Reconfigurable Processor for Energy-Efficient Computational Photography

Rahul Rithe, *Student Member, IEEE*, Priyanka Raina, *Student Member, IEEE*, Nathan Ickes, *Member, IEEE*, Srikanth V. Tenneti, *Student Member, IEEE*, and Anantha P. Chandrakasan, *Fellow, IEEE*

***Abstract*—This paper presents an on-chip implementation of a scalable reconfigurable bilateral filtering processor for computational photography applications such as HDR imaging, low light enhancement and glare reduction. Careful pipelining and scheduling has minimized the local storage requirement to tens of kB. The 40 nm CMOS test chip operates from 98 MHz at 0.9 V to 25 MHz at 0.5 V. The test chip processes 13 megapixels/s while consuming 17.8 mW at 98 MHz and 0.9 V, achieving significant energy reduction compared to software implementations on recent mobile processors.**

***Index Terms*-Computational Photography, High-Dynamic Range Imaging, Bilateral Filtering, Bilateral Grid, Low-Power Electronics, Low-Voltage Operation, Voltage Scaling**

I. Introduction

Computational photography is transforming digital photography by significantly enhancing and extending the capabilities of a digital camera. The field encompasses a wide range of techniques such as high dynamic range (HDR) imaging [1], low-light enhancement [2,3], panorama stitching [4], image deblurring [5] and light field photography [6], which allow users to not just capture a scene flawlessly, but also reveal details that could otherwise not be seen.

Recent research has focused on specialized image sensors to capture information that is not captured by a regular CMOS image sensor. An image sensor with multi-bucket pixels is proposed in [7] to enable time multiplexed exposure that improves the image dynamic range and detects structured light illumination. A back-illuminated stacked CMOS sensor is proposed in [8] that uses spatially varying pixel exposures to support HDR imaging. An approach to reduce the temporal readout noise in an image sensor is proposed in [9] to improve low-light-level imaging. However, computational photography applications using regular CMOS image sensors that are currently used in the commercial cameras have so far been software based. Such CPU/GPU based implementations lead to high energy consumption and typically do not support real-time processing.

Non-linear filtering techniques like bilateral filtering [10] form a significant part of computational photography. These techniques have a wide range of applications, including HDR imaging, low-light enhancement, tone management [11], video enhancement [12] and optical flow estimation [13]. The high computational complexity of such multimedia processing applications necessitates fast hardware implementations [14,15] to enable real-time processing. In addition, energy-efficient operation is a critical concern for portable multimedia applications. Voltage and frequency scaling is an important technique for reducing power consumption while achieving high peak computational performance [16]. The energy efficiency of digital circuits is maximized at low supply voltages [17], which makes ability to operate at low voltage ($V_{DD} \sim 0.5$ V) a key component of achieving low power operation.

This work implements a reconfigurable multi-application processor for computational photography by exploring power reduction techniques at various design stages, such as algorithms, architectures and circuits. The algorithms are optimized to reduce the computational complexity and memory requirement. A parallel and pipelined architecture enables high throughput while operating at low frequencies, which allows real-time processing on HD images. Circuit design for low voltage operation ensures reliable performance down to 0.5 V.

The reconfigurable hardware implementation performs

This work was funded by the Foxconn Technology Group. The authors thank the TSMC University Shuttle Program for chip fabrication and Prof. Fredo Durand and Jonathan Regan-Kelley for valuable feedback and suggestions.

R. Rithe, P. Raina, N. Ickes and A. P. Chandrakasan are with the Microsystems Technology Laboratories at the Massachusetts Institute of Technology, 50 Vassar Street, Cambridge, MA 02139, USA (e-mail: {rjrithe, praina, nickes, anantha}@mtl.mit.edu).

S. V. Tenneti is with the California Institute of Technology, Pasadena, CA 91106, USA (e-mail: stenneti@caltech.edu).

HDR imaging, Low-light enhanced imaging and glare reduction, as shown in Fig. 1. The filtering engine can also be accessed from off-chip and used with other applications. The input images are pre-processed for the specific functions. The core of the processing unit are two bilateral filter engines that operate in parallel and decompose an image into a low frequency base layer and a high frequency detail layer. The bilateral filtering is performed using a bilateral grid structure that converts an input image into a three dimensional data structure and filters it by convolving with a three dimensional Gaussian kernel. Parallel processing allows enhanced throughput while operating at low frequency and low voltage. The bilateral filtered images are post processed to generate the outputs for the specific functions.

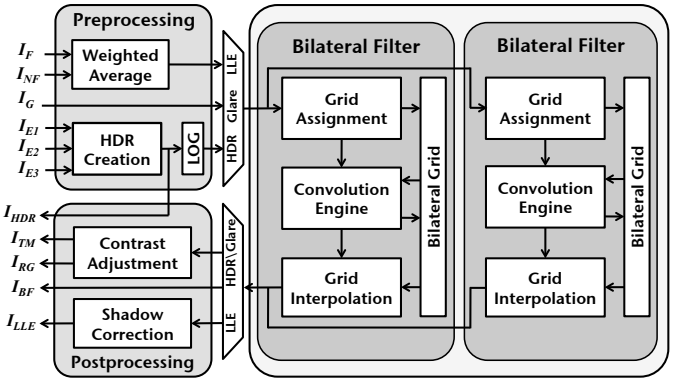


Fig. 1. System block diagram for the reconfigurable computational photography processor

This paper describes bilateral filtering and its efficient implementation using the bilateral grid [18]. A scalable hardware architecture for the bilateral filter engine is described. Implementation of HDR imaging, low-light enhancement and glare reduction using bilateral filtering is discussed in Section IV. The challenges of low voltage operation and approaches to address process variation are described in Section V. Section VI provides measurement results for the testchip.

II. Bilateral Filtering

Bilateral filtering is a non-linear filtering technique that takes into account the difference in the pixel intensities as well as the pixel locations while assigning weights, as opposed to linear Gaussian filtering that assigns filter weights based solely on the pixel locations. For an image I at pixel position p, the bilateral filtered output, I_B, is defined by eq. (1).

$$I_B(p) = \sum_{n=-N}^{N} G_S(n) \cdot G_I(I(p) - I(p-n)) \cdot I(p-n) \quad (1)$$

The output value at each pixel in the image is a weighted average of the values in a neighborhood, where the weight is the product of a Gaussian on the spatial distance (G_S) with standard deviation σ_s and a Gaussian on the pixel intensity/range difference (G_I) with standard deviation σ_r. In linear Gaussian filtering, on the other hand, the weights are determined solely by the spatial term. In bilateral filtering, the range term $G_I(I(p) - I(p-n))$ ensures that only those pixels in the vicinity that have similar intensities contribute significantly towards filtering. This avoids blurring across edges and results in an output that effectively reduces the noise while preserving the scene details. Fig. 2 compares Gaussian filtering and bilateral filtering in reducing image noise and preserving details. However, non-linear

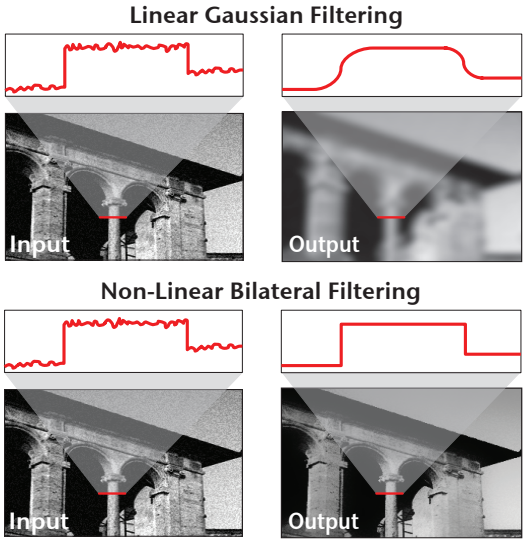


Fig. 2. Comparison of Gaussian filtering and bilateral filtering. Bilateral filtering effectively reduces noise while preserving scene details.

filtering is inefficient and slow to implement because the filter kernel is spatially variant. A direct implementation of bilateral filtering can take on the order of several minutes to process HD images. Faster approaches for bilateral filtering have been proposed that reduce the processing time by filtering subsampled versions of the image with discrete intensity kernels and reconstructing the filtered results using linear interpolation [1,19]. A fast approach to bilateral filtering based on a box spatial kernel, which can be iterated to yield smooth spatial falloff is proposed in [20]. However real-time processing of HD images requires further speed-up.

A. Bilateral Grid

A software-based bilateral grid structure is described in [18], which enables fast bilateral filtering but requires a large amount of storage (65 MB for a 10 megapixel image) for processing. In this work, we implement bilateral filtering using a reconfigurable grid, which reduces the storage requirement to 21.5 kB by scheduling the filtering engine so that only two grid rows need to be stored at a time. The implementation is flexible to allow varying grid sizes for energy/resolution scalable image processing.

The bilateral grid structure used by this chip is constructed as follows. The input image is partitioned into blocks of size $\sigma_s \times \sigma_s$ and a histogram of pixel intensity values is generated for each block. Each histogram has $256/\sigma_r$ bins. This results in a 3D representation of the 2D image, as shown in Fig. 3, referred to as the bilateral grid. Each grid cell (i, j, r) stores the number of pixels in a block corresponding to that intensity bin (W_r^{ij}) and their summed intensity (I_r^{ij}). The processor supports block sizes ranging from 16×16 to 128×128 pixels with 4 to 16 intensity bins in the histogram.

The bilateral grid has two key advantages:

- *Aggressive Down-sampling*: The size of the blocks ($\sigma_s \times \sigma_s$) used while creating the grid and the number of intensity bins ($256/\sigma_r$) determine the amount by which the image is down-sampled. The grid merges blocks of 16×16 to 128×128 pixels into 4 to 16 grid cells. This significantly reduces the number of computations required for processing as well as the amount of on-chip storage required.
- *Built-in Edge Awareness*: Two pixels that are spatially adjacent but have very different intensities end up far apart in the grid in the intensity dimension. Filtering the grid level-by-level using a linear Gaussian kernel, only the intensity levels that are near each other influence the filtering and the levels that are far apart do not contribute in each others filtering. This is equivalent to performing bilateral filtering on the 2D image.

III. Bilateral Filter Engine

The bilateral filter engine using the bilateral grid is implemented as shown in Fig. 4. It consists of three components – the grid assignment engine, the grid filtering engine and the grid interpolation engine.

The image is scanned pixel by pixel in a block-wise manner. The size of the block is scalable from 16×16 pixels to 128×128 pixels. Depending on the intensity of the input pixel, it is assigned to one of the intensity bins. The number of intensity bins is also scalable from 4 to 16.

A. Grid Assignment

The pixels are assigned to the appropriate grid cells by the grid assignment engines. The hardware has 16 grid assignment (GA) engines that can operate in parallel to process 16 intensity levels in the grid. But 4 or 8 grid assignment engines could be activated if the grid uses fewer intensity levels. Fig. 5 shows the architecture of the grid assignment engine. For each pixel from each block, its intensity is compared with the boundaries of the intensity bins using digital comparators. If the pixel intensity is within the bin boundaries, it is assigned to that intensity bin. Intensities of all the pixels assigned to a bin are summed by an accumulator. A weight counter maintains the count of number of pixels assigned to the bin. Both the summed intensity and weight are stored for each bin in on-chip memory.

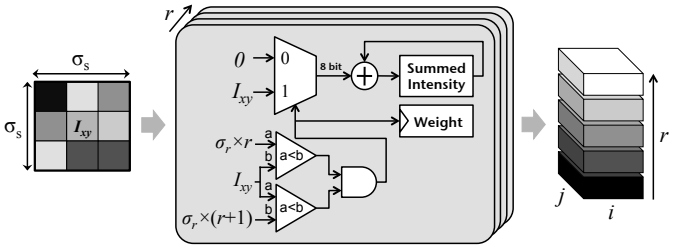


Fig. 5. Architecture of the grid assignment engine.

B. Grid Filtering

The convolution (Conv) engine, shown in Fig. 6, convolves the grid intensities and weights with a $3 \times 3 \times 3$ Gaussian kernel, which is equivalent to bilateral filtering in the image domain, and returns the normalized intensity. The convolution is performed by multiplying the 27 coefficients of the filter kernel with the 27 grid cells and adding them using a 3-stage adder tree. The intensity and weight are convolved in parallel and the convolved intensity is normalized with the convolved weight by using a fixed point divider to make sure that there is no intensity scaling during filtering. The hardware has 16 convolution engines that can operate in parallel to filter a grid with 16 intensity levels. But 4 or 8 of them can be activated if fewer intensity levels are used.

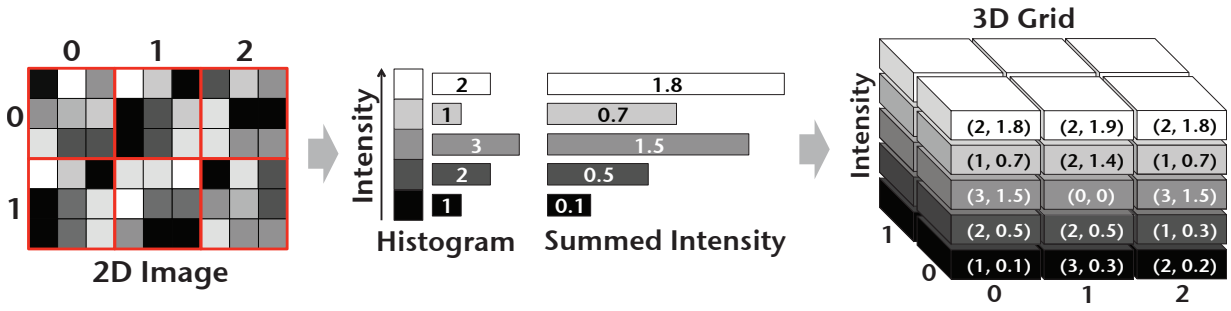


Fig. 3. Construction of a 3D bilateral grid from a 2D image

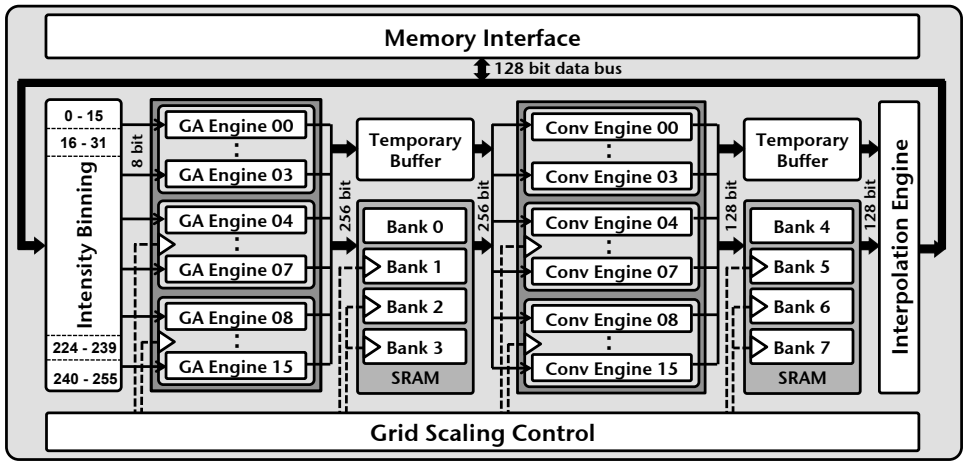


Fig. 4. Architecture of the bilateral filtering engine. Grid scalability is achieved by gating processing engines and SRAM banks

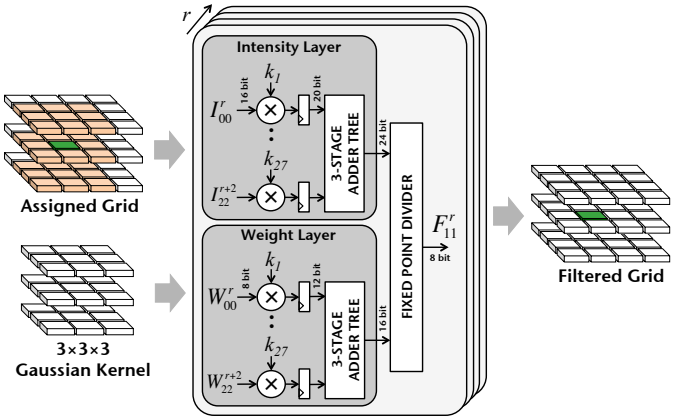


Fig. 6. Architecture of the convolution engine for grid filtering.

C. Grid Interpolation

The interpolation engine, shown in Fig. 7, reconstructs the filtered 2D image from the filtered grid. The filtered intensity value at pixel (x, y) is obtained by trilinear interpolation of $2 \times 2 \times 2$ filtered grid values surrounding the location $(x/\sigma_s, y/\sigma_s, I_{xy}/\sigma_r)$. Trilinear interpolation is equivalent to performing linear interpolations independently across each of the three dimensions of the grid. To meet throughput requirements, the interpolation engine is implemented as three pipelined stages of linear interpolations. The output value $I_{BF}(x, y)$ is calculated from filtered grid values F^r_{ij} using four parallel linear interpolations along the i dimension, given by eq. (2):

$$\begin{aligned} F^r_j &= F^r_{i,j} \times w^i_1 + F^r_{i+1,j} \times w^i_2 \\ F^r_{j+1} &= F^r_{i,j+1} \times w^i_1 + F^r_{i+1,j+1} \times w^i_2 \\ F^{r+1}_j &= F^{r+1}_{i,j} \times w^i_1 + F^{r+1}_{i+1,j} \times w^i_2 \\ F^{r+1}_{j+1} &= F^{r+1}_{i,j+1} \times w^i_1 + F^{r+1}_{i+1,j+1} \times w^i_2 \end{aligned} \tag{2}$$

followed by two parallel linear interpolations along the j dimension, given by eq. (3):

$$\begin{aligned} F^r &= F^r_j \times w^j_1 + F^r_{j+1} \times w^j_2 \\ F^{r+1} &= F^{r+1}_j \times w^j_1 + F^{r+1}_{j+1} \times w^j_2 \end{aligned} \tag{3}$$

followed by an interpolation along the r dimension, given by eq. (4):

$$I_{BF}(x, y) = F^r \times w^r_1 + F^{r+1} \times w^r_2 \tag{4}$$

The interpolation weights, given by eq. (5), are computed based on the output pixel location (x, y), the intensity

of the original pixel in the input image I_{xy} at location (x, y), and the grid cell index (i, j, r).

$$w_1^i = \frac{x}{\sigma_s} - i; \qquad w_2^i = i + 1 - \frac{x}{\sigma_s}$$
$$w_1^j = \frac{y}{\sigma_s} - j; \qquad w_2^j = j + 1 - \frac{y}{\sigma_s}$$
$$w_1^r = \frac{I_{xy}}{\sigma_r} - r; \qquad w_2^r = r + 1 - \frac{I_{xy}}{\sigma_r} \tag{5}$$

The pixel location (x, y) and the grid cell index (i, j, r) are maintained in internal counters. The original pixel intensity I_{xy} is read from the DRAM in chunks of 32 pixels per read request to fully utilize the memory bandwidth.

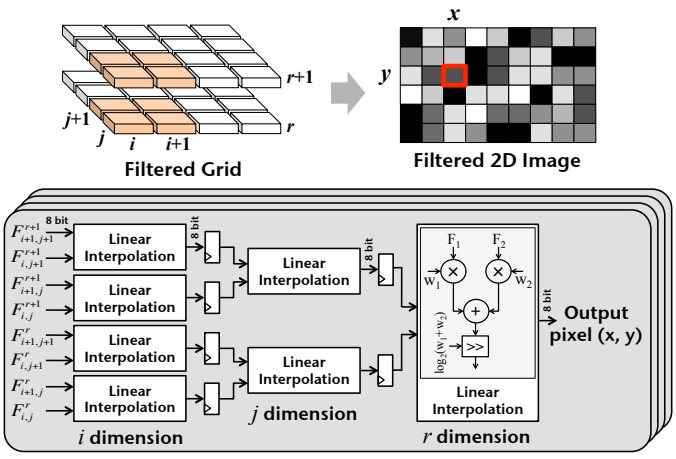


Fig. 7. Architecture of the interpolation engine. Trilinear interpolation is implemented as three pipelined stages of linear interpolations.

The assigned and filtered grid cells are stored in the on-chip memory. Last three assigned blocks are stored in a temporary buffer and two previous rows of grid blocks are stored in the SRAM. Last two filtered blocks are stored in the temporary buffer and one filtered grid row is stored in the SRAM.

D. Memory Management

The grid processing tasks are scheduled to minimize local storage requirements and memory traffic. Fig. 8 shows the memory management scheme by task scheduling. Grid processing is performed cell-by-cell in a row-wise manner. The last three blocks are stored in the temporary buffer and the last two rows are stored in the SRAM. Once a 3×3×3 block is available, the convolution engine begins filtering the grid. When block A, shown in Fig. 8, is being assigned, the convolution engine is filtering block F. As filtering proceeds to the next block in the row, the first assigned block, stored in the SRAM, becomes redundant and is replaced by the first assigned block in the temporary buffer. Last two filtered blocks are stored in the temporary buffer and the previous row of filtered blocks are stored in the SRAM. As 2×2×2 filtered blocks become available, the interpolation engine begins reconstructing the output 2D image. When block F, shown in Fig. 8, is being filtered, the interpolation engine is reconstructing the output 2D image from block I. As interpolation proceeds to the next block in the row, the first filtered block, stored in the SRAM, becomes redundant and is replaced by the first filtered block in the temporary buffer. Boundary rows and columns are replicated for processing boundary cells. This scheduling scheme allows processing without storing the entire grid. Only two assigned grid rows and one filtered grid row need to be stored locally at a time. Memory management reduces the memory requirement to 21.5 kB for processing a 10 megapixel image and allows processing grids of arbitrary height using the same amount of on-chip memory.

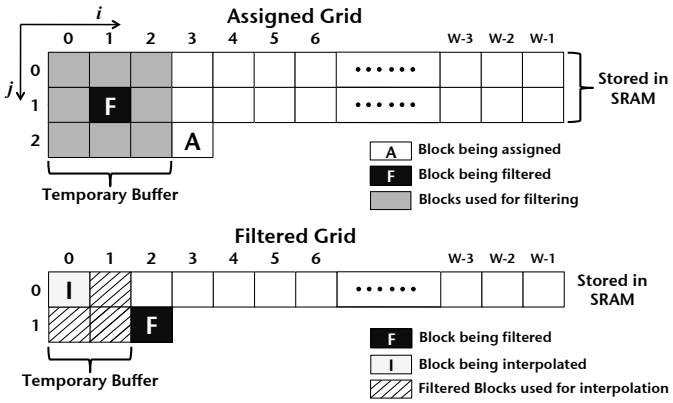


Fig. 8. Memory management by task scheduling.

E. Scalable Grid

The size of the grid is determined by the image size and the downsampling factors. For an image of size $I_W \times I_H$ pixels with the spatial and intensity/range downsampling factors σ_s and σ_r respectively, the grid width (G_W) and height (G_H) are given by eq. (6) and the number of grid cells (N_G) is given by eq. (7).

$$G_W = \frac{I_W}{\sigma_s}; \qquad G_H = \frac{I_H}{\sigma_s} \tag{6}$$

$$N_G = G_W \times G_H \times \left(\frac{256}{\sigma_r}\right) \tag{7}$$

The number of computations as well as storage depends directly on the size of the grid. Selecting the down-sampling factors the same as the standard deviations of

the spatial and intensity/range Gaussians in the bilateral filter (eq. (1)) provides a good trade-off between the output quality and processing complexity. The choice of downsampling factors is guided by the image content and the application. Most applications work well with a coarse grid resolution on the order of 32 pixels with 8 to 12 intensity bins. If the image has high spatial details, a smaller σ_s would result in better preservation of those details in the output. Similarly, a smaller σ_r would help preserve fine intensity details. The grid size is configurable by adjusting σ_s from 16 to 128, which scales the block size from 16×16 to 128×128 pixels, and σ_r from 16 to 64, which scales the number of intensity levels from 16 to 4. For a 10 megapixel (4096×2592) image, the number of grid cells scales from 663552 $(\sigma_s = 16, \sigma_r = 16)$ to 2592 $(\sigma_s = 128, \sigma_r = 64)$. The architecture achieves energy scalability by activating only the required number of hardware units for a given grid resolution.

The 21.5 kB of on-chip SRAM used to store two rows of created grid cells and one row of filtered grid cells. The SRAM is implemented as 8 banks supporting a maximum of 256 cells in each row of the grid with 16 intensity levels, corresponding to the worst case of $\sigma_s = 16, \sigma_r = 16$. Each bank is clock & input gated to save energy when a lower resolution grid is used. Only one bank is used when $\sigma_s = 128$ and all 8 banks are used when $\sigma_s = 16$. The bilateral filter engine achieves scalability by activating only the required number of processing engines and SRAM banks for the desired grid resolution.

IV. Applications

The testchip has two bilateral filter engines, each processing 4 pixels/cycle. The processor performs HDR imaging, low-light enhanced imaging and glare reduction using the bilateral filter engines.

A. High Dynamic Range Imaging

High dynamic range (HDR) imaging is a technique for capturing a greater dynamic range between the brightest and darkest regions of an image than a traditional digital camera. It is done by capturing multiple images of the same scene with varying exposure levels, such that the low exposure images capture the bright regions of the scene well without loss of detail and the high exposure images capture the dark regions of the scene. These differently exposed images are then combined together into a high dynamic range image, which more faithfully represents the brightness values in the scene.

The first step in HDR imaging is to create a composite HDR image from multiple differently exposed images which represents the true scene radiance value at each pixel of the image [21]. The true scene radiance value at each pixel is recovered from the recorded intensity I and the exposure time Δt as follows. The exposure E is defined as the product of sensor irradiance R (which is the amount of light hitting the camera sensor and is proportional to the scene radiance) and the exposure time Δt. The intensity I is a nonlinear function of the exposure E, given by eq. 8.

$$I = f(R \times \Delta t) \tag{8}$$

We can then obtain the sensor irradiance as given by eq. 9.

$$\log(R) = g(I) - \log(\Delta t) \tag{9}$$

where, $g = \log f^{-1}$.

The mapping g is knows as the camera curve [21]. Fig. 9 shows the camera curves for the RGB color channels of a typical camera sensor.

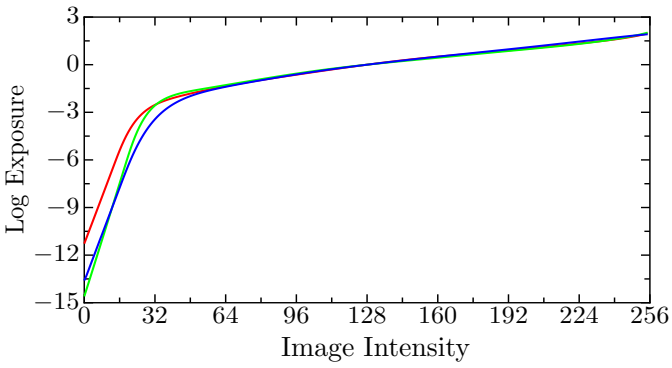


Fig. 9. Camera curves that map the pixel intensity values on to the incident exposure.

The HDR creation module, shown in Fig. 10 takes values of a pixel from three different exposures (I_{E1}, I_{E2}, I_{E3}) and generates an output pixel which represents the true scene radiance value (I_{HDR}) at that location. Since we are working with a finite range of discrete pixel values (8 bits per color), the camera curves are stored as combinational look-up tables to enable fast access. The true (log) exposure values are obtained from the pixel intensities using the camera curves, followed by exposure time correction to obtain (log) scene radiance. The three resulting (log) radiance values obtained from the three images represent the radiance values of the same location in the scene. A weighted average of

these three values is taken to obtain the final (log) radiance value. The weighting function gives a higher weight to the exposures in which pixel value is closer to the middle of the response function (thus avoiding the high contributions from images where the pixel value is saturated). In the end an exponentiation is performed to get the final radiance value (16 bits per pixel per color). Processing in log domain simplifies the computations to additions and subtractions instead of multiplications and divisions.

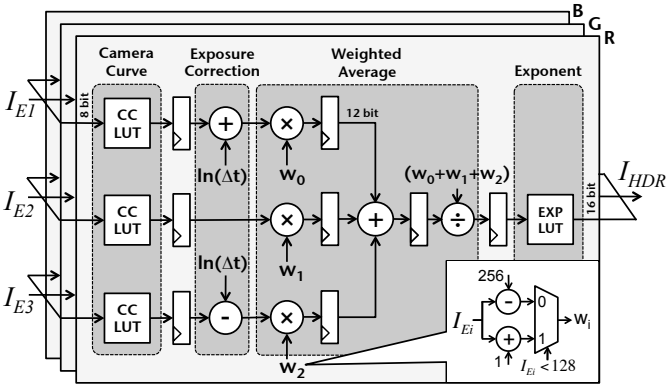


Fig. 10. HDR creation module.

Displaying HDR images on LDR media (8b/pixel) requires tone mapping that compresses image dynamic range by non-linear filtering [1]. The 16b/pixel HDR image is split into intensity and color channels. A low-frequency base layer and a high-frequency detail layer are created by bilateral filtering the HDR intensity in log domain. The dynamic range of the base layer is compressed by a scaling factor in the log domain. The detail layer is untouched to preserve the details and the colors are scaled linearly to 8b/pixel. Merging the compressed base layer, the detail layer and the color channels results in a tone-mapped HDR image (I_{TM}). In HDR mode, both bilateral grids are configured to perform filtering in an interleaved manner, where each grid processes alternate blocks in parallel.

Fig. 11 shows a set of input low-dynamic range exposures and the tonemapped HDR output image.

B. Glare Reduction

Glare reduction is similar to performing single image HDR tone-mapping. The input image is split into intensity and color channels. A low-frequency base layer and a high-frequency detail layer are created by bilateral filtering the intensity. The contrast of the base layer is enhanced using the contrast adjustment module, shown in Fig. 12, which is also used in HDR tone-mapping. The contrast can be increased or decreased depending on the adjustment factor. The scaled color data is merged with the contrast enhanced base layer and the detail layer to obtain a glare reduced output image.

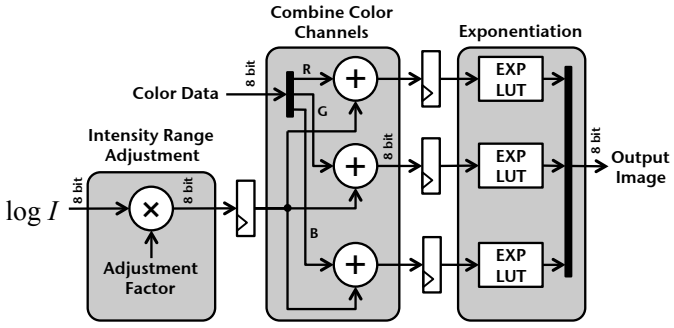


Fig. 12. Contrast adjustment module. Contrast is increased or decreased depending on the adjustment factor.

Fig. 13 shows an input image with glare and the glare reduced output image. Glare reduction recovers details that are white-washed in the original image and enhances the image colors and contrast.

Fig. 13. Input images: (a) image with glare. Output image: (b) image with reduced glare.

C. Low-Light Enhanced Imaging

Low-light enhanced imaging is performed by merging two images captured in quick succession, one taken without flash (I_{NF}) and one with flash (I_F) [2,3]. The bilateral grid is used to decompose both images into base and detail layers. The scene ambience is captured in the base layer of the non-flash image and details are captured in the detail layer of the flash image. In this

Fig. 11. Input low-dynamic range images: (a) under exposed image, (b) normally exposed image, (c) over exposed image. Output image: (d) tonemapped HDR image.

mode, one grid is configured to perform bilateral filtering on the flash image and the other to perform cross-bilateral filtering, given by eq. (10), on the non-flash image using the flash image. The location of the grid cell is determined by the flash image and the intensity value is determined by the non-flash image.

$$I_{CB}(p) = \sum_{n=-N}^{N} \left[G_S(n) \cdot G_I(I_F(p) - I_F(p-n)) \cdot I_{NF}(p-n) \right] \quad (10)$$

The image taken with flash contains shadows that are not present in the non-flash image. A shadow correction module is implemented which merges the details from the flash image with base layer of the cross-bilateral filtered non-flash image and corrects for the flash shadows to avoid artifacts in the output image. A mask representing regions with high details in the filtered non-flash image is created, as shown in Fig. 14. Gradients are computed at each pixel for blocks of 4×4 pixels. If the gradient at a pixel is higher than the average gradient for that block, the pixel is assigned as an edge pixel. This results in a binary mask that highlights all the strong edges in the scene but no false edges due to the flash shadows. The details from the flash image are added to the filtered non-flash image, as shown in Fig. 15, only in the regions represented by the mask. A linear filter is used to smooth the mask to ensure that that the resulting image does not have discontinuities. This implementation of the shadow correction module handles shadows effectively to produce LLE images without artifacts.

Fig. 16 shows a set of input flash and non-flash images and the low-light enhanced output image. The enhanced output effectively reduces noise while preserving details.

Another set of images is shown in Fig. 17. The flash image has shadows that are not present in the non-flash

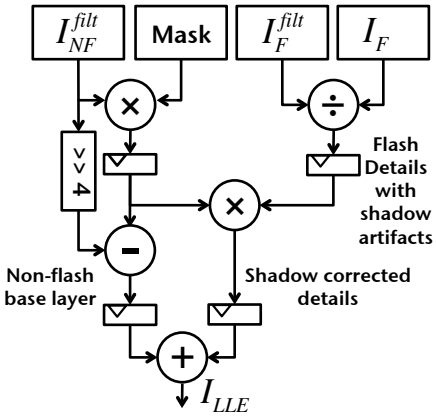


Fig. 15. Merging flash and non-flash images with shadow correction.

image. The bilateral filtered non-flash image reduces the noise but lacks details. The enhanced output, created by adding the details from the flash image, effectively reduces noise while preserving details and corrects for flash shadows without creating artifacts.

V. Low-Voltage Operation

The energy consumed by a digital circuit can be minimized by operating at the optimal V_{DD}, which requires the ability to operate at low voltage. Random Dopant Fluctuation (RDF) is a dominant source of local variation at low voltage, causing random, local threshold voltage shifts [22]. To maintain sufficient reliability and performance at low voltage, significant attention needs to be given to the effects of local variation.

Performance of logic circuits is highly sensitive to variation in V_T in this region of operation, and can also result in functional failures at the extremes of V_T variation. To quantify the functionality of a combinational cell at low voltage, we use the standard cell characterization approach described in [23]. A subset of standard cells from the 40 nm CMOS logic library are analyzed to ensure functionality and quantify the performance at

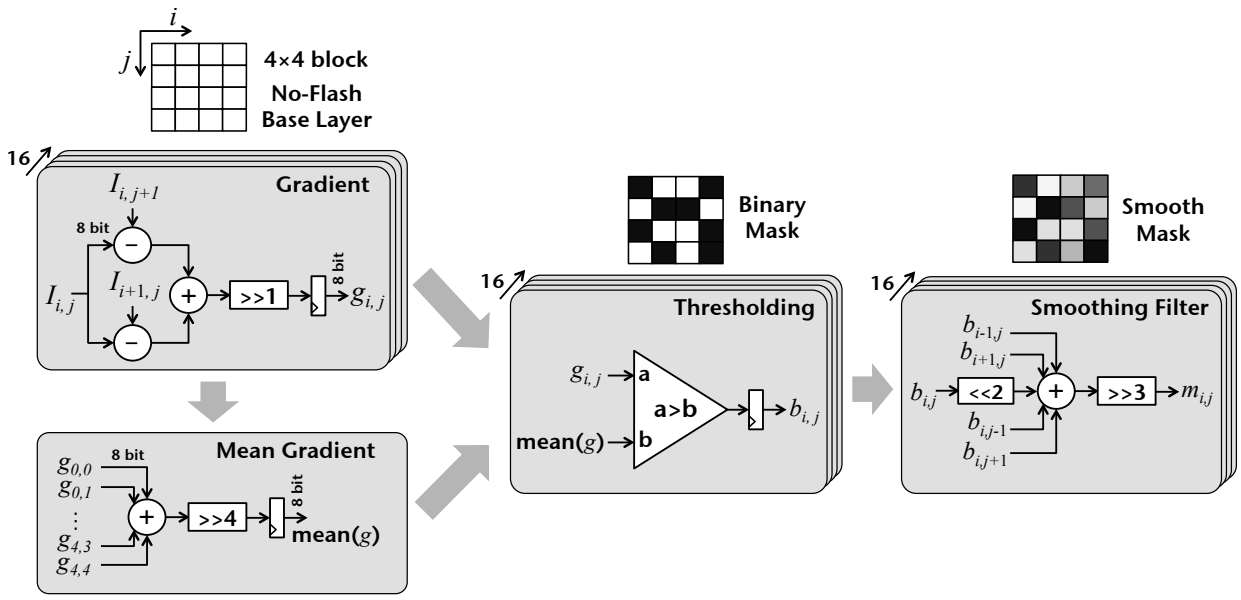


Fig. 14. Generating a mask representing regions with high scene details.

Fig. 16. Input images: (a) image with flash, (b) image without flash. Output image: (c) low-light enhanced image.

0.5 V. Standard cells that fail the functionality or do not satisfy the performance requirement are not used in the design. The functionality and setup/hold performance of flip-flops are also verified using the cell characterization approach.

At nominal voltage, local variations in V_T may result in 5%–10% variation in the logic timing. However, at low voltage, these variations can result in timing path delays with standard deviation comparable to the global corner delay, and must be accounted for during timing closure in order to ensure a robust, manufacturable design. The Probability Density Function (PDF) of delay at 0.5 V for a representative path from the design is shown in Fig. 18. The global corner delay for this path is 21.9 ns, but after accounting for the local variations the 3σ delay becomes 36.1 ns.

At nominal voltage, paths that fail the setup/hold requirement are determined using the corner-based analysis and timing closure is achieved by performing setup/hold fix on these paths. However, at low voltage, it is not possible to consider only the paths that fail the setup/hold requirement in the corner analysis and determine their 3σ setup/hold performance, since a path with larger corner delay need not have a larger stochastic variation. We use the nonlinear operating point analysis (OPA) approach [23] to perform timing analysis at $V_{DD} = 0.5$ V. The potentially critical paths in the design, in presence of variations, are determined by the approach described in

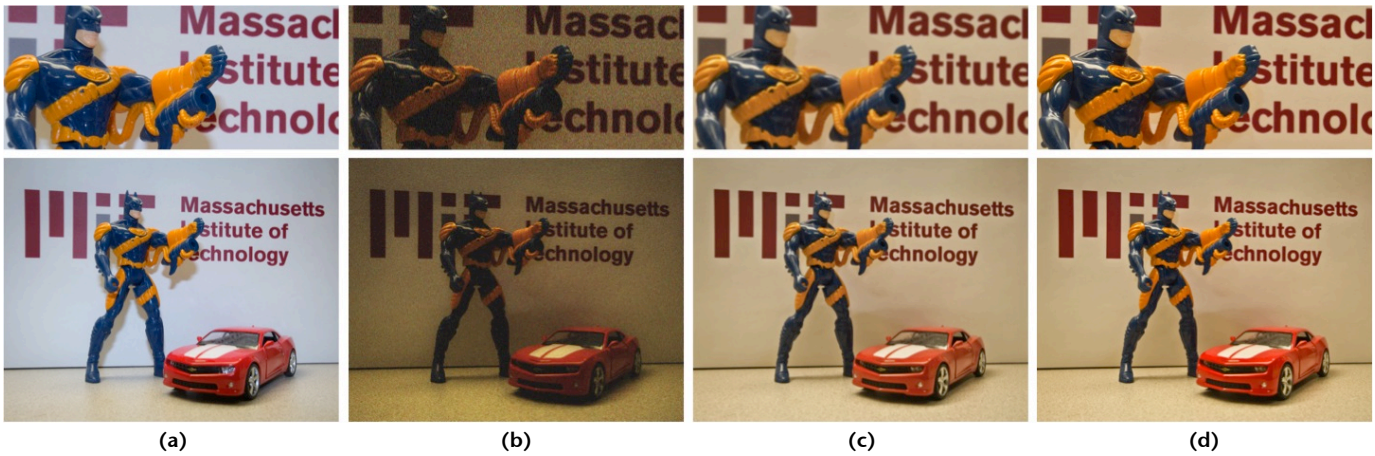


Fig. 17. Input images: (a) image with flash, (b) image without flash. Output images: (c) filtered no-flash image, (d) low-light enhanced image.

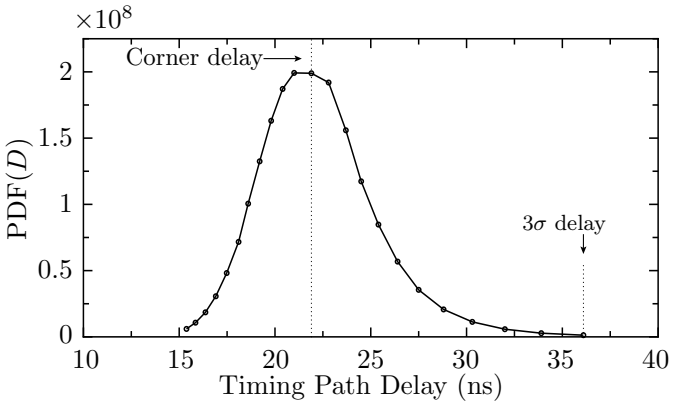


Fig. 18. Delay PDF of a representative timing path from the design at 0.5 V. The global corner delay is 21.9 ns and the 3σ delay, after accounting for local variations, is 36.1 ns.

[23] and 3σ setup and hold performance is computed at $V_{DD} = 0.5$ V using OPA. The three step approach is summarized below.

1) All paths are analyzed with traditional static timing analysis (STA). $+/-3\sigma$ delay (corner delay plus the stochastic $+/-3\sigma$ delay) is used for each cell in the path. This is a pessimistic analysis, so those paths that pass the setup/hold requirement can be removed from further consideration.
2) Paths that fail the first step are re-analyzed. OPA based analysis is applied to the clock paths to determine their 3σ performance. Data paths are analyzed with STA as in the first step with $+/-3\sigma$ delays applied to all the cells in the data path. This is also a pessimistic estimate. The paths that pass the setup/hold requirement can be removed from further consideration.
3) The remaining paths are analyzed with OPA based timing analysis applied to both the data paths and clock paths to determine their true 3σ setup/hold performance.

The paths that fail the 3σ setup or hold performance test are optimized to fix the setup/hold violations. Table I shows statistics on the number of paths analyzed for both setup and hold analysis of the chip. Setup/hold fixing using OPA ensures that cells that are very sensitive to V_T variations are not used in the critical paths. This helps improve the 3σ performance at 0.5 V by 32%, from 17 MHz to 25 MHz.

The functionality and timing characterization for standard cells and the OPA analysis for timing paths ensure reliable functionality, despite statistical variations, with 3σ confidence at 0.5 V.

VI. Measurement Results

The testchip, shown in Fig. 19, is implemented in 40 nm CMOS technology and verified to be operational from 25 MHz at 0.5 V to 98 MHz at 0.9 V.

This chip is designed to function as an accelerator core as part of a larger microprocessor system, utilizing the system's existing DRAM resources. For standalone testing of this chip, a 32 bit wide 266 MHz DDR2 memory controller was implemented using a Xilinx XC5VLX50 FPGA. The performance vs. energy trade-off of the testchip for a range of V_{DD} is shown in Fig. 20. The processor is able to operate from 25 MHz at 0.5 V with 2.3 mW power consumption to 98 MHz at 0.9 V with 17.8 mW power consumption. The run-time scales linearly with the image size with 13 megapixel/s throughput.

TABLE I
SETUP/HOLD TIMING ANALYSIS AT 0.5 V

Phase	Data Path	Clock Path	Paths Analyzed	Worst Slack (ns)	% Fail
Setup Analysis @ 25MHz					
1	STA ($+3\sigma$)	STA (-3σ)	95k	-10.7	3.6%
2	STA ($+3\sigma$)	OPA	3.4k	-2.9	1.5%
3	OPA	OPA	52	-0.05	13.4%
Paths requiring fixing (before timing closure)			7		
Hold Analysis					
1	STA (-3σ)	STA ($+3\sigma$)	95k	-8.2	2.8%
2	STA (-3σ)	OPA	2.7k	-1.8	2.4%
3	OPA	OPA	65	-0.13	13.8%
Paths requiring fixing (before timing closure)			9		

TABLE II
PERFORMANCE COMPARISON WITH MOBILE PROCESSOR IMPLEMENTATIONS

Processor	Technology (nm)	Frequency (MHz)	Power (mW)	Runtime* (s)	Energy* (mJ)
Intel Atom [24]	32	1800	870	4.96	4315
Qualcomm Snapdragon [25]	28	1500	760	5.19	3944
Samsung Exynos [26]	32	1700	1180	4.05	4779
TI OMAP [27]	45	1000	770	6.47	4981
This Work	40	98	17.8	0.771	13.7

*Image size: 10 megapixel

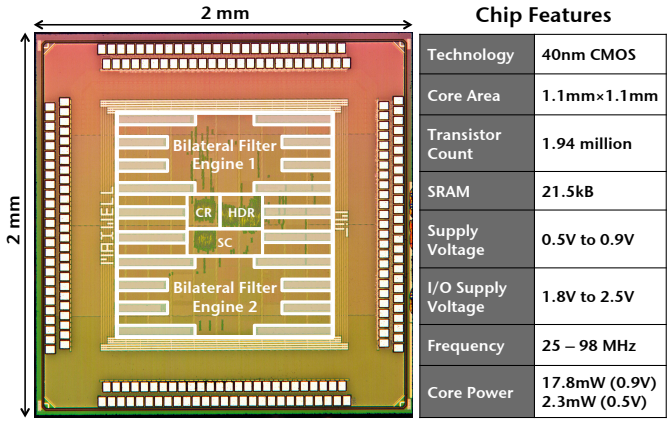


Fig. 19. Die photo of the testchip. Highlighted boxes indicate SRAMs. HDR, CR and SC refer to HDR create, contrast reduction and shadow correction modules respectively.

Table II shows a comparison of the processor performance with implementations on other mobile processors. Software that replicates the functionality of the testchip and maintains identical image quality is implemented on the mobile processors. The implementations are optimized for multi-threading and multi-core processing. Processing runtime and power consumption during software execution are measured. The processor achieves more than 5.2× faster performance than the fastest software implementation and consumes less than 40× power compared to the most power efficient one, resulting in an energy reduction of more than 280× compared to software implementations on some of the recent mobile processors while maintaining the same output image quality. Flexible bit width computations, along with high amount of parallelism and pipelining, enable an optimized processor implementation that achieves higher performance at a lower frequency and significant improvement in energy efficiency compared to software implementations.

The processor is integrated, as shown in Fig. 21, with

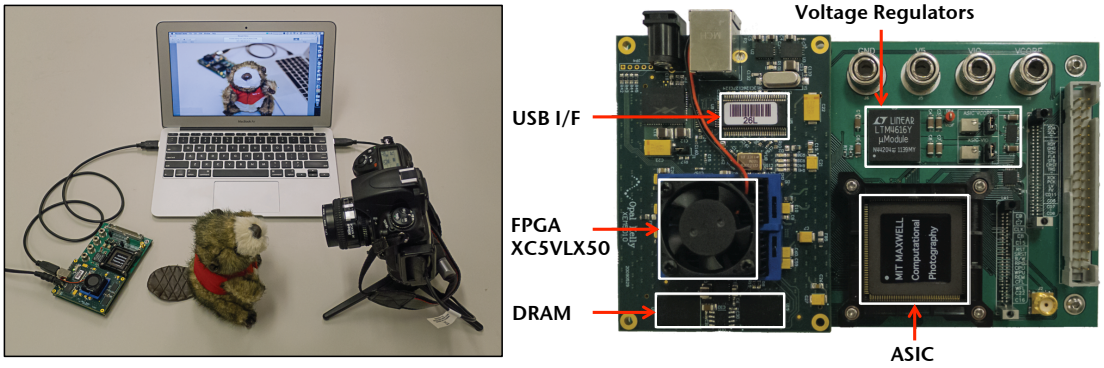


Fig. 22. Demo board and setup integrated with camera and display.

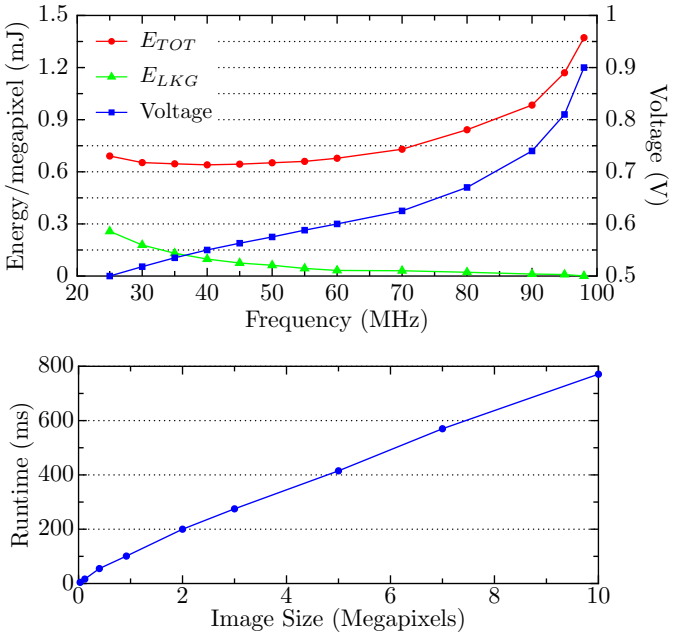


Fig. 20. Processor performance: trade-off of total energy (E_{TOT}) and leakage (E_{LKG}) vs. performance for varying V_{DD}, and processing run-time for different image sizes

a camera and a display through a host PC using the USB interface. A software application, running on the host PC, is developed for processor configuration, image capture, activating processing and result display. The system, shown in Fig. 22, provides a portable platform for live computational photography.

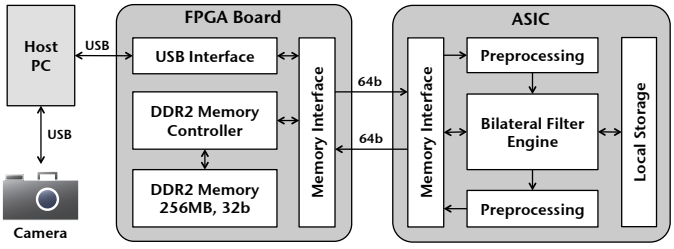


Fig. 21. Processor integration with external memory, camera and display.

VII. Conclusions

We have described the development and implementation of a reconfigurable processor for computational photography using 40 nm CMOS technology. The processor performs HDR imaging, low-light enhancement and glare reduction using a reconfigurable bilateral grid. Highly parallel architecture enables real-time processing of HD images while operating at less than 100 MHz. The ability to operate at low supply voltages is important for maximizing energy efficiency. Circuit design for low voltage operation ensures reliable performance down to 0.5 V. The processor achieves 280× energy reduction compared to software implementations on recent mobile processors. The energy scalable implementation proposed in this work enables efficient integration into portable multimedia devices for real time computational photography.

References

[1] F. Durand, J. Dorsey, "Fast Bilateral Filtering for the display of high-dynamic-range images," *ACM Transactions on Graphics,* Vol. 21, No. 3, 257–266, July 2002.

[2] G. Petschnigg, M. Agrawala, H. Hoppe, R. Szeliski, M. Cohen, K. Toyama, "Digital Photography with flash and no-flash image pairs," *ACM Transactions on Graphics,* Vol. 23, No. 3, 664–672, August 2004.

[3] E. Eisemann, F. Durand, "Flash photography enhancement via intrinsic relighting, *ACM Transactions on Graphics,* Vol. 23, No. 3, 673–678, August 2004.

[4] M. Brown and D. G. Lowe, "Automatic Panoramic Image Stitching using Invariant Features, *International Journal of Computer Vision,* Vol. 74, No. 1, 59–73, August 2007.

[5] A. Levin, Y. Weiss, F. Durand, and W. T. Freeman, "Efficient marginal likelihood optimization in blind deconvolution, *IEEE Conference on Computer Vision and Pattern Recognition,* 2657–2664, June 2011.

[6] R. Ng, M. Levoy, M. Bredif, G. Duval, M. Horowitz, P. Hanrahan, "Light-field photography with a handheld plenoptic camera," *Stanford University Computer Science Tech Report,* CSTR 2005–02, April 2005.

[7] G. Wan, X. Li, G. Agranov, M. Levoy, M. Horowitz, "CMOS Image Sensors With Multi-Bucket Pixels for Computational Photography," IEEE Journal of Solid-State Circuits, Vol. 47, No. 4, 1031–1042, April 2012.

[8] S. Sukegawa, T. Umebayashi, T. Nakajima, H. Kawanobe, K. Koseki, I. Hirota, T. Haruta, M. Kasai, K. Fukumoto, T. Wakano, K. Inoue, H. Takahashi, T. Nagano, Y. Nitta, T. Hirayama, N. Fukushima, "A 1/4-inch 8Mpixel back-illuminated stacked CMOS image sensor," *IEEE International Solid-State Circuits Conference,* 484–485, Feb. 2013.

[9] Y. Chen, Y. Xu, Y. Chae, A. Mierop, X. Wang, A. Theuwissen, "A 0.7e-rms-temporal-readout-noise CMOS image sensor for low-light-level imaging," *IEEE International Solid-State Circuits Conference,* 384–385, Feb. 2012.

[10] C. Tomasi, R. Manduchi, "Bilateral Filtering for Gray and Color Images," *IEEE International Conference on Computer Vision,* 839–846, Jan 1998.

[11] S. Bae, S. Paris, F. Durand, "Two-scale tone management for photographic look, *ACM Transactions on Graphics,* Vol. 25, No. 3, 637–645, July 2006.

[12] E. Bennet, L. McMillan, "Video enhancement using per-pixel virtual exposures, *ACM Transactions on Graphics,* Vol. 24, No. 3, 845–852, July 2005.

[13] J. Xiao, H. Cheng, H. Awhney, C. Rao, M. Isnardi, "Bilateral filtering based optical flow estimation with occlusion detection, *European Conference on Computer Vision,* 211–224, 2006.

[14] J.-H. Woo, J.-H. Sohn, H. Kim, H.-J. Yoo, "A 195 mW, 9.1 MVertices/s Fully Programmable 3-D Graphics Processor for Low-Power Mobile Devices," *IEEE Journal of Solid-State Circuits,* Vol. 43, No. 11, 2370–2380, Nov. 2008

[15] F. Sheikh, S. K. Mathew, M. A. Anders, H. Kaul, S. K. Hsu, A. Agarwal, R. K. Krishnamurthy, S. Borkar, "A 2.05 GVertices/s 151 mW Lighting Accelerator for 3D Graphics Vertex and Pixel Shading in 32 nm CMOS," *IEEE Journal of Solid-State Circuits,* Vol. 48, No. 1, 128–139, Jan. 2013

[16] T. Burd, R. Broderson, "Design Issues for Dynamic Voltage Scaling," *IEEE International Symposium on Low Power Electronics and Design,* 9–14, 2000.

[17] B. H. Calhoun, A. P. Chandrakasan, "Characterizing and Modeling Minimum Energy Operation for Subthreshold Circuits," *IEEE International Symposium on Low Power Electronics and Design,* 90–95, 2004.

[18] J. Chen, S. Paris, F. Durand, "Real time edge-aware image processing with the bilateral grid," *ACM Transactions on Graphics,* Vol. 26, No. 3, article 103, July 2007.

[19] S. Paris, F. Durand, "A Fast Approximation of the Bilateral Filter Using a Signal Processing Approach," *International Journal of Computer Vision,* Vol. 81, No. 1, 24–52, January 2009.

[20] B. Weiss, "Fast median and bilateral filtering," *ACM Transactions on Graphics,* Vol 25, No. 3, 519–526, July 2006.

[21] P. E. Debevec, J. Malik, "Recovering high dynamic range radiance maps from photographs, *ACM Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '97),* 369–378, August 1997.

[22] B. Zhai, S. Hanson, D. Blaauw, D. Sylvester, "Analysis and mitigation of variability in subthreshold design," *International Symposium on Low Power Electronics and Design (ISLPED),* 20–25, August 2005.

[23] R. Rithe, S. Chao, J. Gu, A. Wang, S. Datla, G. Gammie, D. Buss, A. Chandrakasan, "The effect of random dopant fluctuations on logic timing at low voltage," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* Vol. 20, No. 5, 911–924, May 2012.

[24] "Intel Atom Processor Z2760," http://www.intel.com/content/www/us/en/processors/atom/atom-z2760-datasheet.html.

[25] "DragonBoard Snapdragon S4 Plus APQ8060A Mobile Development Board," https://developer.qualcomm.com/mobile-development/development-devices/dragonboard.

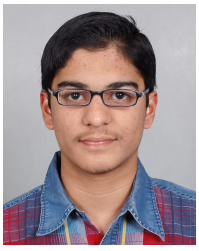
[26] "Samsung Exynos 5 Dual Arndale Board," http://www.arndaleboard.org/wiki/index.php/Main_Page.

[27] "PandaBoard: Open OMAP 4 mobile software development platform," http://pandaboard.org/content/platform.

Rahul Rithe (S'08) received the B. Tech. degree in electronics and electrical communication engineering from Indian Institute of Technology Kharagpur in 2008 and the S.M. degree in electrical engineering and computer science from the Massachusetts Institute of Technology in 2010, where he is currently pursuing his doctoral degree. His research interests include low-power system design for portable multimedia applications. He was awarded the President of India Gold Medal in 2008 and was a recipient of the MIT Presidential Fellowship in 2008.

Priyanka Raina (S'11) received the B.Tech. degree in electrical engineering from Indian Institute of Technology Delhi in 2011 and the S.M. degree in electrical engineering and computer science from Massachusetts Institute of Technology in 2013, where she is currently pursuing her doctoral degree. Her research interests include design of low-power circuits for computational photography applications.

Nathan Ickes (M'11) received his S.B., M. Eng. and Ph.D. from the Massachusetts Institute of Technology in 2001, 2002 and 2008 respectively. He is currently a research scientist in the Digital Integrated Circuits and Systems Group at MIT. His research interests include micropower digital circuits and system architectures, software support for power management, high-reliability and fault tolerance, and biomedical applications of micropower systems.



Srikanth V. Tenneti received the B.Tech degree in Electrical Engineering from Indian Institute of Technology (IIT) Bombay in 2012. He is currently pursuing a PhD degree in electrical engineering at the California Institute of Technology. His research interests include Sub-Nyquist sampling schemes for Parametric Analog Signals. He has worked with the Digital Integrated Circuits and Systems Group at MIT and the Signal Processing and Artificial Neural Networks (SPANN) Lab at IIT-Bombay in the past. He received the best undergraduate thesis award in Electrical Engineering from IIT Bombay in 2012.

Anantha P. Chandrakasan (F'04) received the B.S, M.S., and Ph.D. degrees in electrical engineering and computer sciences from the University of California, Berkeley, CA, USA, in 1989, 1990, and 1994, respectively. Since September 1994, he has been with the Massachusetts Institute of Technology, Cambridge, where he is currently the Joseph F. and Nancy P. Keithley Professor of Electrical Engineering.

He was a co-recipient of several awards including the 1993 IEEE Communications Society's Best Tutorial Paper Award, the IEEE Electron Devices Society's 1997 Paul Rappaport Award for the Best Paper in an EDS publication during 1997, the 1999 DAC Design Contest Award, the 2004 DAC/ISSCC Student Design Contest Award, the 2007 ISSCC Beatrice Winner Award for Editorial Excellence and the ISSCC Jack Kilby Award for Outstanding Student Paper (2007, 2008, 2009). He received the 2009 Semiconductor Industry Association (SIA) University Researcher Award. He is the recipient of the 2013 IEEE Donald O. Pederson Award in Solid-State Circuits.

His research interests include micro-power digital and mixed-signal integrated circuit design, wireless microsensor system design, portable multimedia devices, energy efficient radios and emerging technologies. He is a co-author of Low Power Digital CMOS Design (Kluwer Academic Publishers, 1995), Digital Integrated Circuits (Pearson Prentice-Hall, 2003, 2nd edition), and Sub-threshold Design for Ultra-Low Power Systems (Springer 2006). He is also a co-editor of Low Power CMOS Design (IEEE Press, 1998), Design of High-Performance Microprocessor Circuits (IEEE Press, 2000), and Leakage in Nanometer CMOS Technologies (Springer, 2005).

He has served as a technical program co-chair for the 1997 International Symposium on Low Power Electronics and Design, VLSI Design '98, and the 1998 IEEE Workshop on Signal Processing Systems. He was the Signal Processing Sub-committee Chair for ISSCC 1999–2001, the Program Vice-Chair for ISSCC 2002, the Program Chair for ISSCC 2003, the Technology Directions Sub-Committee Chair for ISSCC 2004–2009, and the Conference Chair for ISSCC 2010–2012. He is the Conference Chair for ISSCC 2013. He was an Associate Editor for the IEEE Journal of Solid-State Circuits from 1998 to 2001. He served on SSCS AdCom from 2000 to 2007 and he was the meetings committee chair from 2004 to 2007. He was the Director of the MIT Microsystems Technology Laboratories from 2006 to 2011. Since July 2011, he is the Head of the MIT EECS Department.