

JCT3V-D0165

CE1.h: VSP Complexity Analysis and Constrained VSP (CVSP)

Feng Zou, Dong Tian, Anthony Vetro, Huifang Sun
Mitsubishi Electric Research Labs (MERL)
Cambridge, MA

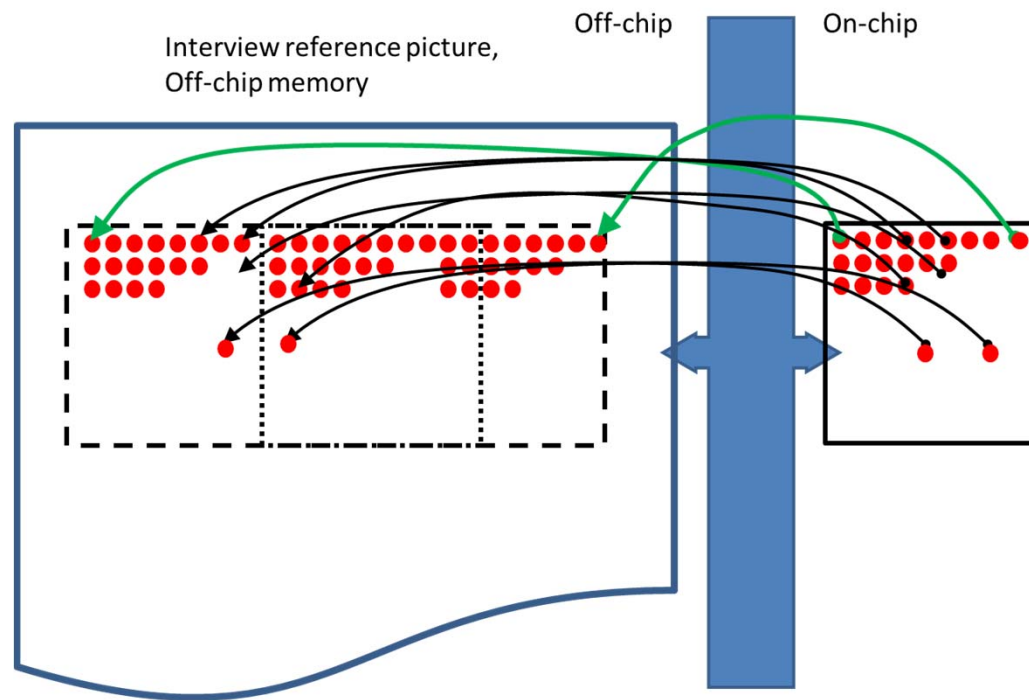
4th JCT-3V meeting, Incheon, KR, April 2013

Assumptions for HW implementations

- This contribution
 - Study both off-chip and on-chip complexity
 - Propose constraints to simplify VSP implementation
- DPB buffer stores in off-chip memory (SDRAM)
- HEVC interpolation process
 - 8- or 7- tap filters
 - On-chip processing elements (PE)
 - Supported by on-chip local memory (LM)
- Prediction block (PB) is represented as MxN
 - M: Width
 - N: Height

Problem: Off-chip memory access

- VSP imposes irregular addressing
- Also, off-chip memory bandwidth increased!
 - 1/3 more data fetching
 - 8 times addressing



Proposal: Constrained VSP (CVSP)

Constrain VSP to maintain the same amount of pixel fetching from DPB as traditional MC

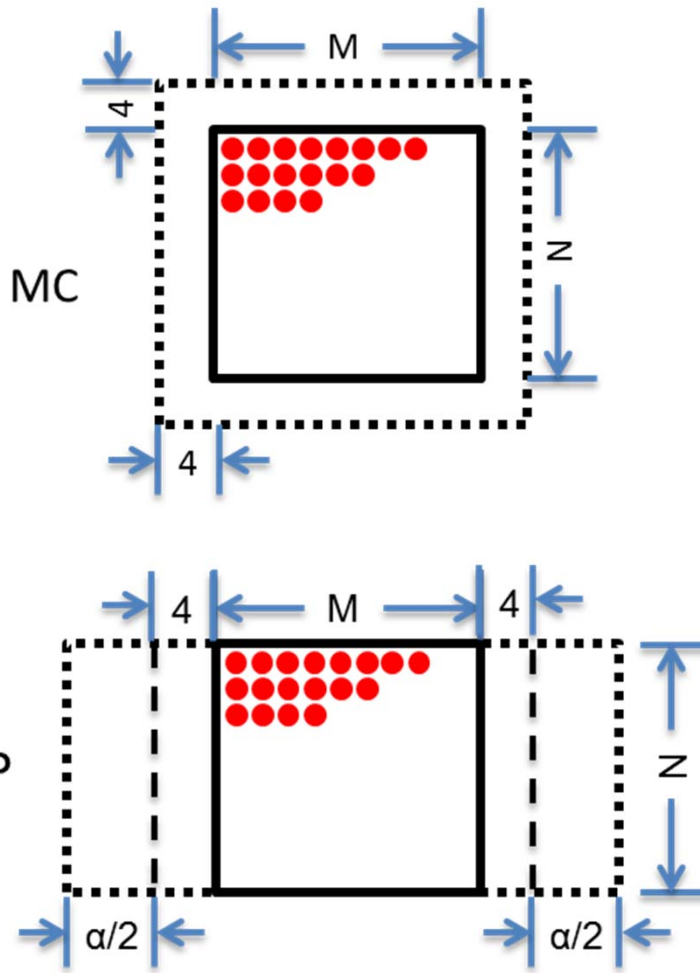
$$(M+7) \times (N+7)$$

Traditional MC



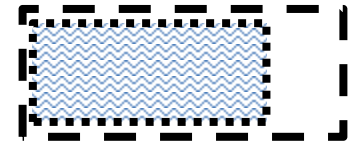
$$(M+\alpha+7) \times N$$

1x1 CVSP



Determine α to keep the number of referenced pixels **the same as before**

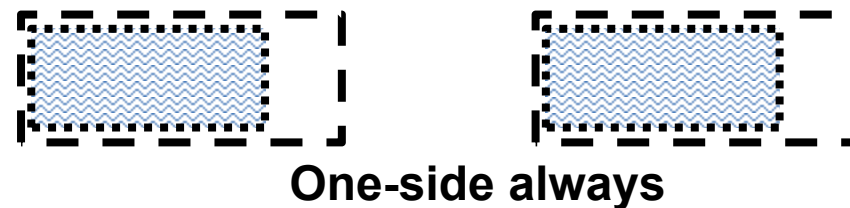
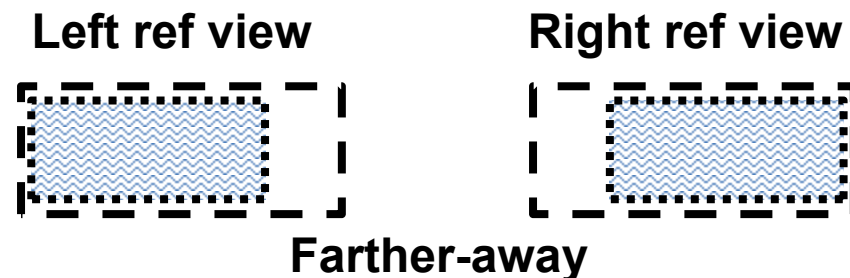
Proposed CVSP – Reference window size



M	N	Traditional MC (M+7)x(N+7)	CVSP (M+α+7)xN	α	M+α
4	8	192	165	10	14
8	4	192	165	26	34
8	8	256	225	13	21
8	16	384	345	7	15
16	8	384	345	20	36
16	16	576	529	10	26
16	32	960	897	5	21
32	16	960	897	17	49
32	32	1600	1521	9	41
32	64	2880	2769	4	36
64	32	2880	2769	16	80
64	64	5184	5041	8	72

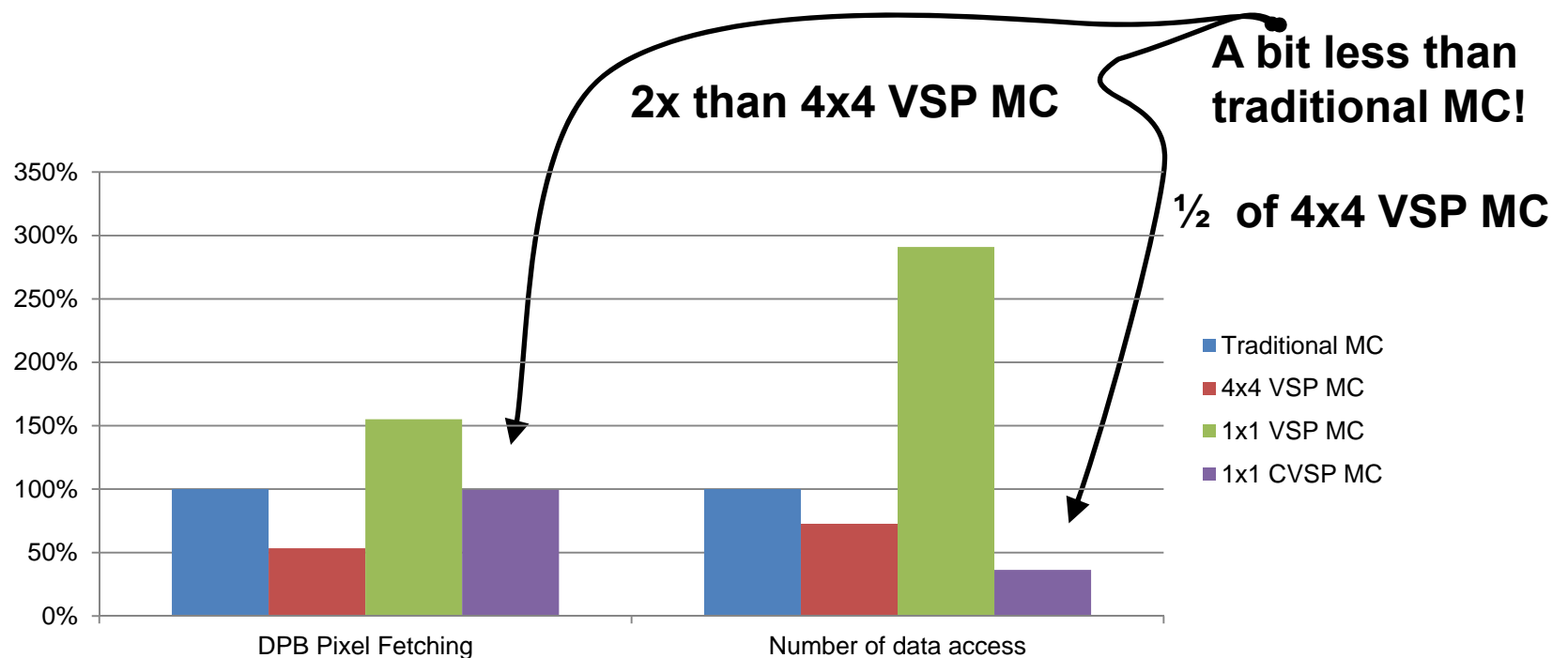
Proposed CVSP – apply constraints on depth

- Get the window width of the reference pixels
 - Determine α to keep the number of pixels not increased
- Get a DV for the reference pixels at PU level
 - Method 1: Farther-away alignment
 - Method 2: Left-side alignment



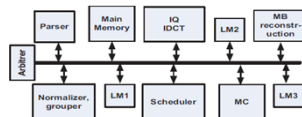
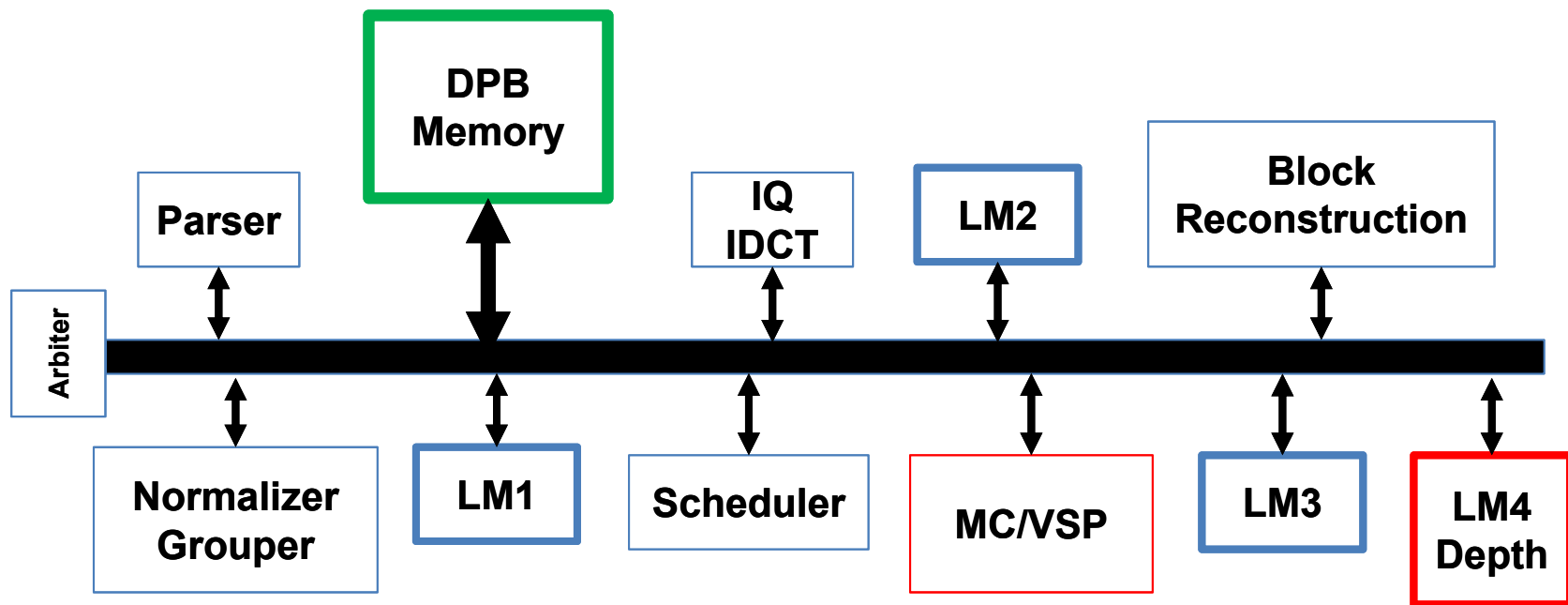
Off-chip memory access – same as traditional MC

- The proposed CVSP achieves the same off-chip complexity!
 - Amount of DPB pixel fetching
 - Number of data access (off-chip)
 - DV for reference pixels can be determined at PU level



HW block diagram for a decoder

- Block diagram of a typical decoder design
 - Modified to incorporate depth



H. Habli, et.al., Optimizing Off-Chip Memory Access Costs in Low Power MPEG-4 Decoder, *ICICS'12*, April 3–5, 2012,

Modified /Added blocks

- MC module
 - Modified to support both traditional MC and VSP MC
- New local memory*
 - LM4 for depth, ($\leq 4K$, 64×64 , or $64 +$ memory reuse)
- Convert the depth values in LM4 to disparity values*
- Convert the disparities to memory vector
 - May re-use LM4 memory
- Linear memory reading from DPB with proposed CVSP
 - Processing memory vectors in LM4*
 - Similar to processing memory vectors in LM1

* Shared with DoNBDV procedure

High level factors

- Total number of interpolation
 - Computation
 - On-chip memory bandwidth
- On-chip pixel loading
 - On-chip memory bandwidth

- DPB pixel fetching
 - Off-chip memory bandwidth
- DPB addressing
 - Computation

**On-chip
complexity**

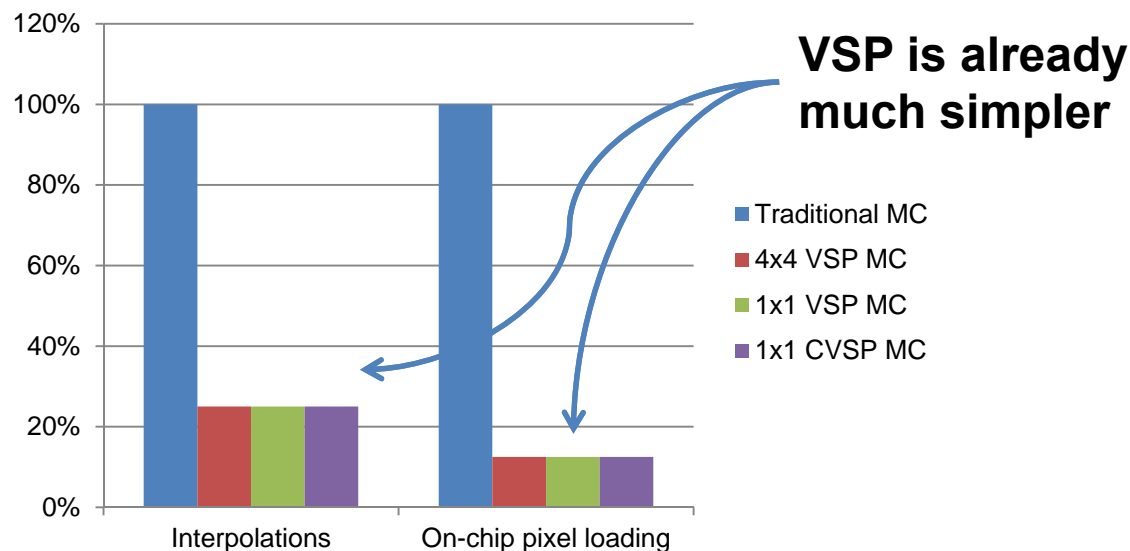
**Off-chip
complexity**

On a 64x64 block with all 8x4 PB partitions

On-chip complexity

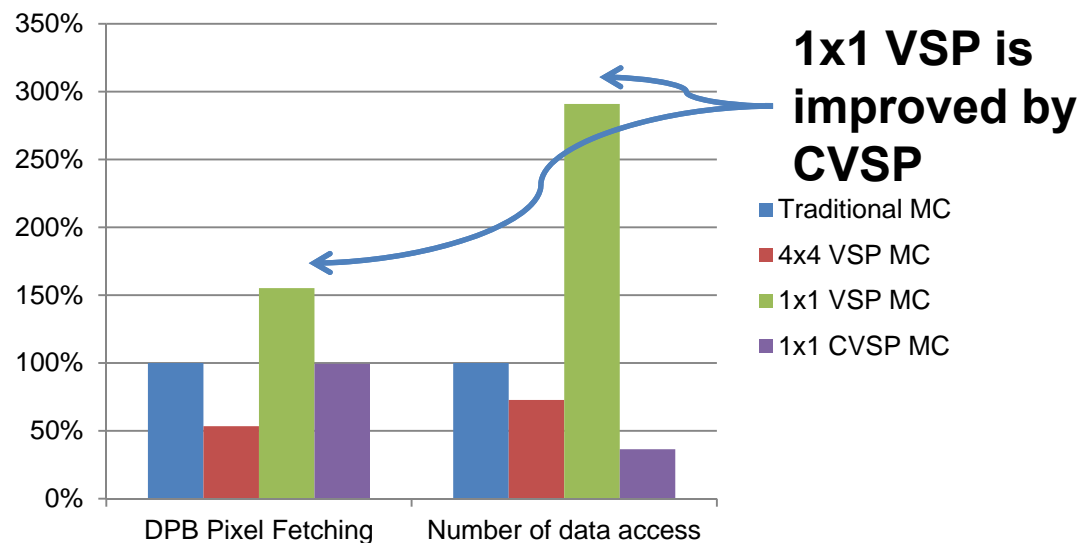
- Memory bandwidth
- Computation

Gap for on-chip pixel loading may be closer with data reusing



Off-chip complexity

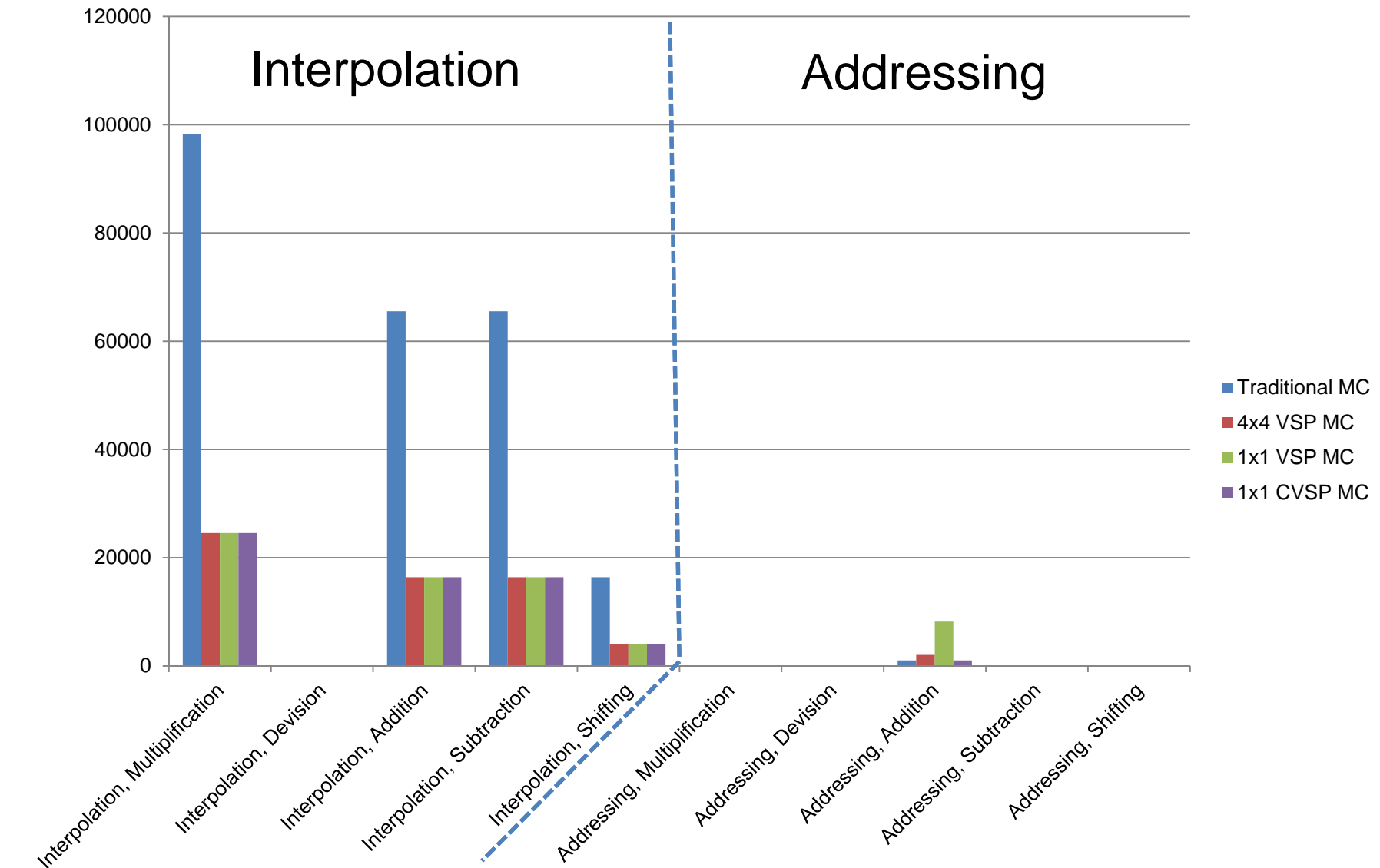
- Memory bandwidth
- Computation



Low level factors for arithmetic calculations

- Factors
 - Multiplication
 - Division
 - Addition
 - Subtraction
 - Shifting
- Arithmetic calculations
 - Interpolation
 - Address calculation

On a 64x64 block with all 8x4 PB partitions



Simulations on CVSP

- Results of “further-away” method
 - Overall bitrate saving is 0.2%
 - Almost without coding time changes

	video 0	video 1	video 2	video PSNR / video bitrate	video PSNR / total bitrate	synth PSNR / total bitrate	enc time	dec time	ren time
Balloons	0.0%	-0.3%	0.1%	0.0%	0.0%	-0.1%	105.3%	101.7%	106.8%
Kendo	0.0%	-0.1%	0.0%	0.0%	0.0%	0.0%	103.0%	91.6%	102.0%
Newspaper_CC	0.0%	-0.1%	-0.1%	0.0%	0.0%	-0.1%	101.7%	98.5%	102.0%
GT_Fly	0.0%	-0.3%	-0.4%	-0.1%	-0.1%	-0.1%	100.1%	99.6%	100.5%
Poznan_Hall2	0.0%	-0.6%	-0.2%	-0.2%	-0.2%	-0.1%	99.4%	101.8%	100.9%
Poznan_Street	0.0%	-0.1%	-0.1%	0.0%	0.0%	0.0%	100.3%	104.2%	96.5%
Undo_Dancer	0.0%	-3.7%	-3.0%	-1.0%	-0.9%	-1.1%	100.0%	103.7%	97.6%
1024x768	0.0%	-0.2%	0.0%	0.0%	0.0%	-0.1%	103.3%	97.3%	103.6%
1920x1088	0.0%	-1.2%	-0.9%	-0.3%	-0.3%	-0.3%	100.0%	102.3%	98.9%
average	0.0%	-0.7%	-0.5%	-0.2%	-0.2%	-0.2%	101.4%	100.2%	100.9%

1x1 VSP performance, well underestimated

- Though a full synthetic picture looks satisfactory, the use of VSP pixels still very low, only ~3%!
- In concept, VSP should be a very powerful tool to exploit interview redundancy
 - Why not more pixels VSP skipped? ~30%? ...



**Synthetic pictures
@ QP 40**

- Subjective measurements should be given high priority

Side-by-Side Comparisons

Evaluation aspects	Traditional MC	1x1 VSP	4x4 VSP MC	1x1 CVSP
Off-chip memory data rate	100%	155%	53%	99%
Number of data access (off-chip)	100%	300%	72%	36%
Off-chip random access	Yes	Yes	Yes	Yes
Off-chip line memory loading	Yes	No	Yes	Yes
Off-chip irregular addressing	No	Yes	No	No
Off-chip overall	Anchor	Much more complex	A bit more	Much less
On-chip interpolation computation	100%	25%	25%	25%
Number of data access (on-chip)	100%	25%	25%	25%
On-chip data rate (8x4)	100%	12%	12%	12%
On-chip data rate (8x4) w/ reuse	100%	142%	49%	142%
On-chip additional storage	0	4K	4K	4K
On-chip data reading	Uniform	Non-uniform	Half-uniform	Non-uniform
On-chip overall	Anchor	Simpler	Simpler	Simpler or similar
Encoder impacts	Motion estimation	No ME	No ME	No ME
Chip changes	Anchor	Yes	Yes	Yes
Bitrate saving	0.0%	1.0%	0.8%	1.0%

Conclusions

- 1x1 CVSP MC is much less complex than traditional MC
- 1x1 CVSP MC is much less complex than unconstrained 1x1 VSP MC
- 1x1 CVSP MC is less complex or similar to 4x4 VSP MC
- 1x1 CVSP MC maintains the performance of 1x1 VSP MC
- Recommend to adopt 1x1 CVSP
 - Higher performance with significantly reduced complexity

THANKS

Especially to
NTT, JCT3V-D0225
& **MediaTek, JCT3V-D0293**
for their valuable time in cross checking

Complexity comparison on an MxN block

Decoder Side Analysis	Traditional MC	4x4 VSP MC	1x1 VSP MC	Proposed CVSP MC (1x1)
A: Number of fetched pixels from DPB	$(M+7) \times (N+7)$	$4 \times (4+7) \times M \times N / (4 \times 4)$	$M \times N \times 8$	$(M+7+\alpha) \times N = (M+7) \times (N+7)$, α is the maximum disparity constraint
B: Number of data access (off-chip)	$N+7$ (line buffer read)	$N \times (M/4)$ (line buffer read)	$M \times N$	N (line buffer read)
C: Maximum total number of interpolation	$(8+1) \times M \times N$ (not consider data reuse) $(N+7) \times M + N \times M$ $= (2N+7) \times M$ (consider data reuse)	$M \times N$,	$M \times N$	$M \times N$ (interpolation used only for horizontal direction)
D: Number of read out pixels	$8 \times 8 \times M \times N = 64 \times M \times N$	$8 \times M \times N$	$8 \times M \times N$	$8 \times M \times N$

On a 64x64 block with all 8x4 PB partitions

High level

Decoder Side Analysis	Traditional MC	4x4 VSP MC	1x1 VSP MC	Proposed CVSP MC (1x1)
A: Number of fetched pixels from DPB	165*128=21,120 1x	88*128=11,264 ~1/2	256*128=32,768 1.55x	164*128=20,992 .99x
B: Number of data access (off-chip)	11*128=1408 (line buffer read) 1x	8*128=1,024 (line buffer read) 72%	32*128=4,096 3x	4*128=512 (line buffer read) 36%
C: Maximum total number of interpolation	288*128=36,864 (not consider data reuse) 128*128=16,384 (consider vertical interpolated 1x intermediate fractional data reuse)	32*128=4,096 ~1/8 or 1/4	32*128=4,096 ~1/8 or 1/4	32*128=4,096 (interpolation used only for horizontal direction) ~1/8 or 1/4
D: Number of read out pixels	2,048*128=262,144 1x	256*128=32,768 1/8	256*128=32,768 1/8	256*128=32,768 1/8

On a 64x64 block with all 8x4 PB partitions

Low level

		Traditional MC	4x4 VSP MC	1x1 VSP MC	Proposed CVSP MC (1x1)
Interpolation	X	$768 * 128 = 98,304$	$192 * 128 = 24,576$	$192 * 128 = 24,576$	$192 * 128 = 24,576$
	/	0	0	0	0
	+	$512 * 128 = 65,536$	$128 * 128 = 16,384$	$128 * 128 = 16,384$	$128 * 128 = 16,384$
	-	$512 * 128 = 65,536$	$128 * 128 = 16,384$	$128 * 128 = 16,384$	$128 * 128 = 16,384$
	>>	$128 * 128 = 16,384$	$32 * 128 = 4,096$	$32 * 128 = 4,096$	$32 * 128 = 4,096$
Addressing calculation	X	0	0	0	0
	/	0	0	0	0
	+	$8 * 128 = 1,024$	$16 * 128 = 2,048$	$64 * 128 = 8,192$	$8 * 128 = 1,024$
	-	0	0	0	0
	>>	0	0	0	0

About LM4, 4K or 64

- In NBDV, 64 bytes required, it is not necessary to buffer the whole depth block
- NBDV returns a DV from the fetched depth block
- CVSP returns a DV from the fetched depth block as well
 - Used for DPB pixel fetching
 - Indicating the starting address
- The above two processes can be harmonized
 - Can be line based processing
- The fetched depth line is required for MC
 - Can be line based processing
- If the 4K is allowed, one time depth fetching
- Otherwise, two times depth fetching ($64 \times 64 / 8 / 4 \times 2 = 256$ LM1)

On-chip pixel loading

- On-chip pixel loading without data reuse
 - 1x1 VSP MC is 1/8 of traditional MC
- On-chip pixel loading with data reuse
 - 8x4 traditional MC: 15 pixels/line * (4 lines + 8 padded lines) = 180 + data reuse logic
 - 4x4 VSP MC: 11 pixels/line * (4 lines + 0 padded lines) * 2 block = 88 + data reuse logic
 - 1x1 VSP MC: 8 pixels * 8 columns * 4 lines = 256 w/o data reuse logic