# A 1.3 megapixel FPGA-based smart camera for high dynamic range real time video

Pierre-Jean Lapray*, Barthélémy Heyrman*, Matthieu Rossé* and Dominique Ginhac*
*LE2I UMR 6306, Univ Burgundy, Dijon, France
Email: plapray@gmail.com, (heybar,dginhac,mrosse)@u-bourgogne.fr

***Abstract*—A camera is able to capture only a part of a high dynamic range scene information. The same scene can be fully perceived by the human visual system. This is true especially for real scenes where the difference in light intensity between the dark areas and bright areas is high. The imaging technique which can overcome this problem is called HDR (High Dynamic Range). It produces images from a set of multiple LDR images (Low Dynamic Range), captured with different exposure times. This technique appears as one of the most appropriate and a cheap solution to enhance the dynamic range of captured environments. We developed an FPGA-based smart camera that produces a HDR live video colour stream from three successive acquisitions. Our hardware platform is build around a standard LDR CMOS sensor and a Virtex 6 FPGA board. The hardware architecture embeds a multiple exposure control, a memory management unit, the HDR creating, and the tone mapping. Our video camera enables a real-time video at $60$ frames per second for a full sensor resolution of $1,280 \times 1,024$ pixels.**

## I. Introduction

Conventional cameras have a limited dynamic range, much less than the human vision system. In many imaging systems, saturated zones in the dark and illuminated areas of the captured image still a problem. These limitations are due to large variations of real scene radiances, with over and under-exposed areas appeared in the single image. The sequential capture of several images with different exposure times can deal with the lack of information in extreme lightning conditions. This way is called the HDRi (High Dynamic Range imaging) system.

### A. HDR imaging system

Creating an HDR image is done in three steps:

- Recover the response curve of the system,
- Blend pixels into radiance values,
- Perform Tone Mapping to display the resulting HDR images onto a standard monitor.

In an ideal configuration, create an HDR image is done by simply reducing low dynamic images in the same configuration by dividing each image by their exposure time (normalization), then by summing the pixels of the images. However, due to the presence of complex electronic circuits, most of the cameras have a non-linear processing about the transformation of the incident light. This non-linear transformation is transposed by the inverse of the response curve of the camera, noted $g$. This feature must be known before the estimation of the real radiance values, because it determines the relationship between the light striking the sensor and the corresponding pixel values. For most devices, the function f is specific to each system (lens + opening + sensor), and must be estimated as best accurate as possible to reconstruct the HDR image. Since 1995, several algorithms [1], [2], [3], [4] have emerged, to evaluate as precisely as possible the function $g$.

The estimation of the real radiance values can be followed by a dynamic range mapping, in order to visualize the result on an LDR display. This method is called the tone mapping and it is used to render the HDR data to match dynamic of conventional hardware display. For example, it can convert 32-bit wide pixels to 8-bit wide pixels ([0,255]). There are two types of tone mapping operators (TMO): spatially uniform TMO (i.e. global TMO) and spatially non-uniform TMO (i.e. local TMO). In our case, several algorithms seem to be implementable in real time due to fast computation capabilities, whether global or local. Following is a list of software methods (from the fastest to the slowest algorithm) whose efficiency and simplicity of the calculations are shown in the literature:

- Drago et al. [5] (*Adaptive Logarithmic Mapping For Displaying High Contrast Scenes*)
- Duan et al. [6] (*Tone-mapping high dynamic range images by histogram novel Adjustment*)
- Reinhard et al. [7] (*Photographic Tone Reproduction for Digital Images (global algorithm)*)
- Durand et al. [8] (*Fast Bilateral Filtering for the Display of High-Dynamic-Range Images*)
- Fattal et al. [9] (*Gradient Domain High Dynamic Range Compression*)
- Tumblin et al. [10] (*Time-dependent visual adaptation for fast realistic image display*)

In this paper, we propose a HDR smart camera based on a parallel architecture dedicated to real-time HDR video content. What is new in this paper is shown through four steps. First, we capture images from the sensor with alternating three exposure times, selected by our Multiple Exposure Control (MEC). Then, we manage reading and writing operations in memory in order to have several video streams in parallel, corresponding to the different exposure times. Under a highly parallel context, we blend the three video streams together with a modified version of a HDR technique. Finally, an hardware implementation of a global tone mapping technique is performed. We will begin by describing existing works about HDR video technique in Section II. Then, in Section III, we will describe our system in detail. Finally, some experiments and results will follow this discussion in Section IV. Concluding remarks are then presented.

| Method | Hardware | Capture | HDR Fusion | Frames used for HDR | Tone Mapping | Resolution | FPS |
|---|---|---|---|---|---|---|---|
| Akyüz et al. [11] | GPU | no | yes | 9 | yes | - | 65 |
| Mann et al. [12] | FPGA | yes | yes | 3 | yes | $1,280 \times 720$ | 120 |
| Ureña et al. [13] | GPU/FPGA | no | no | - | yes | $640 \times 480$ | 30/60 |
| Guthier et al. [14] | CPU+GPU | yes | no | - | no | $640 \times 480$ | 25 |
| Ching-Te et al. [15] | ARM SOC | no | no | 3 | yes | $1,024 \times 768$ | 60 |
| Bachoo et al. [16] | CPU+GPU | no | yes | 3 | - | $1,600 \times 1,200$ | 20 |

TABLE I. SUMMARY OF THE MAIN EMBEDDED REAL-TIME ARCHITECTURES DEDICATED TO HDR.

## II. RELATED WORK

We detail here the existing hardware architecures. We limit ourselves to recent systems which can operate in real time, whether they are focused exclusively on capture, HDR creating or tone mapping. Table I summarizes these methods.

In 2012, Akyuz et al. [17] developed a complete system on a GPU platform. The tasks are performed in parallel with a pipelined structure. Generating HDR and the tone mapping are done without knowing the response curve of the camera. They use the algorithm originally proposed by Debevec et al. [2] to estimate the radiance values. Regarding to the operation of the tone mapping, it is the Reinhard et al. [7] algorithm which has been chosen and implemented. Some results are identical compared to other methods implemented on CPU. They reach a framerate of 65 fps for producing HDR images, and 103 frames per second for performing the tone mapping. However, they do not have time to load textures on the GPU. The majority of time is spent in sending pixels to the GPU. Radiance computations and weighting have little impact on the speed calculation, and the framerate of the final system.

The most popular complete vision system is based on the Mann architecture [12]. In 2012, a welding helmet composed of two computer-controlled video cameras has been presented. The data received by these cameras are recorded line by line in an external memory SDRAM. Several FIFOs store pixels and read them simultaneously line by line. The number of FIFOs depends on the number of images used in the HDR reconstruction. A LUT containing precomputed values is used to combine multiple exposures. This LUT is inspired of the work by Ali et al. [18], the estimation of radiances is done with a CCRF (”Comparametric Camera Response Function”). With this method, they are able to obtain a video with a fixed latency, and a controlled calculation (real-time) on a Xilinx Spartan-6 LX45 FPGA.

Ureña et al. [13] published in 2012 two tone mapping architectures, described both on GPU and FPGA. The implementations were done on a battery portable operating circuit. A new generation of tone mapping is presented in this article, rather than considering existing operators, because they require many computation time and memory. The tone mapping operator includes both local and global calculation. Typically, for the overall look, it highlights areas containing low contrasts, but can also protect areas where the contrast is well. Locally, it reduces the areas that are too bright in order to improve the image details. The overall improvement is based on the brightness histogram adaptation of each channel in the HSV colour space. On the other hand, the local enhancement is based on the retina-like technique. To summarize, the Gaussian filters, the weighting and the human visual system consideration are the main advantages of the operator. The FPGA implementation produced a video with a high frame rate, consuming little electric power, while the GPU implementation provides greater sensibility in the calculation of HDR pixels, but uses a lot of resources. The two systems can be reduced to a mobile application running on batteries.

In 2012 Guthier et al. [19] introduced an algorithm with a good HDR quality, that can be implemented with the same number of capture LDR. The choice of exposures is performed optimally selecting the better shutter speeds that will add the more useful information to contribute to the final HDR image. Their context can be real-time, by minimizing the number of catches. Basically, the exposure times are chosen so that the brightness value at a position $i$, $j$, where at least one LDR image captured has a well exposed pixel. First, a good approximation of the value radiance $E$ is calculated taking into account the response function of the camera and a contributing function. A weighting function, also contributes to the value of the final radiance. The histogram of radiances is used to calculate a sequence of shutter speeds choosing it corresponding to and the peaks of the contribution function. A useful relationship is made between the histogram of radiance vector and the contribution that indicate potentially changes in the scene. A stability criterion is also introduced to the sequence which allows each frame to be adjusted until a stable shutter sequence is found. Finally, with this algorithm, they save capturing time and are able to reduce the number of LDR exposures without loss of quality at the end of the computation.

Ching-Te et al. [15] suggests a methodology to develop a tone mapping processor optimized using an ARM SOC platform (System On Chip). Their processor evaluates both photographic compression method by Reinhard et al. [7], and the gradient compression method by Fattal et al. [9], for different applications. The new processor can compress $1,024 \times 768$ HDR images at 60 fps. The core needs $8,1mm^2$ of physical area with $0.13m$ TSMC technology.

Bachoo [16] developed a dedicated technical application of exposure fusion (initiated by Mertens et al. [20]), to merge a real-time $1600 \times 1200$ video at 20 fps using three black and white videos. They are able to control the speed of image generation, to have a constant frame rate, relative to the defined processing block size. They perform an alternative Goshtasby algorithm [21]. The implementation is done on CPU and GPU. The algorithm is divided into two parts so that the power of the CPU processing and GPU (”Graphics Processing Unit”) is used wisely. The CPU perform massively sequential operations such as calculating entropy blocks. The GPU is used to merge the blocks together, operation which can be parallelized to increase execution speed of the fusing process. The speed can be increased if the video resolution is reduced or if the size of processing blocks increases. As this, a compromise between calculation speed and quality can be chosen. Nothing is said about the choice of exposure time and no method is proposed to estimate exposures. It is recorded that the use of additional exposures may produce a bottleneck in the fusing process.

## III. A Dedicated HDR Smart Camera

The dedicated hardware system is built around a CMOS sensor board and an evaluation board from Xilinx (see Figure 1(a)). The parallel architecture presented in this paper operates in several stages. At the first stage, an FPGA input interface receives sequentially three pixel streams (produced by the e2v sensor), and stores them to a SDRAM memory as colour frames. A Multiple Exposure Control (MEC) based on the histogram computation also operates in parallel to select the proper exposure times. It changes the sensor configuration each time an image is captured. At the same time, a memory management core reads the previous frames stored into the SDRAM, and delivers it as a three live parallel video outputs. At the third stage, the different pixel streams are combined using the Debevec et al. algorithm [2], knowing the response curve of the imaging system and the exposure times. This stage produces a complete radiance map of the captured scene. Finally, the High Dynamic Range frame is tone mapped by the Reinhard et al. algorithm [7] and can be displayed on a standard LCD monitor. This full process is continuously updated in order to perform a real time HDR live video at 60 fps with a $1280 \times 1024$-pixel resolution. Our real-time constraint is that the availability of a new HDR data from the LDR captures must not exceed a fixed latency of $1ms$, guaranteeing that the HDR process is imperceptible to the viewer.

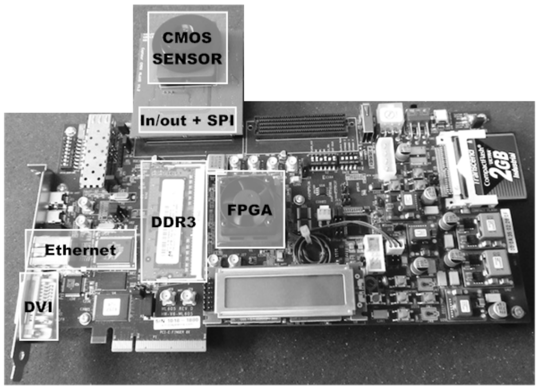
### A. Multiple Exposure Control

Our camera must be automatically adapted to the illumination level, just as the human eye do. So, a best set of exposures have to be captured. But, when we perform HDR stitching, traditional auto exposure algorithms fail. We present a similar approach of a previous state of the art algorithm, adapted to our real-time hardware requirements. Our sensor is able to send us the complete image histogram. Using the histogram of each image will allow to have a real-time preview of the total range of brightness that will be recorded. Gelfand et al. [22] use the FCam programmable API to capture two images, alternating short and long exposure. They require that fewer than 10% of the pixels in the images are bright for the short exposure, and require that fewer than 10% of pixels have values less than 16 for the long exposure. When the two exposures are stable, the metering is complete and they perform pseudo-HDR computation. They do not take into account future changes in light conditions in the captured scene, which could corrupt their dynamic range value. We use a similar approach to select three proper exposures $\Delta t_L$, $\Delta t_M$ and $\Delta t_H$ related to our scene.

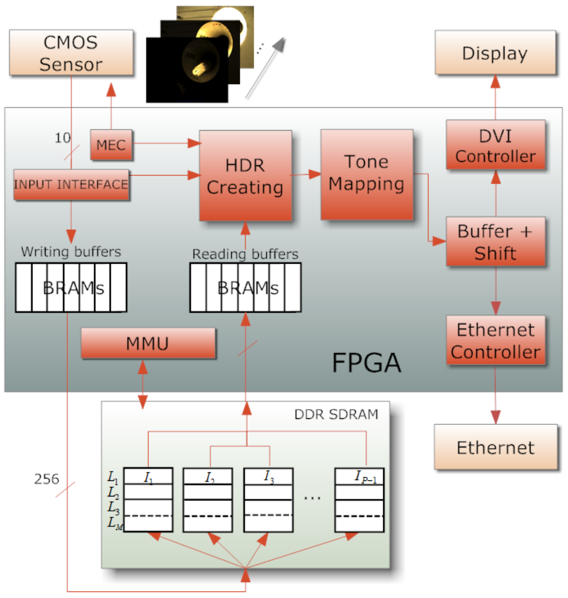
Histogram comes through 64 categories, coded with 16-bit depth representation. Depending on which image is receiving (low ($I_L$) or high exposure ($I_H$)) we apply these functions:

$$Q_L = \sum_{h=1}^{h=4} \frac{q(h)}{N} \qquad Q_H = \sum_{h=60}^{h=64} \frac{q(h)}{N} \tag{1}$$

where $Q_L$ and $Q_H$ are respectively the proportion of pixels present on a specific part of the histogram (among $N$ pixels



(a)



(b)

Fig. 1. Overview of our HDR smart camera

which compose an image). $q_h$ is the quantity of pixel in each histogram category $h$. Calculation is made with the first four and the last four category histograms of images $I_H$ and $I_L$. As output pixels are coded with 10-bit (coded between 0 to 1023), four categories correspond to a range of 64 pixel values. Then, we calculate two parameters for the two extreme exposure times like this:

$$\delta Q_{L/H} = |Q_{L/H} - Q_{L/H,req}| \tag{2}$$

where $Q_{L/H,req}$ is the required quantity of pixel for the part of histogram concerned. $\delta Q_{L/H}$ determines how far is the quantity of pixel with the desired quantity. Once we have these parameters, we can perform a series of decisions:

$$\Delta t_{L/M/H,t+1} \leftarrow MEC(\Delta t_{L/H,t}) \tag{3}$$

$$\Delta t_{L,t+1} = \begin{cases} \Delta t_{L,t} \pm 1x \text{ for } \delta Q_L > thr_{Lm} \\ \Delta t_{L,t} \pm 10x \text{ for } \delta Q_L > thr_{Lp} \end{cases} \tag{4}$$

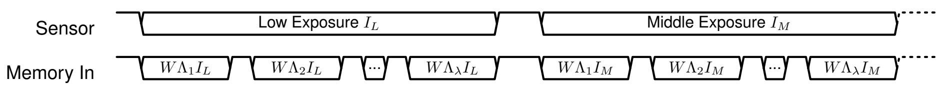


Fig. 2. Memory Management Core initialization. The sensor send sequentially low ($I_L$) and middle ($I_M$) exposure times. Writing operations into memory of each rows $\Lambda$ indexed by $\lambda$ of the first two frames.

$$\Delta t_{H,t+1} = \begin{cases} \Delta t_{H,t} \pm 1x \text{ for } \delta Q_H > thr_{Hm} \\ \Delta t_{H,t} \pm 10x \text{ for } \delta Q_H > thr_{Hp} \end{cases} \quad (5)$$

$$\Delta t_{M,t+1} = \sqrt{\Delta t_{L,t} \Delta t_{H,t}} \quad (6)$$

where $thr_m$ (minus) an $thr_p$ (plus) are the threshold values in order to have 2 levels of precision on how we will adjust the exposure values. $x$ is an extra exposure time corresponding to the integration time of one sensor row. $\Delta t_{L/M/H,t+1}$ are the exposure times which will be programmed into the sensor for the next frames $I_{L/M/H}$. The middle exposure time is a combination of $\Delta t_L$ and $\Delta t_H$. As we work with highly parallelizable architecture, we can do estimations of the dynamic range of the scene during HDR processing. If needed, we correct the exposure times if lighting conditions have changed.

## *B. MMU*

The use of external off-chip memories is judicious for our application that processes large amount of data and high data rates. For our case, video processing requires two frames of data to be stored (the third frame is given by the sensor). In practice, this storage is implemented using DDR3 SDRAM chip which is a part of our hardware development platform. It requires fast and efficient direct memory access logic to achieve high dynamic range video in real-time.

The sensor is able to send full-resolution images at 60 frames/s. Initialization of our specific HDR Memory Management Core is shown in Fig. 2. $I_L$ and $I_M$ are first stored in DDR3 memory. The first frame ($I_L$) is stored row by row $W\Lambda_\lambda I_L$, where $\lambda$ indexes row number ($1 <= \lambda <= 1024$). For example $W\Lambda_1 I_L$ means "writing of the first row $\Lambda 1$ of $I_L$ into memory". Each row write operation is followed by inter-row delay, due to horizontal sensor synchronization. For the second frame $I_M$, the image is also stored row by row ($W\Lambda_\lambda I_M$). This initialization step is required before the generation of the first HDR frame. We can't avoid waiting for these two first exposures. After this step, the Memory Management Core can start (see Fig. 3).

During the capture of the last frame ($I_H$), rows of the two previous frames stored are synchronously read from the memory during inter-frame ($R\Lambda_\lambda I_L$, $R\Lambda_\lambda I_M$) and buffered into Block RAMs (BRAMs) while each new captured row ($W\Lambda_\lambda I_H$) is stored in memory. It's important to notice that the design is a pure-hardware system which is processor-free and must be able to absorb a continuous pixel flow of about 80 MegaPixels per second from the sensor (called "Memory In" in Fig. 2 and in Fig. 3) while reading two other pixel flows corresponding to the two stored images (respectively called "Memory Out 1" and "Memory Out 2" in Fig. 3).

The HDR content is computed with the methods described in Sections III-C and III-D. The HDR process needs a continuous stream of pixels of three images and then can only be performed while receiving the third frame $I_H$. Then, the process can iterate throughout the capture of the fourth frame (low exposure $I'_L$) and the readout of the second and third frame ($I'_M$ and $I'_H$). Finally, our memory management system is able to deliver two parallel pixel streams that have been acquired and stored into the memory and a third pixel stream directly from the sensor. With this technique, each HDR pixel only requires three memory accesses (one write and two read operations during one row interval), saving many memory access operations. The main advantages of such a technique are (1) to store only two images in memory, and (2) to avoid the waiting for the three images to compute an HDR image. A latency corresponding to 136 clock rising-edges (i.e. $1.2us$ for a $114MHz$ system clock) is required by the system to create HDR tone mapped data (grey part of HDR output in Fig. 3) from the three captured lines. And then, it delivers an HDR video stream at 60 fps directly updated at each time the sensor sends an image.

## *C. HDR creating*

The evaluation of the inverse of the response curve of the system $g$ only requires the evaluation of a finite number of values (typically $1,024$ values for a 10-bit precision sensor), as depicted in the paper of Debevec et al. [2]. These values can be preliminary evaluated from a sequence of several images, then stored in the camera, and reused further to convert pixel values. For our case, the curve $g$ has been calculated, in a first step, using the Matlab code provided with the Debevec paper. In a second step, this curve has been stored in LUTs on the hardware platform. For recovering the HDR luminance value $E_{ij}$ of a particular pixel, all the available exposures of this pixel are combined using the following equation:

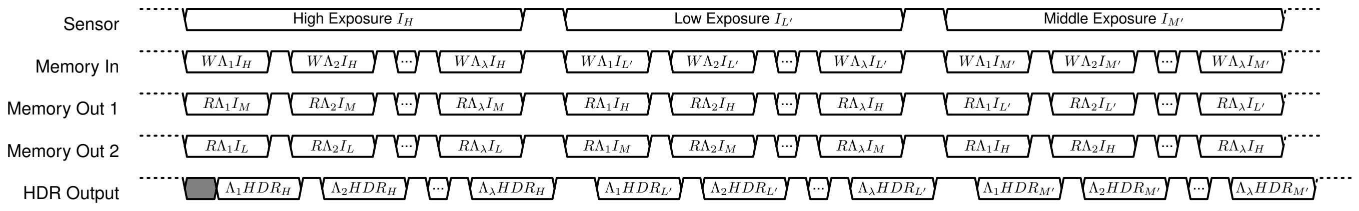


Fig. 3. Memory Management Core. Performing three parallel streaming videos with low ($I_L$), middle ($I_M$) and high ($I_H$) exposure times. The delayed HDR row output is shown after HDR and tone mapping computations (related to Section III-C and III-D.

$$\ln E_{ij} = \frac{\sum_{p=1}^{p=3} \omega'(Z_{p,ij})[g(Z_{p,ij}) - \ln \Delta t_p]}{\sum_{p=1}^{p=3} \omega'(Z_{p,ij})} \quad (7)$$

where $p$ indexes the image number, $i$ and $j$ indexes pixel position and $\omega'(z)$ is a weighting function giving higher weight to values closer to the middle of the function:

$$\omega'(z) = \begin{cases} z - Z_{min} \text{ for } z \leq \frac{1}{2}(Z_{min} + Z_{max}) \\ Z_{max} - z \text{ for } z > \frac{1}{2}(Z_{min} + Z_{max}) \end{cases} \quad (8)$$

where $Z_{min}$ and $Z_{max}$ values depend on the sensor output dynamic (typically $1,024$ values for a 10-bit precision sensor). Considering $Z_{1,ij}$, $Z_{2,ij}$ and $Z_{3,ij}$ as $Z_{L,ij}$, $Z_{M,ij}$ and $Z_{H,ij}$ in a 3-frame HDR system, the overall scheme is visible in the pipeline architecture depicted in Fig. 4

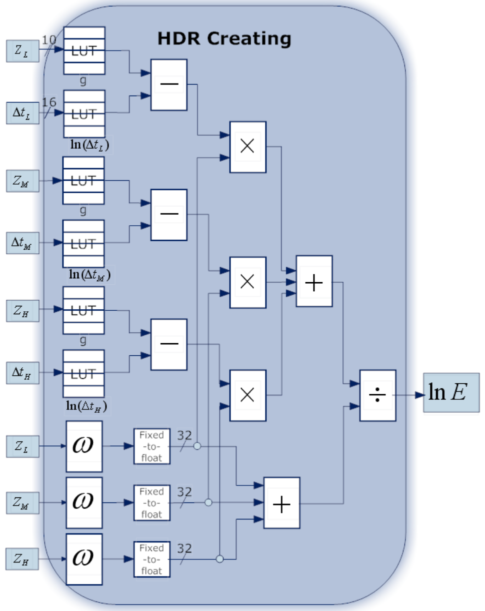


Fig. 4. HDR creating using three different pixel streams.

Computation of luminance values requires the use of 32-bit arithmetic operators (substractors, multipliers etc.) and transition from 10-bit to IEEE754 32-bit wide (called "Fixed-to-Float" in Fig. 4). LUTs are used to store the inverse of the response curve $g$ and make the transition from exposure time values $\Delta t_{L/M/H}$ to neperian logarithm field.

### D. Tone Mapping

Once the radiance map is recovered, HDR image pixels have to be mapped to the display range of a selected material, typically a LCD monitor. In our case, the displayable range is $2^8$ values. The global part of Reinhard et al. [7] algorithm requires one global computation: the log average luminance found in the image, calculated by:

$$\bar{E}_{ij} = \exp\left(\frac{1}{N}\sum_{i,j} \ln(\delta + E_{ij})\right) \quad (9)$$

where $E_{i,j}$ is the scene luminance for pixel $(i,j)$, $N$ is the total number of pixels in the image. For our case, due to signal noise, we cannot have zero pixel value, so we can assume that $\delta = 0$. The summation is only over non-zero pixels. Then, we want to map the middle grey scene luminance to the middle-grey of the displayable image. For the photographic tone reproduction operator, an approach is to scale the input data such that the log average luminance is mapped to the estimated key of the scene $a$, where $a$ is a scaling constant appropriate to the illumination range of the image scene:

$$D_{ij} = 255 \cdot \frac{a\frac{E_{ij}}{\bar{E}_{ij}}}{1 + a\frac{E_{ij}}{\bar{E}_{ij}}} = 255 \cdot \frac{1}{1 + \frac{\bar{E}_{ij}}{a \cdot E_{ij}}} \quad (10)$$

The tone mapping pipeline implemented is shown in Fig. 5. Frame enable permit to calculate the log average luminance each time a HDR image is receiving by the HDR creating pipeline.

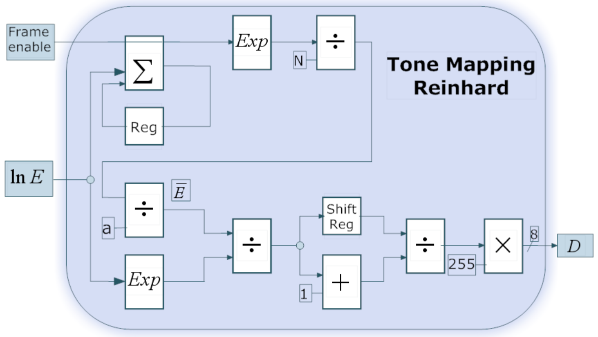


Fig. 5. Tone mapping hardware pipeline.

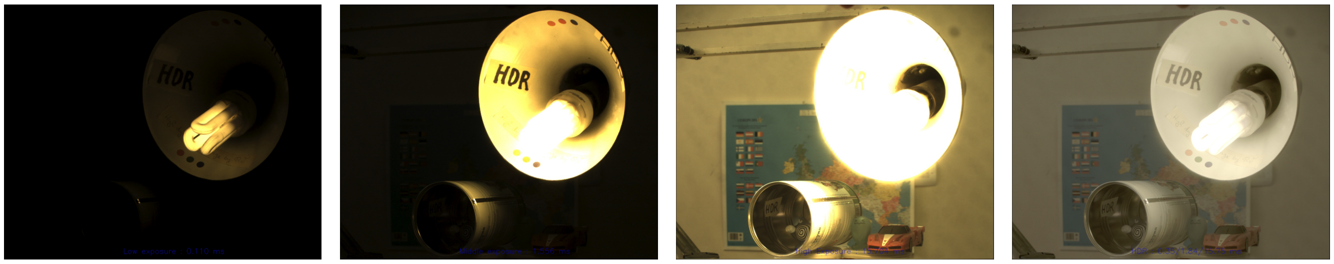
## IV. Implementation

Our work has been implemented on a Virtex-6 platform. We show the hardware implementation results in Table IV. Usually, FPGA-based image processing requires many specific devices such as SRAM memory, multi-port memory, video direct memory access, dedicated processors, and consequently, consumes many DSP blocks. This is not the case for our implementation. It consumes relatively low hardware complexity since the number of occupied slices is $6,692$ (about $17\%$ of the device) and the number of LUTs is $16,880$ (i.e. $11\%$ of the device).

| Metric | Utilization | Availability |
|---|---|---|
| Estimated supply power | 6.039 W | |
| Maximum frequency | 125.0 MHz | |
| Number of occupied Slices | 6,692 | 17% |
| LUTs | 16,880 | 11% |
| Registers | 20,192 | 6% |
| Number of bonded IOBs | 196 | 32% |
| 36K BRAMs | 17 | 4% |

TABLE II. Summary of hardware implementation results on the Virtex-6 platform.

Captures of still images from the different video LDR stream are shown in Fig. 6. You can see at the same time the contributions from the different LDRs frames (Fig. 6-a, 6-b, and 6-c) in the HDR image (Fig. 6-d). As an example, we can



(a) (b) (c) (d)

Fig. 6. Result of the complete system. Our Multiple Exposure Control can select the three proper exposures, and the specific memory management core permits us to display 3 bracketed images at the same time.

distinguish the word ”HDR” inside the lamp (high brightness), and the word ”HDR” inside the tube (low brightness).

## V. Conclusion

An HDR smart camera, with a complete hardware system from capture to display, has been designed for rendering HDR content at a 1.3-megapixel resolution and a high frame rate of 60 fps. Such a smart camera demonstrates that HDR video is feasible and exceeds other state of the art hardware platforms both in terms of resolution, speed, image quality and control. However, some effort has to be done in standardization, compression and sharing HDR data in order to provide an efficient HDR smart camera able to dynamically manage any variation in the scene. As an illustration, the multiple exposure technique can cause problems due to scene motion and generate artefacts in the resulting HDR video. In our case, our platform is not very affected with such a problem because extremely rapid scene motion does not happen in our captured scenes. This is partly due to the fact that we used a dedicated memory management core which delivers multiple videos in parallel at 60 frames per second, and that our pixel stream is continuously updated with the current frame sent by the sensor. However, in extremely rapid scene motion, real-time implementation of specific ghost removal algorithms need to be investigated in order to enhance the HDR video quality.

## References

[1] Mann, Picard, S. Mann, and R. W. Picard, “On being ‘undigital’ with digital cameras: Extending dynamic range by combining differently exposed pictures,” in *Proceedings of IS&T*, 1995, pp. 442–448.

[2] P. E. Debevec and J. Malik, “Recovering high dynamic range radiance maps from photographs,” in *SIGGRAPH*, 1997, pp. 369–378.

[3] T. Mitsunaga and S. Nayar, “Radiometric Self Calibration,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, Jun 1999, pp. 374–380.

[4] E. Reinhard, G. Ward, S. Pattanaik, and P. Debevec, *High Dynamic Range Imaging: Acquisition, Display, and Image-Based Lighting (The Morgan Kaufmann Series in Computer Graphics)*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2005.

[5] T. A. F. Drago, K. Myszkowski and N. Chiba, “Adaptive logarithmic mapping for displaying high contrast scenes,” *EUROGRAPHICS 2003 / P. Brunet and D. Fellner*, vol. Volume 22, no. 3, pp. 419–426, 2003.

[6] C. Jiang Duan, MarcoBressan and GuopingQiu, “Tone-mapping high dynamic range images by novel histogram adjustment,” *Pattern Recognition*, vol. 43, pp. 1847–1862, 2010.

[7] E. Reinhard, M. Stark, P. Shirley, and J. Ferwerda, “Photographic tone reproduction for digital images,” *ACM Transactions on Graphics*, vol. 21, no. 3, pp. 267–276, 2002.

[8] F. Durand and J. Dorsey, “Fast bilateral filtering for the display of high-dynamic-range images,” Laboratory for Computer Science, Massachusetts Institute of Technology, Tech. Rep., 2002.

[9] M. W. Raanan Fattal, Dani Lischinski, “Gradient domain high dynamic range compression,” School of Computer Science and Engineering The Hebrew University of Jerusalem, Tech. Rep., 2002.

[10] S. N. Pattanaik, J. Tumblin, H. Yee, and D. P. Greenberg, “Time-dependent visual adaptation for fast realistic image display,” in *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, ser. SIGGRAPH ’00. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 2000, pp. 47–54.

[11] A. Akyz, “High dynamic range imaging pipeline on the gpu,” *Journal of Real-Time Image Processing*, pp. 1–15, 2012.

[12] S. Mann, R. Lo, K. Ovtcharov, S. Gu, D. Dai, C. Ngan, and T. Ai, “Realtime hdr (high dynamic range) video for eyetap wearable computers, fpga-based seeing aids, and glasseyes (eyetaps),” in *25th IEEE Canadian Conference on Electrical Computer Engineering (CCECE)*, may 2012, pp. 1–6.

[13] R. Ureña, P. Martinez-Cañada, J. M. Gómez-López, C. A. Morillas, and F. J. Pelayo, “Real-time tone mapping on gpu and fpga,” *EURASIP J. Image and Video Processing*, vol. 2012, p. 1, 2012.

[14] B. Guthier, S. Kopf, and W. Effelsberg, “Optimal shutter speed sequences for real-time hdr video,” in *Imaging Systems and Techniques (IST), 2012 IEEE International Conference on*, july 2012, pp. 303 –308.

[15] C.-T. Chiu, T.-H. Wang, W.-M. Ke, C.-Y. Chuang, J.-S. Huang, W.-S. Wong, R.-S. Tsay, and C.-J. Wu, “Real-time tone-mapping processor with integrated photographic and gradient compression using 0.13um technology on an arm soc platform,” *Journal of Signal Processing Systems*, pp. 1–15, 2010.

[16] A. K. Bachoo, “Real-time exposure fusion on a mobile computer,” in *20th Annual Symposium of the Pattern Recognition Association of South Africa (PRASA)*, december 2009, pp. 111–115.

[17] A. Gençtav and A. O. Akyüz, “Evaluation of radiometric camera response recovery methods,” in *SIGGRAPH Asia 2011*, ser. SA ’11. New York, NY, USA: ACM, 2011, pp. 15:1–15:1.

[18] M. Ali and S. Mann, “Comparametric image compositing: Computationally efficient high dynamic range imaging,” in *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, march 2012, pp. 913 –916.

[19] B. Guthier, S. Kopf, and W. Effelsberg, “A real-time system for capturing hdr videos,” in *Proceedings of the 20th ACM international conference on Multimedia*, ser. MM ’12. New York, NY, USA: ACM, 2012, pp. 1473–1476.

[20] F. V. R. Tom Mertens, Jan Kautz, “Exposure fusion,” Hasselt University EDM transationale Universiteit Limburg Belgium, University College London UK, Tech. Rep., 2007.

[21] A. A. Goshtasby, “Fusion of multi-exposure images,” *Image and Vision Computing*, vol. 23, pp. 611–618, 2005.

[22] M. Tico, N. Gelfand, and K. Pulli, “Motion-blur-free exposure fusion.” in *ICIP’10*, 2010, pp. 3321–3324.