Hardware Architecture of Motion Estimation (ME)

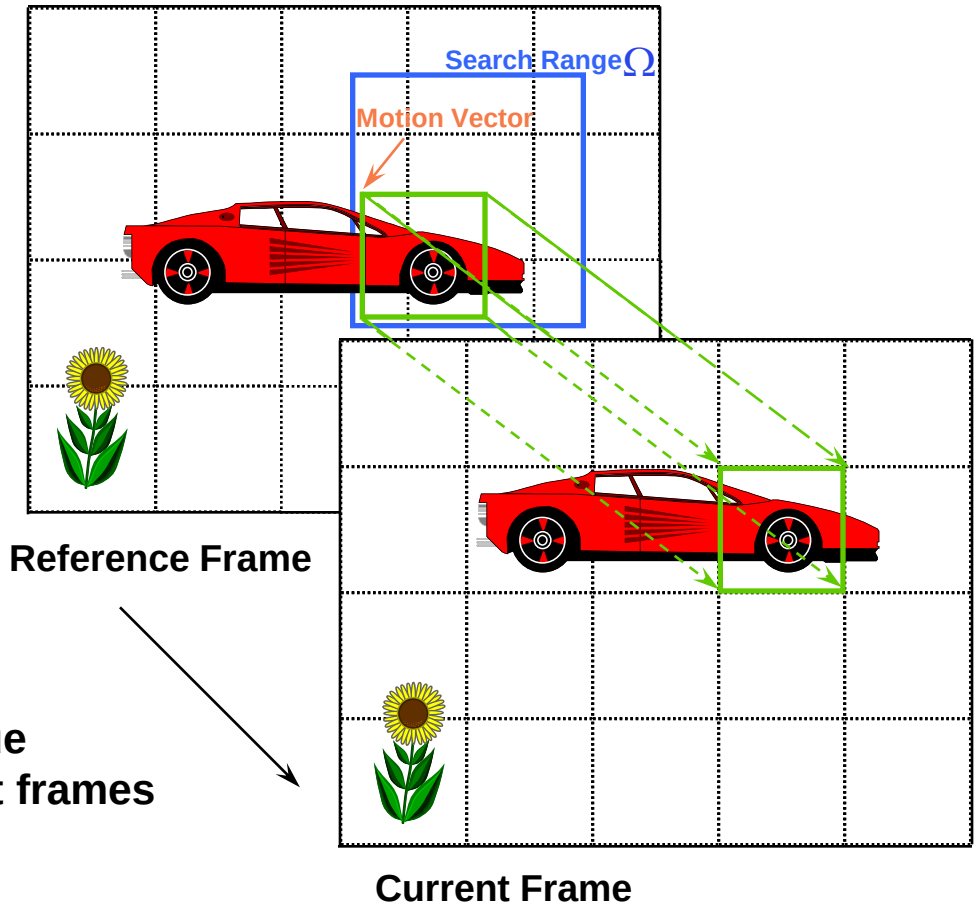
Shao-Yi Chien

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

Block Matching Algorithm (BMA) (1/3)

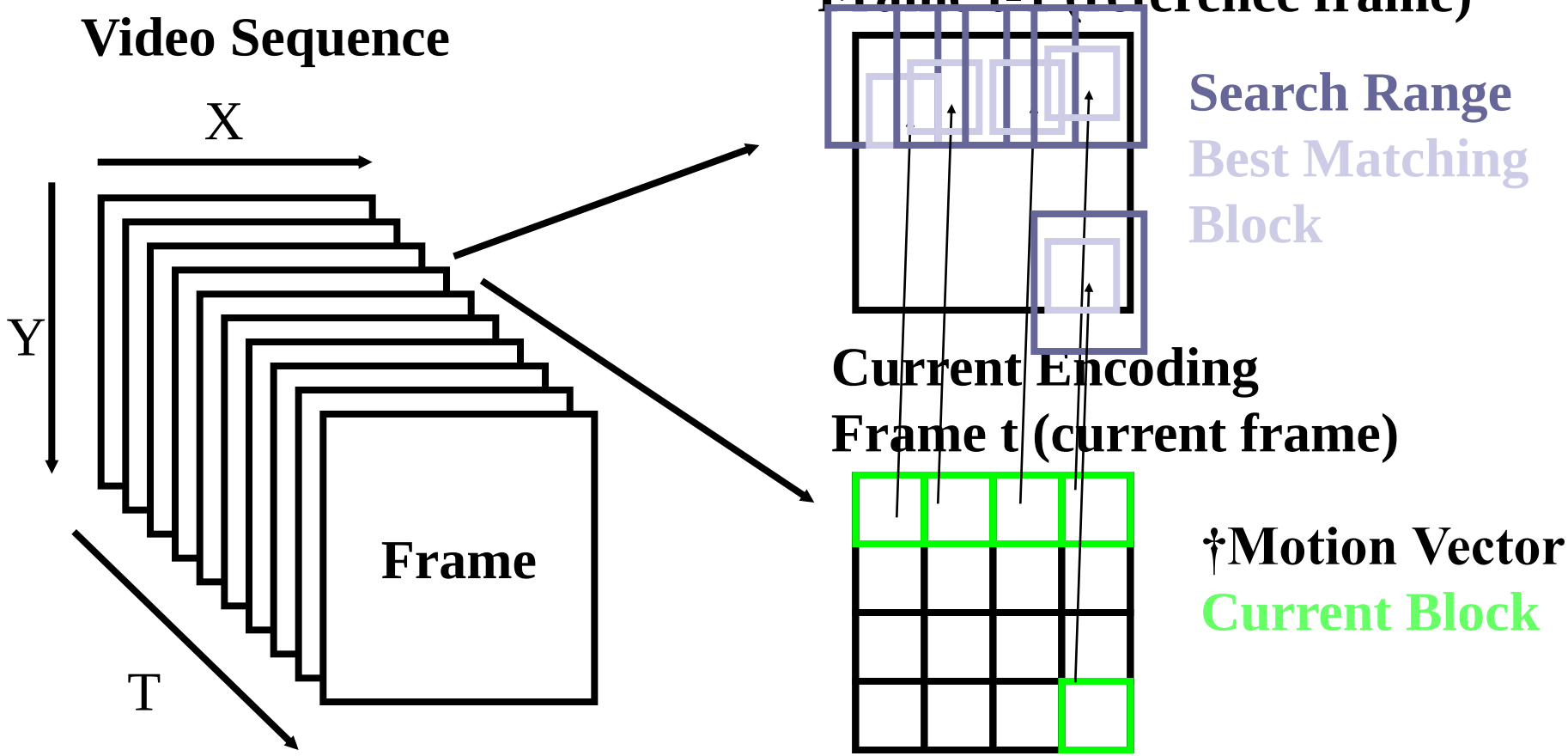
- Track block regions from frame to frame
- Compromise between efficiently removing temporal redundancy and computation cost
- Usually, motion estimation is only performed at the **encoder side** to avoid the huge computation at the decoder side, so the motion data have to be transmitted from the encoder 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)



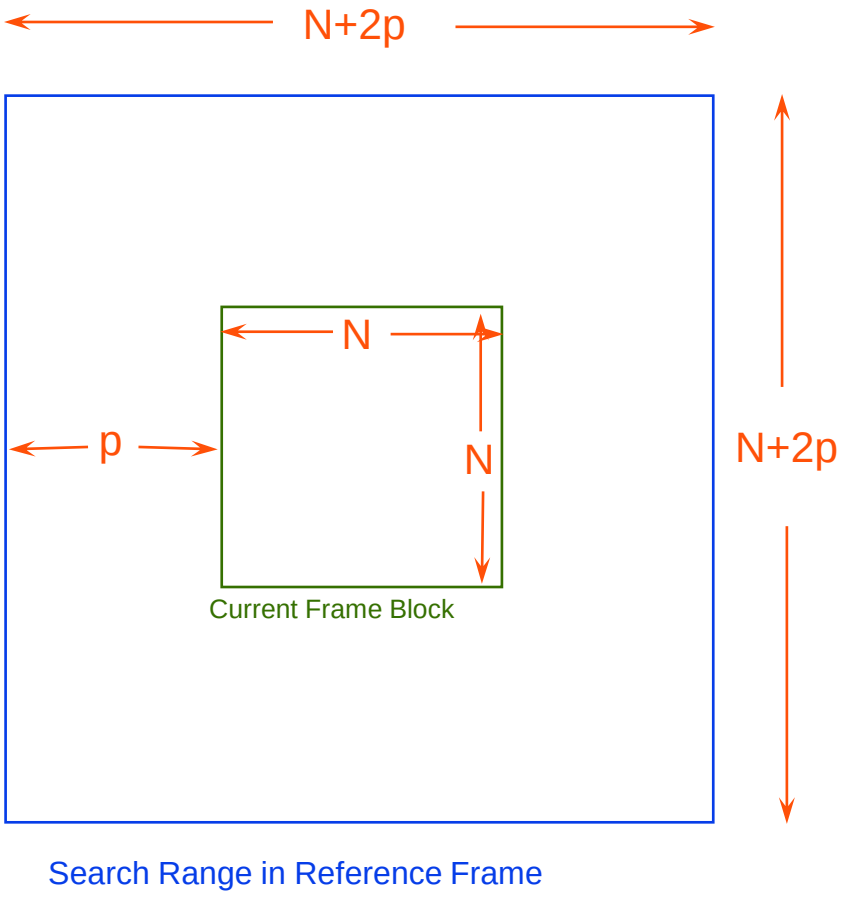
Motion Vector
$V_t(p.q) = (Veci, Vecj)$
the location in the search range Ω
that has the maximum correlation value
between blocks in temporally adjacent frames

Block Matching Algorithm (BMA) (3/3)



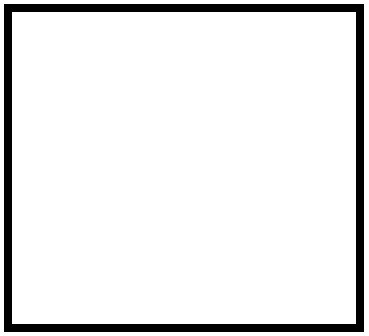
Factors of Affecting BMA

- 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]



Search Range in Reference Frame

Full-Search Block Matching Algorithm

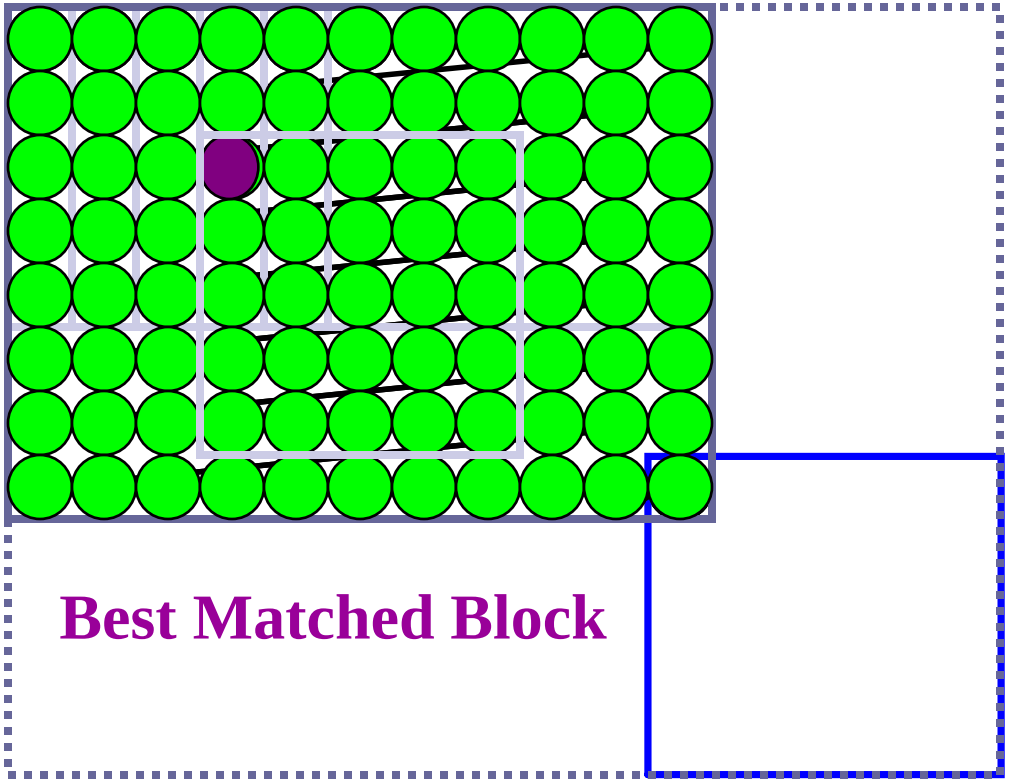


Current Block

Search Range

Reference Block (Candidate Block)

Candidate Search Position (Search Location)



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

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

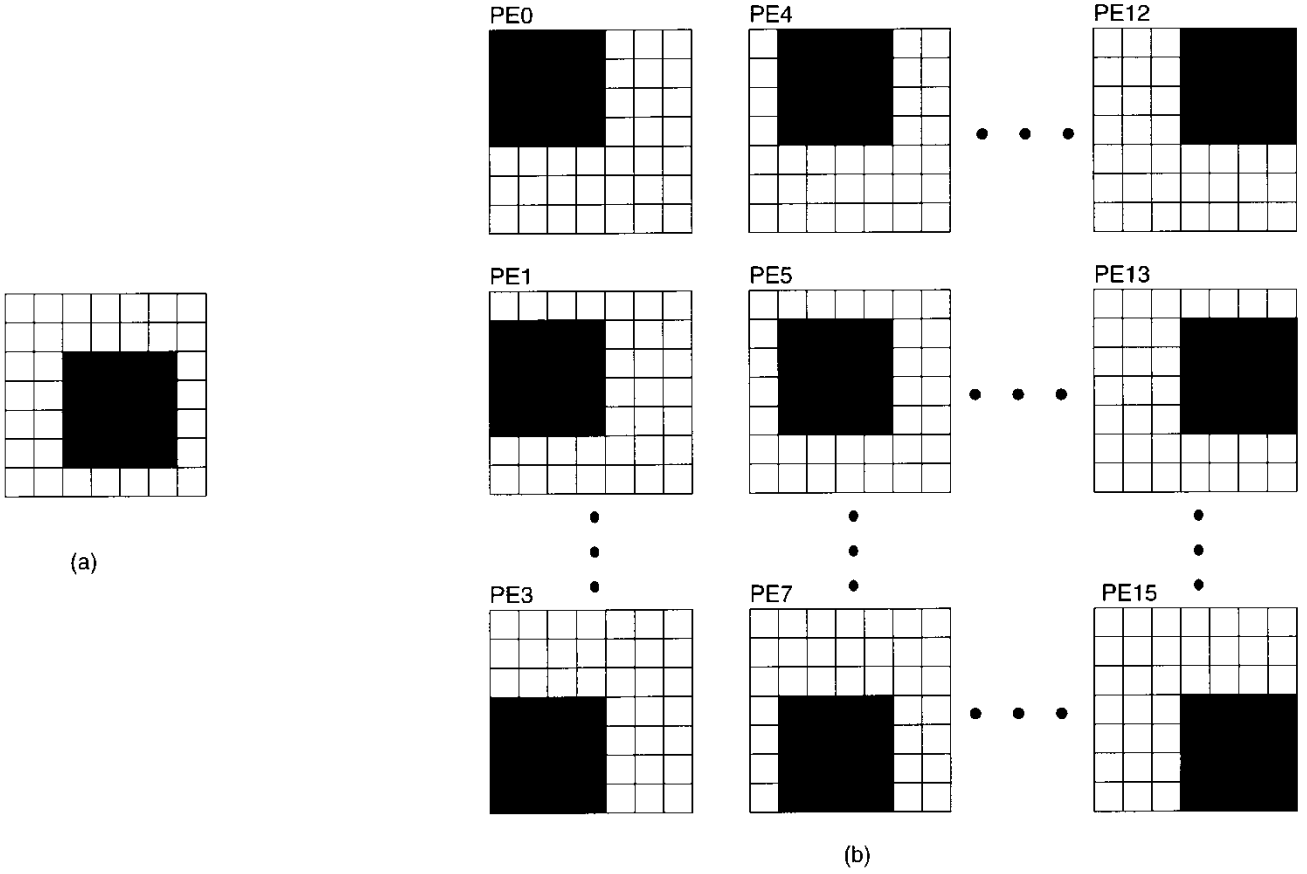
Inter-Level Parallelism (1/2)

- **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
                       SAD(k,l) = 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 (2px2p PE's), at least (NxN) cycles are required.
For 1-D arrays (2px1 PE's), at least (2pxNxN) cycles are required.

Inter-Level Parallelism (2/2)



current block candidate blocks

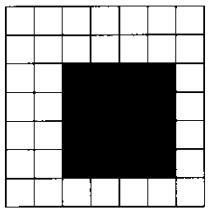
Intra-Level Parallelism (1/2)

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

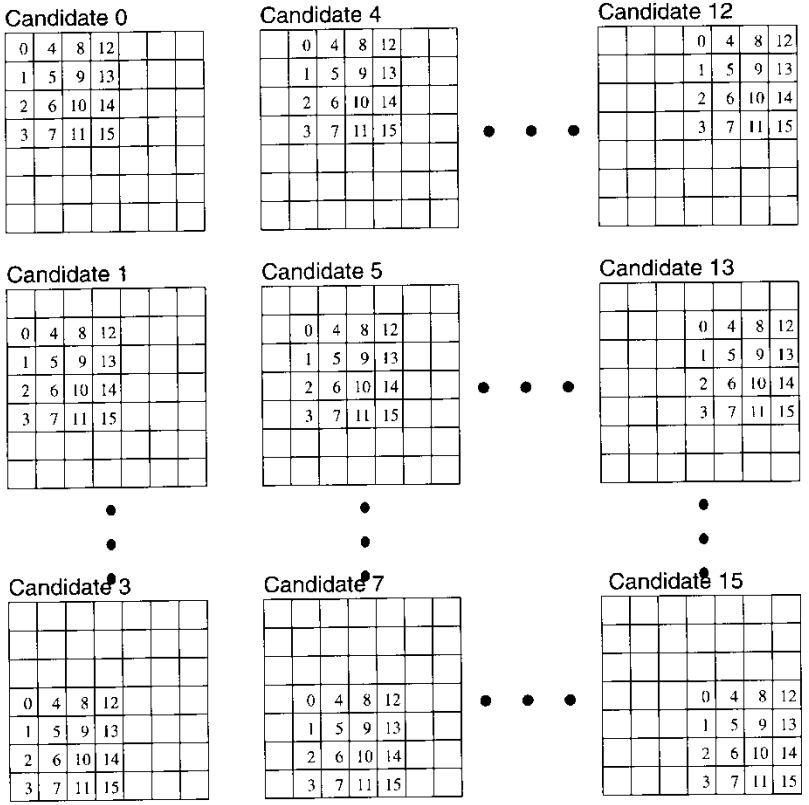
```
Loop 3:          For m = 0 to N-1
Loop 4:               For n = 0 to N-1
                           SAD(k,l) = SAD(k,l) + |X(m,n)-Y(m+k,n+l)|
                      End (Loop 4)
                 End (Loop 3)
            End (Loop 2)
       End (Loop 1)
```

Each PE is responsible for the SAD of one pixel in all candidates.
For 2-D arrays (NxN PE's), at least (2px2p) cycles are required.
For 1-D arrays (Nx1 PE's), at least (Nx2px2p) cycles are required.

Intra-Level Parallelism (2/2)



(a)



(b)

current block

candidate blocks

1-D Systolic Array

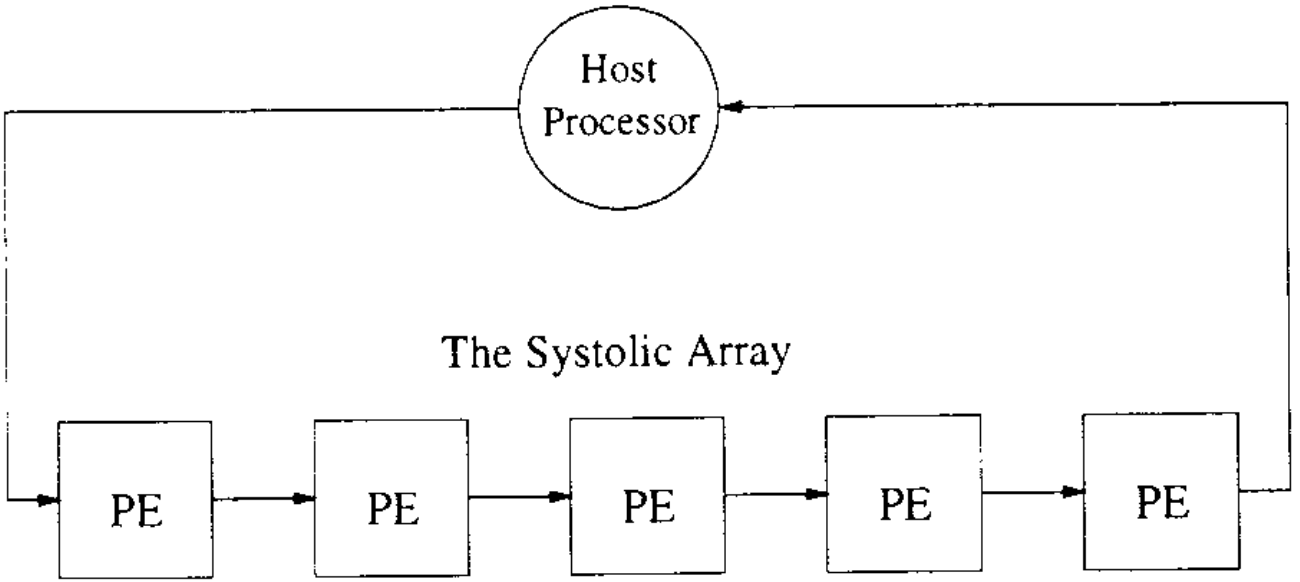
K.-M. Yang, M.-T. Sun, L. Wu, “A family of VLSI design for the motion compensation block matching algorithm,” *IEEE Transactions on Circuits and Systems*, vol. 36, no. 10, October 1989

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

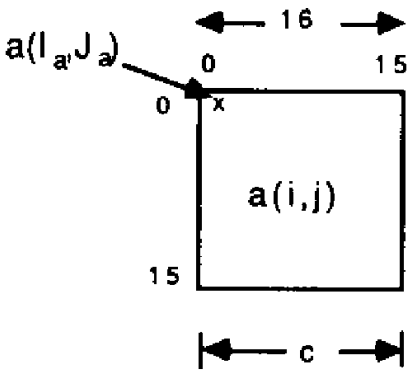
- The first chip design for block matching motion estimation in the world
- Two kinds of dataflow
  - Broadcasting reference frame, move current
  - Broadcasting current frame, move reference
- PE number = 1-D search range
- Each PE computes the SAD of a candidate macroblock
- Flexible block size (simply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

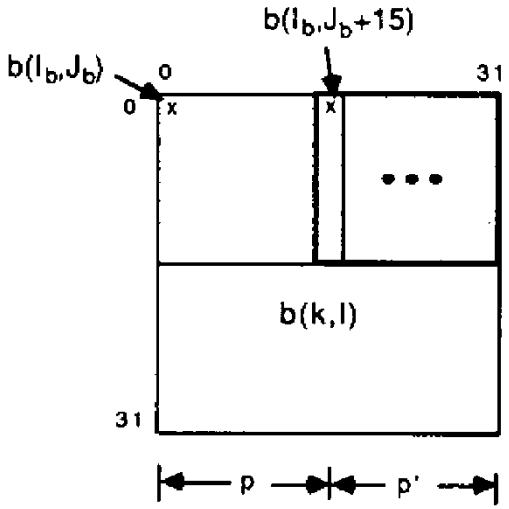
Block Matching Algorithm

- $$S(m_j) = \sum\sum |a(I_a + i, J_a + j) - b(I_b + k, J_b + l + m_j)|,$$

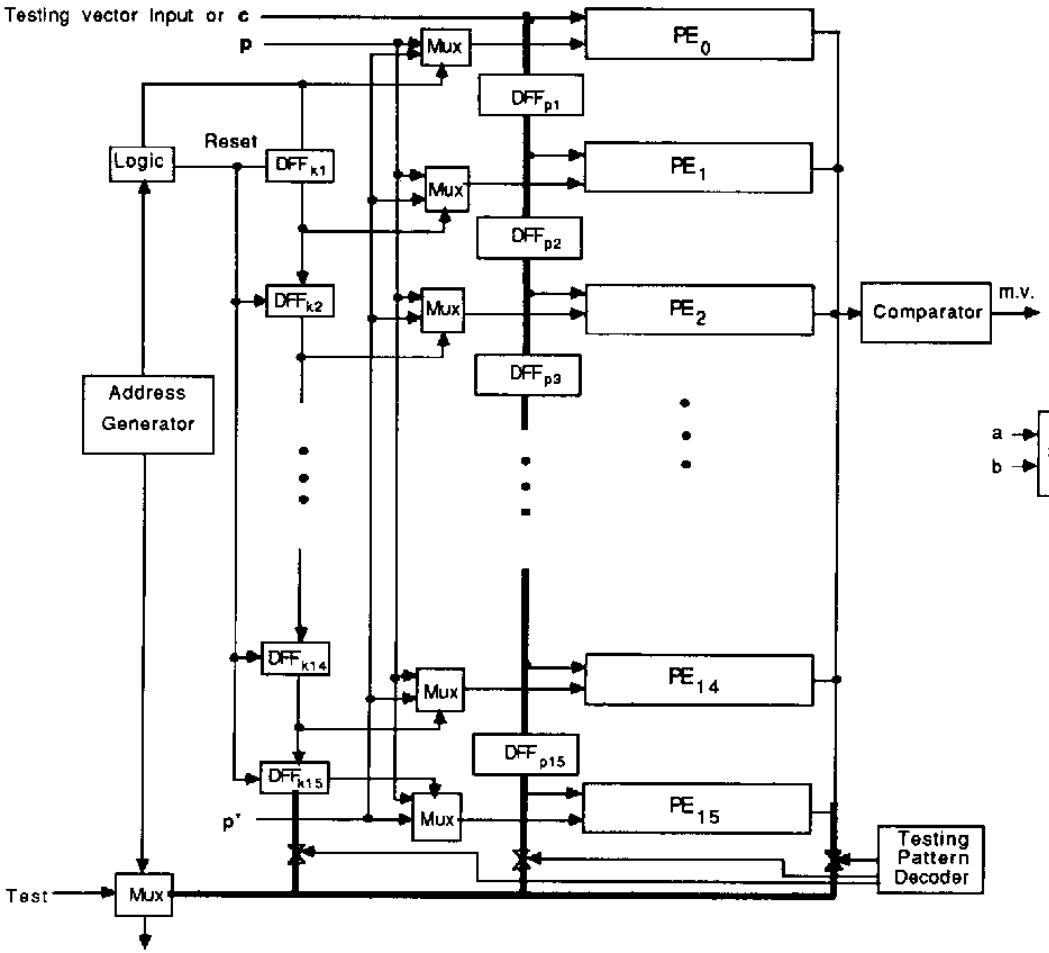
  $$\text{for } m_j = 0, 1, \cdots, 15.$$

  - a: current frame
  - b: reference frame



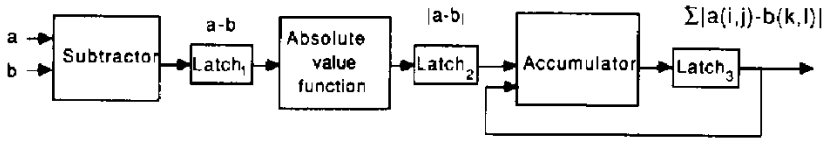


Broadcasting Reference Frame



search range [-8, +7]

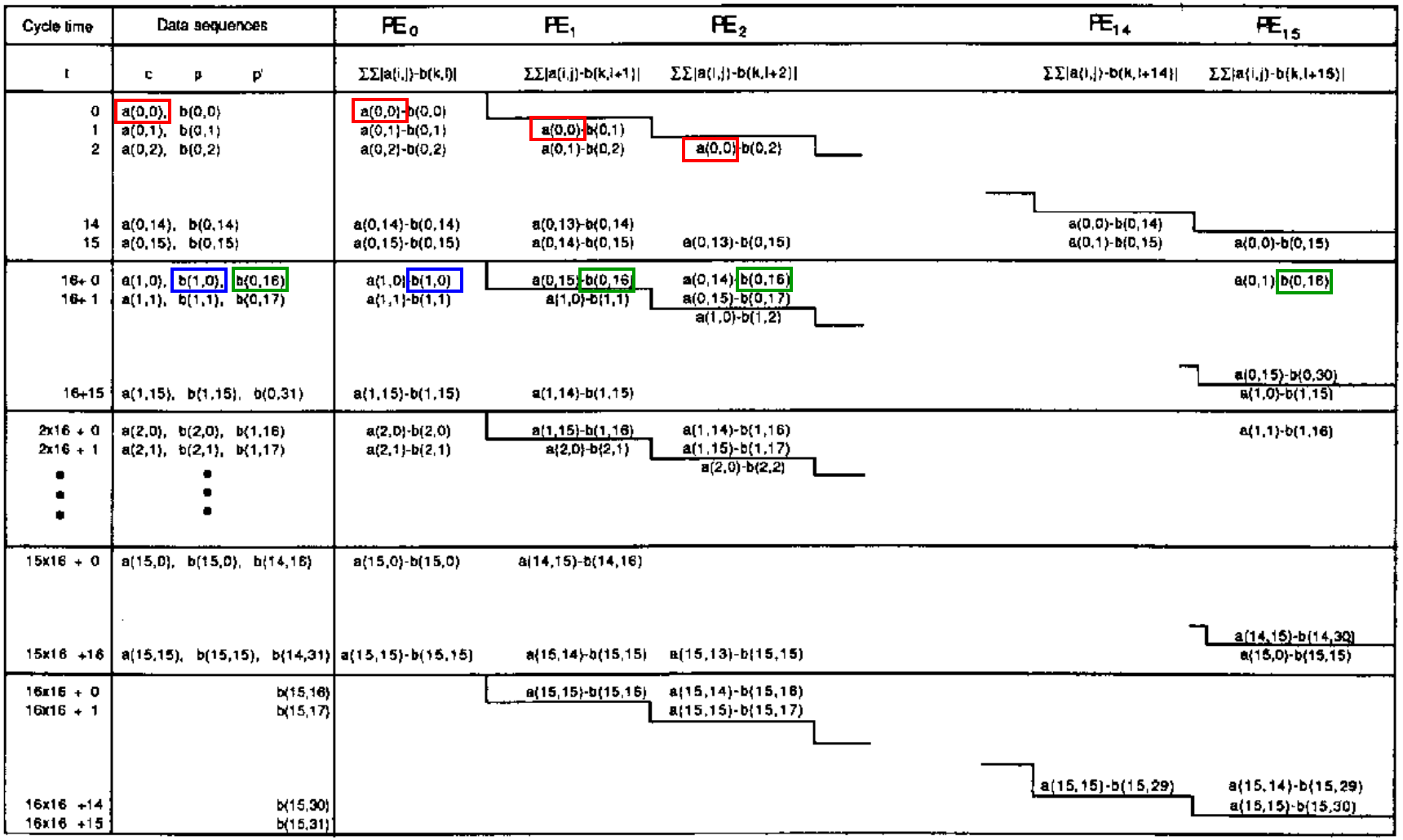
block size 16x16



Processing Element

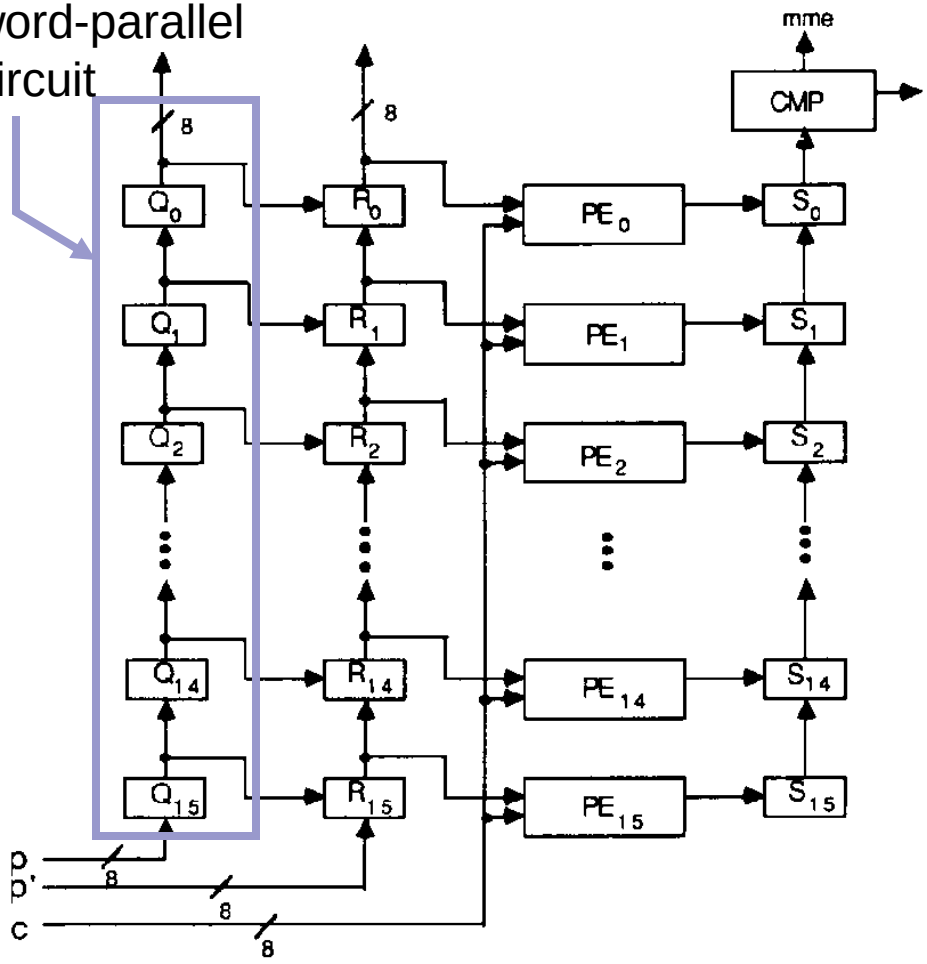
Accumulate the SAD of a candidate

Basic Data Flow



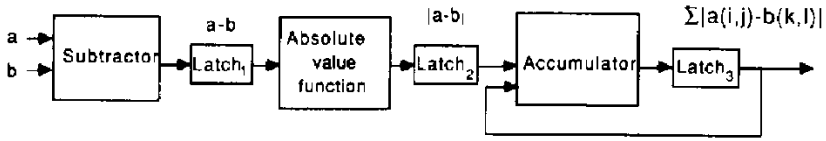
Broadcasting Current Frame

Word-serial to word-parallel circuit



search range [-8, +7]

block size 16x16



Processing Element

Accumulate the SAD of a candidate

Basic Data Flow

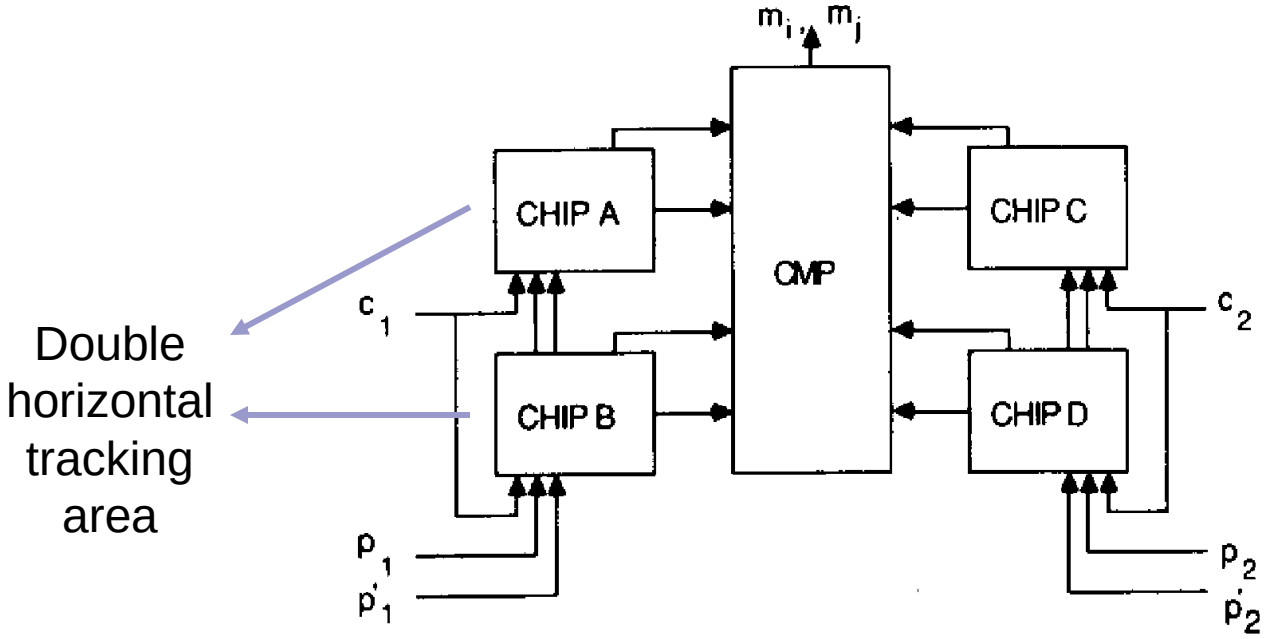
b(0,0)
b(0,1)
.
b(0,15)

Cycle time	Data seque			PE_0	PE_1		PE_{14}	PE_{15}
t	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	a(0,0)	b(0,16)	b(1,0)	a(0,0)-b(0,0)	a(0,0)-b(0,1)		a(0,0)-b(0,14)	a(0,0)-b(0,15)
0x16+1	a(0,1)	b(0,17)	b(1,1)	a(0,1)-b(0,1)	a(0,1)-b(0,2)		a(0,1)-b(0,15)	a(0,1)-b(0,16)
.	.	.	.	.	.	...	.	.
0x16+14	a(0,14)	b(0,30)	b(1,14)	a(0,14)-b(0,14)	a(0,14)-b(0,15)		a(0,14)-b(0,28)	a(0,14)-b(0,29)
0x16+15	a(0,15)	b(0,31)	b(1,15)	a(0,15)-b(0,15)	a(0,15)-b(0,16)		a(0,15)-b(0,29)	a(0,15)-b(0,30)
1x16+0	a(1,0)	b(1,16)	b(2,0)	a(1,0)-b(1,0)	a(1,0)-b(1,1)		a(1,0)-b(1,14)	a(1,0)-b(1,15)
1x16+1	a(1,1)	b(1,17)	b(2,1)	a(1,1)-b(1,1)	a(1,1)-b(1,2)		a(1,1)-b(1,15)	a(1,1)-b(1,16)
.	.	.	.	.	.	...	.	.
1x16+14	a(1,14)	b(1,30)	b(2,14)	a(1,14)-b(1,14)	a(1,14)-b(1,15)		a(1,14)-b(1,28)	a(1,14)-b(1,29)
1x16+15	a(1,15)	b(1,31)	b(2,15)	a(1,15)-b(1,15)	a(1,15)-b(1,16)		a(1,15)-b(1,29)	a(1,15)-b(1,30)
14x16+0	a(14,0)	b(14,16)	b(15,0)	a(14,0)-b(14,0)	a(14,0)-b(14,1)		a(14,0)-b(14,14)	a(14,0)-b(14,15)
14x16+1	a(14,1)	b(14,17)	b(15,1)	a(14,1)-b(14,1)	a(14,1)-b(14,2)		a(14,1)-b(14,15)	a(14,1)-b(14,16)
.	.	.	.	.	.	...	.	.
14x16+14	a(14,14)	b(14,30)	b(15,14)	a(14,14)-b(14,14)	a(14,14)-b(14,15)		a(14,14)-b(14,28)	a(14,14)-b(14,29)
14x16+15	a(14,15)	b(14,31)	b(15,15)	a(14,15)-b(14,15)	a(14,15)-b(14,16)		a(14,15)-b(14,29)	a(14,15)-b(14,30)
15x16+0	a(15,0)	b(15,16)	b(0,0)	a(15,0)-b(15,0)	a(15,0)-b(15,1)		a(15,0)-b(15,14)	a(15,0)-b(15,15)
15x16+1	a(15,1)	b(15,17)	b(0,1)	a(15,1)-b(15,1)	a(15,1)-b(15,2)		a(15,1)-b(15,15)	a(15,1)-b(15,16)
.	.	.	.	.	.	...	.	.
15x16+14	a(15,14)	b(15,30)	b(0,14)	a(15,14)-b(15,14)	a(15,14)-b(15,15)		a(15,14)-b(15,28)	a(15,14)-b(15,29)
15x16+15	a(15,15)	b(15,31)	b(0,15)	a(15,15)-b(15,15)	a(15,15)-b(15,16)		a(15,15)-b(15,29)	a(15,15)-b(15,30)

Load reference frame data from Q_0-Q_{15} to R_0-R_{15} in parallel for every 16 cycles

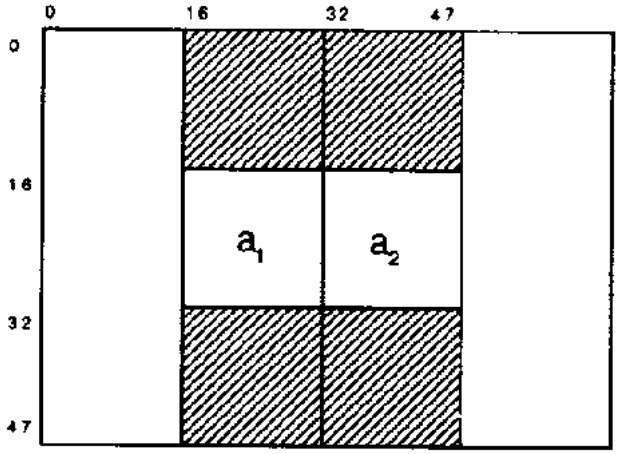
Cascaded Chip Design

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

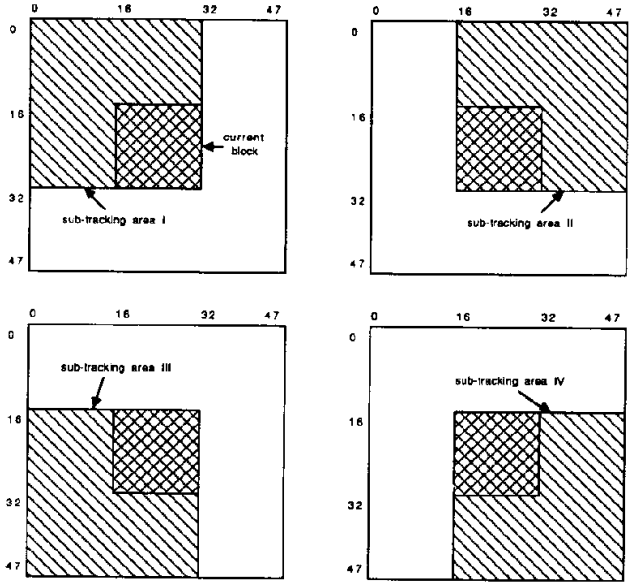


Overlapped Search Area

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



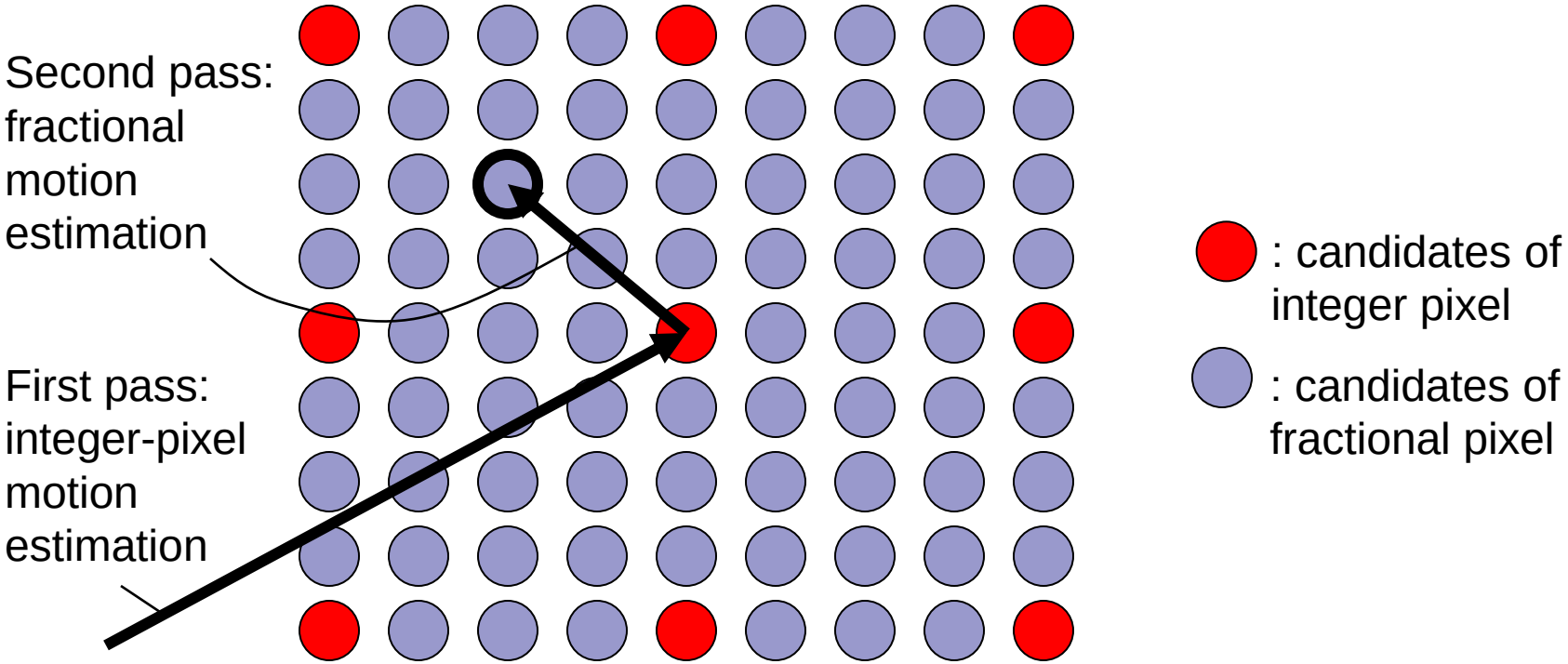
Overlapped tracking area of two adjacent blocks



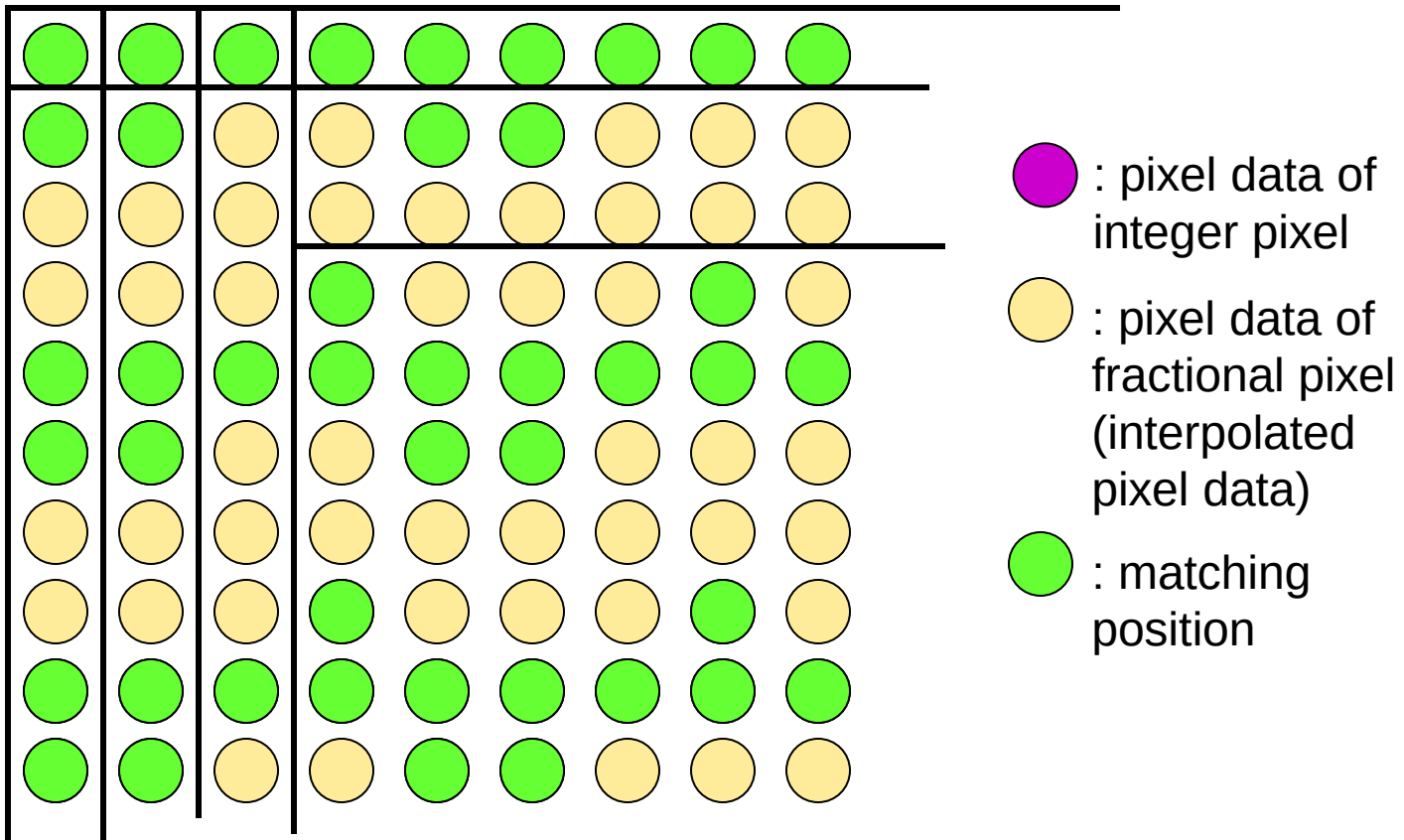
Overlapped sub-tracking area

Motion Estimation with Fractional Precision (1/2)

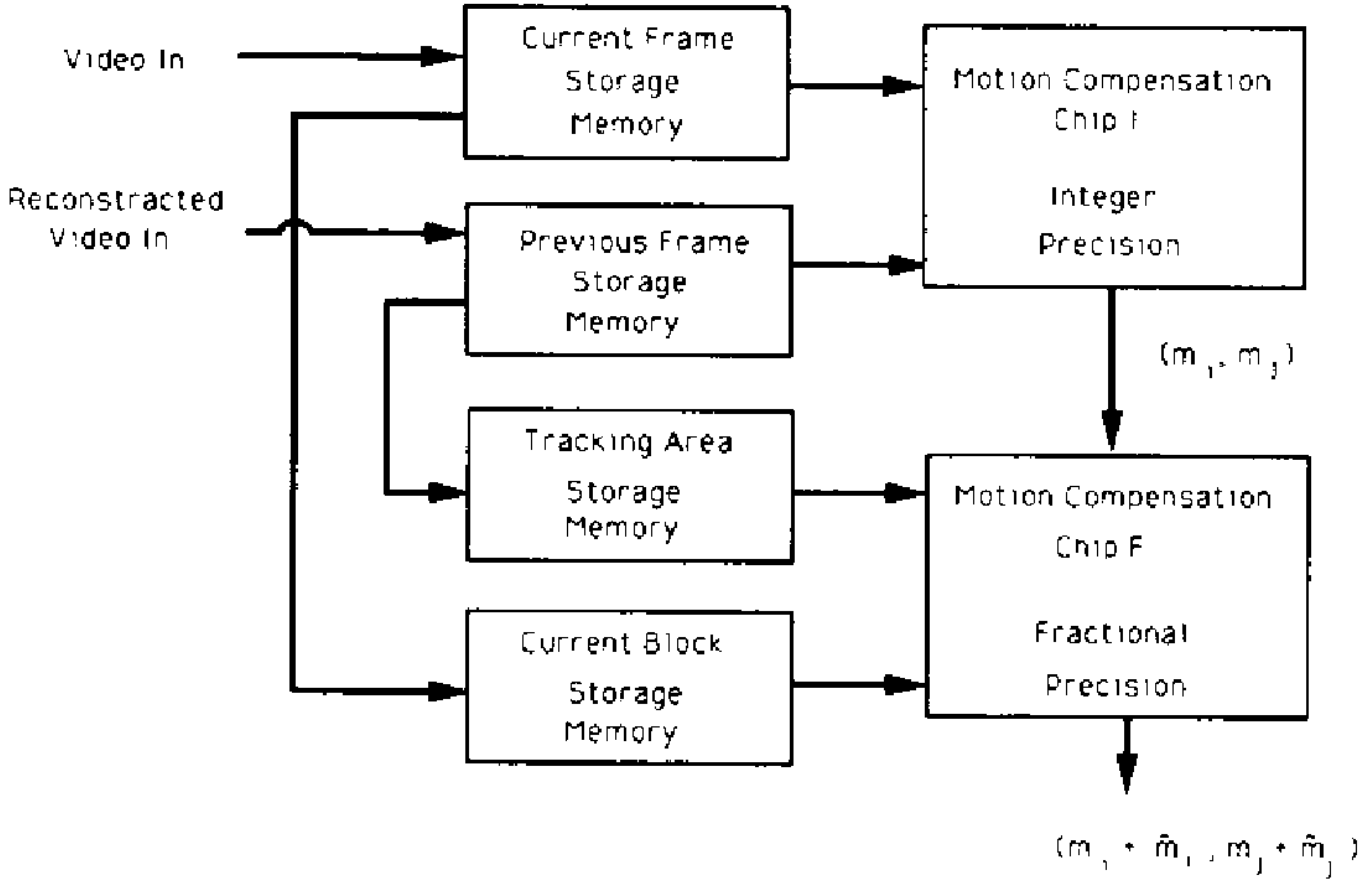
- Quarter-pel precision



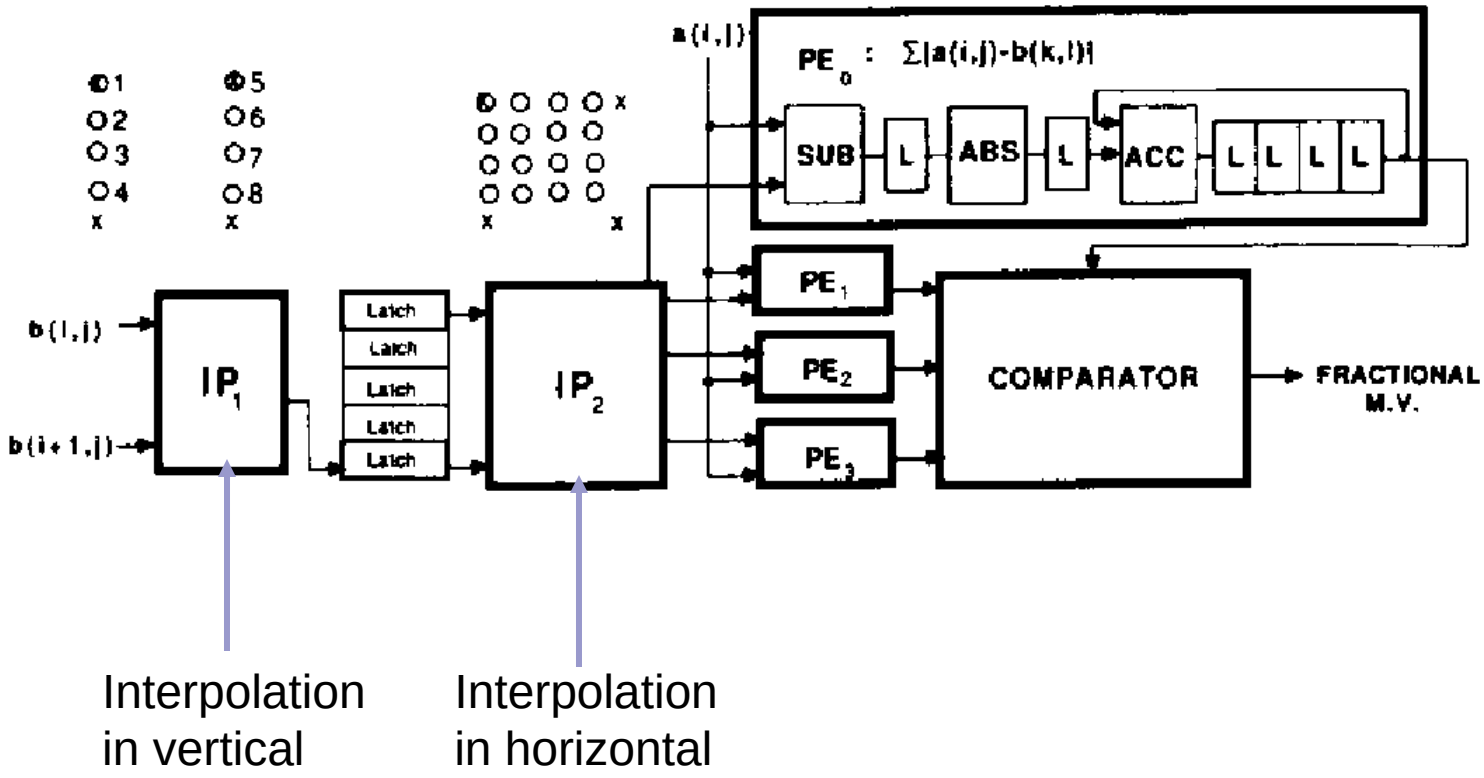
Motion Estimation with Fractional Precision (2/2)



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 N_v − 1
    do h = 0 to N_h − 1
        MV(h,v) = (0,0)
        D_min(h,v) = ∞
        do m = −p to p − 1
            don = −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 D_min(h,v) > MAD(m,n)
                    D_min(h,v) = MAD(m,n)
                    MV(h,v) = (m,n)
                endif
            enddo n
        enddo m
    enddo h
enddo v
```

$$\begin{cases} b = vN_h + h & 0 \le b < N_{hv} = N_h N_v \\ h = b \bmod N_h \\ v = \left\lfloor \dfrac{b}{N_h} \right\rfloor \end{cases}$$

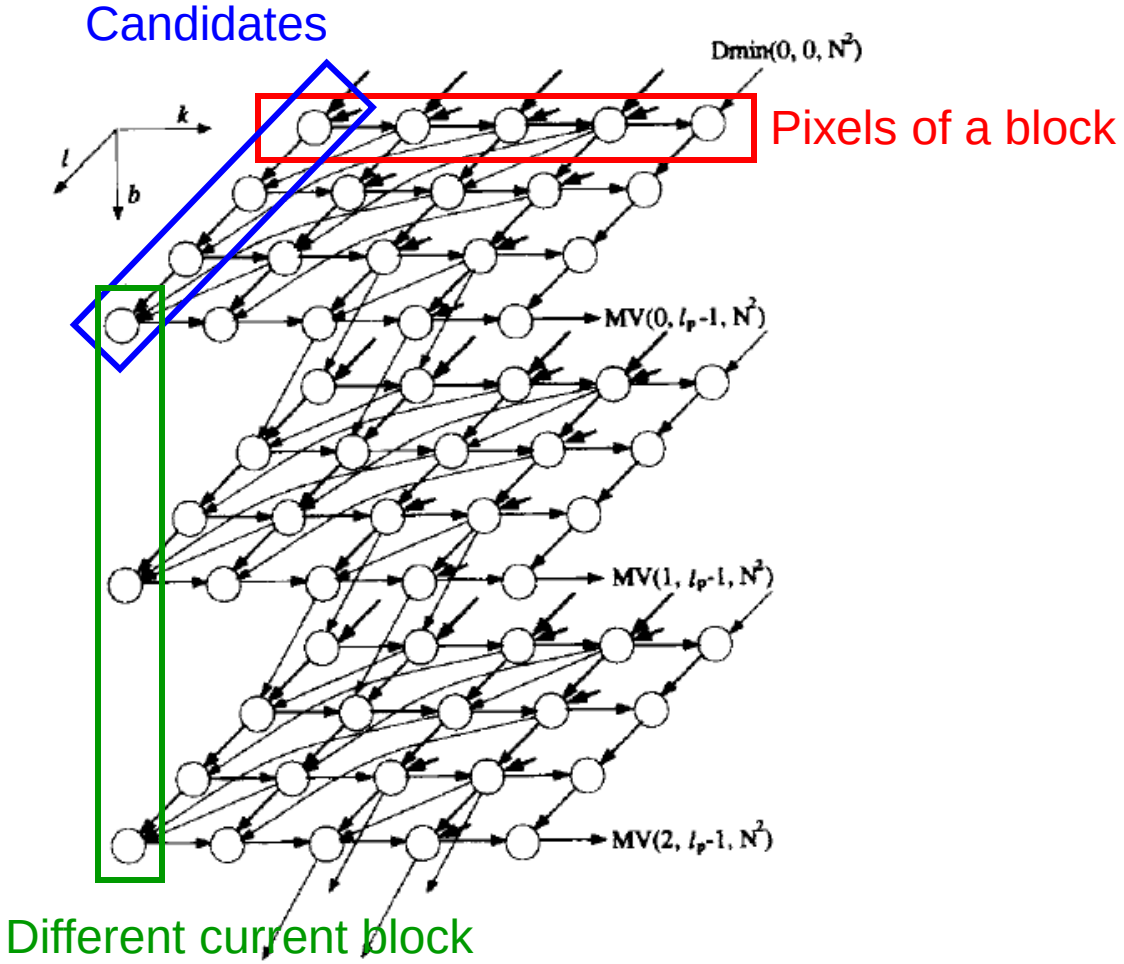
$$\begin{cases} l = 2p(m+p) + n + p & 0 \le l < l_p = (2p)^2 \\ m = \left\lfloor \dfrac{l}{2p} \right\rfloor - p \\ n = l \bmod 2p - p \end{cases}$$

$$\begin{cases} k = iN + j & 0 \le k < N^2 \\ i = \left\lfloor \dfrac{k}{N} \right\rfloor \\ j = k \bmod N. \end{cases}$$

```
do b = 0 to N_hv − 1
    MV(b) = 0
    D_min(b) = ∞
    do l = 0 to l_p − 1
        MAD(l) = 0
        do k = 0 to N^2 − 1
            MAD(l) = MAD(l) + |x_s(k) − y_s(k + l)|
        enddo k
        if D_min(b) > MAD(l)
            D_min(b) = MAD(l)
            MV(b) = l
        endif
    enddo l
enddo b
```

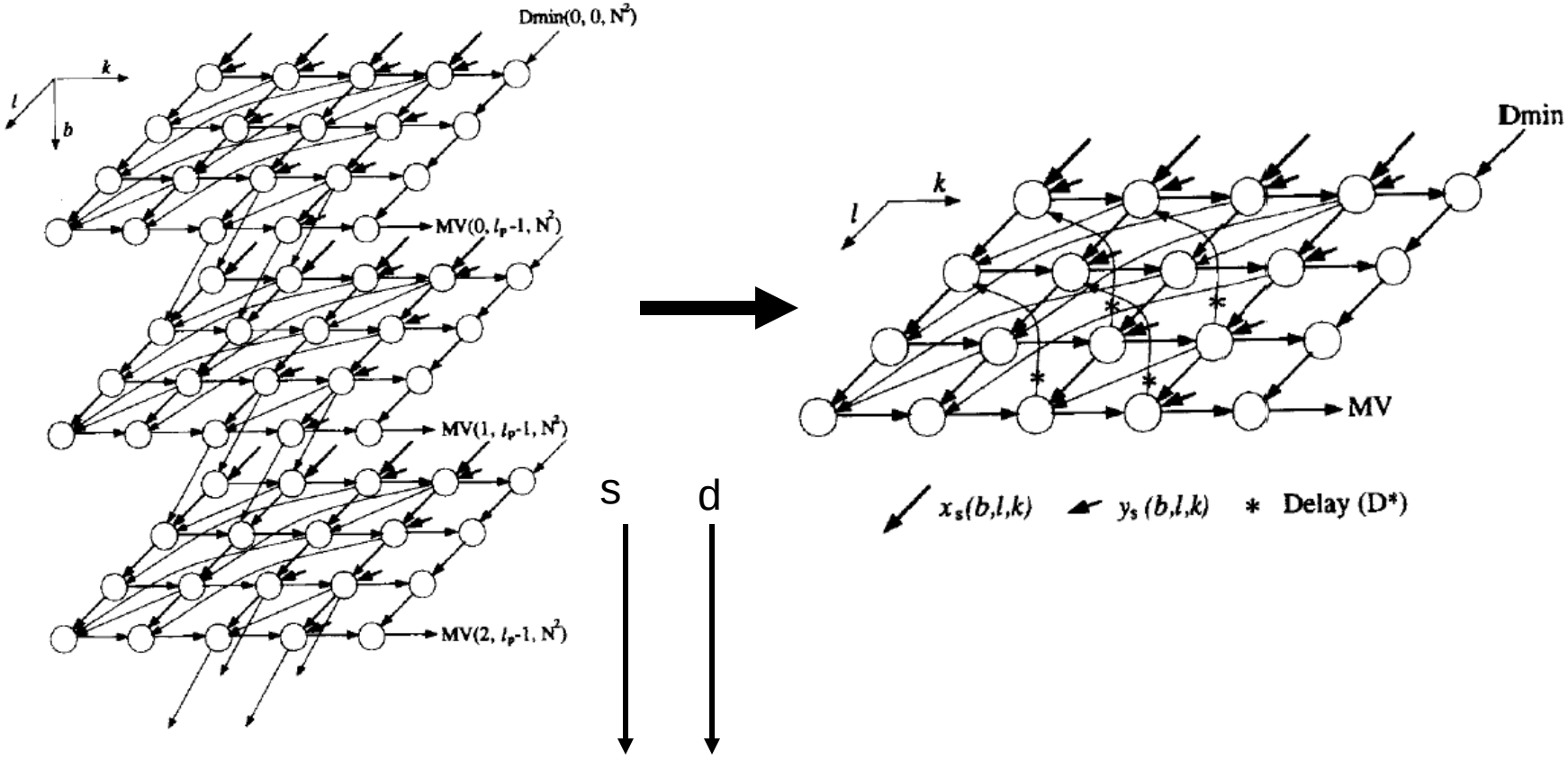
b: block index
l: search candidate index
k: pixel index

3-D DG of Motion Estimation

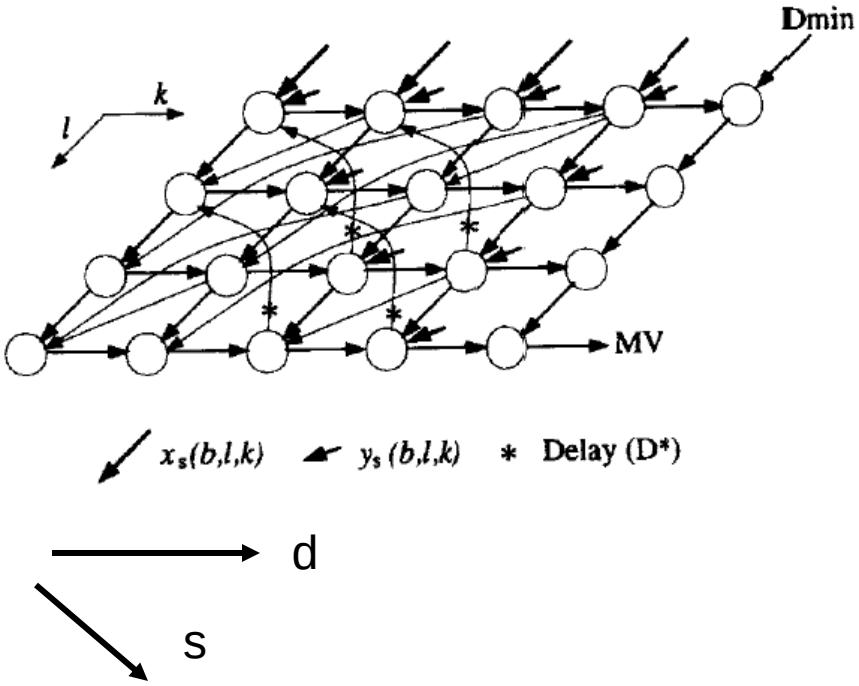


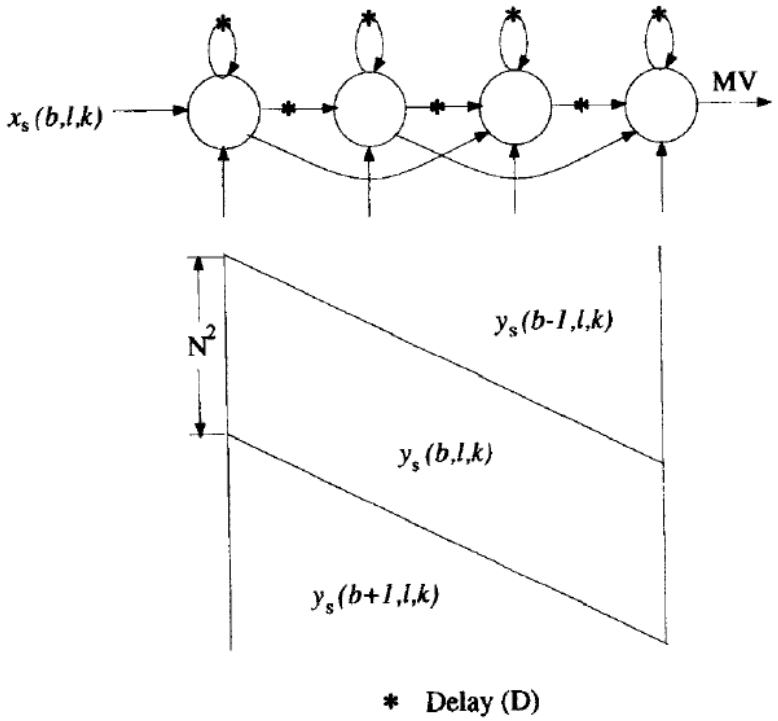
- Three axis
  - b
  - l
  - k
- Data reuse of reference frame is also considered

3-D DG to 2-D SFG



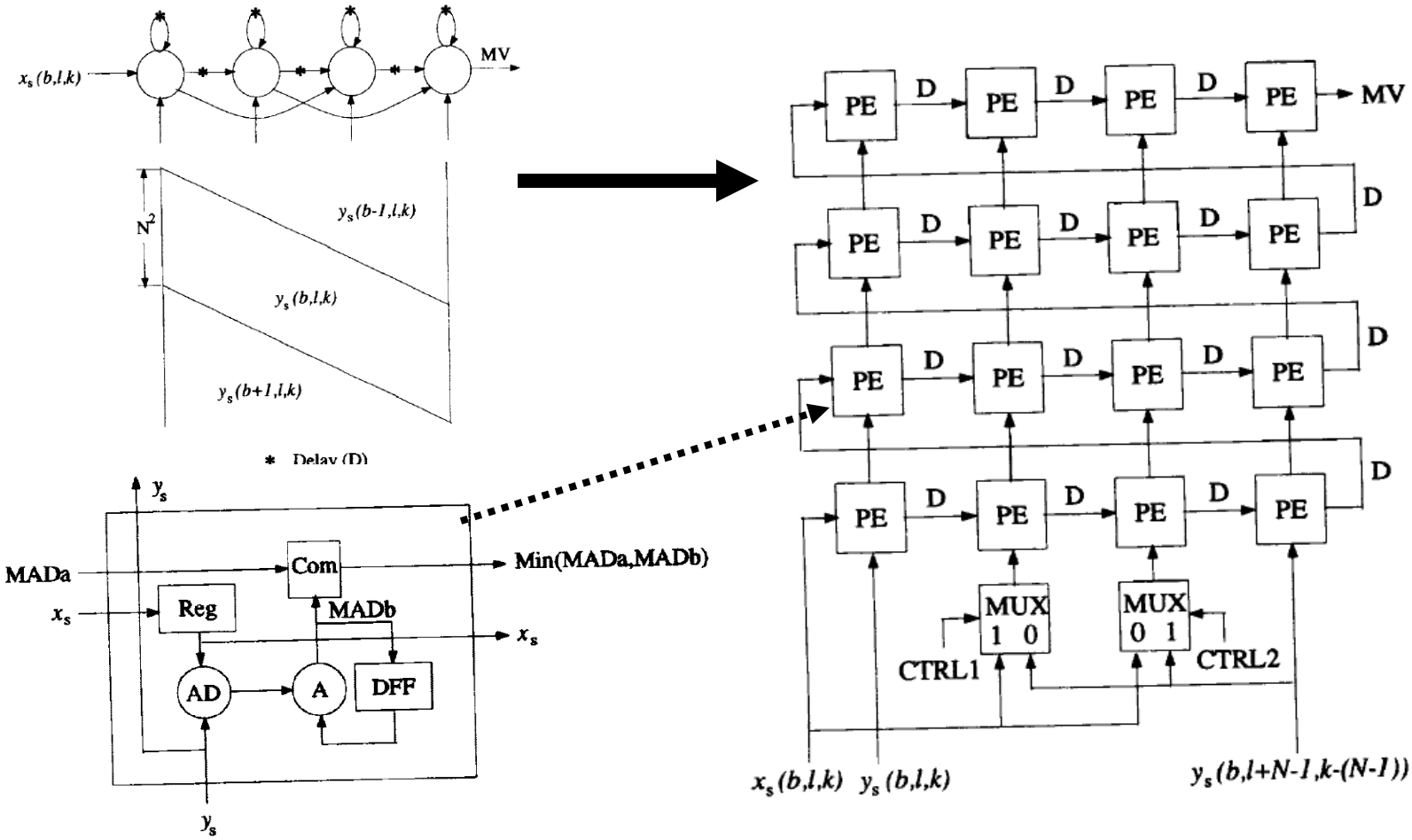
2-D SFG to 1-D SFG



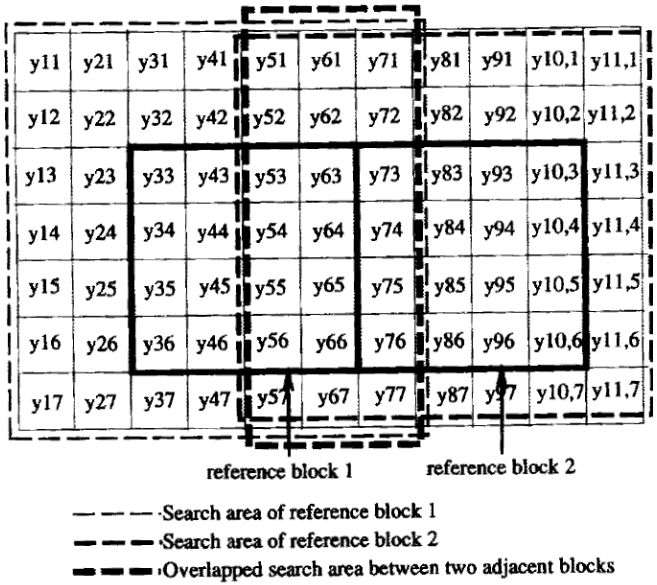


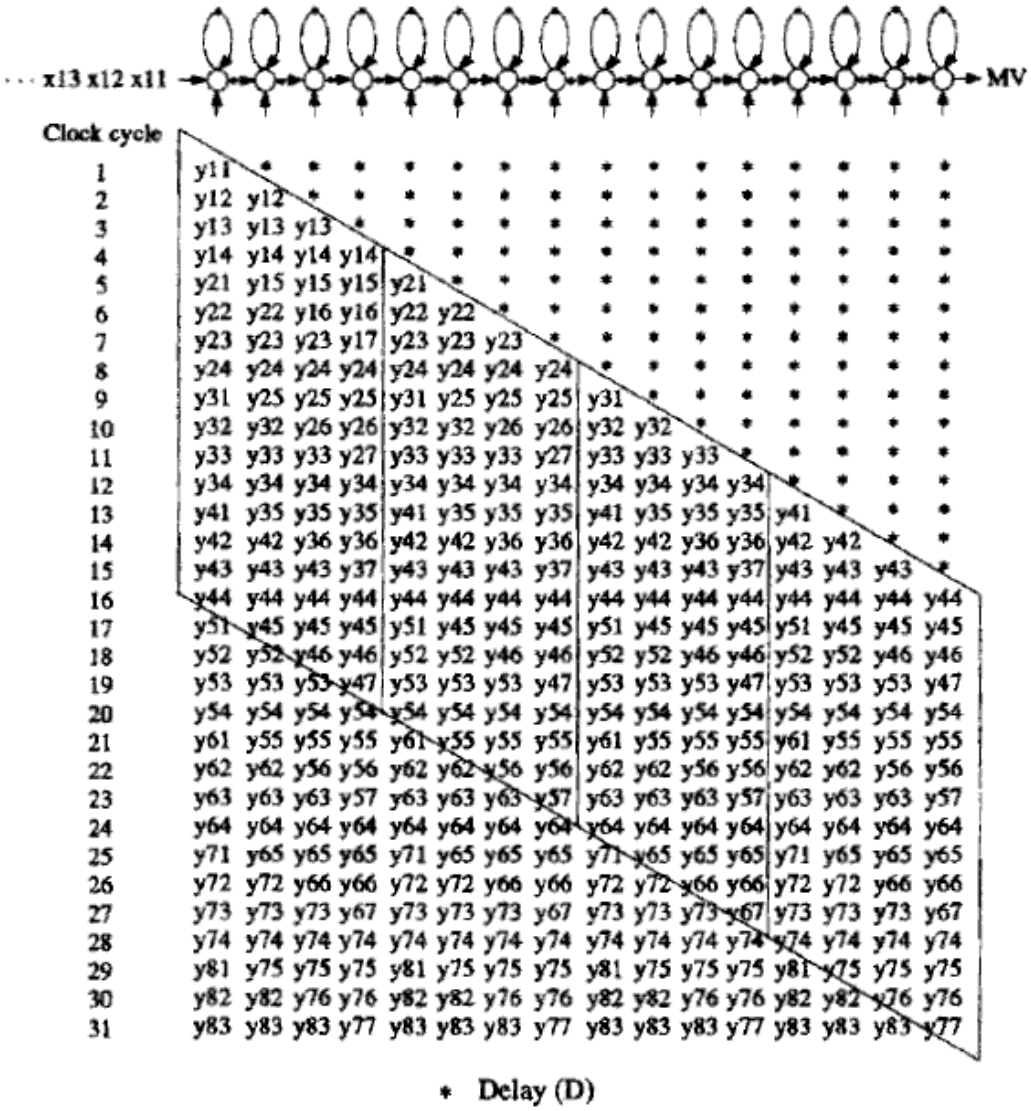
$D^*=N^2D$

1-D SFG to 2-D Mesh



Scheduling


reference block 1
reference block 2
Search area of reference block 1
Search area of reference block 2
Overlapped search area between two adjacent blocks


x13 x12 x11
MV
Clock cycle
Delay (D)

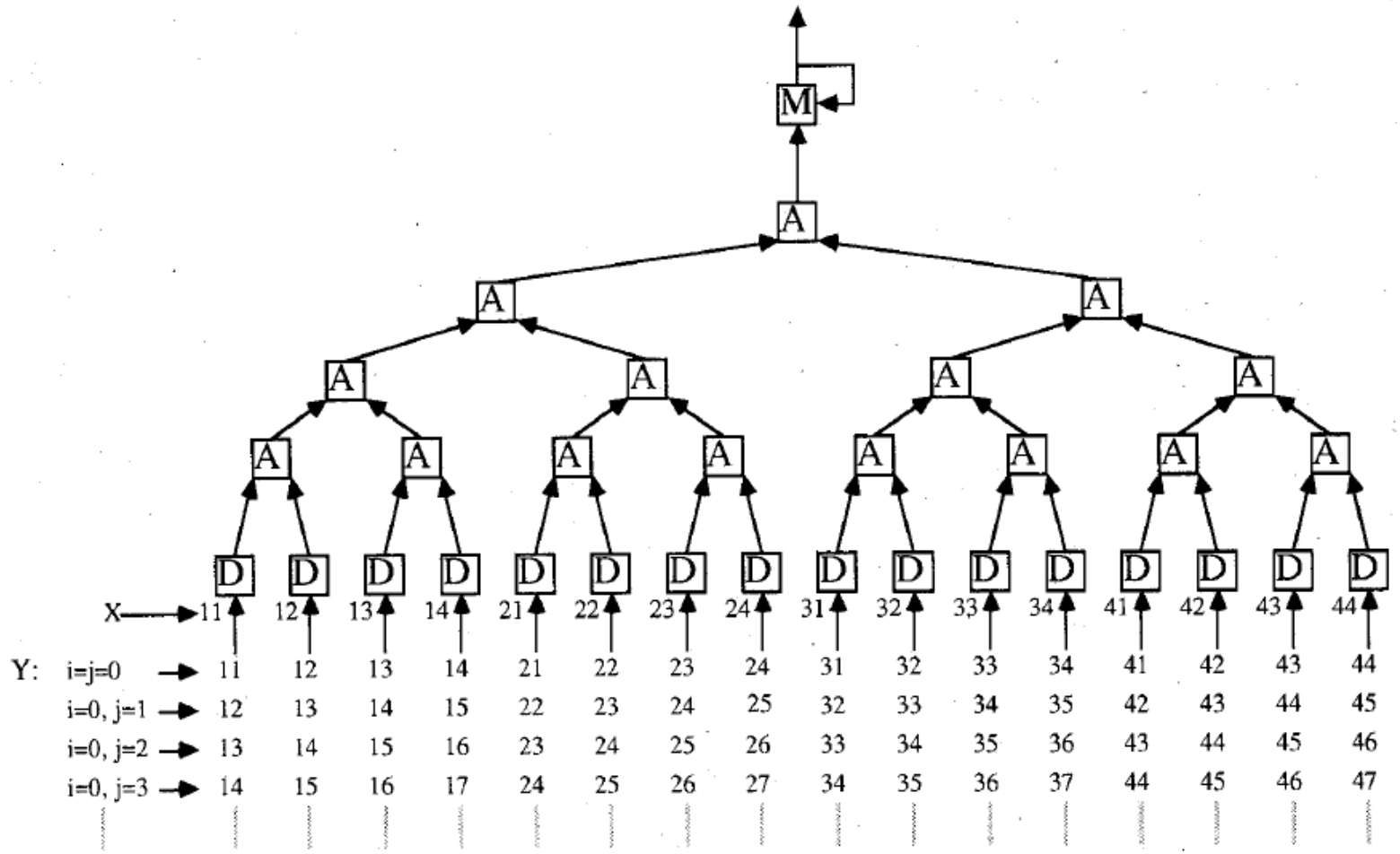
Tree-Based Architecture

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

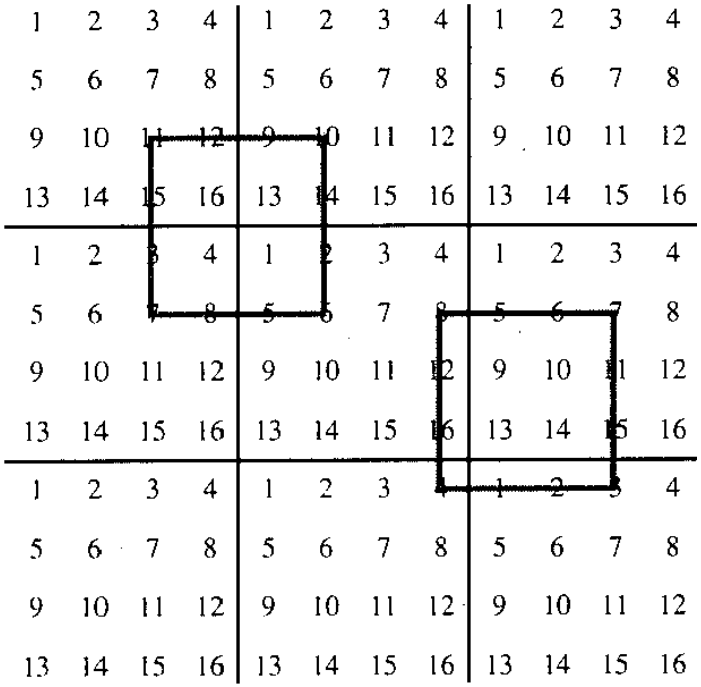
Features

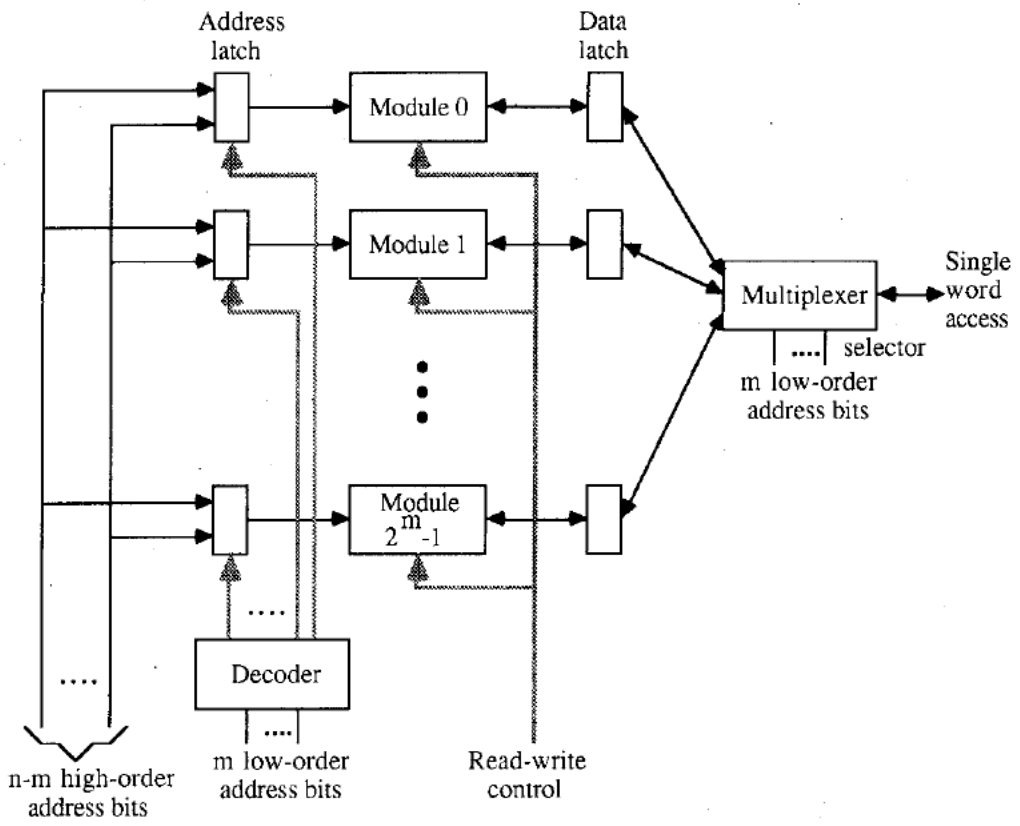
- High throughput
- Parallel computing
- Short data path length, low-latency delay
- Independent data flow computation that benefits the **irregular block matching** especially for the realization of **three-step hierarchy search algorithm**

Tree-Based Architecture

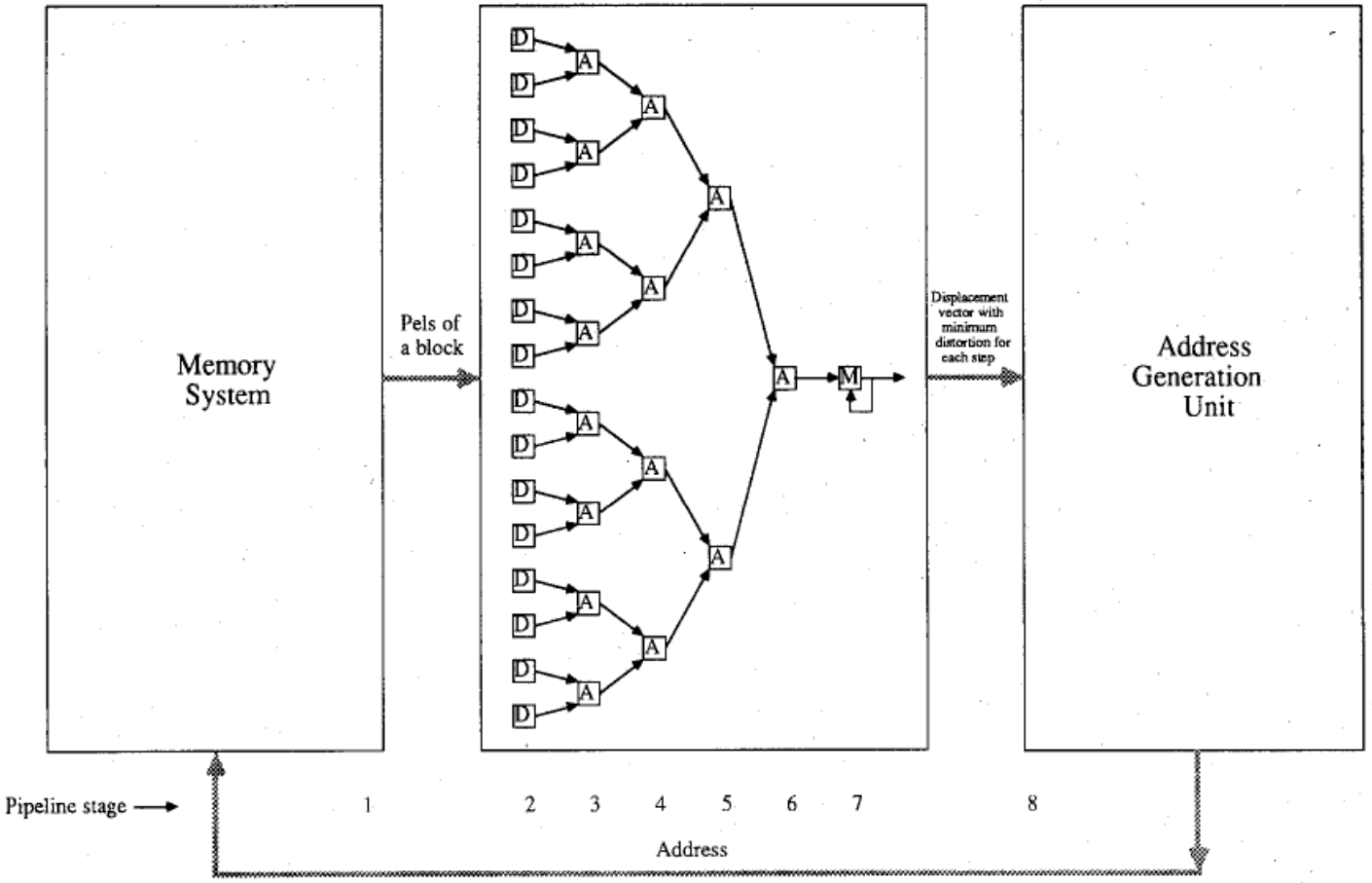


Memory Interleaving



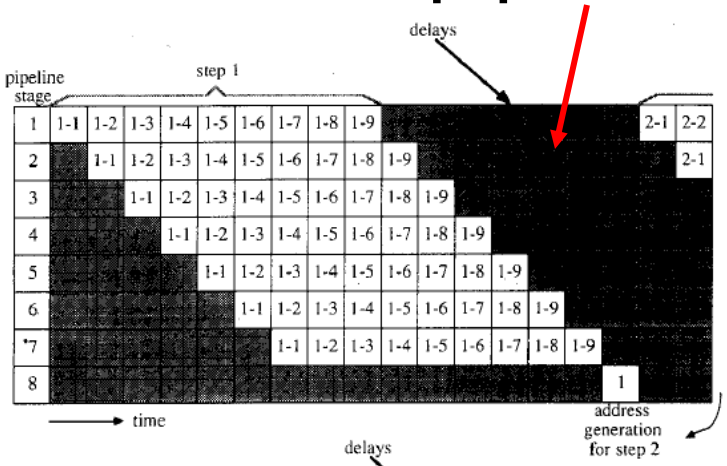


Whole Architecture

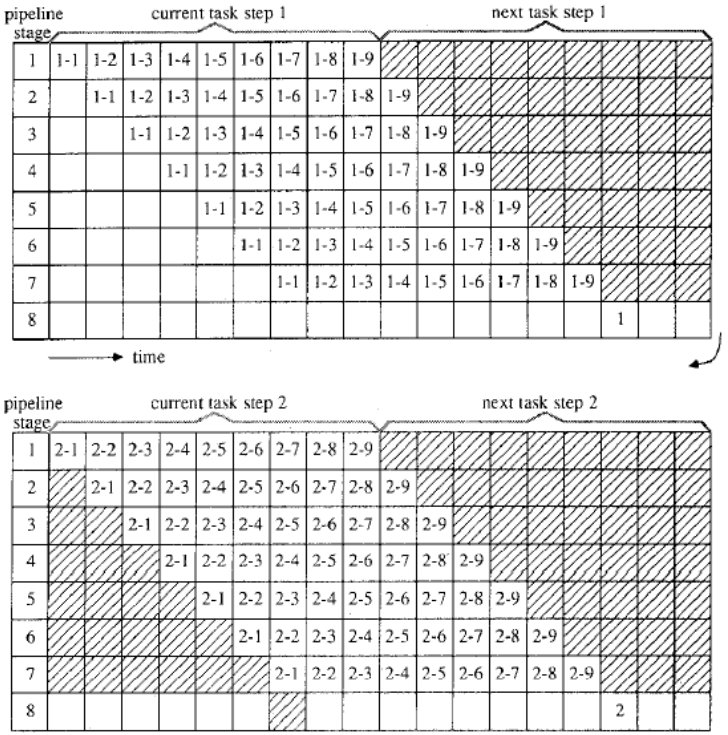


Pipeline Interleaving

- Remove pipeline bubbles (step hazards)

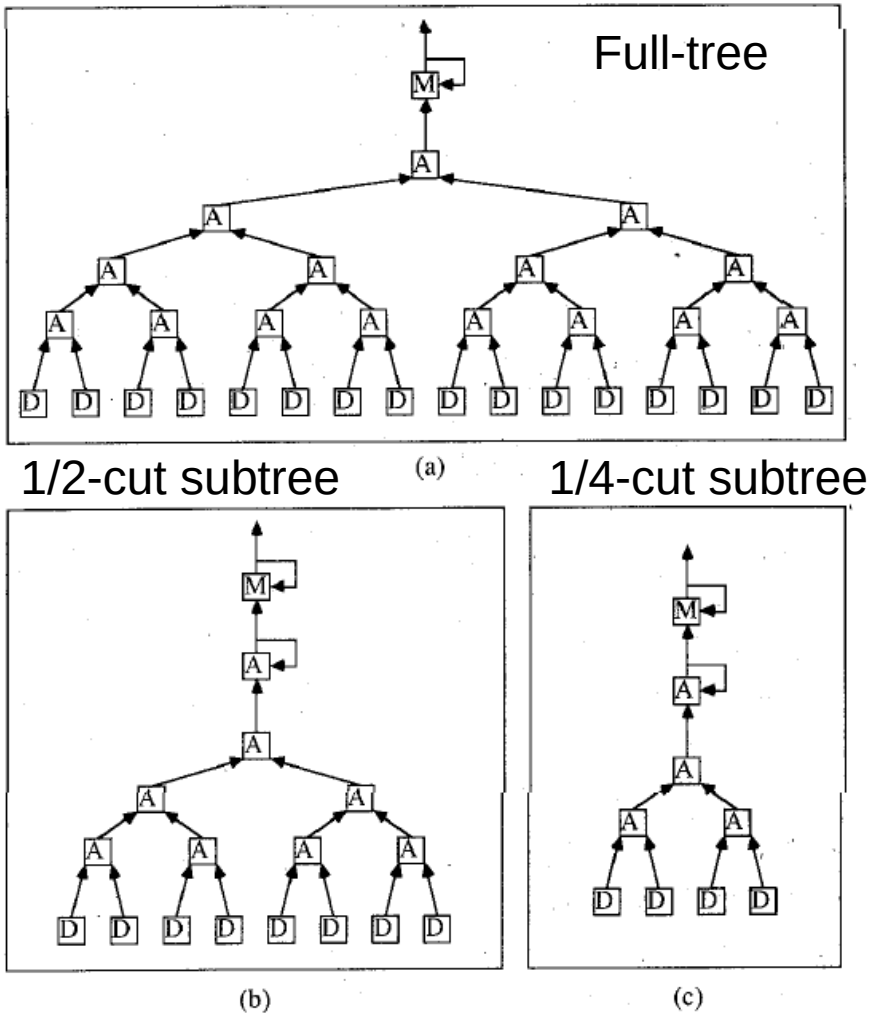


Interleave with adjacent block



Tree-Cut Technique

- Applying folding to
  - Processing elements
  - Memory



Comparison

TABLE I
TREE-CUT TECHNIQUE FOR HARDWARE REDUCTION

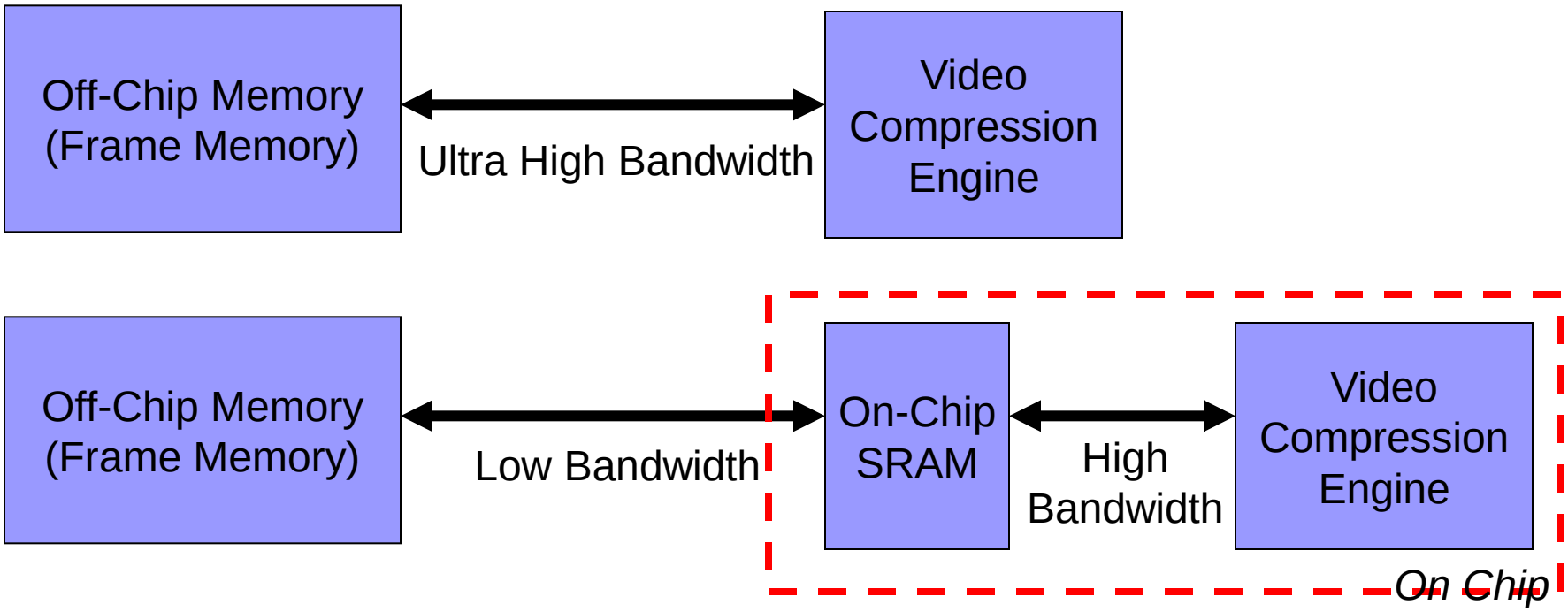
Configuration	#ADDER	N = 8	N = 16	Time Instances Required (FBMA)	N = 8 d = 3	N = 16 d = 7
Systolic mesh	$2N^2 + N + 1$	137	529	$(2d + 1)(N + 2d)$	98	450
Full tree	$2N^2$	128	512	$(2d + 1)^2$	49	225
$\frac{1}{2}$ cut	$2N^2/2 + 1$	65	257	$2(2d + 1)^2$	98	450
$\frac{1}{4}$ cut	$2N^2/4 + 1$	33	129	$4(2d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

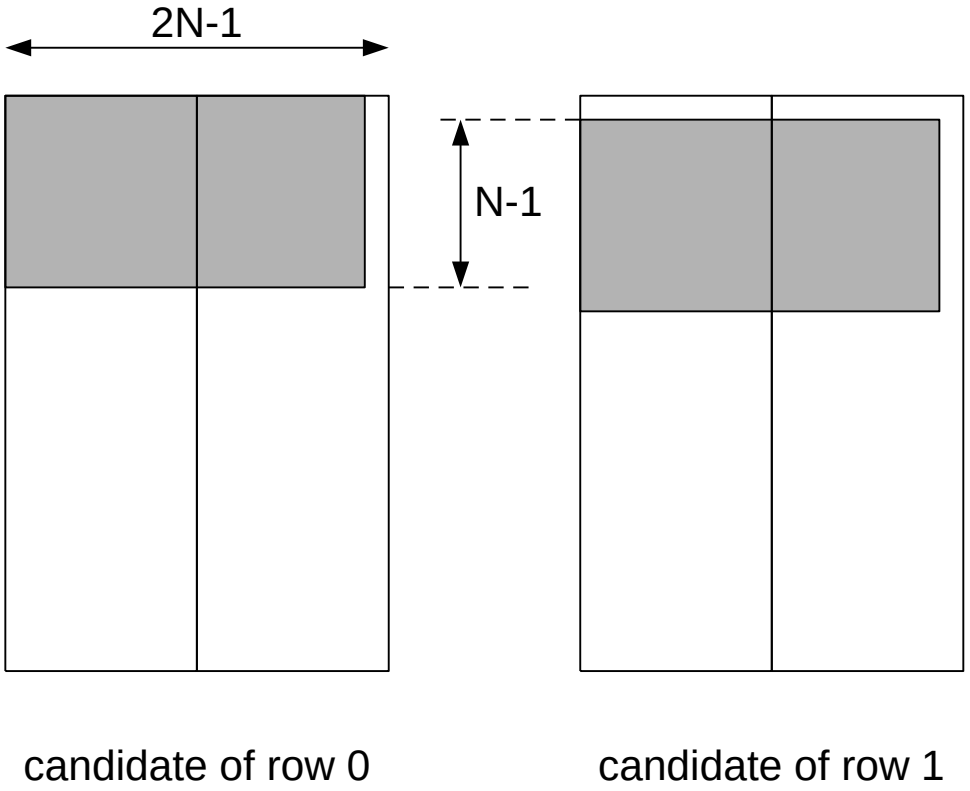


Illustration of Scheme B

- Data reuse between adjacent columns of candidates in a block

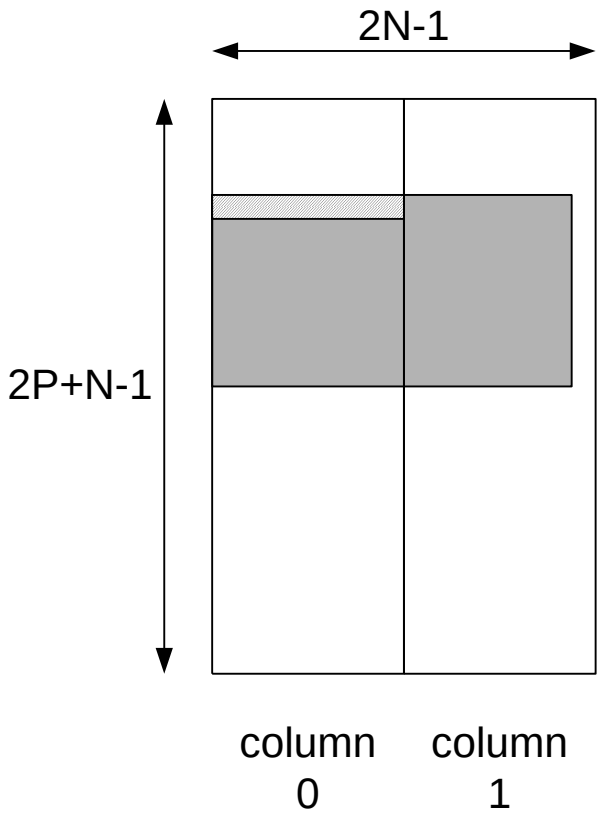


Illustration of Scheme C

- Data reuse between adjacent blocks in one row of block

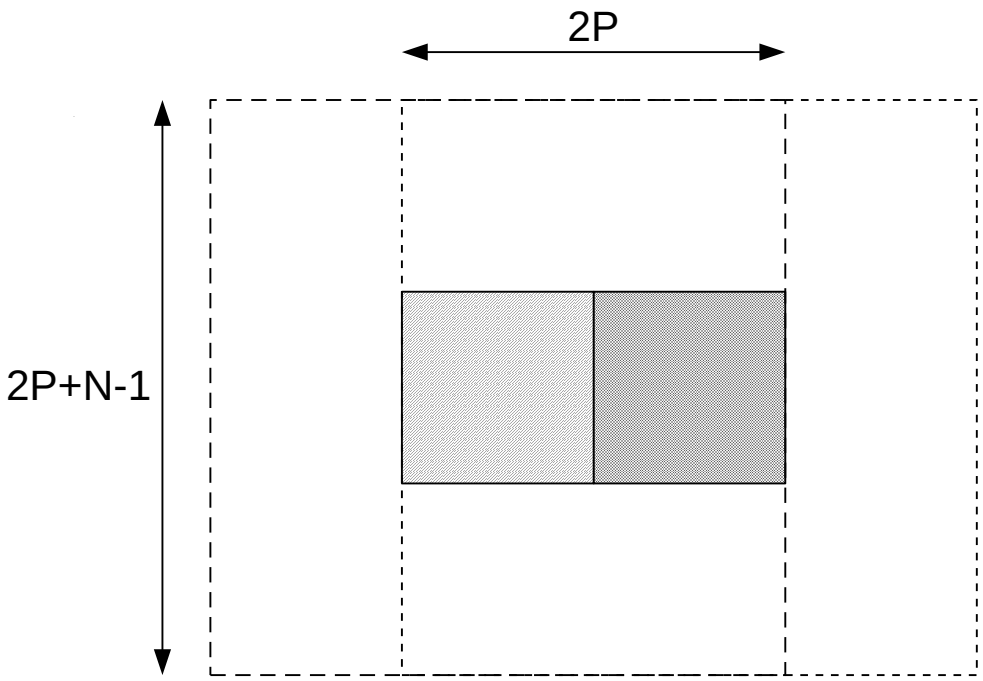
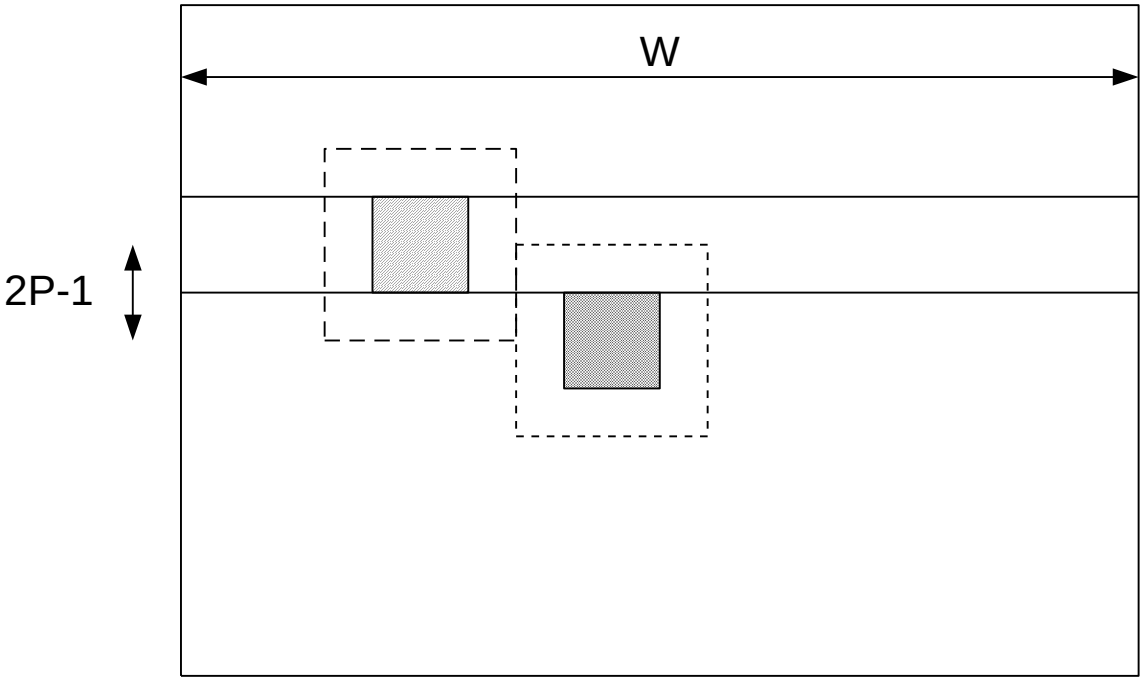


Illustration of Scheme D

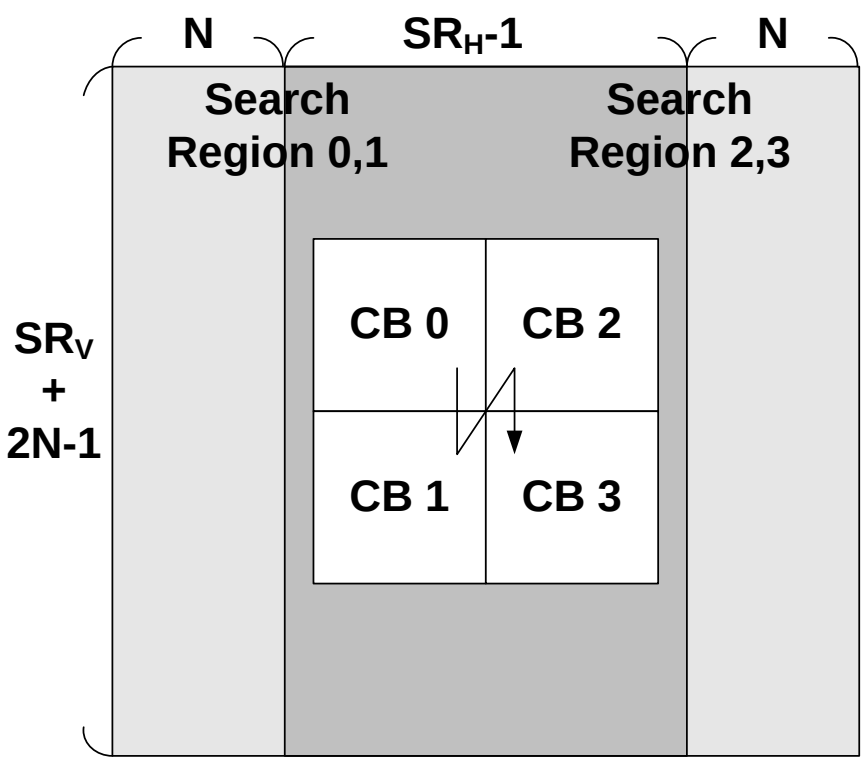
- Data reuse between different rows of block



Comparison of Different Schemes of Search Area Data Reuse

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

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

- On-chip memory size (SRB)

Fast-Algorithm-Based Architecture

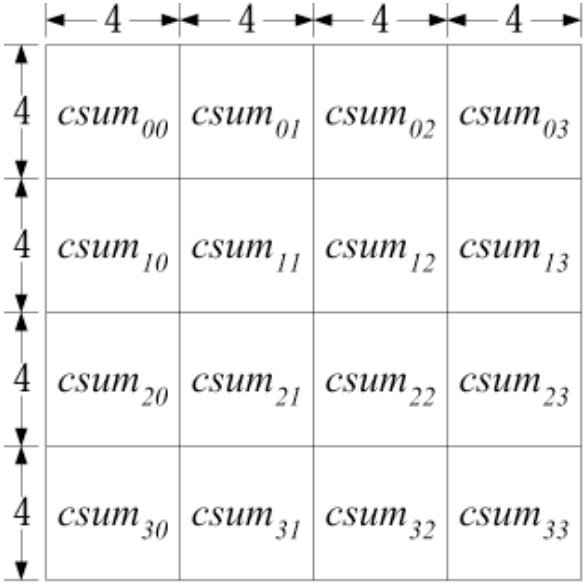
Y.-W. Huang, S.-Y. Chien, B.-Y. Hsieh, and L.-G. Chen, “An efficient and low power architecture design for motion estimation using global elimination algorithm,” *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2002

Multi-Level SEA (MSEA)

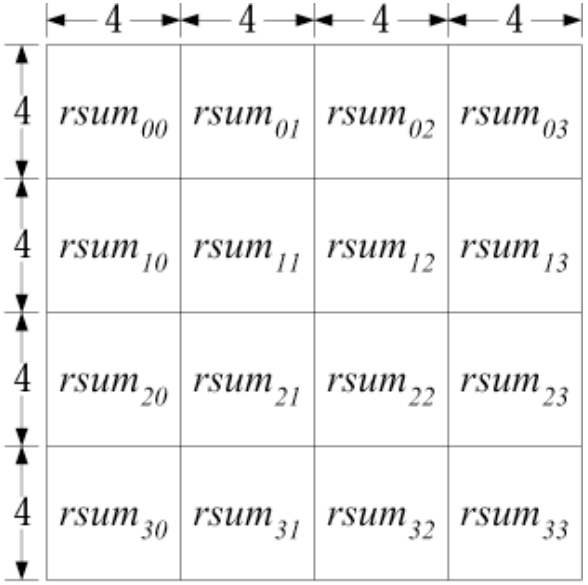
- Convert the sea value to **msea** value.
- Split the 16x16 macro-block into sub-blocks.
- **SAD** $\geqq$ **msea** $\geqq$ **sea**, which means MSEA can skip more unnecessary SAD calculations than SEA under the same scan order.
- However, the computation of msea value is heavier than that of sea value.

Example of MSEA at Level = 3

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



16 x 16 current block



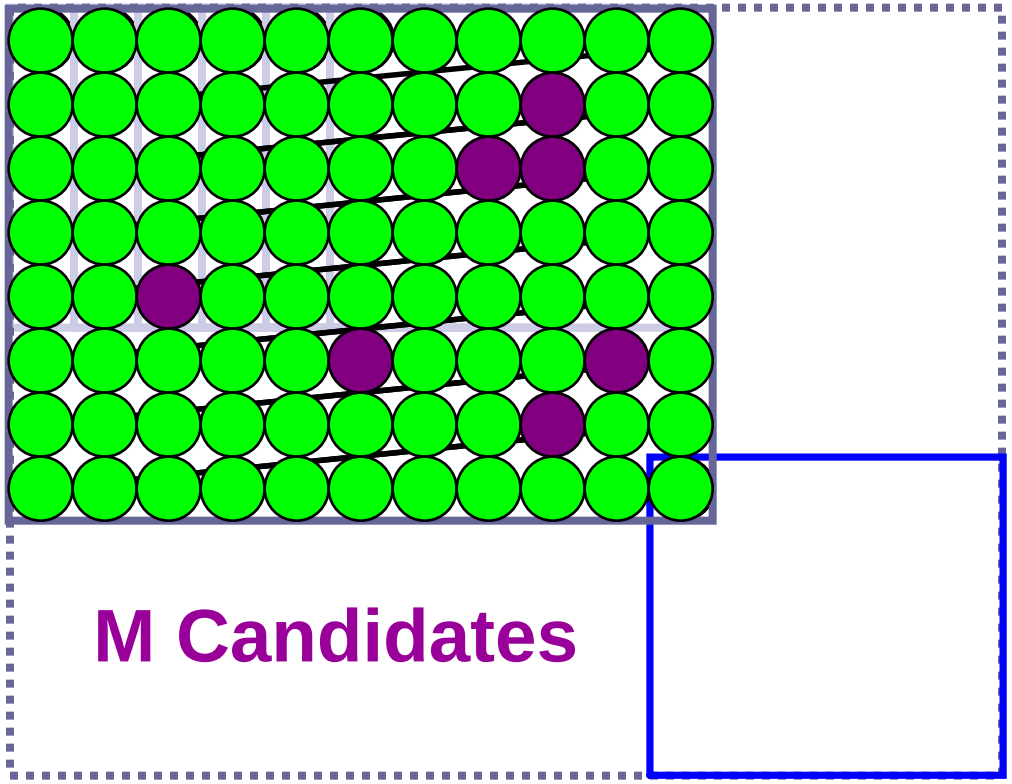
16 x 16 candidate block

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

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

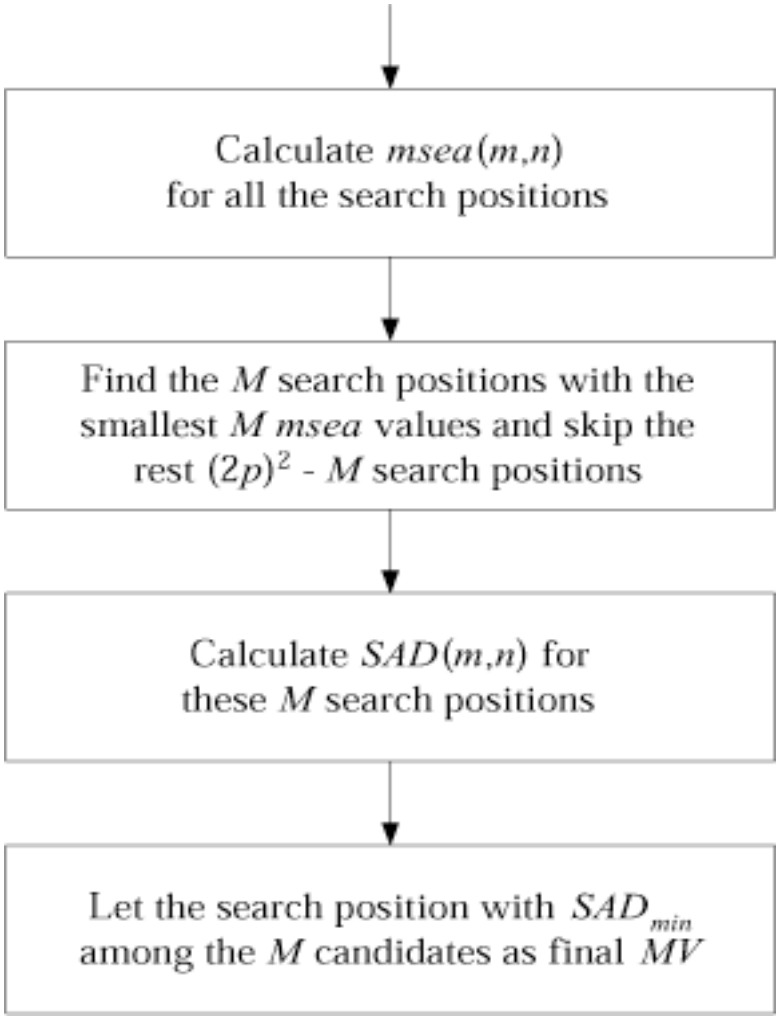
$$msea(m,n) = |csum_{00} - rsum_{00}| + |csum_{01} - rsum_{01}| + \ldots\ldots + |csum_{33} - rsum_{33}|$$

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

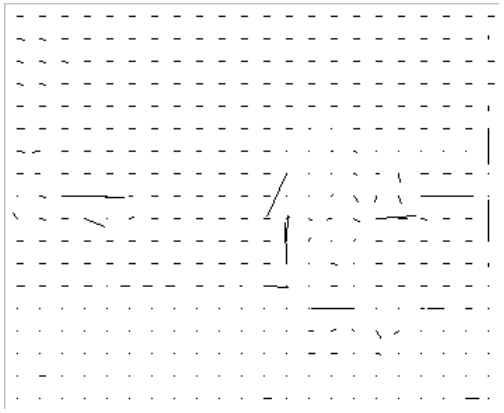
Motion Compensated Subjective View

Previous Frame

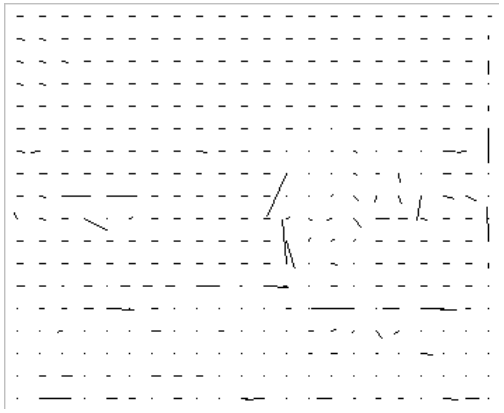
Current Frame

Compensated Frame **MV Plot**

FSBMA



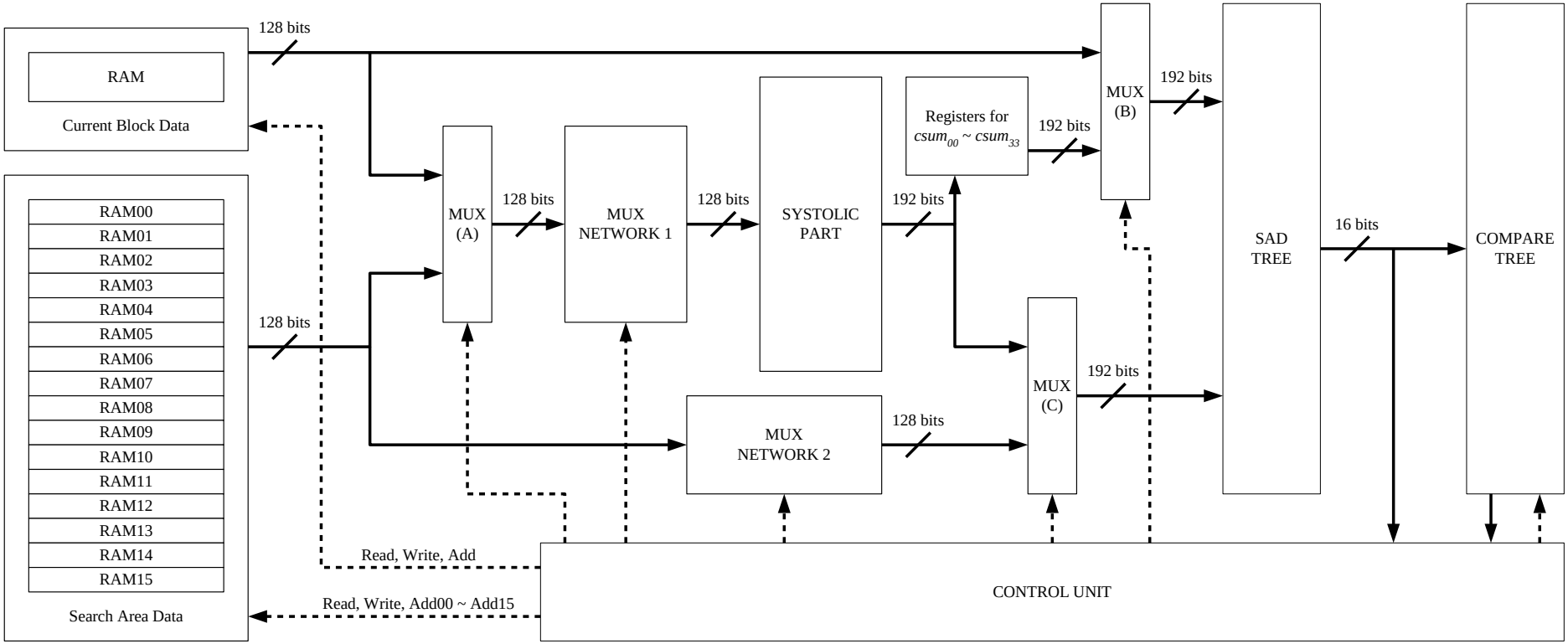
GEA



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

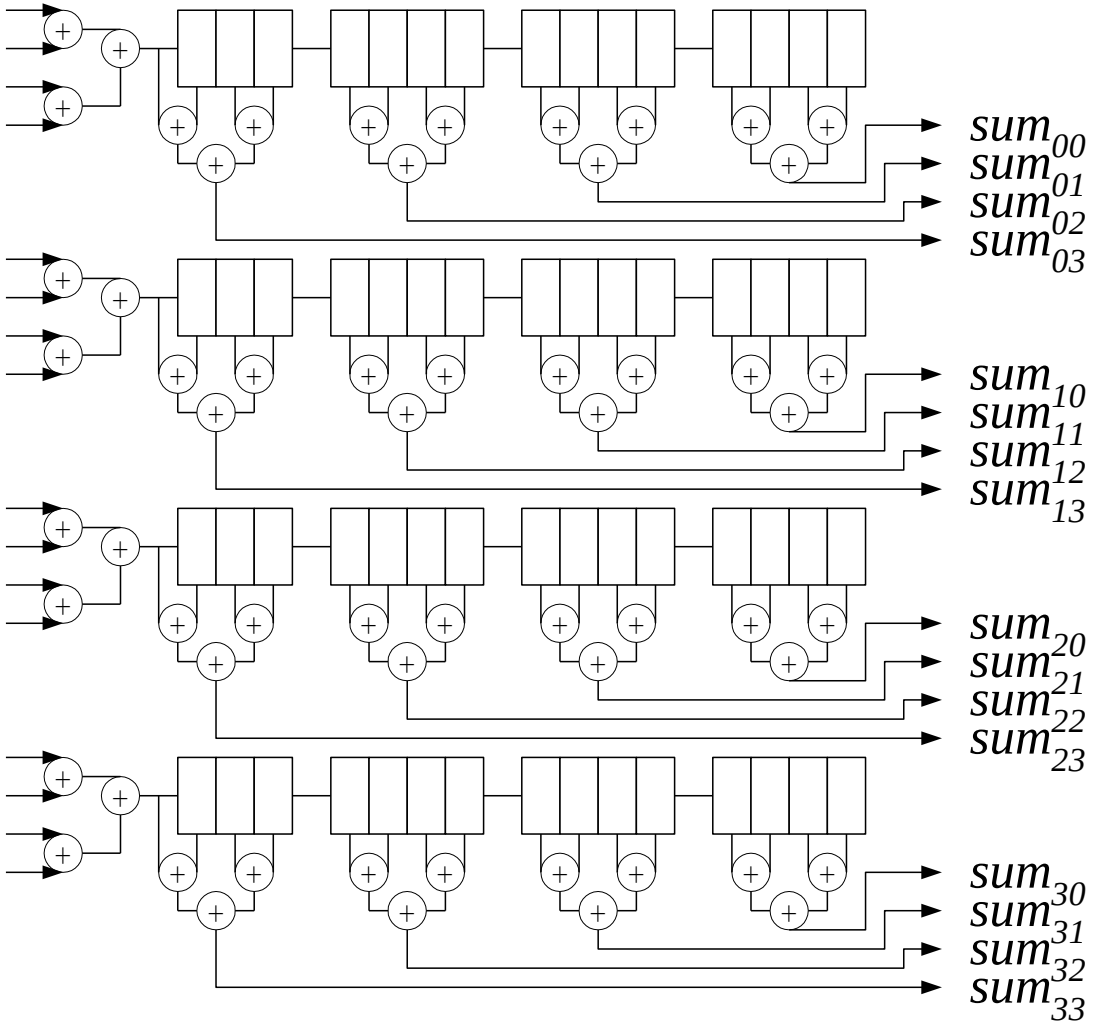
VLSI Architecture Design for GEA



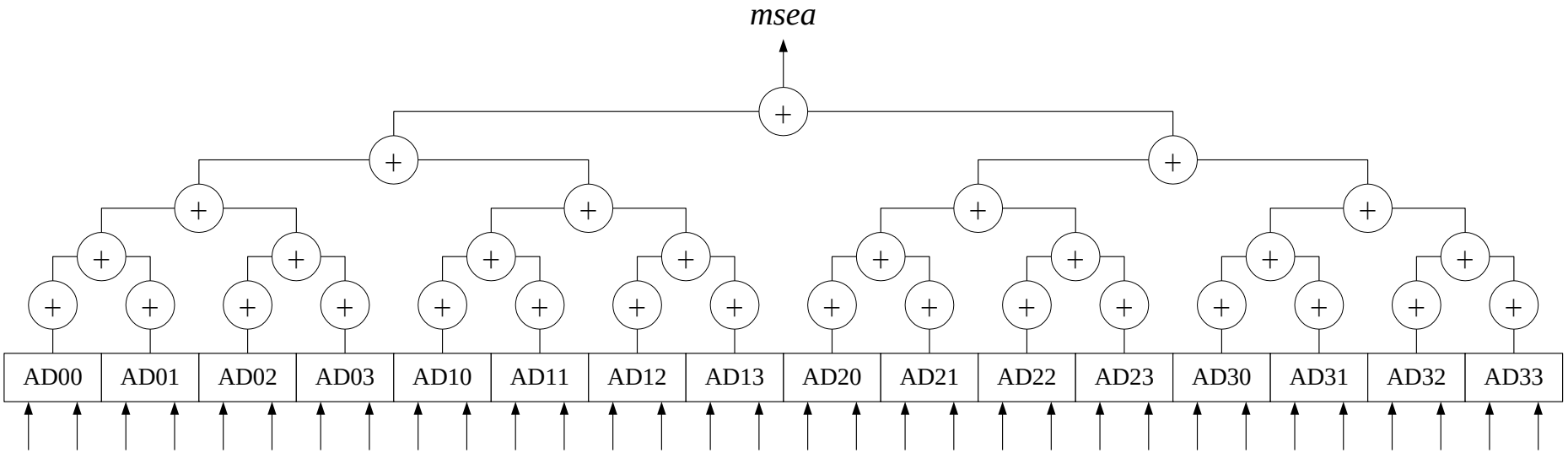
Systolic Part

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.

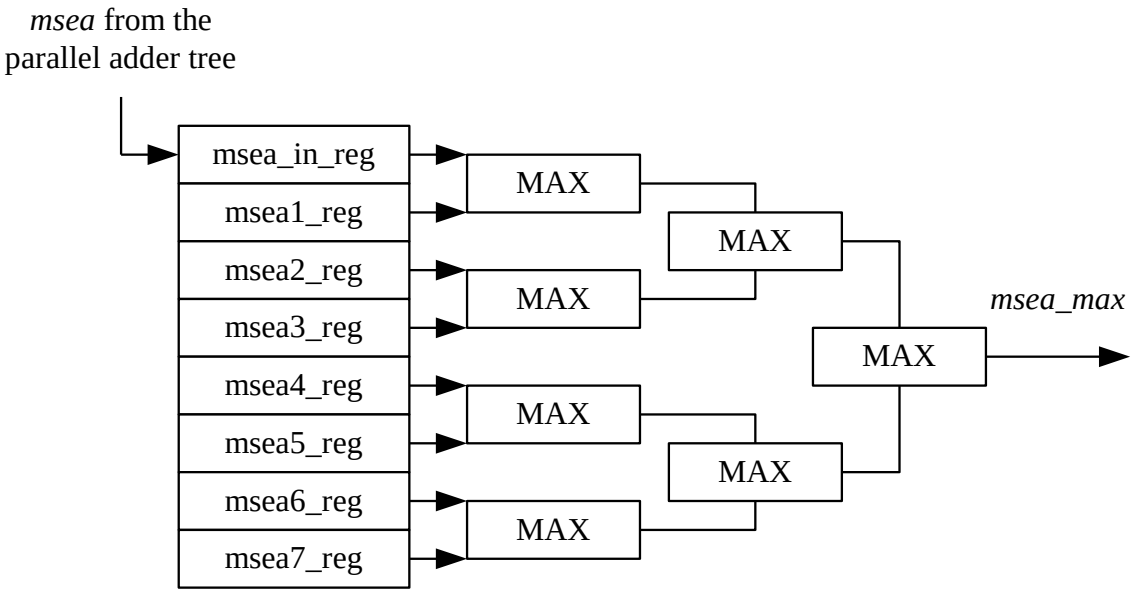


Parallel Adder Tree



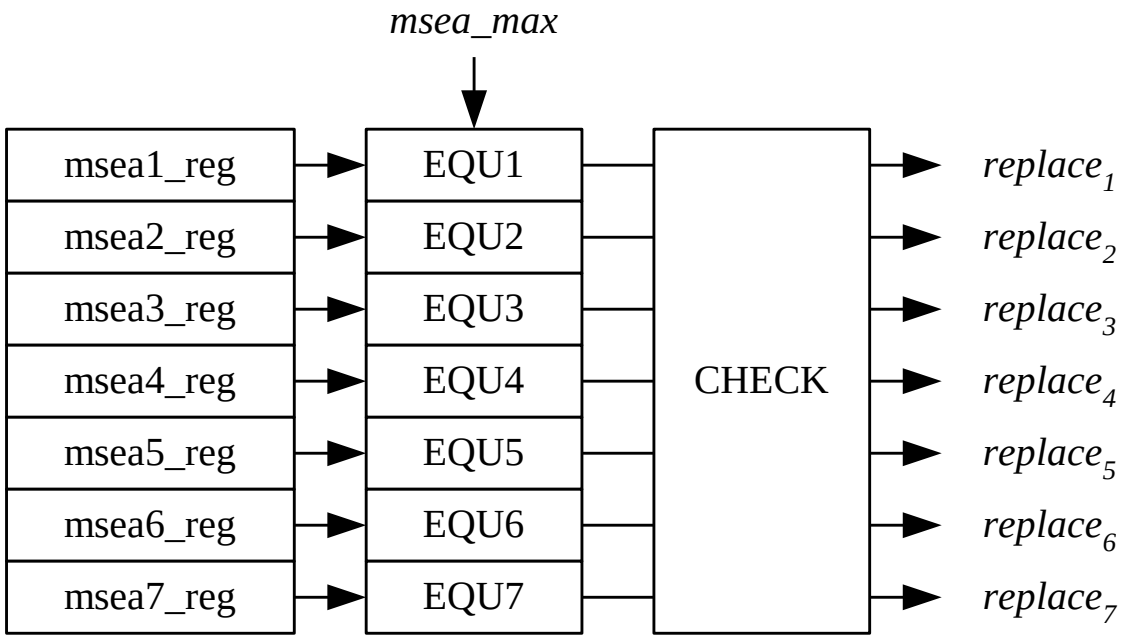
1. **"AD" unit stands for "Absolute Difference."**
2. **The sixteen sub-block sums of current block (stored in register right after they were previously calculated by the systolic part) and the sixteen sub-block sums of a candidate block (directly inputted from the systolic part) are inputted in parallel.**
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)



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=1~7) 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.**

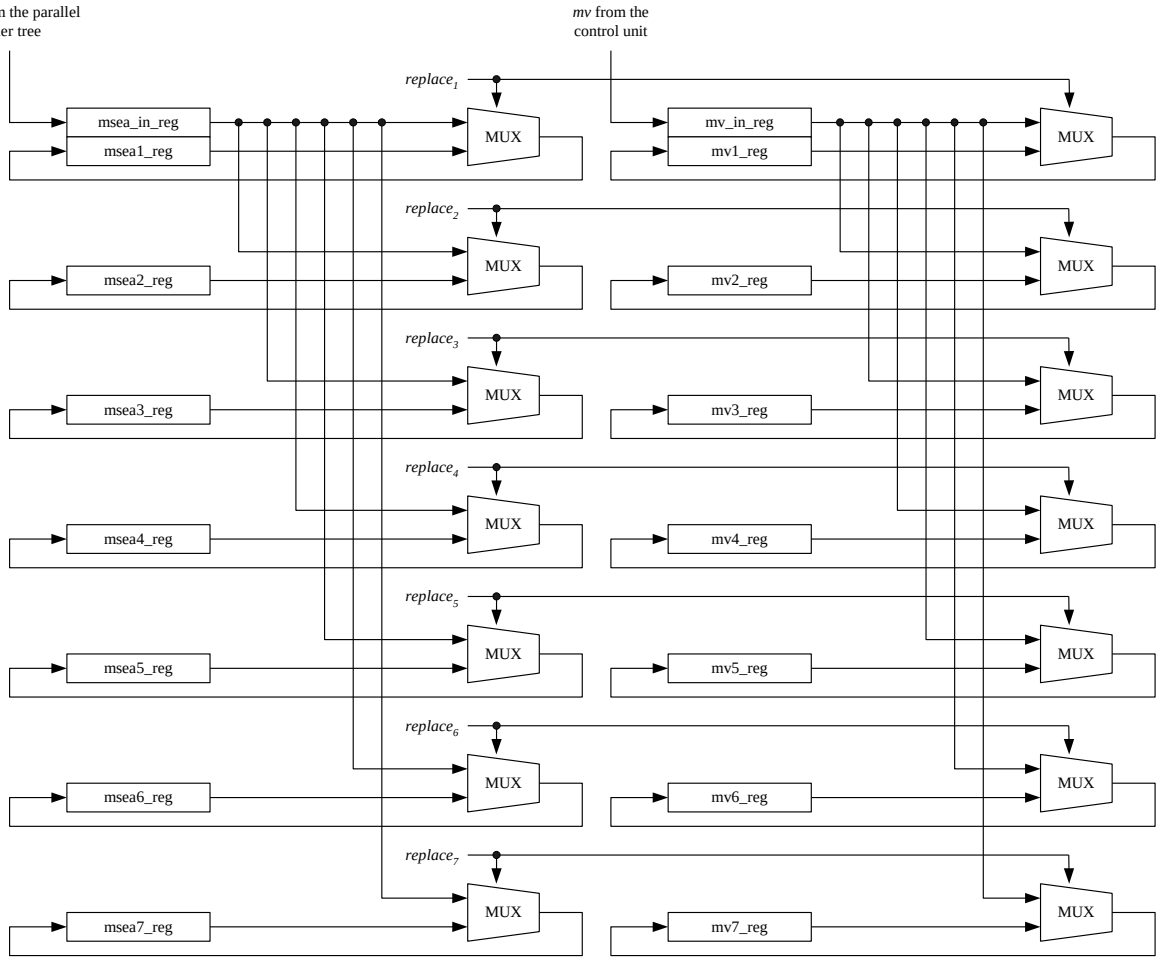
Parallel Comparator Tree (2/3)



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.**



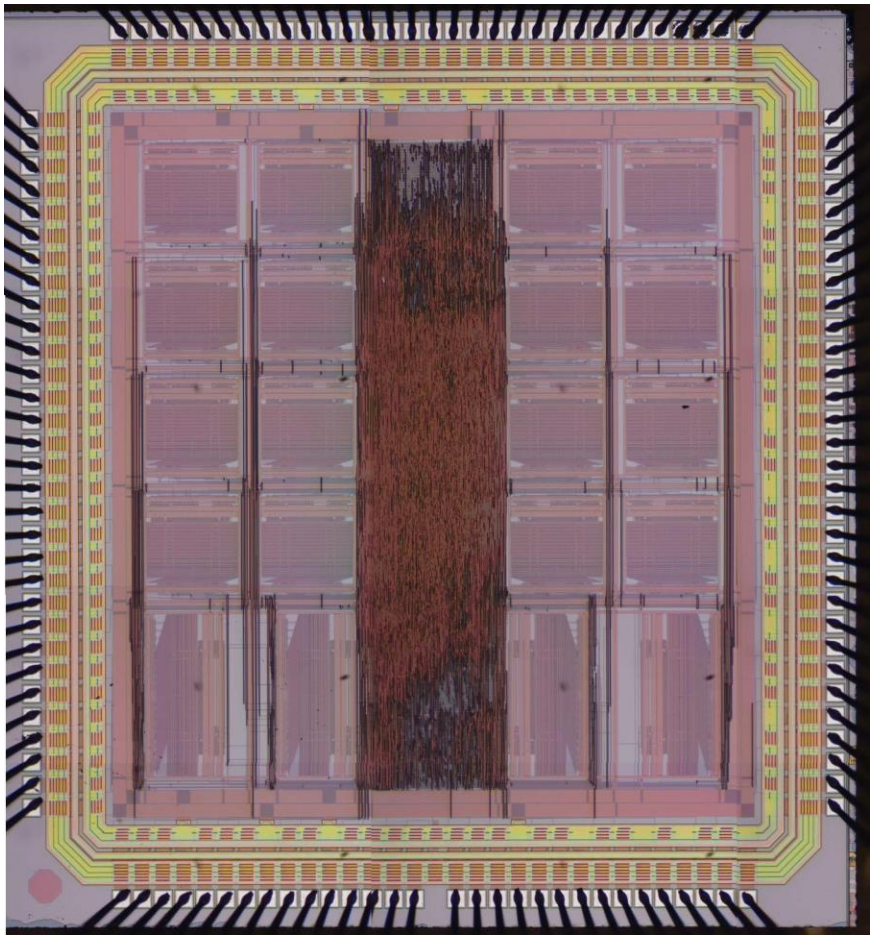
Comparison with FSBMA [-16,+15]

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

Comparison with FSBMA [-32,+31]

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

Chip Photo & Spec.



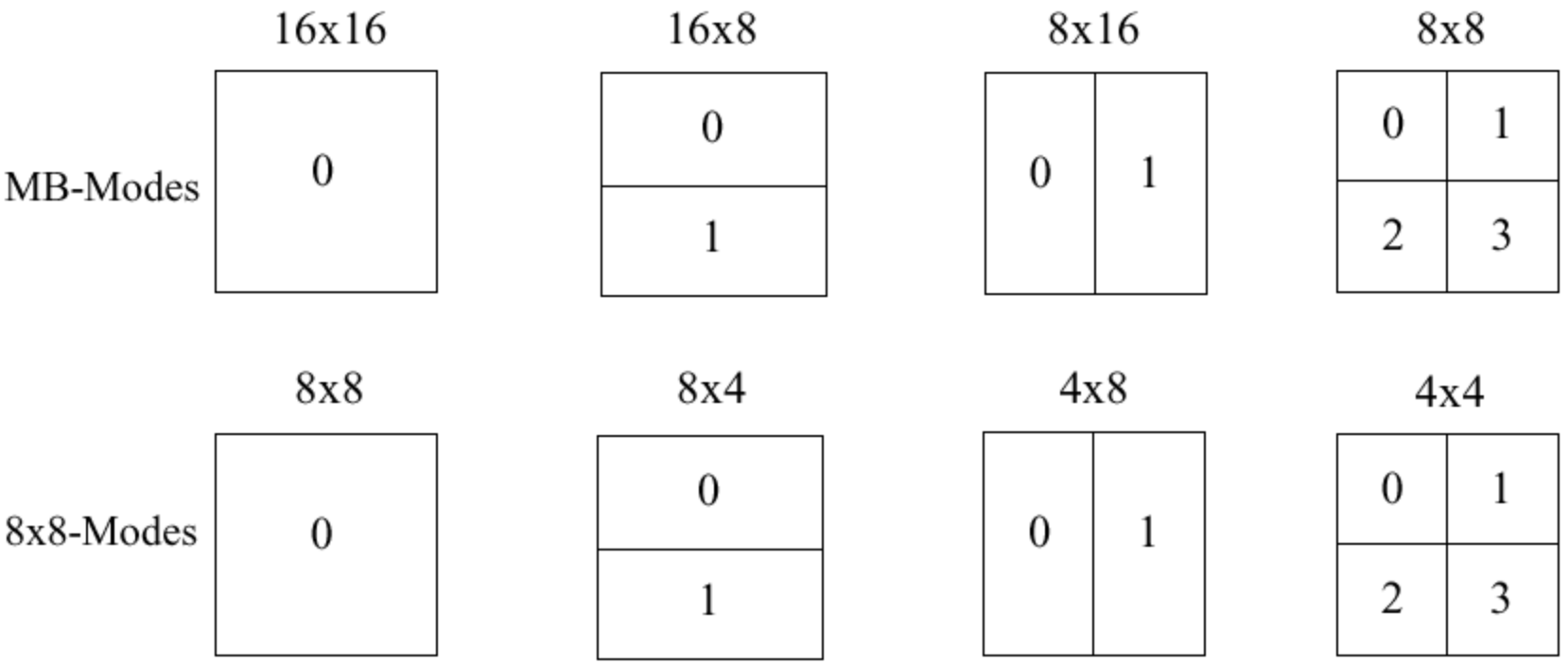
Process	TSMC 1P4M 0.35 um
Package	128 CQFP
Die Size	3.679 x 4.001 mm^2
Core Size	2.591 x 2.879 mm^2
Max. Frequency	27.8 MHz
Logic Gate Count	25,997
On-Chip SRAM	20,480 bits
Transistor Count	357,551
Search Range	[-16,+15]
Processing Speed	152 QCIF (176 x 144) fps 38 CIF (352 x 288) fps
Power Consumption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

Motion Estimation in H.264

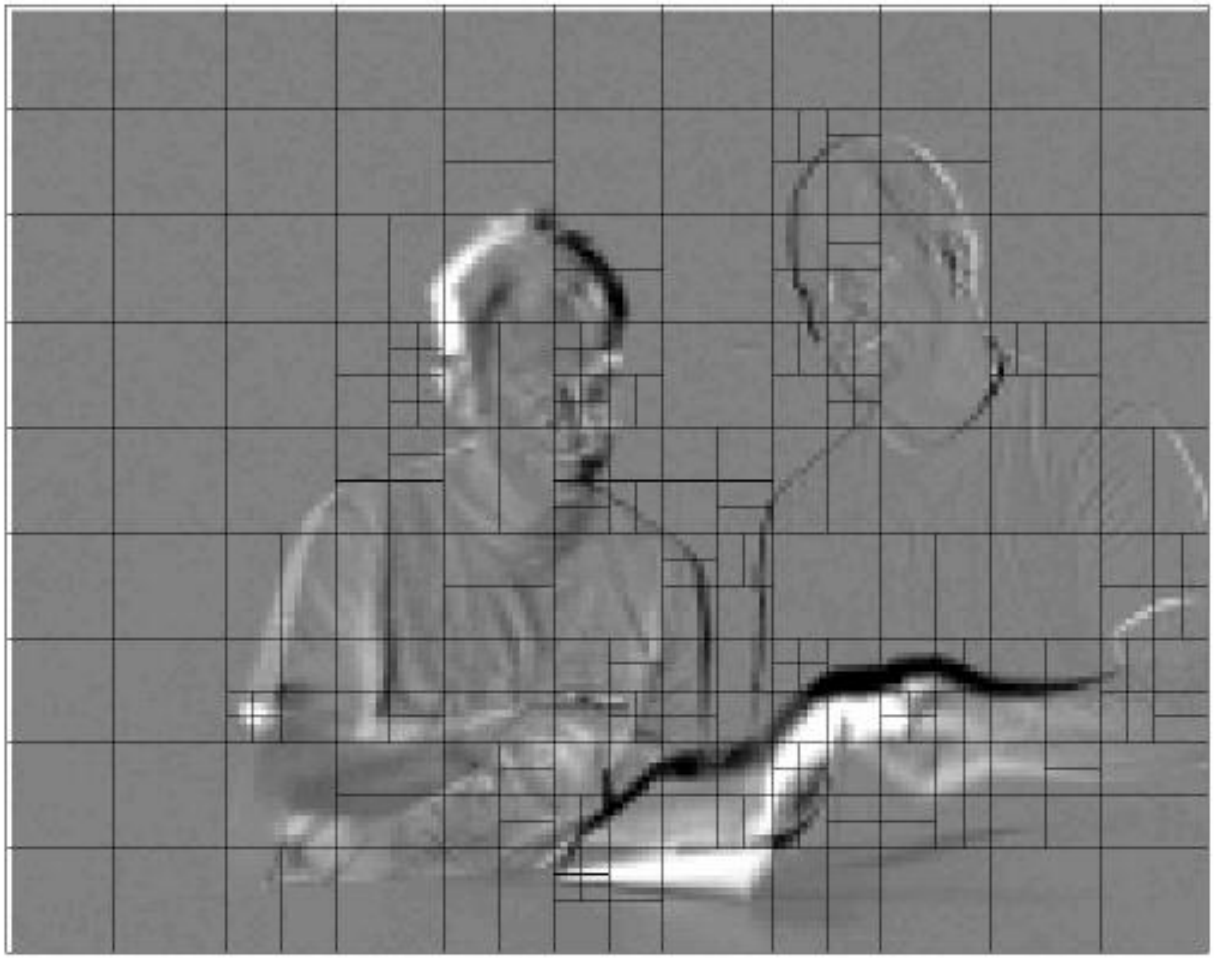
Motion Compensation

- 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

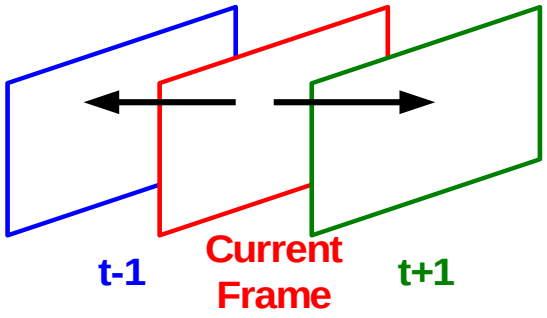


Example of Variable Block Sizes

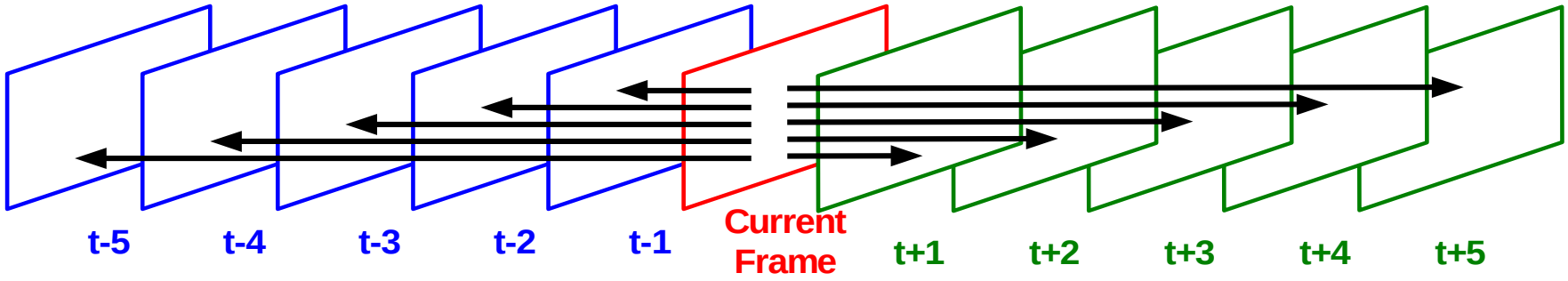


Multiple Reference Frames

MPEG-1, MPEG-2, MPEG-4



H.264/JVT/AVC

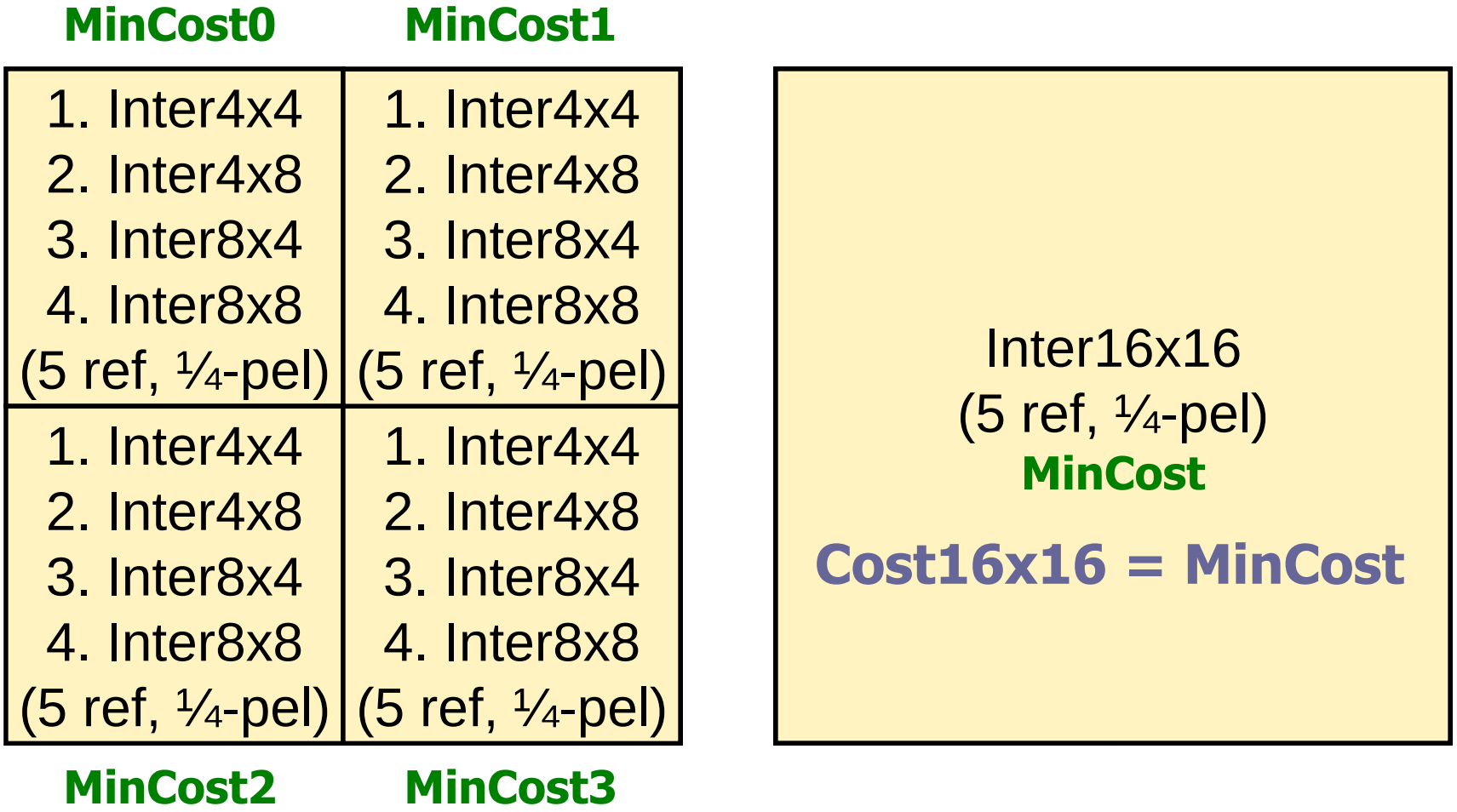


Note: the whole 8x8 sub-partition must be predicted by the same reference frame.

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.
- λ is called Lagrangian multiplier.
  - Theoretically, assume D is a function of R, denoted as D(R), then λ should be obtained by differentiating J with respect to R.
  - Setting the first derivative to zero and solve λ.
  - In the reference software, λ is a function of QP.

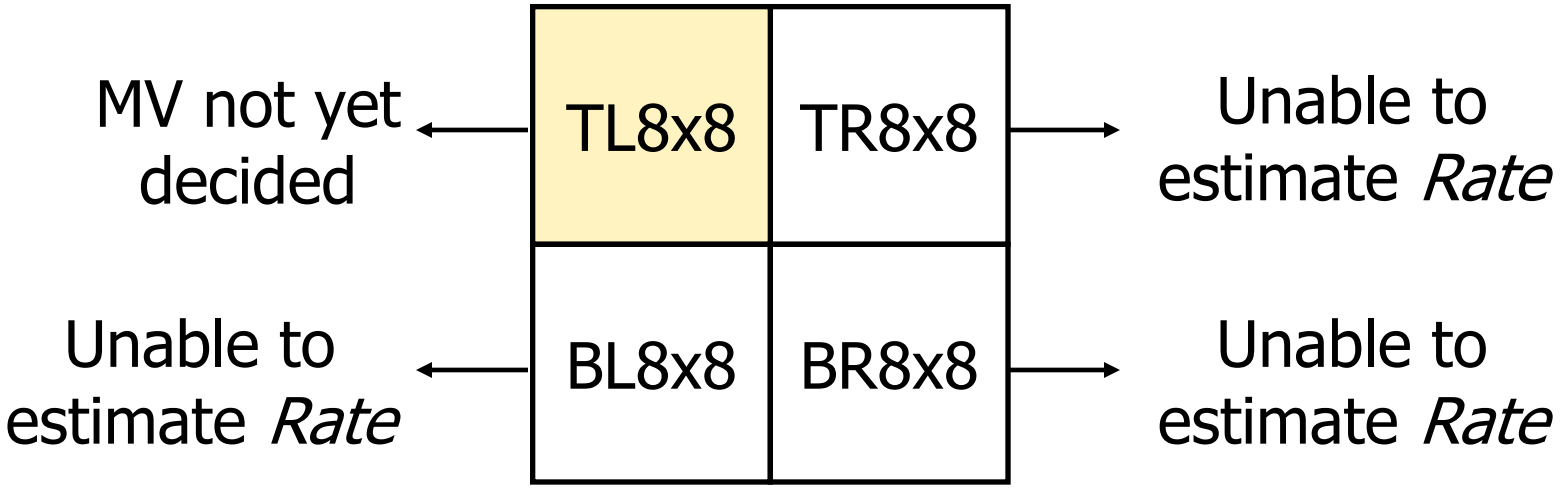
Macroblock Mode Decision



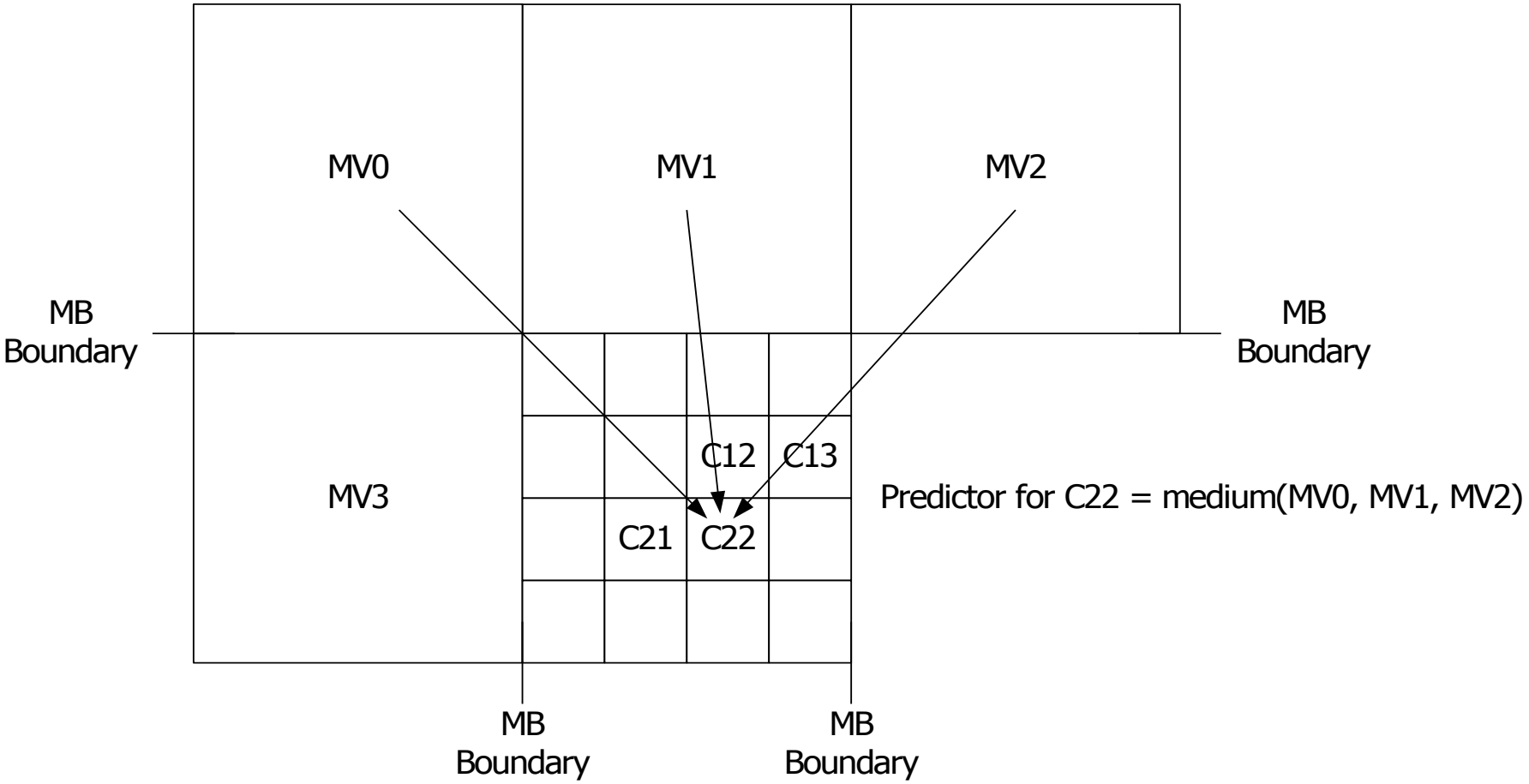
Cost8x8 = MinCost0+MinCost1+MinCost2+MinCost3

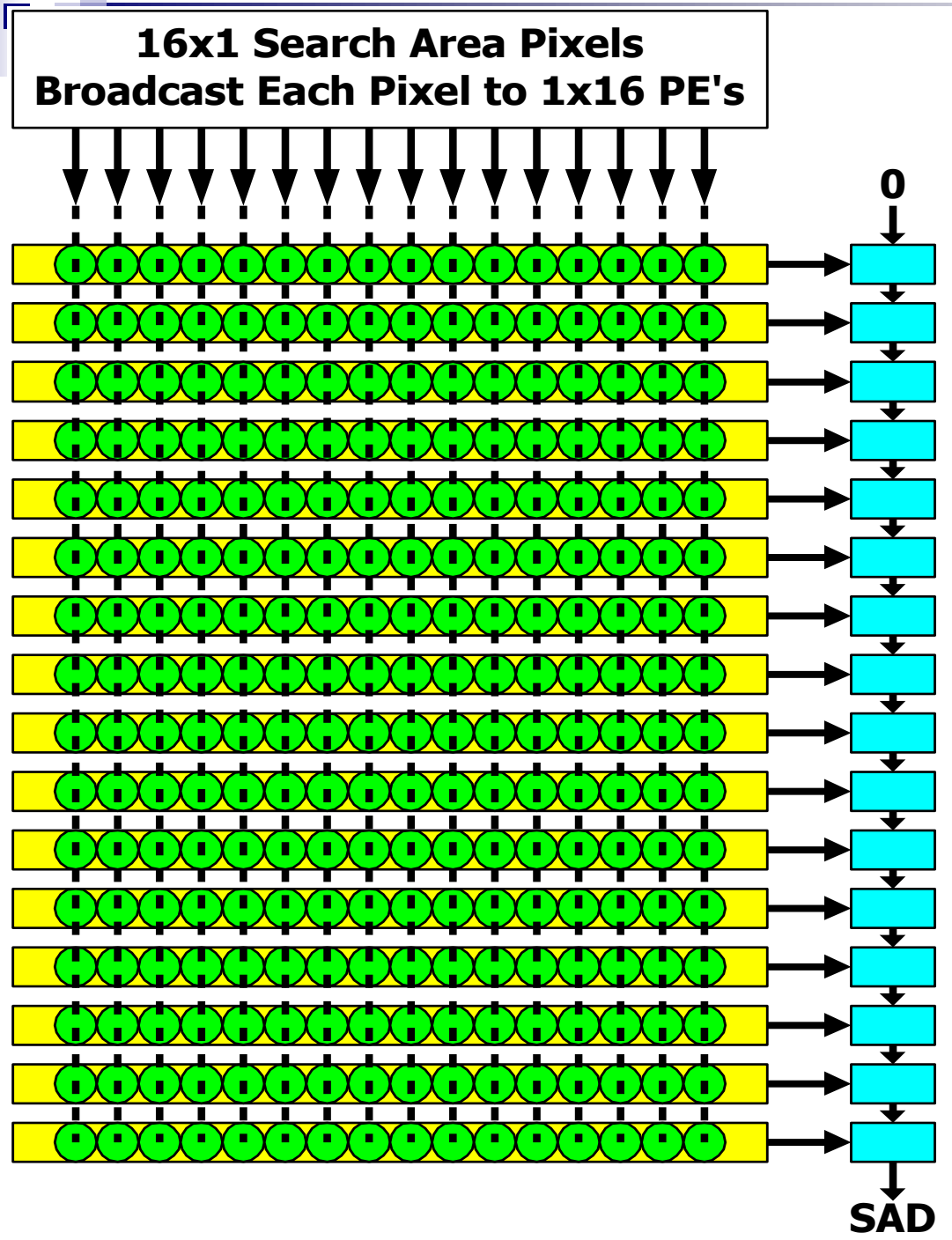
Encountered Problem

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



Modified Macroblock Mode Decision

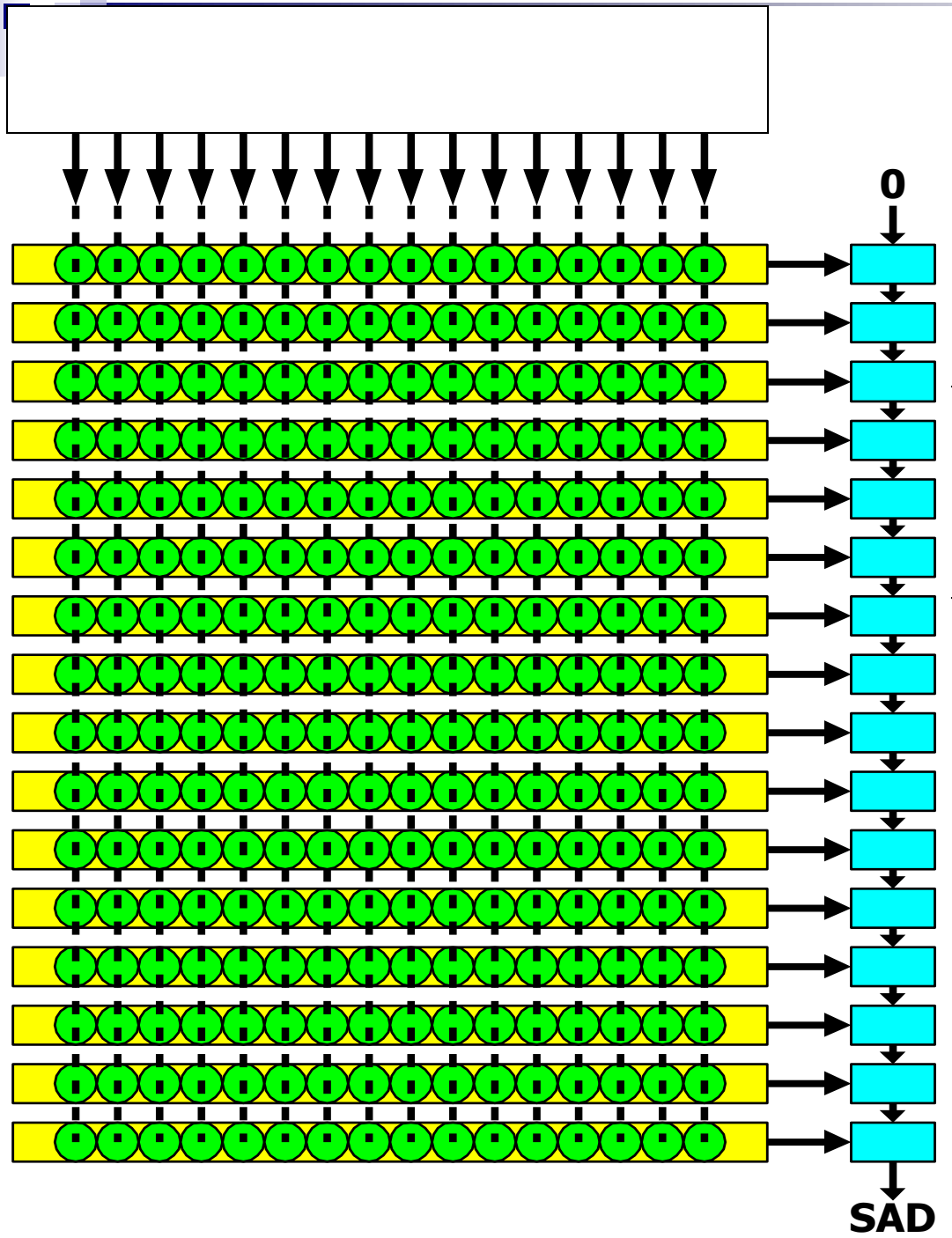


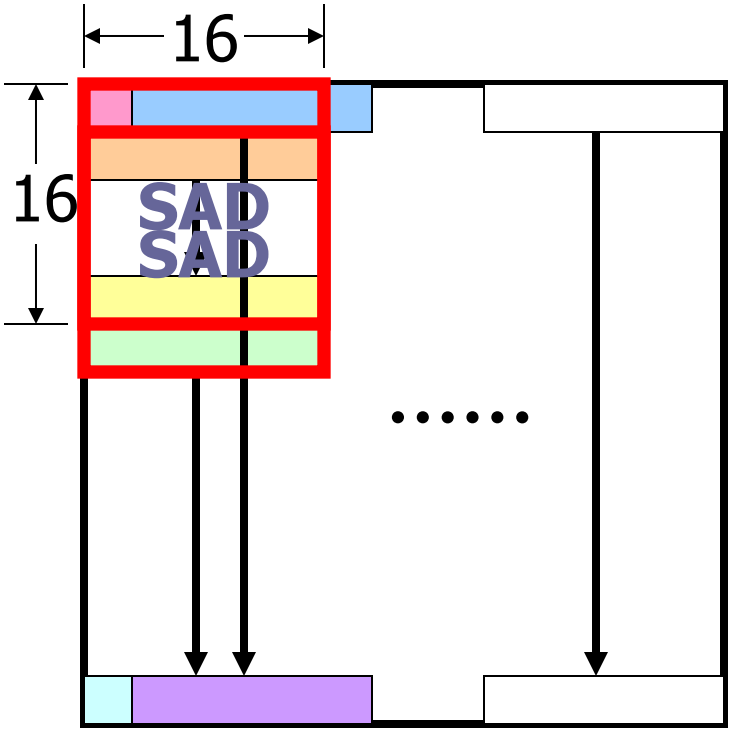


Basic Architecture

- 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

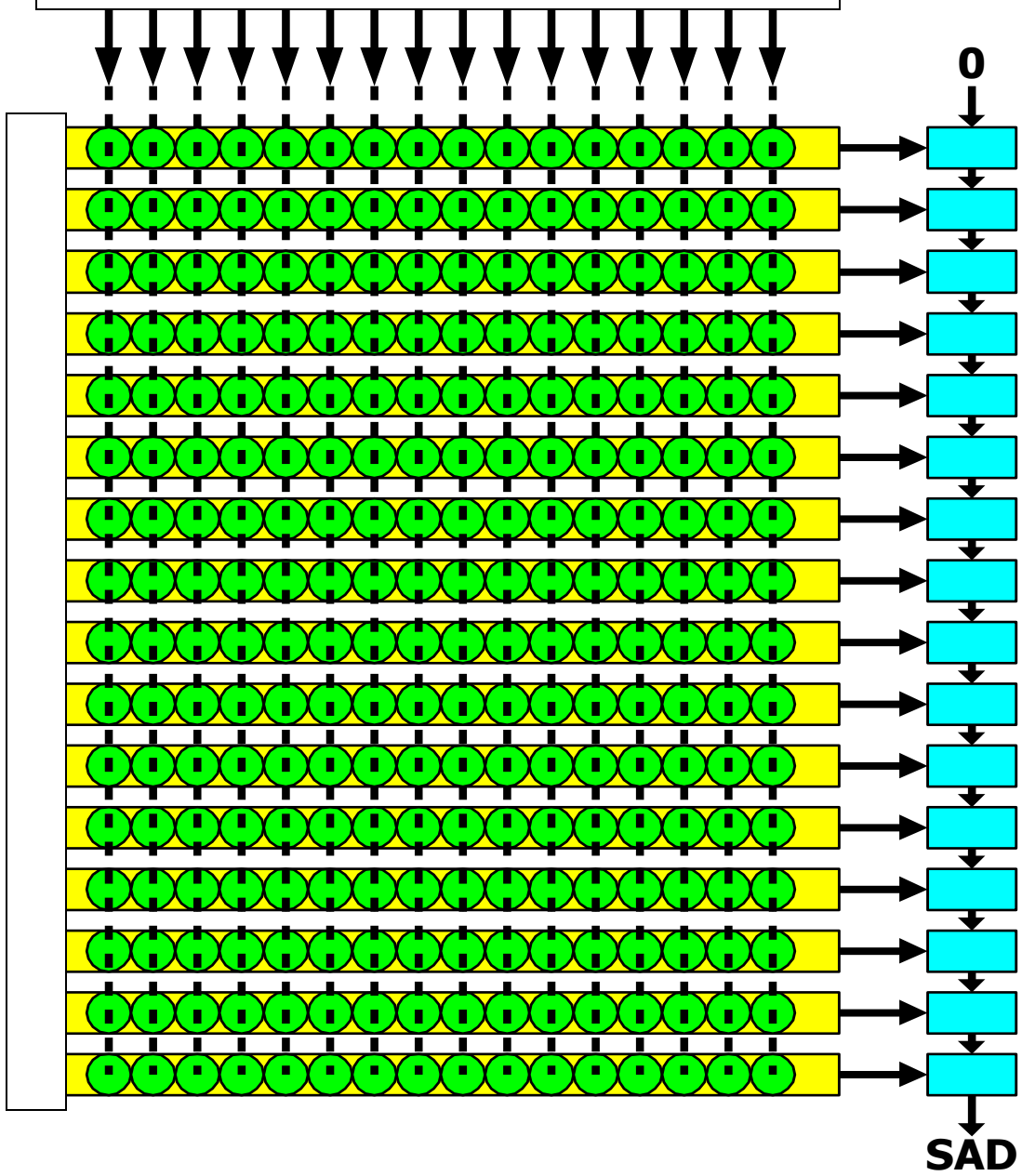
Basic Data Flow

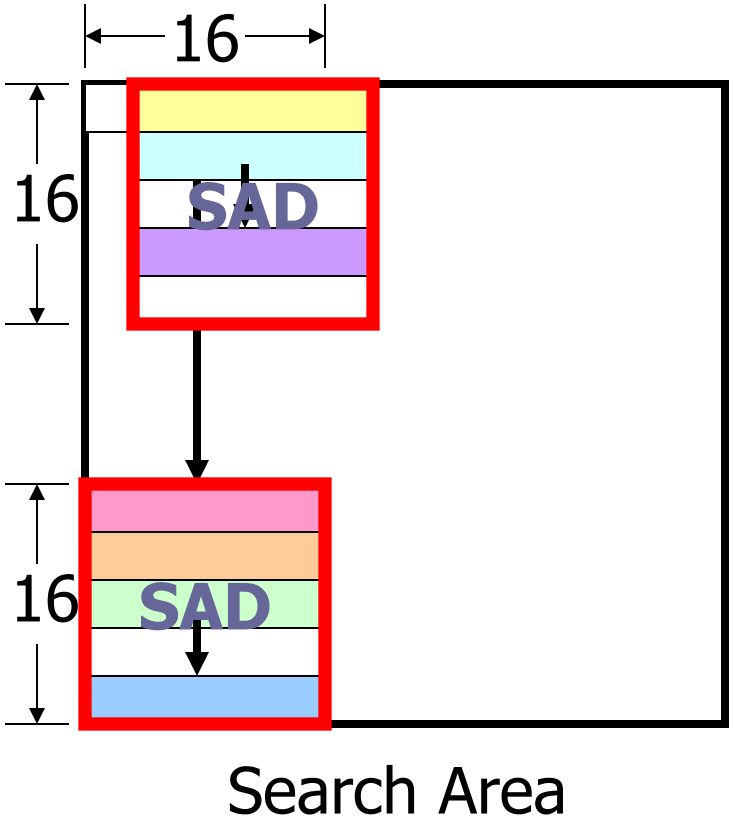




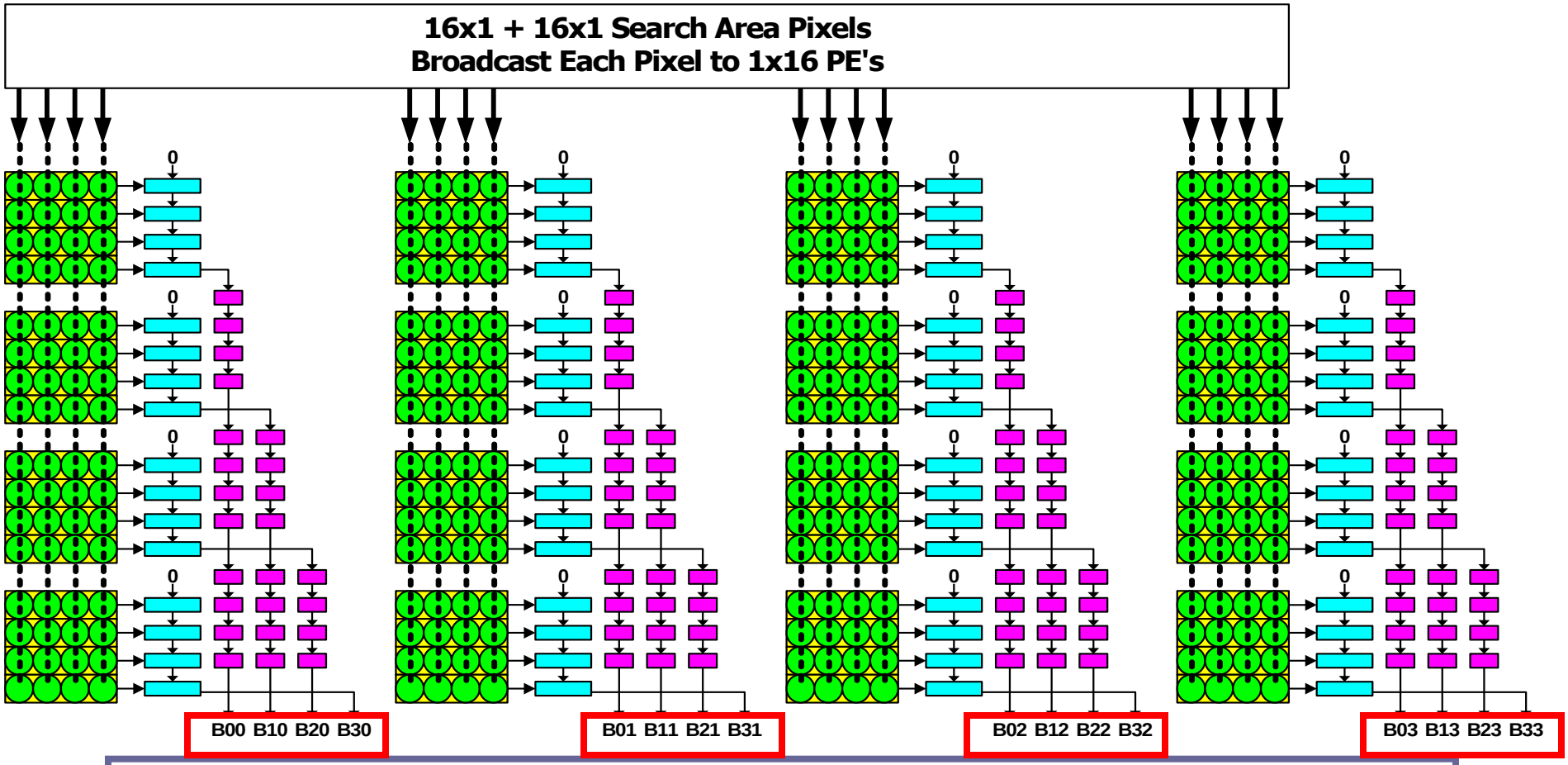
Search Area

Advanced Data Flow



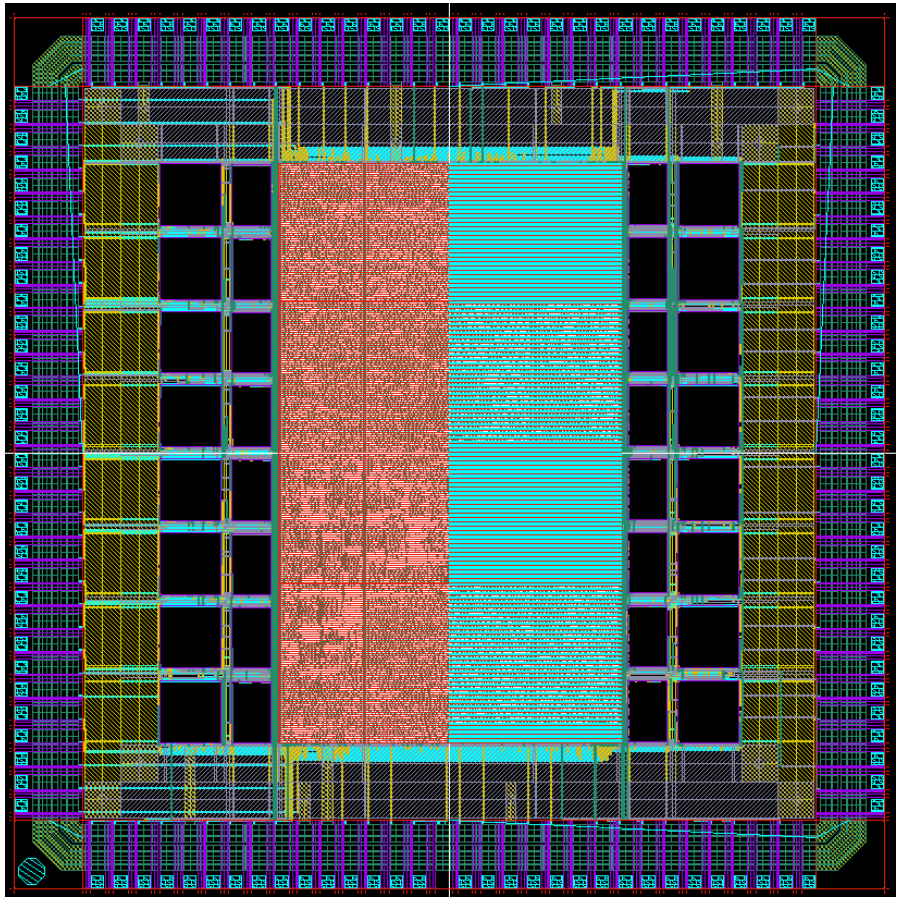


Variable Block Size Architecture



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^2
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 D1 (720x480) 30fps 1Ref. SIF (352x240) 30fps 4Ref.

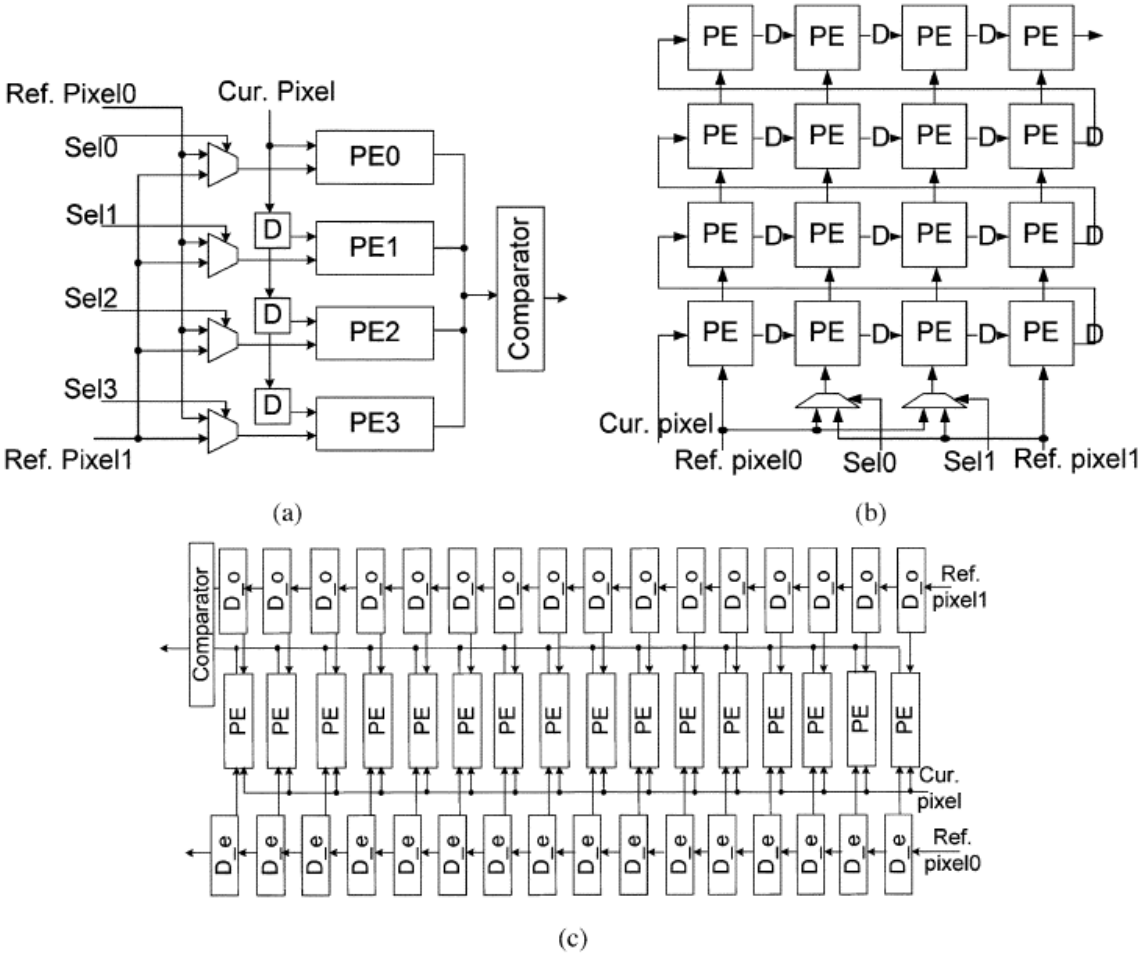
Variable-Block-Size Motion Estimation

- 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.

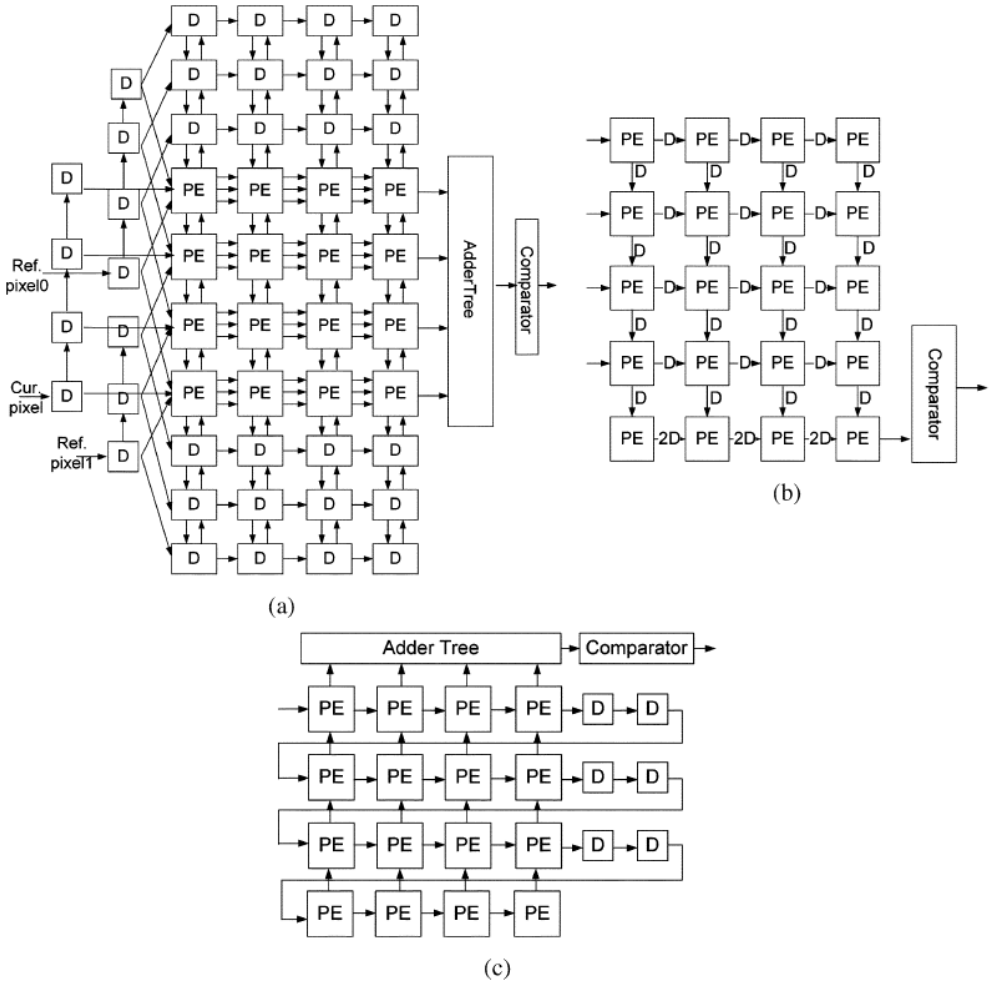
Six Major Reference Architecture

- **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.

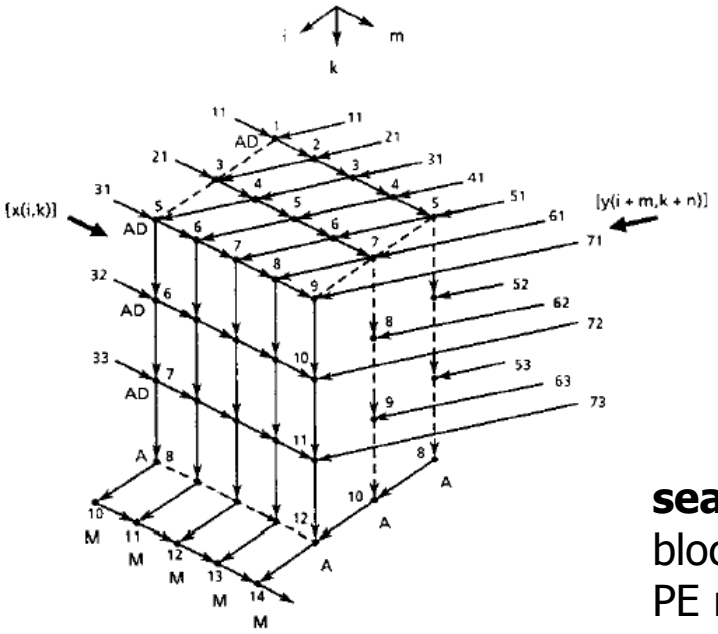
Inter-Parallel Architecture

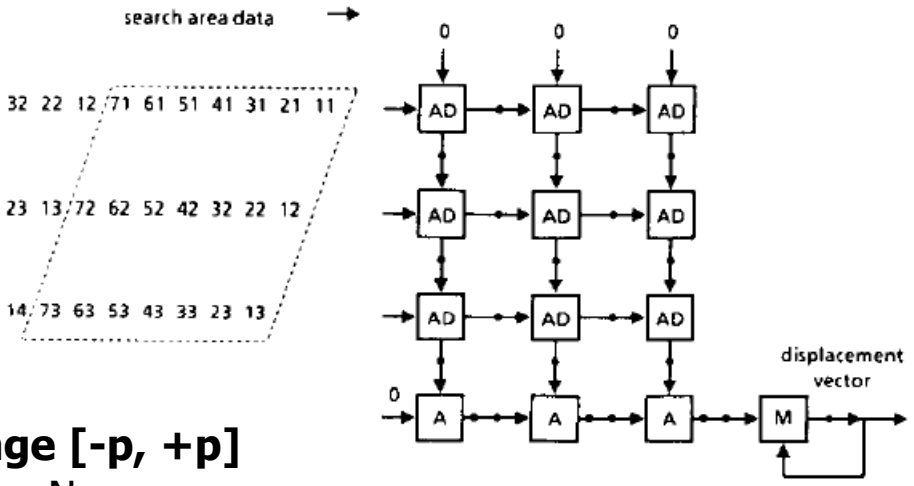


Intra-Parallel Architecture



AB2 Architecture





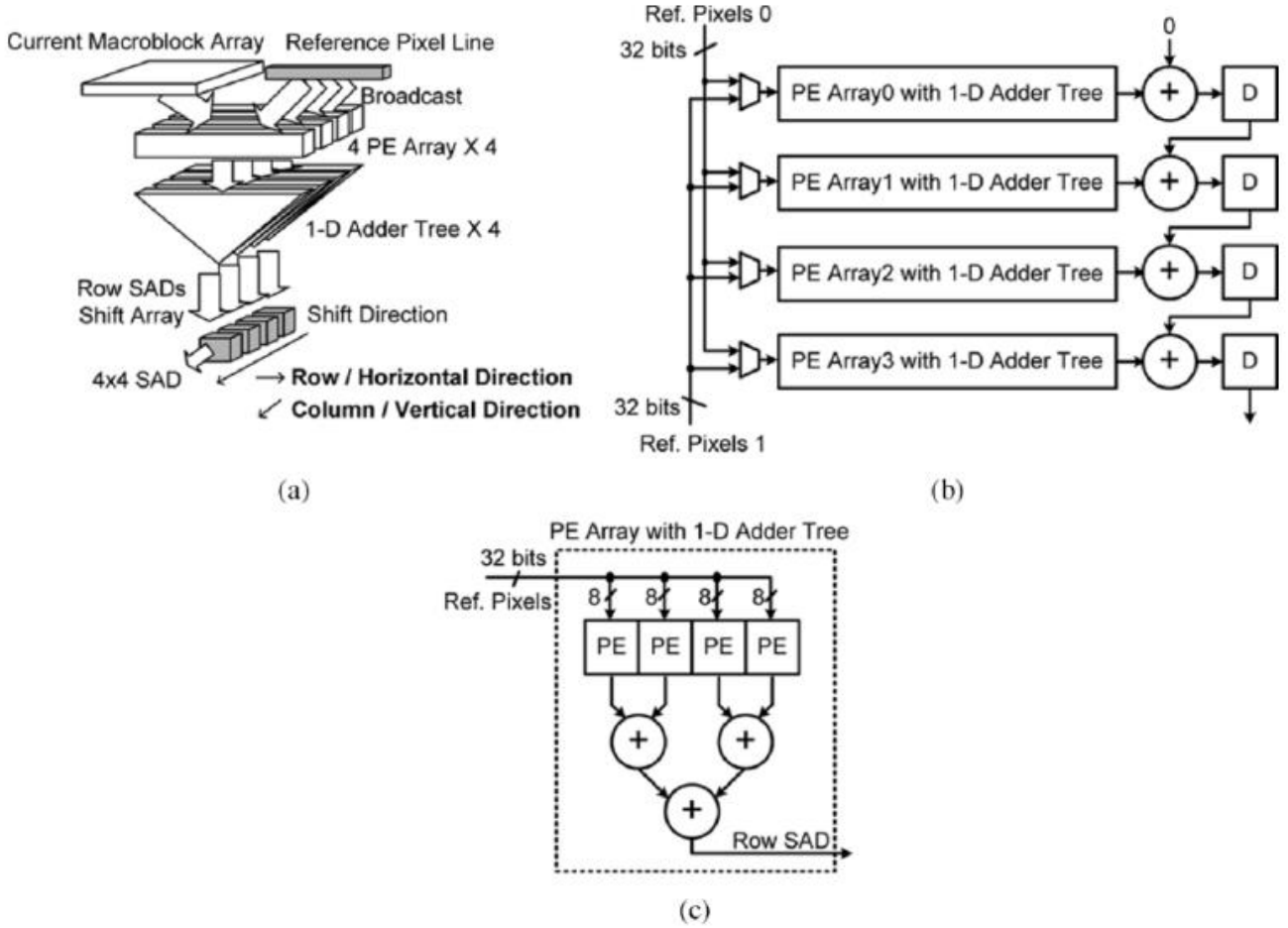
search range [-p, +p]
block size N x N
PE no. = N^2 = 2-D block size

Dependence Graph
projection on i,k plane

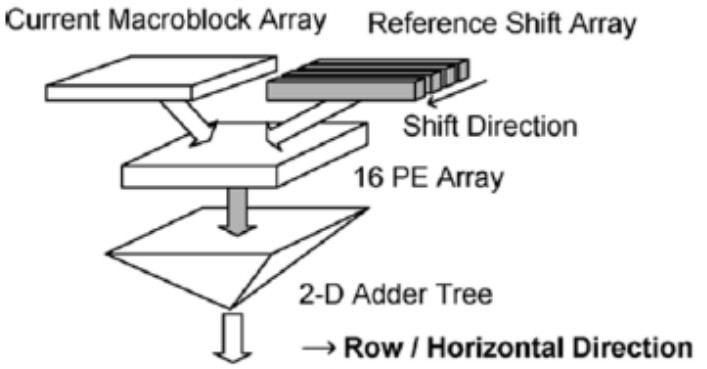
Systolic Architecture
(N=3, p=2)

Number of cycles for a macroblock = (2p+1) x (2p+N)

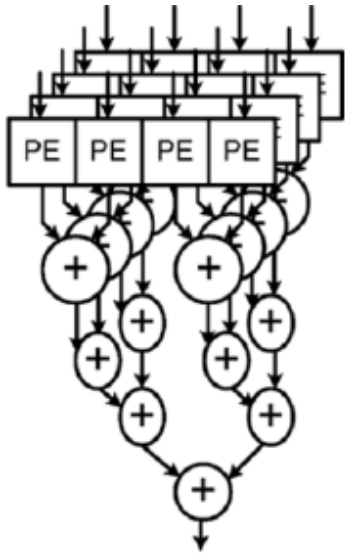
Propagation Partial SAD Architecture



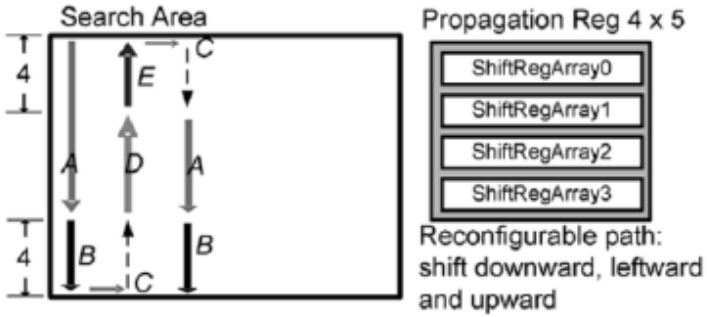
SAD Tree Architecture



(a)



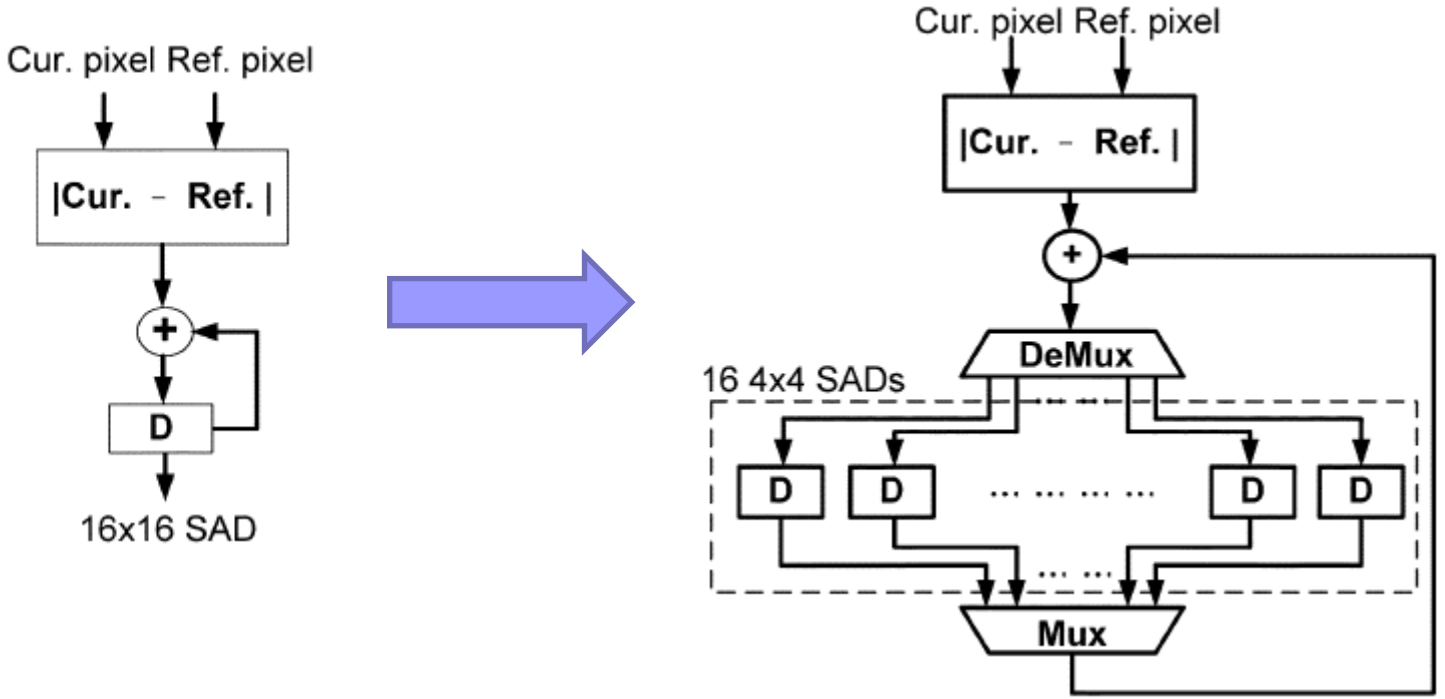
(b)



A : Shift downward and fetch 4 pixels in each cycle
B : Shift downward and fetch 5 pixels in each cycle
C : Shift leftward and do not fetch pixels
D : Shift upward and fetch 4 pixels in each cycle
E : Shift upward and fetch 5 pixels in each cycle

(c)

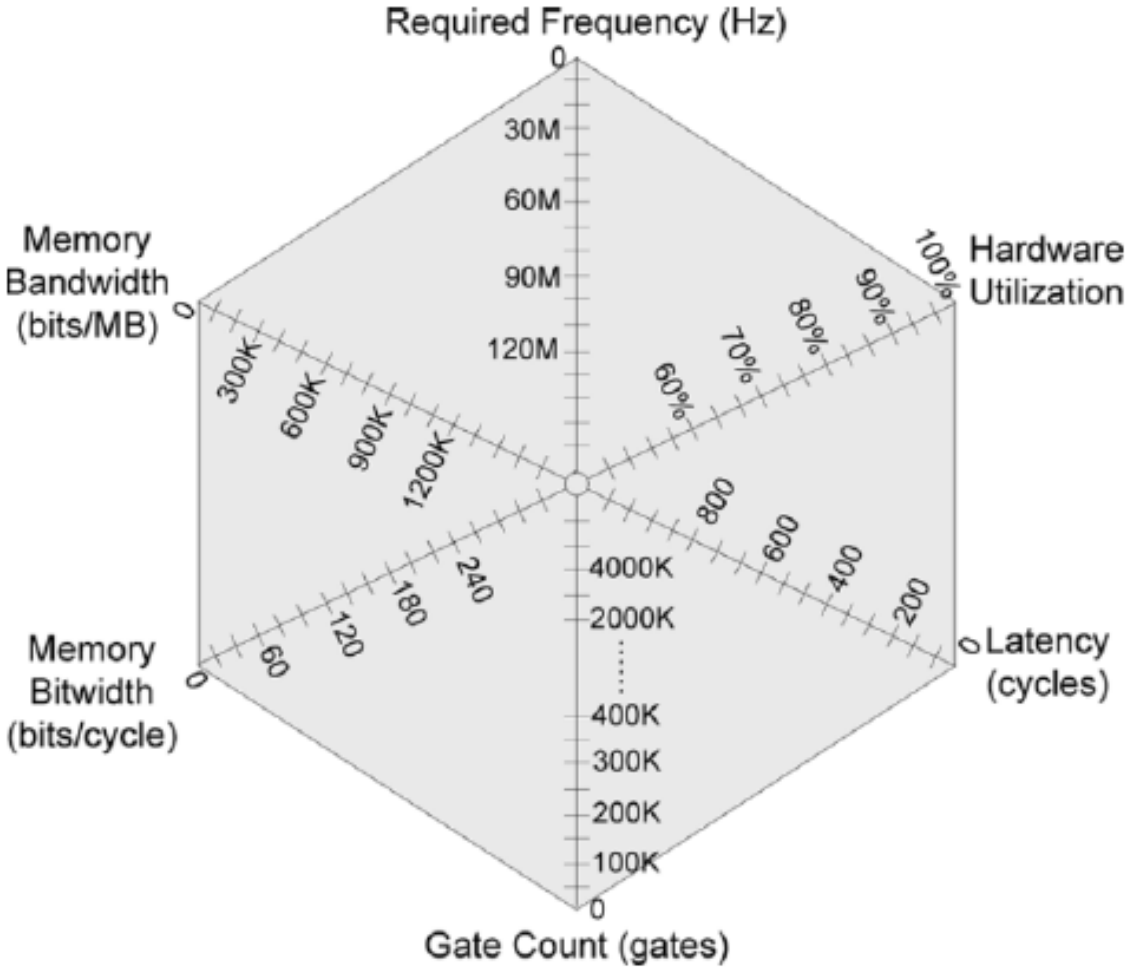
VBSME Version?

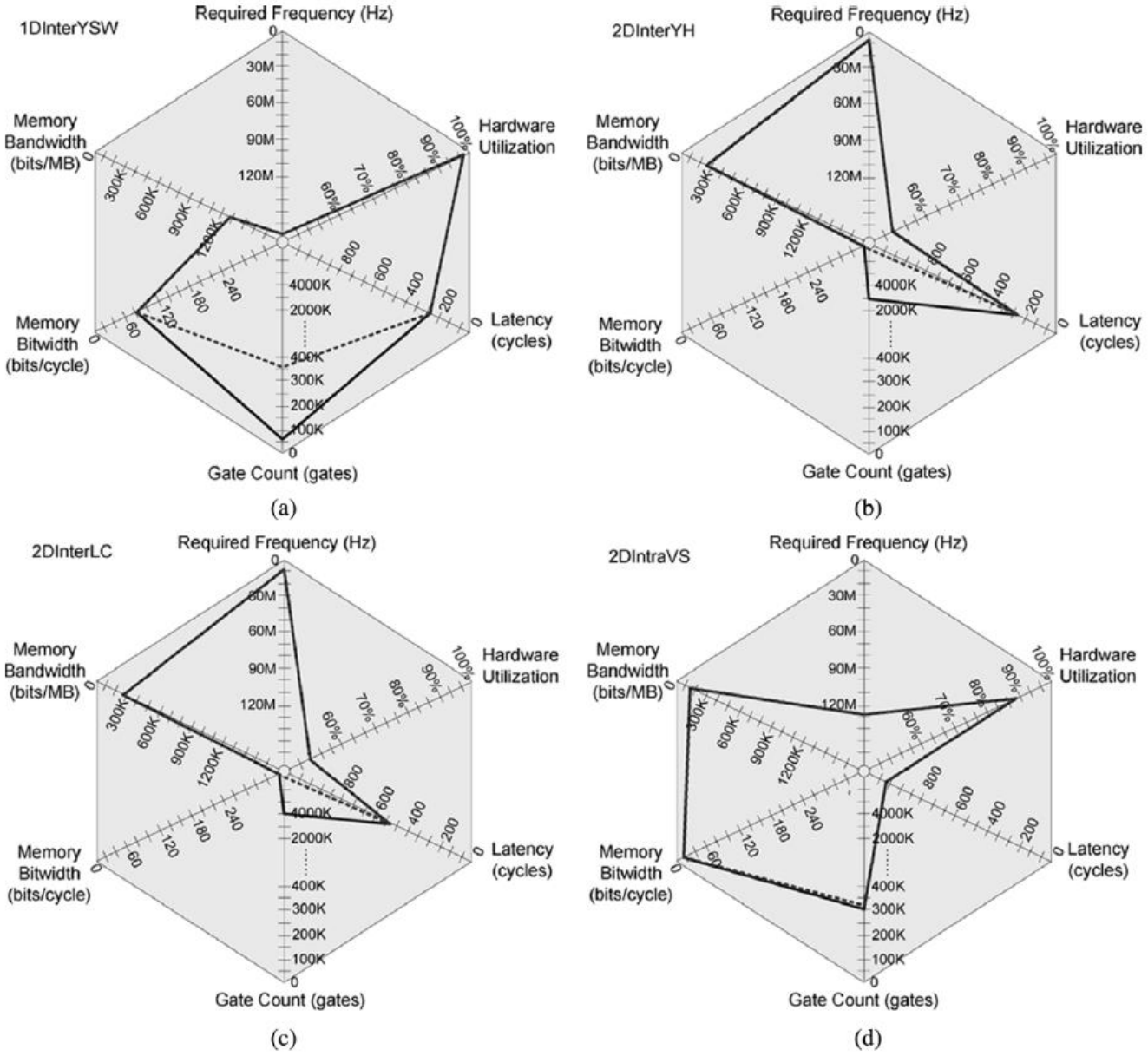


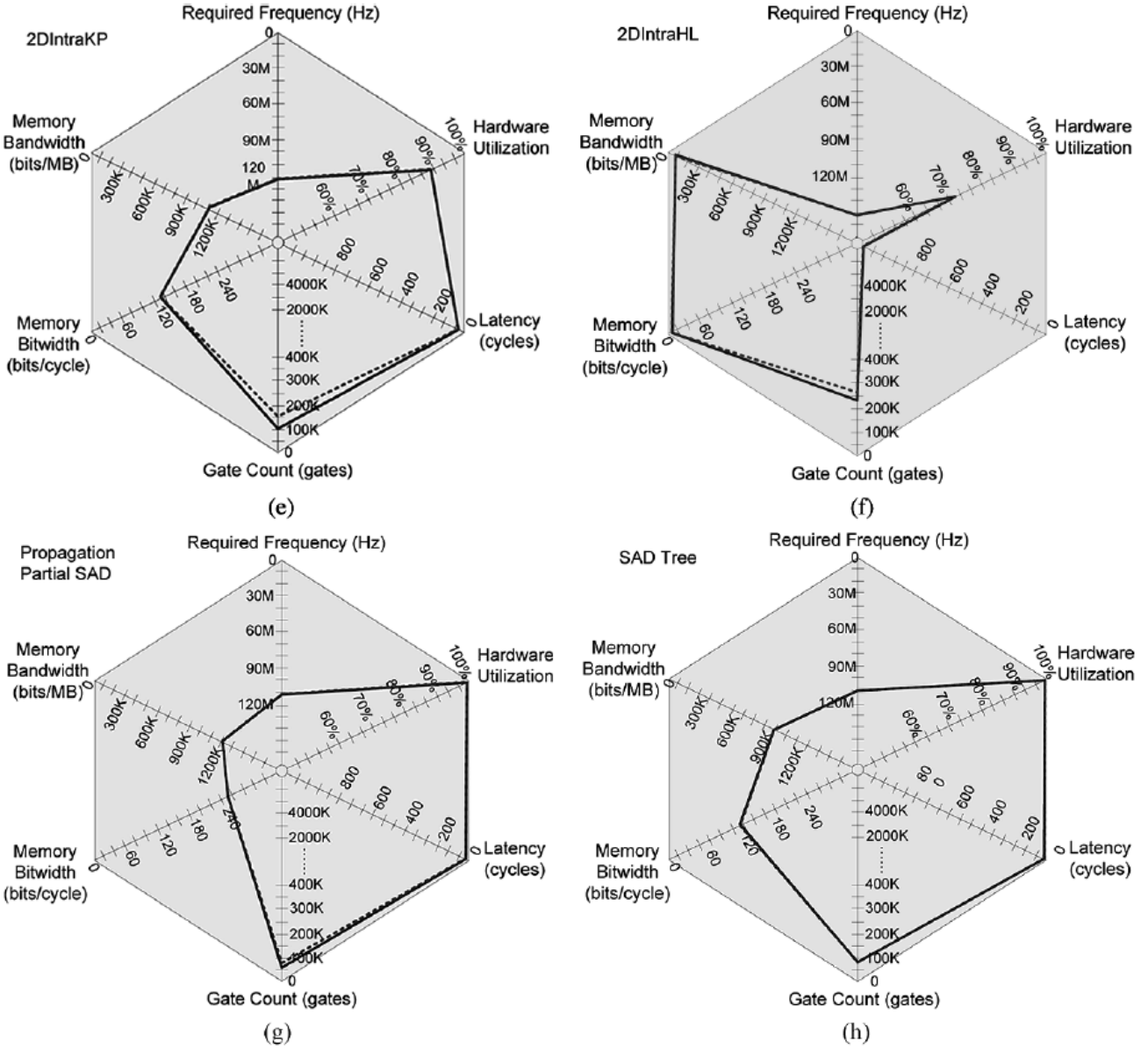
Comparison

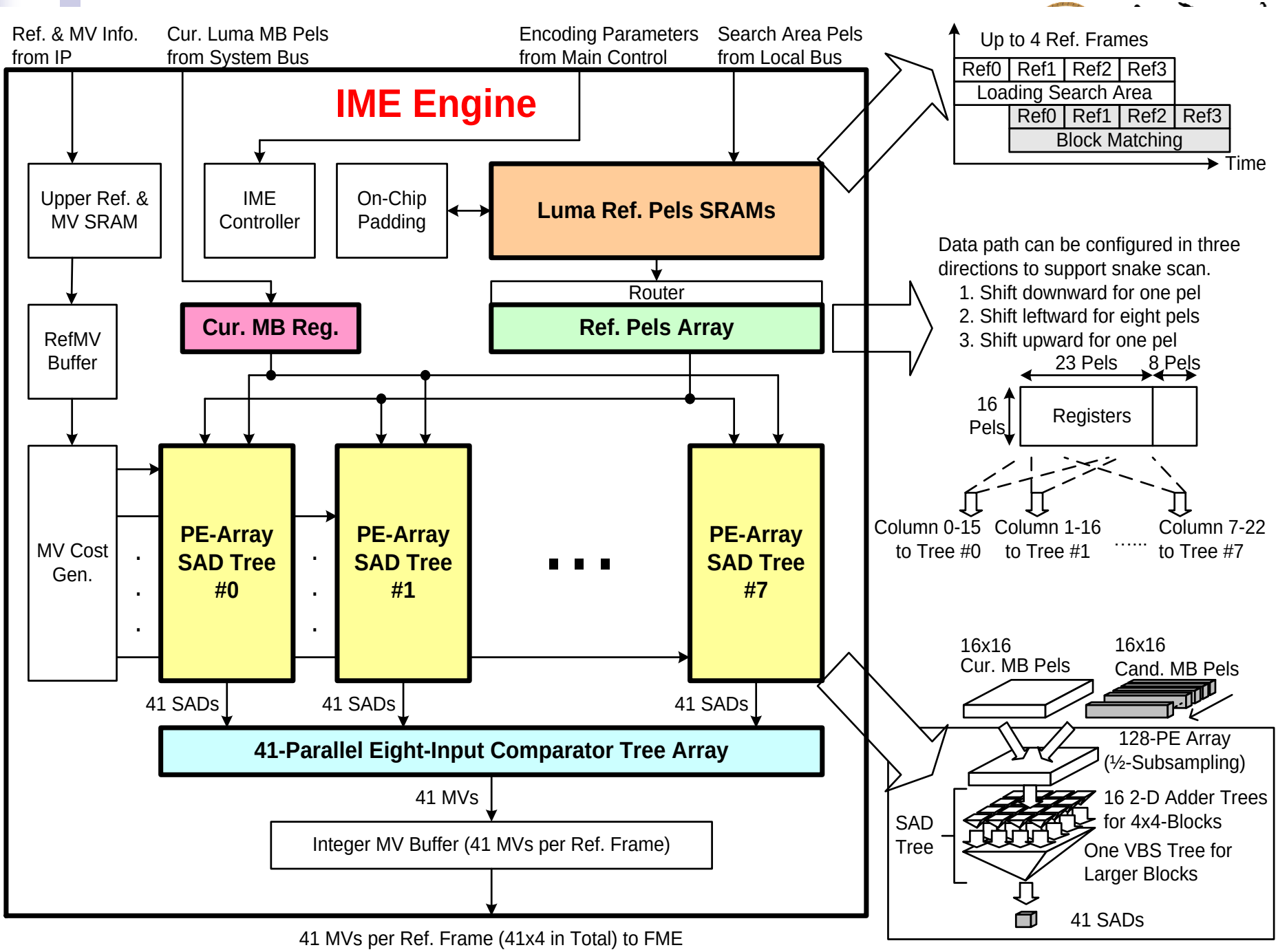
Name	No. of PEs	Operating Cycles (Cycles/Macroblock)	Latency (Cycles)	Data Flow
1DInterYSW [5]	$2P_h$	$N^2 \times 2P_v + 2P_h$	N^2	Data Flow I
2DInterYH [7]	$2P_h \times 2P_v$	$2N^2$	N^2	Data Flow I
2DInterLC [8]	$2P_h \times 2P_v$	$2N^2$	$2N^2$	Data Flow I
2DIntraVS [9]	N^2	$2P_h \times 2P_h + N \times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

Comparison: Hexagonal Plot




1DInterYSW
Required Frequency (Hz)
Hardware Utilization
Latency (cycles)
Gate Count (gates)
Memory Bandwidth (bits/MB)
Memory Bitwidth (bits/cycle)
(a)
2DInterYH
Required Frequency (Hz)
Hardware Utilization
Latency (cycles)
Gate Count (gates)
Memory Bandwidth (bits/MB)
Memory Bitwidth (bits/cycle)
(b)
2DInterLC
Required Frequency (Hz)
Hardware Utilization
Latency (cycles)
Gate Count (gates)
Memory Bandwidth (bits/MB)
Memory Bitwidth (bits/cycle)
(c)
2DIntraVS
Required Frequency (Hz)
Hardware Utilization
Latency (cycles)
Gate Count (gates)
Memory Bandwidth (bits/MB)
Memory Bitwidth (bits/cycle)
(d)


2DIntraKP
Required Frequency (Hz)
Hardware Utilization
Latency (cycles)
Gate Count (gates)
Memory Bitwidth (bits/cycle)
Memory Bandwidth (bits/MB)
(e)
2DIntraHL
Required Frequency (Hz)
Hardware Utilization
Latency (cycles)
Gate Count (gates)
Memory Bitwidth (bits/cycle)
Memory Bandwidth (bits/MB)
(f)
Propagation Partial SAD
Required Frequency (Hz)
Hardware Utilization
Latency (cycles)
Gate Count (gates)
Memory Bitwidth (bits/cycle)
Memory Bandwidth (bits/MB)
(g)
SAD Tree
Required Frequency (Hz)
Hardware Utilization
Latency (cycles)
Gate Count (gates)
Memory Bitwidth (bits/cycle)
Memory Bandwidth (bits/MB)
(h)


Ref. & MV Info. from IP
Cur. Luma MB Pels from System Bus
Encoding Parameters from Main Control
Search Area Pels from Local Bus
IME Engine
Upper Ref. & MV SRAM
IME Controller
On-Chip Padding
Luma Ref. Pels SRAMs
RefMV Buffer
Cur. MB Reg.
Router
Ref. Pels Array
MV Cost Gen.
PE-Array SAD Tree #0
PE-Array SAD Tree #1
PE-Array SAD Tree #7
41 SADs
41 SADs
41 SADs
41-Parallel Eight-Input Comparator Tree Array
41 MVs
Integer MV Buffer (41 MVs per Ref. Frame)
41 MVs per Ref. Frame (41x4 in Total) to FME
Up to 4 Ref. Frames
Ref0 Ref1 Ref2 Ref3
Loading Search Area
Ref0 Ref1 Ref2 Ref3
Block Matching
Time
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
23 Pels
8 Pels
16 Pels
Registers
Column 0-15 to Tree #0
Column 1-16 to Tree #1
.....
Column 7-22 to Tree #7
16x16 Cur. MB Pels
16x16 Cand. MB Pels
128-PE Array (½-Subsampling)
SAD Tree
16 2-D Adder Trees for 4x4-Blocks
One VBS Tree for Larger Blocks
41 SADs