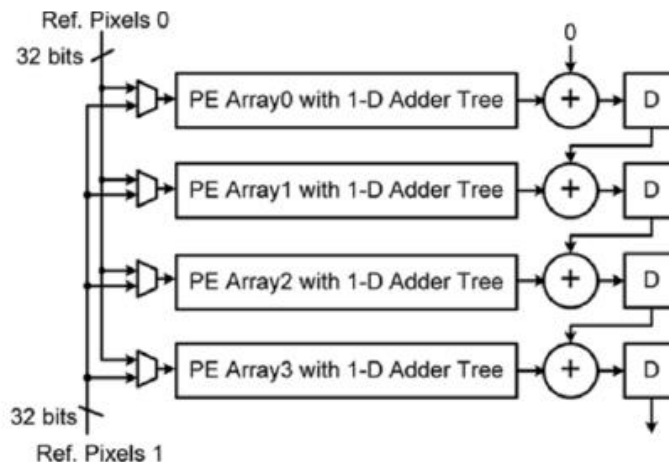
Systolic Architecture  
 ( $N=3, p=2$ )

Number of cycles for a macroblock =  $(2p+1) \times (2p+N)$

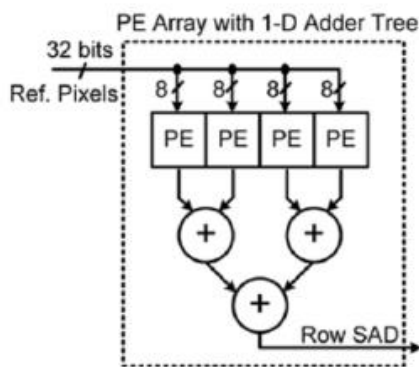
# Propagation Partial SAD Architecture



(a)

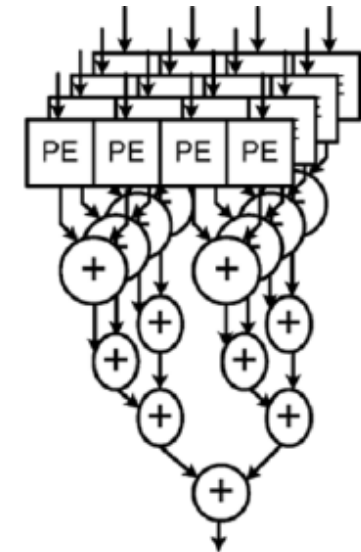
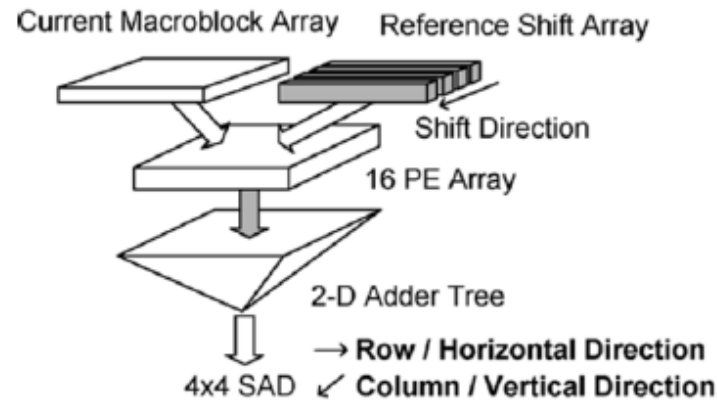


(b)



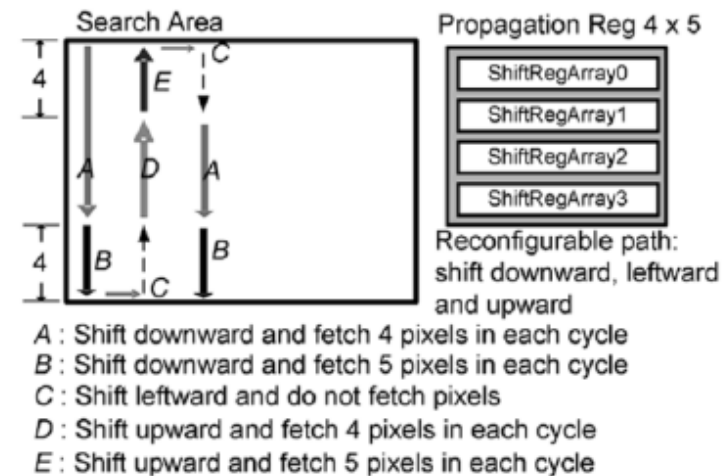
(c)

# SAD Tree Architecture



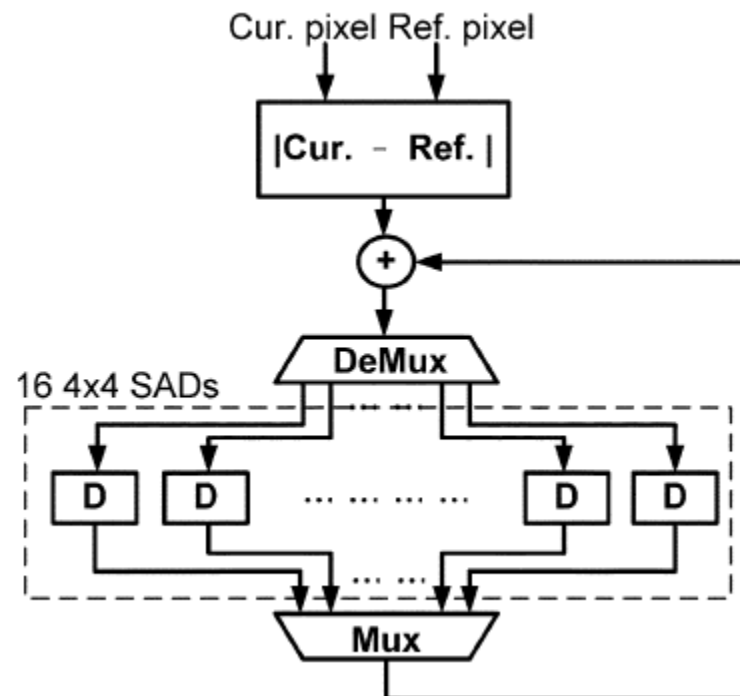
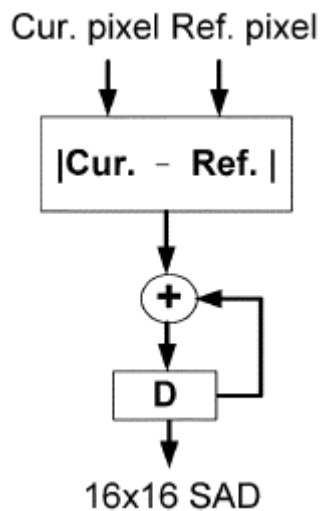
(a)

(b)



(c)

# VBSME Version?

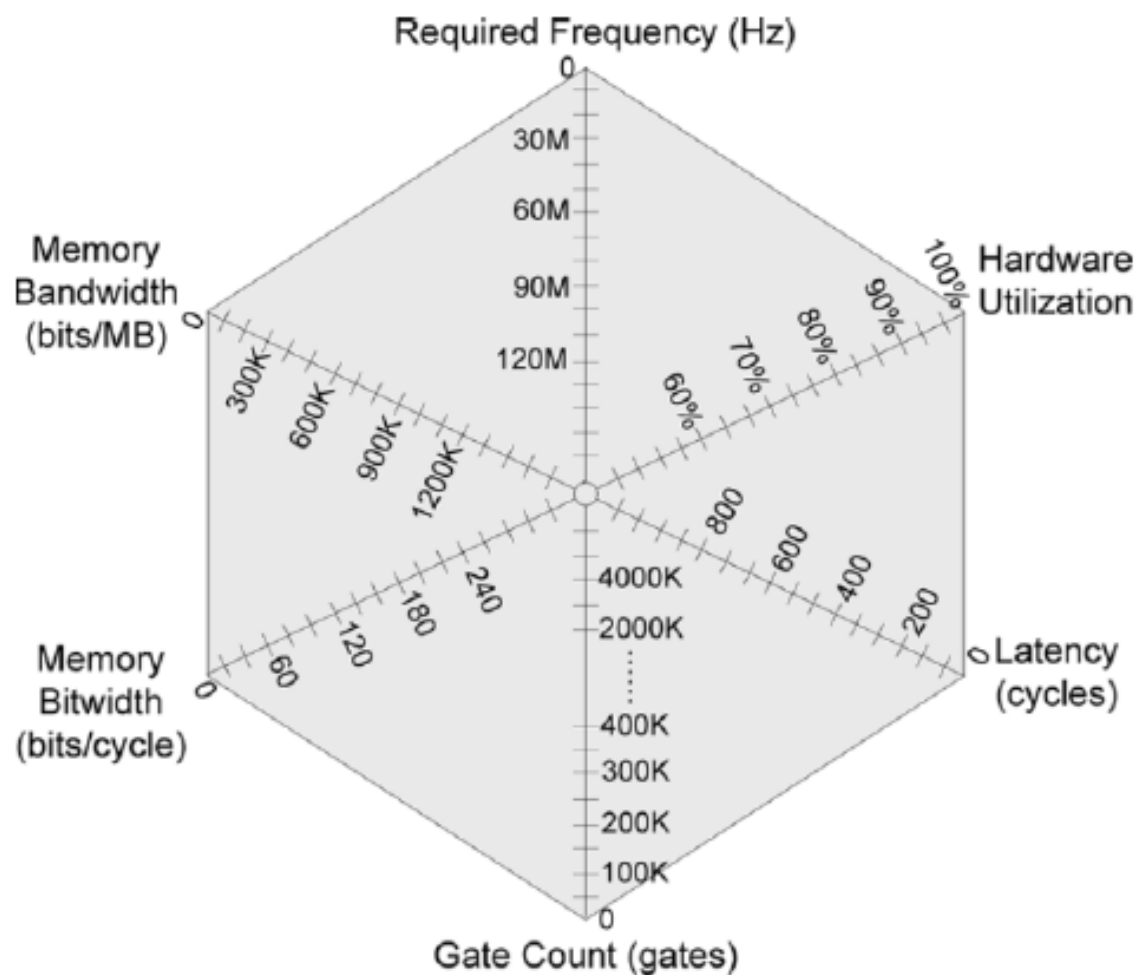


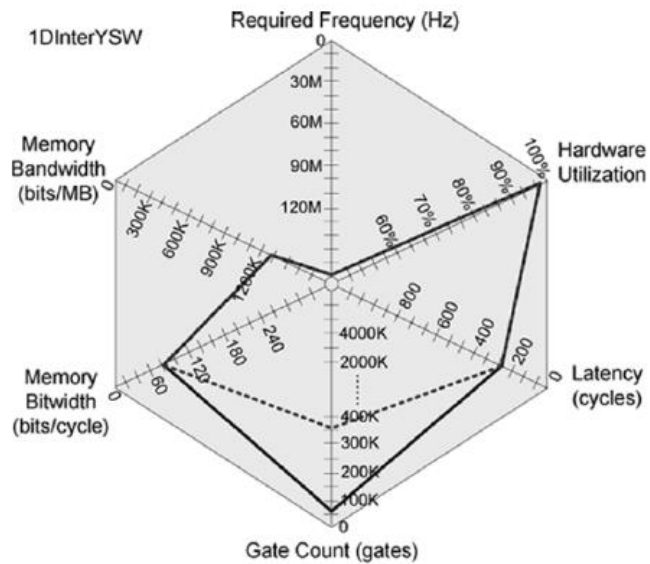


# Comparison

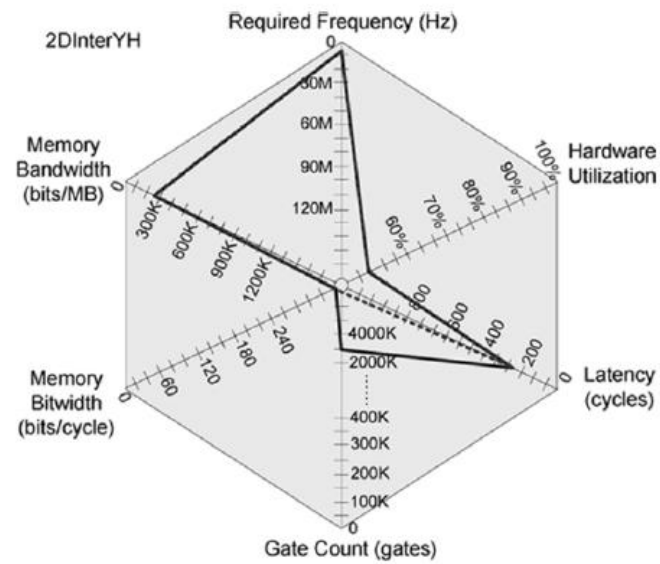
| Name                         | No. of PEs         | Operating Cycles<br>(Cycles/Macroblock) | Latency<br>(Cycles)                  | Data Flow     |
|------------------------------|--------------------|---|--------------------------------------|---------------|
| 1DInterYSW [5]               | $2P_h$             | $N^2 \times 2P_v + 2P_h$                | $N^2$                                | Data Flow I   |
| 2DInterYH [7]                | $2P_h \times 2P_v$ | $2N^2$                                  | $N^2$                                | Data Flow I   |
| 2DInterLC [8]                | $2P_h \times 2P_v$ | $2N^2$                                  | $2N^2$                               | Data Flow I   |
| 2DIntraVS [9]                | $N^2$              | $2P_h \times 2P_h + N \times 2P_v$      | $N \times 2P_v$                      | Data Flow III |
| 2DIntraKP [6]                | $N^2$              | $2P_v \times (N + 2P_h) + N$            | $3N$                                 | Data Flow II  |
| 2DIntraHL [10]               | $N^2$              | $(2P_v + N - 1) \times (2P_h + N - 1)$  | $2N + (N - 1) \times (2P_h + N - 2)$ | Data Flow II  |
| <i>Propagate Partial SAD</i> | $N^2$              | $2P_h \times 2P_v + N - 1$              | $N$                                  | Data Flow II  |
| <i>SAD Tree</i>              | $N^2$              | $2P_h \times 2P_v + N - 1$              | $N$                                  | Data Flow III |

# Comparison: Hexagonal Plot

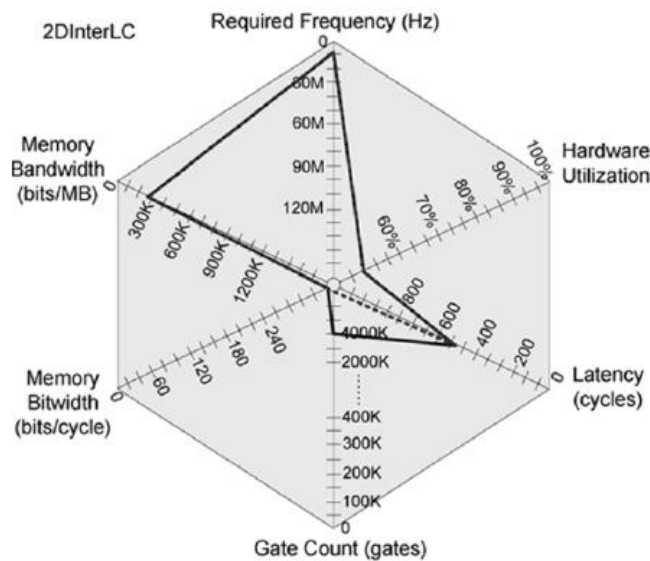




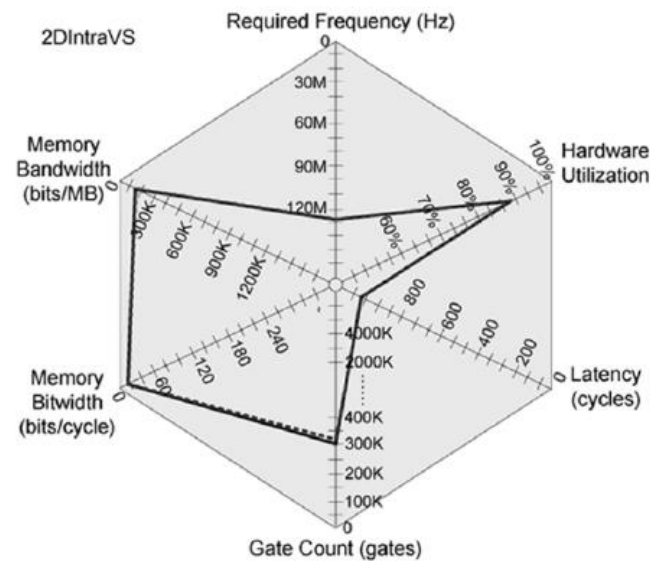
(a)



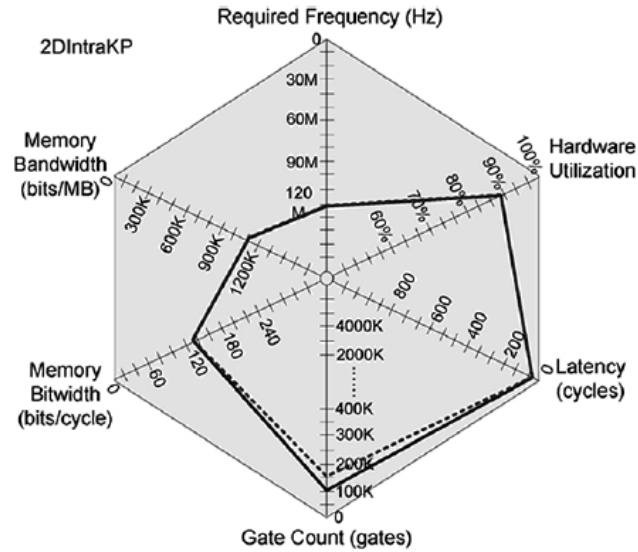
(b)



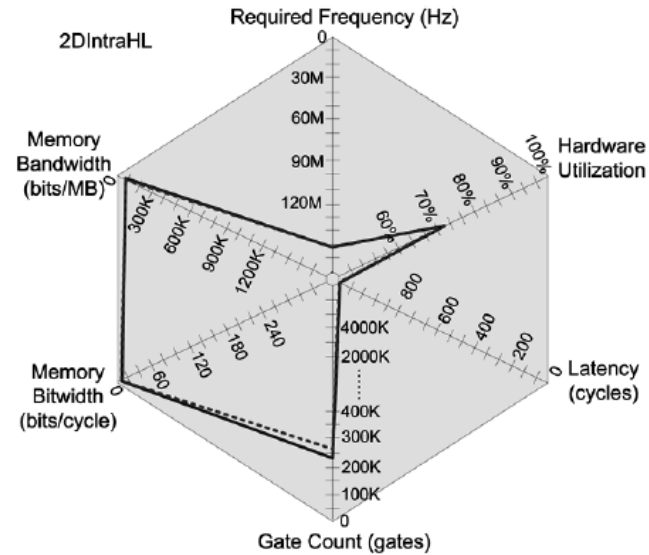
(c)



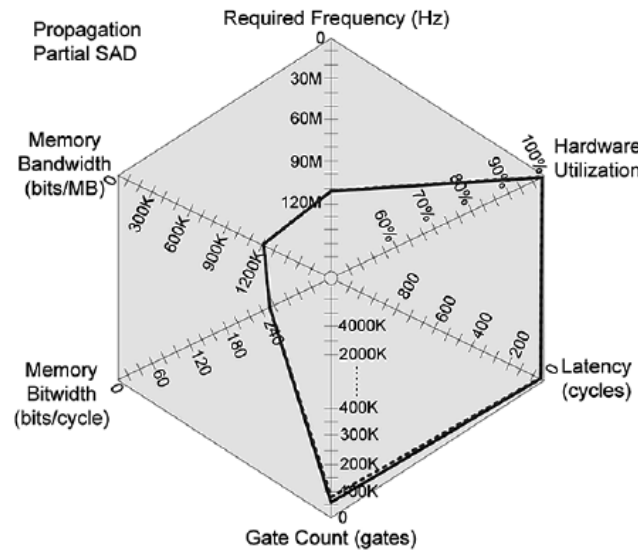
(d)



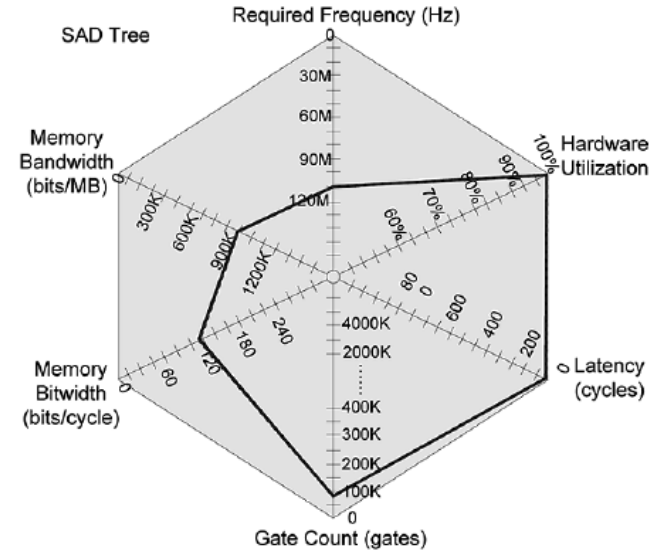
(e)



(f)

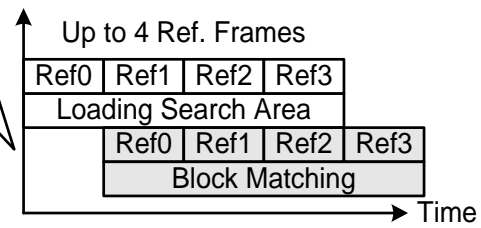
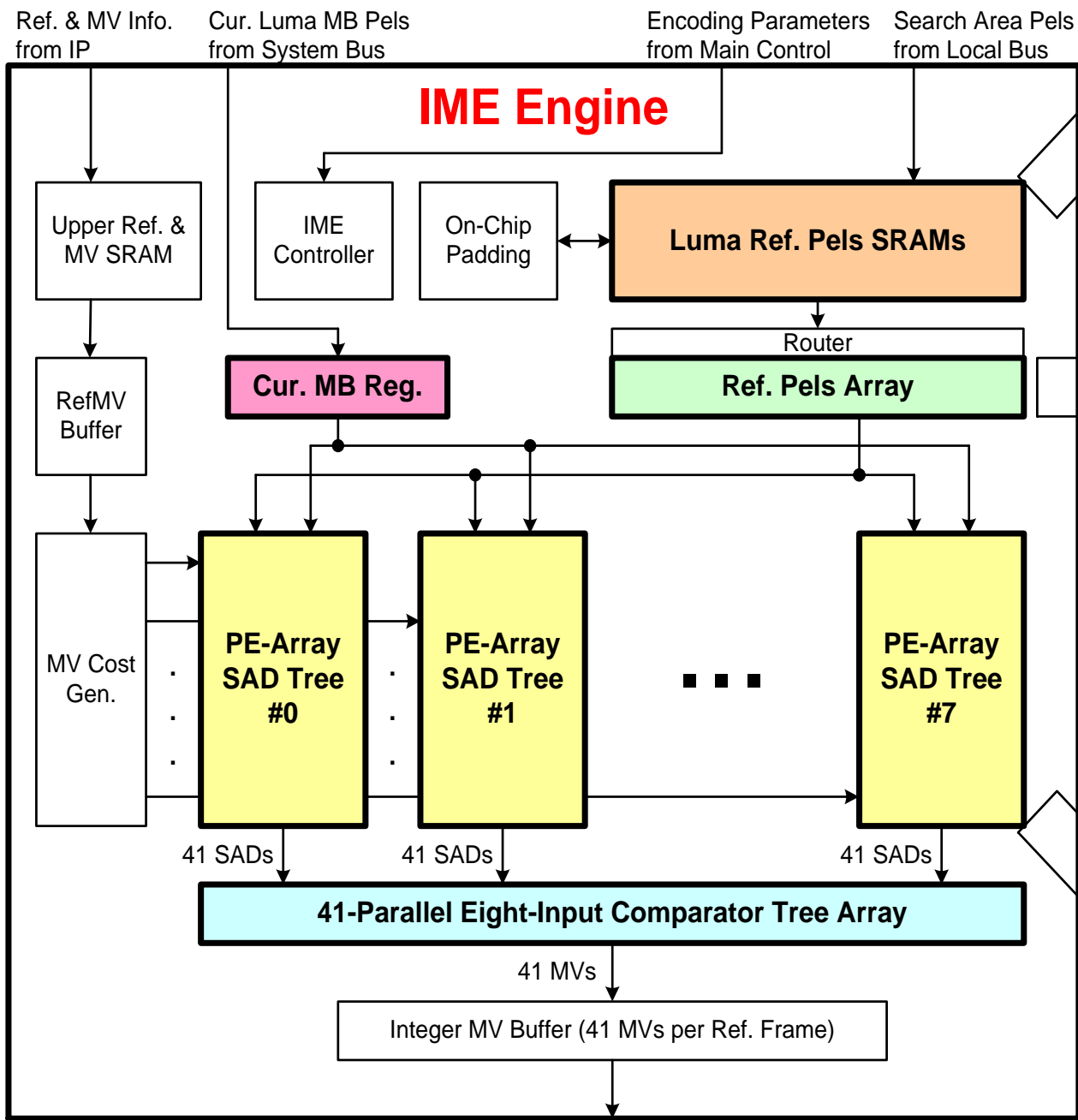


(g)



(h)





Data path can be configured in three directions to support snake scan.

1. Shift downward for one pel
2. Shift leftward for eight pels
3. Shift upward for one pel

