A MONOLITHIC PROGRAMMABLE ULTRA-HD VIDEO CODEC ENGINE

Hetul Sanghvi, Mihir Mody, Niraj Nandan, Mahesh Mehendale, Subrangshu Das, Dipan Kumar Mandal, Vyagrheswarudu Nainala, Vijayavardhan Baireddy, Pavan Shastry
Texas Instruments, Bangalore, India.
Email: {hetul,mihir,niraj,m-mehendale,s-das2,dipan,vyagrhee,viva,pavanvs}@ti.com

ABSTRACT

With advances in video coding standards like H.264 and HEVC coupled with those in the display technology, Ultra HD contents have started taking the mainstream. This is driving the need for high computation and memory bandwidth in current multi-media SOCs. In this paper, we present a monolithic multi-format video codec engine which achieves Ultra HD performance for H.264 High Profile, reduces the external memory bandwidth requirement by 2X as compared to its predecessor and takes only 5.9 mm^2 of silicon area in a low power 28nm process.

Index Terms— H.264, Ultra HD, full-HD, Architecture, VLSI, 4K, Multi-format, Video Engine, Cache.

1. INTRODUCTION

The demand for higher visual quality and immersive visual experience is driving the need to support Ultra-HD resolutions. Newer video coding standards like H.264 and HEVC is making it possible to store and share Ultra HD content in a reasonable size, thus increasing its acceptance. One option to achieve Ultra HD performance is by using 4 instances of a Full-HD video codec engine. This is however not only costly as it does not achieve the sharing of the common video codec engine infrastructure but also more complex as it requires efficient scheduling mechanism to balance the processing across different cores given the fact that it is tough to parallelize video codecs.

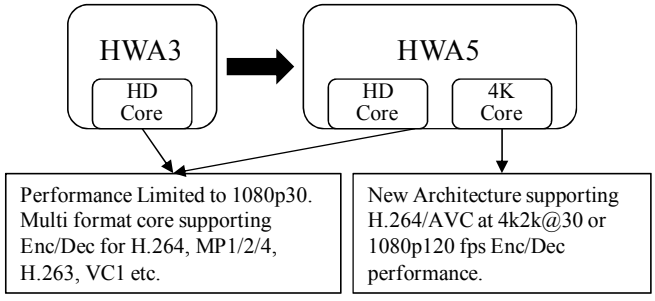


Fig. 1. Multiple format and performance point using two engines

In this paper, we present IVA-HD2, an integrated video codec engine that supports H.264 video codec standard up to Ultra-HD @ 30fps in a low power 28 nm process. It uniquely integrates additional logic over IVA-HD [1] to provide this high performance while still maintaining full-HD @ 30fps performance for other video compression standards including MPEG1/2/4, VC1 and others [Fig. 1]. Besides increasing the processing throughput of the codec engine by 4X, IVA-HD2 also optimizes the required external memory bandwidth by 2X over IVA-HD [1].

Fig. 2 shows the block diagram of IVA-HD2. It contains different hardware accelerators (HWA) for motion estimation (ME), spatial intra prediction (IPE), Transform and Quantization Engine (CALC), Motion compensation (MC [2]), De-blocking filter (LPF [3]) and Entropy codec (ECD). It also has a Video DMA Engine (VDMA) optimized for 2D block-transfers required in video processing [4]. The overall data flow control and interaction with the external CPU host is managed by two specialized Video RISC Processors (ICONT1/2). Inside IVA-HD2, there is also a shared Level-2 memory (SL2) for sharing pixel and control data (primarily MB level) between the HWAs. It also acts as a data buffer for pixel and control data that is fetched (stored) by VDMA from (to) external memory. Message Network is a low latency communication network that is used to indicate MB level task completion between HWAs. This indication is required by HWA to fetch shared data from SL2. Configuration network (CFG Interconnect) is used by ICONT1/2 and external Host Controller to configure the HWAs for different codec and frame level parameters.

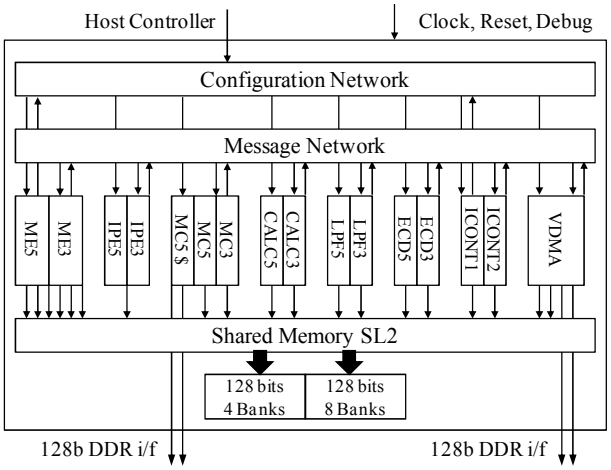


Fig. 2. Proposed block diagram of IVA-HDIVA-HD2 video engine

978-1-4799-2893-4/14/$31.00 ©2014 IEEE

The accelerators marked *v3* in block diagram denote full-HD performance for previous generation standards [1]. The accelerators marked *v5* refers to the ultra-HD H.264 engine done in this work. The integration of *v3* and *v5* is done to reduce area by maximizing infrastructure re-use e.g. SL2, messaging network, configuration network, etc.

2. VIDEO ENGINE ARCHITECTURE

2.1. Video Engine Performance

IVA-HD2 is built on top of IVA-HD [1] and its frequency was therefore fixed at 266 MHz. In order to process Ultra-HD frames (each containing 32K MBs) at 30 fps, a single MB is required to be processed in 270 cycles. Since there is multiple overheads at frame and slice level that are non parallelizable, the MB level budget is further reduced to 180 cycles.

Since the pipeline is switchable at a MB level, each MB operation requires some cycles to bring up the pipeline depending upon the depth of the pipeline. Given the requirement to complete the MB processing in 180 cycles, it is very important to reduce this pipeline bring up cycles. Different techniques used to reduce this include using local storage (L1 memory) for storing temporary data, using flop based storage for storing context information.

2.2. 2D Storage of data in SL2

Pixel data is typically stored in 1D or raster format in a video codec engine. However most video processing steps (like motion compensation) are block based requiring access to 2D block of pixels of arbitrary size and location in a video frame. In IVA-HD2, the pixel data is stored in 2D format [Fig. 3] inside SL2 memory. So a single access of SL2 memory (of 128 bits wide) will fetch a 4x4 pixel block instead of a 16x1 pixel block. This helps to improve the efficiency of data fetches for un-aligned accesses. For 16x16 block access, this 2D format organization reduces the SL2 bandwidth by 22% (32 to 25 fetches) and saves 60 mW of power for Ultra-HD encoder than a 1D formatted IVA-HD [1].

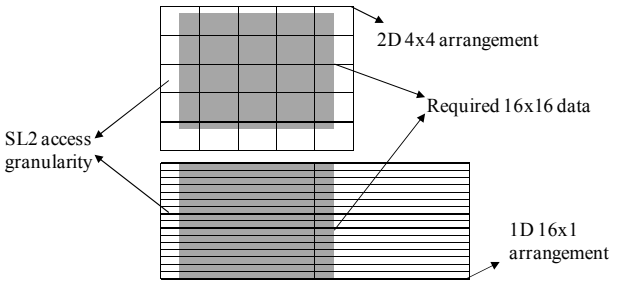


Fig. 3. Proposed 2D Block storage in internal shared (SL2) memory

2.3. Optimized Shared Memory Access

Even with 2D storage of pixel data, meeting UHD performance required close to 4X the shared memory bandwidth than previous. Since this was not feasible, it became very critical to optimize and reduce the shared memory bandwidth from every HWA. Any data that was used only for self consumption of the HWA was moved to local L1 storage inside the HWA like Left MB parameters in CALC, partially filtered pixels in LPF and fractional position interpolated pixels in ME.

2.4. Optimizing SDRAM accesses

Scaling the performance from Full HD to UHD requires a 4X increase of SDRAM bandwidth. Since it was only possible to support twice the memory bandwidth at the system level, it became very important to reduce SDRAM bandwidth requirement by at-least 2X than IVA-HD [1]. Each Macro Block in a frame is characterized by a certain meta-data aka MB Info. MB Info comprises of two parts – header information containing parameters like MB type and motion vector information (MV Info) containing parameters like number of partitions and their MVs. This meta-data is stored in external SDRAM and is read one or more times during video processing. In IVA-HD [1], the MB Info size is fixed whereas in IVA-HD2, the MV Info size is variable and scales with the number of partitions in the MB. This brings a significant reduction in DDR bandwidth and power.

Padding of reference pixels or otherwise extension of boundary pixels is required when the reference region fetches (done by ME and MC) lie outside the reference picture. In [1] the DMA engine performs this padding while storing the reconstructed frame in DDR. In IVA-HD2, the padding is done internally when regions beyond image boundary are referred, eliminating the movement of padded data to and from the SDRAM, thus reducing power.

3. OPTIMIZED ACCELERATORS

3.1. Correlation of ME Accesses from SL2

Increasing throughput from Full-HD to Ultra HD is a big challenge as it requires 4X more parallel processing assuming same frequency in both. Typical motion estimation algorithm for finding the best match for the current MB in the reference image involves hierarchically searching - skipping 4 pixels in the reference frame, then refining around the winner by skipping 2 pixels and further refining every pixel search. This is refined further using sub-pixel interpolation search. Fig. 4 explains this mechanism. These searches result in accessing same reference pixels several times. This correlation is exploited by storing the reference pixel data needed in local L1 memory inside ME. A special custom buffer was designed to enable accessing up to 36x4 pixels aligned at any vertical pixel position simultaneously as shown in Fig. 4. This resulted in 20% reduction in SL2 accesses in case of video encode – thereby saving significant amount of power.

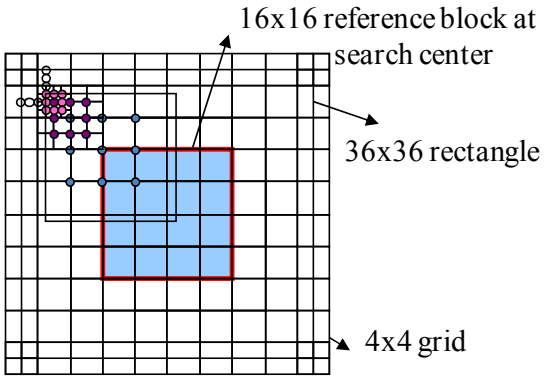


Fig. 4. 36x36 pixel window for 4:2:1 search followed by fractional search

3.2. Optimizing External SDRAM accesses in Encoder

Optimizing external SDRAM bandwidth required for encoder, especially for motion estimation, enables use of higher motion search range enabling a better quality (both subjective and objective) of motion estimation. In motion estimation, reference frame data is fetched from external SDRAM to perform search. It was observed that there is a significant overlap between the reference frame data required by adjacent MBs for motion search. To ensure reference frame data is only fetched once, ME uses a hybrid search window and overlap detection scheme where the search window, maintained inside SL2, has partly sliding and partly growing regions as shown in Fig. 5. Further, a 'bounding box' (BBox) is created of all the necessary reference data for a given MB search and only the non-overlapping regions of it (w.r.t. the previous MB) are fetched from the external SDRAM. This leads to large reduction in average DDR traffic, a key for 4k Ultra-HD encode.

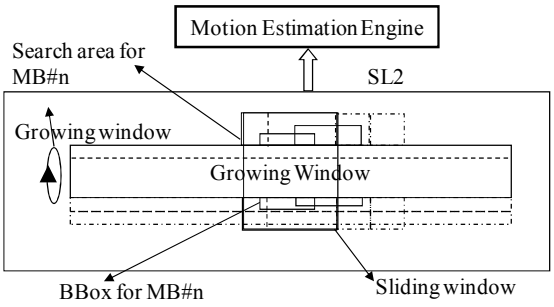


Fig. 5. Hybrid search window with overlap detection

3.3. Optimizing External SDRAM accesses in Decoder

Motion compensation (MC) takes the maximum SDRAM bandwidth in a video decoder. In [1] Bounding Box (BBox) was used to reduce DDR bandwidth. This technique uses ICONT to form a rectangular box encompassing the reference region fetch required by each partition within a MB. The technique fails when MB predicts from more than one reference in a direction - increasing DDR bandwidth and power.

To overcome this, as shown in Fig. 6, we implement a hierarchical reference pixel caching scheme which exploits the overlap in both Horizontal and Vertical direction between the Reference pixels required for every partition within a MB and across MBs. The principle is to de-generate the Reference pixel fetches for every partition into pixel blocks - 4x4 for Luma and 8x2 for Chroma and tag them. There is lot of locality in the data required for adjacent partitions and MBs. This fact is exploited to create a hierarchical caching scheme where multiple levels of comparisons are done to reduce the comparison cost. This is described in detail in [4] and [5]. In case of Ultra-HD content, horizontal caching is not sufficient to exploit overlap in reference data fetches in the vertical direction as it maintains small history. Vertical Cache, a simple 1-way associative cache operating from SL2, is designed to exploit this overlap. With both these knobs, we achieve a 50% reduction in average SDRAM bandwidth in the video decoder case. With these knobs, we achieve a 50% reduction amounting to 800 MBps in average SDRAM bandwidth in the decoder case.

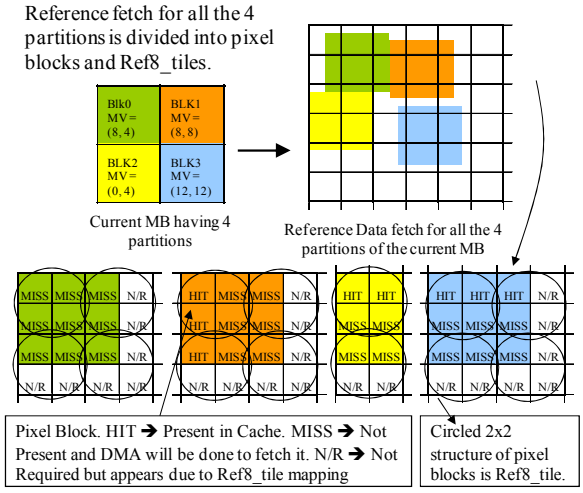


Fig. 6. Proposed 2D MC cache to reduce SDRAM access

3.4. Fast Entropy Codec

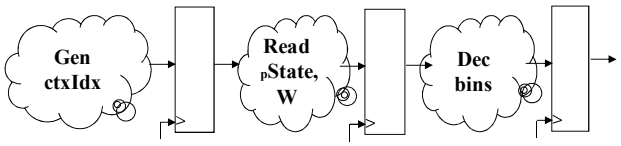
Context-based Adaptive Binary Arithmetic Coding (CABAC) is one of the tools of Entropy coding in H.264. CABAC can achieve bit-rate savings up to 14% but at the cost of higher computational complexity in comparison to CAVLC. Decoder complexity of CABAC engine is much higher that of encoder. The proposed design optimizes the data path to accommodate 3bins/cycle decode.

- Parallelize paths wherever possible
- Equalize the logic levels in all these parallel paths
- Avoid generating the redundant signals

The stages of CABAC decoding in implementation can be summarized as:

- ctxIdx computation
- Reading pState and valMPS
- Decoding the bin

As shown in Fig. 7, each of these stages takes almost one clock period. So we make the design with three pipelines.



CYCLE	0	1	2	3	4	5
ctxIdx Computed	0,1,2	3,4,5	6,7,8	9,10,11	12,13,14	15,16,17
ctxIdx reg		0,1,2	3,4,5	6,7,8	9,10,11	12,13,14
pS,w read		0,1,2	3,4,5	6,7,8	9,10,11	12,13,14
pS,w reg			0,1,2	3,4,5	6,7,8	9,10,11
Bins computed			0,1,2	3,4,5	6,7,8	9,10,11
Bins reg				0,1,2	3,4,5	6,7,8

Fig. 7. CABAC decode stages and cycles

As we can see, the CABAC core takes (16+32+64) ctxIdx's, from which 3 ctxIdx are chosen based on values of bins (6,7,8,9,10) that are available in cycle 5.
Exploiting the similarities between decode and encode, the design can perform both decode and encode operations. This saves the area of the extra hardware for encode.

4. OPTIMIZED CONTROLLER FOR EFFICIENT, LOW POWER PROCESSING

IVA-HD2 uses a customized RISC processor (ARP32) to build an efficient, low power control processing unit (ICONT) specialized to process frame/slice headers, bit rate control algorithms etc. up to 10x times faster than a normal embedded CPU (e.g. an ARM968 CPU). It provides efficient modes with low latency of wake via interrupt and other events allowing energy efficient processing of multi-slice video frames and other tasks of IVA-HD2 sub-system.

5. FIRMWARE LEVEL OPTIMIZATION

Typically, the firmware running on local controller uses simple sequential model for video processing from hardware due to lack of availability of real-time OS (RTOS) as well as tight on-chip memory budgets. This works well for simple video playback scenario. In case of video surveillance and infrastructure, which requires large number of video channels of lower resolutions or video conferencing market with higher number of slice processing results in much lower utilization of video hardware as inefficiencies becomes significant resulting in lower performance. As described in [6], this work implements novel approach by defining software based video codec framework (aka VCF) to enable multi-threading the local ARP32 controller in firmware. The proposed framework defines model for controller firmware (in terms of software acceleration, static model for data flow, state diagram etc). Simple primitives (e.g. threads, queues, semaphores, messaging, scheduling scheme etc) defined by the VCF helps achieve this without using any RTOS. The prior approach was single monolithic firmware component (e.g. H.264 decoder) to control hardware. VCF proposes next level granularity i.e. multiple firmware components which has standard API interface & behaviors as shown in Fig. 8 below.

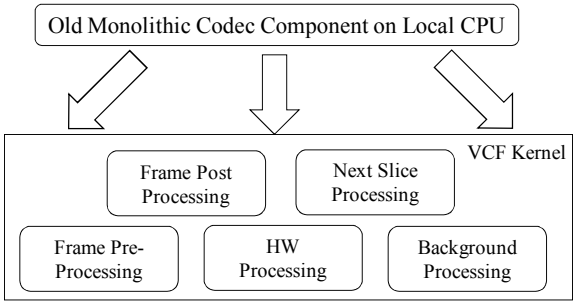


Fig. 8. Proposed Firmware Component model in VCF

6. RESULTS

The table below compares the present work with [1].

	This work	ISSCC 2012[1]
Process	28nm LP CMOS	45nm LP CMOS
Supply	1.0 V & AVS	1.1V & AVS
Full-HD Freq	66 MHz	266MHz
Standards	14(D) + 9 (E)	14 (D) + 9 (E)
H.264 Performance	4Kp60(D) / 30(E)	1080p60(D) / 30(E)
Full-HD Power	9(D)/18(E) mW	65(D)/100(E) mW
Full-HD DDR BW (H.264 HP Dec)	200 MBytes/s	400 MBytes/s

7. CONCLUSION

In this paper, several techniques spanning across hardware and software were described to achieve a UHD video codec engine for mobile applications in a hitherto unseen power envelop enabling 14 hours of video playback time over HDMI. Besides improving the processing capability by 4X, it also significantly reduces the required external memory bandwidth by 2X a key to enable UHD performance at the system level.

8. ACKNOLEDGEMENT

Developing an IP of this complexity requires collaboration of cross-functional teams. We sincerely thank M Sharma, R Reddy, H Tamama, AD Gupte, Y Matsuba, S Sajayan, RR Srinivasan, N Acharya, S Ramaiyan, P Karnamadakala, B Holla, M Sadafale and D Sharma for their contribution in development of this IP.

9. REFERENCES

[1] M. Mehendale, M. Mody, H. Tamama, et.al, "A true multi-standard, programmable, low-power, full HD video-codec engine for smartphone SoC", Proc. IEEE ISSCC, Feb 2012.

[2] H. Sanghvi, "Low Power Architecture for Motion Compensation in a 4K Ultra-HD AVC and HEVC Video Codec System", IEEE ICIIP, 2013.

[3] N. Nandan and M. Mody, "Scalable High performance Loop filter architecture for video codecs", IEEE ICIIP, 2013.

[4] N. Nandan, "High Performance DMA Controller for Ultra HDTV Video Codecs", IEEE ICCE, 2014.

[5] H. Sanghvi, "2D Cache Architecture for Motion Compensation in a 4K Ultra-HD AVC and HEVC Video Codec System", IEEE ICCE, 2014.

[6] Mihir Mody, “Video Codec Framework (VCF): Novel Firmware Architecture for Video Hardware”, National Conference on Communication (NCC), 2014.