

Research Article

A DVP-Based Bridge Architecture to Randomly Access Pixels of High-Speed Image Sensors

Tareq Hasan Khan and Khan A. Wahid

Department of Electrical and Computer Engineering, University of Saskatchewan, 57 Campus Drive, Saskatoon, SK, Canada S7N 5A9

Correspondence should be addressed to Tareq Hasan Khan, tareq_992403@yahoo.com

Received 14 October 2010; Revised 3 January 2011; Accepted 17 January 2011

Academic Editor: Sandro Bartolini

Copyright © 2011 T. H. Khan and K. A. Wahid. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

A design of a novel bridge is proposed to interface digital-video-port (DVP) compatible image sensors with popular microcontrollers. Most commercially available CMOS image sensors send image data at high speed and in a row-by-row fashion. On the other hand, commercial microcontrollers run at relatively slower speed, and many embedded system applications need random access of pixel values. Moreover, commercial microcontrollers may not have sufficient internal memory to store a complete image of high resolution. The proposed bridge addresses these problems and provides an easy-to-use and compact way to interface image sensors with microcontrollers. The proposed design is verified in FPGA and later implemented using CMOS 0.18 μm Artisan library cells. The design costs 4,735 gates and 0.12 mm^2 silicon area. The synthesis results show that the bridge can support a data rate up to 254 megasamples/sec. Its applications may include pattern recognition, robotic vision, tracking system, and medical imaging.

1. Introduction

In recent years, image sensors have increased in quality and capability and at the same time decreased in price, making them desirable to include in small electronic devices and systems. However, these image sensors are difficult to interface with most commercial microcontrollers (MCUs) as these high-speed image sensors produce data at such a high rate that cannot be processed in real time. As a consequence, most high-speed image sensors are difficult to use in low-power and low-speed embedded systems. There is no buffering provided inside the image sensors. Most MCUs have limited internal memory space and may not be able to store a complete frame unless external memory is provided. Moreover, these image sensors send image data in a row-by-row fashion; as a result, the data cannot be accessed randomly; the first row must be read prior to the second row to avoid data loss. Many image processing algorithms, such as transform coding using the Discrete Cosine Transform (DCT) and pattern recognition for robotic vision, need to access pixel values in a

random access fashion. Besides, a high-speed clock must be provided to operate the image sensors properly.

In order to overcome these difficulties, researchers in the past have proposed application-specific design of image sensors with control circuitry and dedicated memory embedded on the same chip, as shown in Figure 1(a). Such sensors are dedicated to particular application and cannot be used for general purpose. In this paper, we present a digital-video-port (DVP) compatible bridge architecture that will “bridge” any general-purpose image sensor with the image processor as shown in Figure 1(b). In this work, our target is the low-speed and low-power MCU that is realized here as the image processor. The proposed bridge aims to overcome the speed gap between the commercially available image sensors and MCUs. By using the bridge hardware, the image processor can easily initialize any DVP-compatible image sensor and capture image frames. The captured pixel values are then accessed by the image processor at a random fashion through a parallel memory access interface at a desired speed for further processing. The proposed design is synthesized and

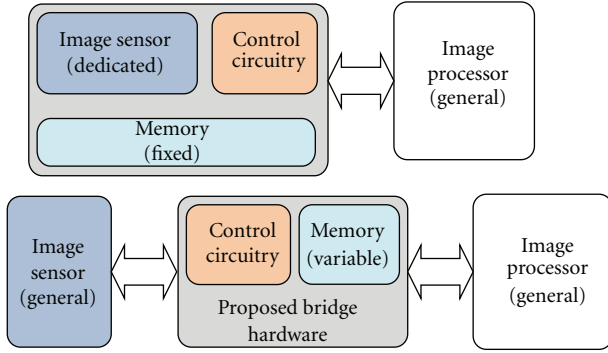


FIGURE 1: Image processor connected with (a) application-specific image sensor and (b) general-purpose image sensor via the proposed bridge.

tested in commercial FPGA board, where the maximum speed achieved is 248 MHz. The VLSI design using standard 0.18 μm CMOS Artisan library cells is also presented. The bridge can be used in various embedded system applications including pattern recognition, robotic vision, biomedical imaging, tracking system, where random access of image pixels is required.

It should be noted that the commercial high-speed image sensors may be interfaced with more advanced MCUs (such as AT91CAP7E, AT91SAM7S512 from Atmel [1]). However, these microcontrollers contain many additional features (such as six-layer advanced high-speed bus (AHB), peripheral DMA controller, USB 2.0 full-speed device, and configurable FPGA Interface) that may not be required for simple imaging applications. Besides, programming such microcontrollers and implementing the required protocols increase the design cycle time. The purpose of the proposed bridge hardware is to provide a compact, ready-made, and easy-to-use solution that enables interfacing of commercial general-purpose image sensors with simple microcontrollers that are low-cost and easy-to-program (such as 8051 [2, 3], AVR [4], and PIC [5]). Thus the bridge hardware helps to shorten the design/development cycle time and facilitates rapid system level prototyping.

2. Background

In [6–10], presented are some VLSI designs on CMOS image sensors with random access. In [11, 12], the authors have presented two different designs of a random access image sensor based on a data-address bus structure. The work in [13] presents a low-power full-custom CMOS digital pixel sensor array designed for a wireless endoscopy capsule [14]. The proposed architecture reduces the on-chip memory requirement by sharing pixel-level memory in the sensor array with the digital image processor. A dental digital radiographic (DDR) system using a high-resolution charge-coupled device (CCD) imaging sensor was developed and its performance for dental clinic imaging was evaluated in [15]. The work in [16] presents a novel smart CMOS image sensor integrating hot pixel correcting readout

circuit to preserve the quality of the captured images for biomedical applications. In [17], an image sensor with an image compression feature using the 4×4 DCT is presented. In [18], a CMOS image sensor has been designed to perform the front-end image decomposition in a Prediction-SPIHT image compression scheme. In [19], an image sensor unit with sensor to detect the gravity direction and a built-in image rotation algorithm is presented. The system rotates the captured image in the direction of gravity for better viewing that can be used in rescue robots. The paper in [20] discusses a range image sensor using a multispot laser projector for robotic applications. In [21], a pointing device using the motion detection algorithm and its system architecture is presented. The proposed motion detection pointing device uses just binary images of the binary CMOS image sensor (BCIS). In [22], a smart image sensor for real-time and high-resolution three-dimensional (3D) measurement to be used for sheet light projection is presented. A facial image recognition system based on 3D real-time facial imaging by using correlation image sensor is discussed in [23]. The differential geometry theory was employed to find the key points of face image. A design of an image sensor focusing on image identification by adjusting the brightness is presented in [24]. It has GPRS connectivity and can be used in vehicle surveillance system. In [25], a single-chip image sensor for mobile applications realized in a standard 0.35 μm CMOS technology is presented. In [26], a solution to reduce the computational complexity of image processing by performing some low-level computations on the sensor focal plane is presented. An autonomous image sensor for real-time target detection and tracking is presented in [27]. In [28], the authors describe and analyse a novel CMOS pixel for high-speed, low-light imaging applications. An 8.3-M-pixel digital-output CMOS active pixel image sensor (APS) for ultra-definition TV (UDTV) application is discussed in [29]. In [30], a hardware accelerator for image reconstruction in digital holographic imaging is presented that focuses to maximize the computational efficiency and minimize the memory transfer overhead to the external SDRAM.

There are some commercial image sensors such as MT9V011 from Aptina [31] and OVM7690 from OmniVision [32] that support partial access of image segments known as “windowing”. By configuring the control resistors, the top-left and bottom-right corners of the desired area can be specified. The image sensor then captures and sends an image of the specified rectangle. However, it is not possible to access (and capture) other segments of the same frame with this feature which is required in several image coding applications such as transform coding. There are two more disadvantages of such approach: firstly, the internal control registers need to be reconfigured every time an image capture request is sent, which is an extra overhead; secondly, because of the time taken for this reconfiguration, the sensor will capture a frame that is different in the time instant. Besides, the “windowing” is limited to rectangles only; the image data cannot be accessed in any other shapes.

In summary, the works mentioned above discuss different designs of image sensors targeted to specific application;

however, they are not available for general-purpose use. In this paper, we present a novel concept—the design of a bridge architecture that connects the commercial MCUs to any commercial DVP-based general-purpose image sensors. The bridge needs to be configured once with a set of addresses (provided by the manufacture as found in the datasheet) in order to communicate with the image sensor, which makes the design universal and for general-purpose use.

3. Design Objectives

Considering the application types (i.e., robotics vision, imaging, video, etc.) and availability of commercial micro-controllers (MCUs), in this work, we have set the following design objectives to facilitate the interfacing of high-speed image sensors with low-performance MCU.

- (i) The bridge hardware should operate at very high speed (over 200 MHz) so that the image pixels can be accessed in real time through high-speed image sensors. As a result, the MCUs (or image processor) using the bridge need not be high performance and high speed.
- (ii) The bridge should contain sufficient memory space to store image frames of different resolutions, such as CIF, QVGA, VGA, full HD, and UHDTV. Thus, an MCU with limited on-chip memory may be able to access image pixels from the buffer memory of the bridge at the desired speed. Moreover, because of the memory buffer, any image segments of the same frame can be accessed without having to reconfigure the image sensor, which is required in many video coding applications. An example of such application is the Discrete Cosine Transform-based image coding, where several 8×8 blocks of image segments of the same frame are required.
- (iii) The bridge should provide an efficient way to access the image pixels randomly. A more convenient way is to access the 2D pixel arrays using parallel interfacing with row and column positions. This will be a significant improvement over the designs with typical data-address structure [11, 12].
- (iv) The usage of the bridge should be robust. As a result, it should provide efficient and easy ways to access image pixels in virtually any shapes, such as rectangles, circles, oval, and points. This facilitates fully random access in any random shapes.
- (v) Commercial image sensors from different vendors have unique device parameters along with internal control registers for proper configuration (such as, to configure frame size, colour, and sleep mode). The bridge should be able to communicate with most available image sensors. As a result, the design should be universal so that it can be configured at the beginning with the proper set of parameters for a particular image sensor.

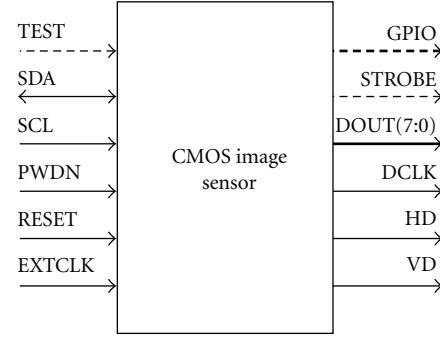


FIGURE 2: DVP interface pins of an image sensor.

- (vi) Commercial image sensors use I2C protocol and DVP interfacing. Hence, the desired bridge hardware must have I2C protocol already configured as well as support DVP interfacing.
- (vii) Most commercial image sensors require high-speed external clock for its operation. It is desirable that the bridge supplies that clock so that the clock can be efficiently controlled during operation (i.e., full clock rate during regular operation, reduced rate during sleep or inactivity, etc.). At the same time, the bridge should be able to detect any inactivity and automatically enable the sleep mode of the image sensor—this will result in power savings.

4. The DVP Interface

Most leading commercial CMOS image sensors, both standard-definition (SD) and high-definition (HD), send image data using a common standard interface, known as the DVP interface. The common I/O pins of a typical CMOS image sensor are shown in Figure 2.

The *VD* (or *VSYNC*) and *HD* (or *HSYNC*) pins indicate the *end of frame* and *end of row*, respectively. Pixel data bytes are available for sampling at the *DOUT(0:7)* bus at the positive edge of the *DCLK* signal. The *EXTCLK* is the clock input for the image sensor. The frequency of *DCLK* is half or quarter of the frequency of *EXTCLK* depending on the configuration of the image sensor. The initialization and configuration of the image sensor is done by the 2-wire (*SCL* and *SDA*) I2C protocol. In the context of image sensor, it is often called as Serial Camera Control Bus (SCCB) interface [32]. The frame size, colour, sleep mode, and wake up mode can be controlled by sending I2C commands to the image sensor. The *RESET* is an active low reset signal for the image sensor. Some image sensors have a pin (*PWDN*) to control the active-sleep mode. Some HD image sensors may contain additional control pins (as shown as dotted line in Figure 2), which are used in special modes; however, these extra pins may be tied to V_{DD} or GND or left unconnected in normal operation.

4.1. Standard-Definition (SD) CMOS Image Sensors. The DVP interface is widely used in most commercially available

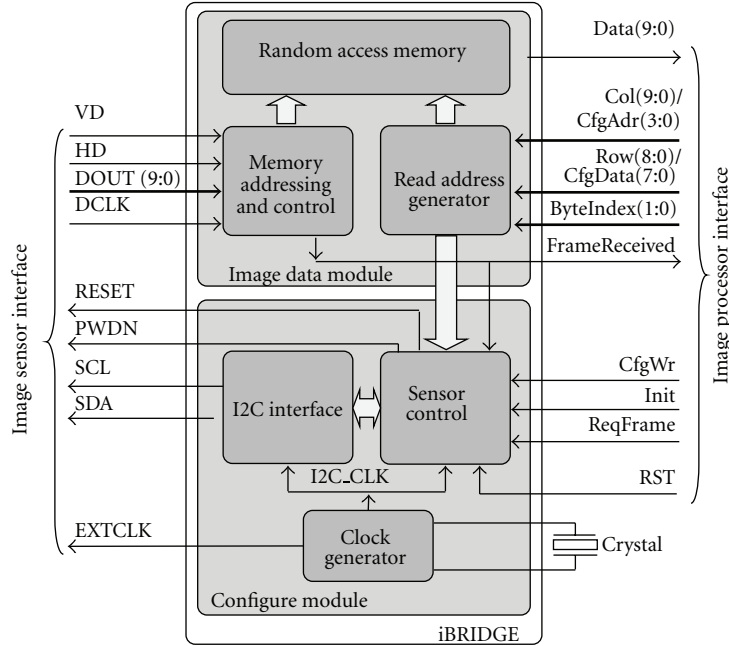


FIGURE 3: Block diagram of the iBRIDGE.

SD CMOS image sensors, such as TCM8230MD from Toshiba [33], OVM7690 from OmniVision [32], MT9V011 from Aptina [31], LM9618 from National [34], KAC-9630 from Kodak [35], and PO6030K from Pixelplus [36].

4.2. High-Definition (HD) CMOS Image Sensors. Most native HD (720p and 1080p) image sensors such as OV10131 from OmniVision [32] and MT9P401 from Aptina [31] use the DVP interface. Some higher-resolution HD image sensors such as OV9810 [32] use an additional interface, called the mobile industry processor interface (MIPI) along with the typical DVP interface. The data output bus DOUT is generally wider than 8 bits in these HD image sensors.

5. The iBRIDGE Architecture

The proposed bridge (referred as iBRIDGE from now) is placed in between the image sensor and the image processor or the microcontroller. Figure 3 shows the interfacing of the iBRIDGE and its internal blocks. The pins on the left hand side are to be connected with an image sensor while those on the right hand side are to be connected to the image processor (or the MCU). There are two types of signals coming from the Image Processor Interface: configuration signals (CfgWr, Init, ReqFrame, and RST) and frame access signals (Data, Col, Row, etc.). The configuration signals are asynchronous in nature whereas the frame access signals depend on the speed of the image processor requesting the “access”; hence these incoming signals do not need to be synchronized with iBRIDGE’s internal clock.

5.1. Configuring the iBRIDGE. The operation starts by first configuring the iBRIDGE’s internal registers with a set of

TABLE 1: Configuration register mapping.

<i>CfgAdr</i> (3:0)	<i>CfgData</i> (7:0)	<i>CfgAdr</i> (3:0)	<i>CfgData</i> (7:0)
0000	Device ID	1000	Cmd2 Reg. Adr.
0001	Total Cmd.	1001	Cmd2 Reg. Data
0010	Sleep Reg. Adr.	1010	Cmd3 Reg. Adr.
0011	Sleep Reg. Data	1011	Cmd3 Reg. Data
0100	Wake Reg. Adr.	1100	Cmd4 Reg. Adr.
0101	Wake Reg. Data	1101	Cmd4 Reg. Data
0110	Cmd1 Reg. Adr.	1110	ImageWidth/4
0111	Cmd1 Reg. Data	1111	Bytes-per-pixel

predefined addresses so that it can properly communicate with the image sensor. Image sensors of different manufacturers have different device ID (or slave address) which is used for such communication using the I2C protocol [37]. The image sensors also have internal control registers used to configure the functionality, such as frame size, colour, and sleep mode, and so forth. These registers are controlled by I2C protocol. The register mapping is also different for different manufacturers; so, the iBRIDGE needs to be configured as well with the proper configuration mapping of the image sensor (found on the datasheet). Table 1 shows the configuration registers implemented inside the iBRIDGE that are required in normal operation. The table may be extended to accommodate additional special features. The content of these registers can be modified by two multiplexed input ports: *CfgAdr*(3:0) and *CfgData*(7:0). In order to write to a register of iBRIDGE, the register address is placed on *CfgAdr*(3:0) bus, the data on *CfgData*(7:0) bus, and

TABLE 2: Data and blank bytes sent by Toshiba image sensor.

Image size	Init BlankBytes	PixelBytes/Row (RGB)	BlankBytes/Row	TotalRow	n
SubQCIF	157	256	1304	254	2
QQVGA	157	320	1240	254	2
QVGA	157	640	920	254	2
VGA	157	1280	280	507	1

the *CfgWr* pin is asserted high. Thus, the iBRIDGE can be configured for any DVP-compatible image sensor which makes it universal.

5.2. Operation of the iBRIDGE. An external crystal needs to be connected with the iBRIDGE to generate the necessary clock signals. The frequency of the crystal should be the required *EXTCLK* frequency of the image sensor. In order to capture one frame, at first image processor asserts the *RST* pin to high and then makes it low. The image capturing process can be started by asserting the *Init* pin to high. The image sensor will set the frame size and colour according to the information provided in the configuration registers and then image data will began to store in the bridge's memory block. During the image capturing process, *Data(9:0)* goes to high-impedance state. As soon as the image capturing process is completed, the *FrameReceived* pin goes from low to high. At this time, the image sensor is taken to sleep mode to save power. The *Col(9:0)*, *Row(8:0)*, and *ByteIndex(1:0)* buses are then used by the image processor to access any pixel of the frame from iBRIDGE's RAM at the desired speed and in a random access fashion. After placing the column and row value on the *Col(9:0)* and *Row(8:0)* bus, the data (pixel value) of that particular location of the frame appears on the *Data(9:0)* bus after a certain time delay, called T_{access} , which is given by (1). Note that, for an SD image sensor, only the lower 8 bits (*Data(7:0)*) are used:

$$T_{\text{access}} = T_{\text{cal.adr}} + T_{\text{mem}}, \quad (1)$$

where $T_{\text{cal.adr}}$ is the time required to calculate the physical memory address from column and row positions and T_{mem} is the access time of the memory. The typical value of T_{access} is 35 nanosecs for the current design constraints. If the image has more than one bytes-per-pixel (such as RGB888, RGB565, and YUV422), then the other consecutive bytes can be accessed by placing the offset on the *ByteIndex(1:0)* bus. After the desired pixel values are read, the process of capturing the next frame with the same configuration can be repeated by asserting *ReqFrame* pin from low to high. Most image sensors send some invalid blank bytes and invalid rows while capturing a frame. The time needed to capture a frame, $T_{\text{Req-Recieved}}$, and the maximum frame-rate, $\text{FPS}_{\text{iBRIDGE}}$, can be calculated from the following:

$$\text{FPS}_{\text{iBRIDGE}} = \frac{1}{T_{\text{Req-Recieved}}}, \quad (2)$$

where

$$T_{\text{Req-Recieved}} = T_{\text{Wakeup}} + T_{\text{FrameStore}}, \quad (3)$$

$$T_{\text{Wakeup}} = \text{I2C_WriteCommandBits} \times \frac{1}{f_{\text{SCL}}}$$

$$= 30 \times \frac{1}{400 \text{ KHz}} = 75 \times 10^{-6} \text{ sec},$$

$$T_{\text{FrameStore}} = \left[\text{InitBlankBytes} + \left(\frac{\text{PixelBytes}}{\text{Row}} + \frac{\text{BlankBytes}}{\text{Row}} \right) \times \text{TotalRow} \right] \times \frac{n}{f_{\text{DCLK}}}, \quad (4)$$

where T_{Wakeup} is the time required for the image sensor to wake up from sleep mode, $T_{\text{FrameStore}}$ is the time required to store a complete frame in memory from the image sensor, *I2C_WriteCommandBits* is the required number of bits that need to be sent to write in the image sensor's internal registers, f_{SCL} is the frequency of the SCL pin of the I2C interface, *InitBlankBytes* is the number of blank bytes sent by the image sensor at the beginning of a new frame, *PixelBytes/Row* is the number of pixel bytes sent by the image sensor for one row, *BlankBytes/Row* is the number of blank bytes sent by the image sensor for one row, *TotalRow* is the number of rows sent by the image sensor for one frame, and n is a constant for dividing the frequency of *DCLK*.

The maximum FPS achieved by an image processor or microcontroller can be calculated using the following:

$$T_{\text{processing}} = T_{\text{mem.access}} + T_{\text{algorithm}},$$

$$T_{\text{mem.access}} = N \times \text{BPP} \times \text{CPI} \times \frac{1}{f_{\text{mcu}}}, \quad (5)$$

$$\text{FPS}_{\text{max}} = \frac{1}{T_{\text{Req-Received}} + T_{\text{processing}}},$$

where $T_{\text{mem.access}}$ is the time needed to access the required pixel bytes from the iBRIDGE's memory, $T_{\text{algorithm}}$ is the time required for implementing any desired image processing algorithm by the image processor, N is the number of random pixels that need to be accessed (in the worst case, $N_{\text{max}} = W \times H$, where W and H are image width and height, namely), BPP is the number of bytes per pixel, CPI is the number of clock cycle required by the image processor to read a byte from iBRIDGE's memory, and f_{mcu} is the frequency of the image processor's clock.

Table 2 shows the above mentioned parameters for a Toshiba image sensor [33]. Similar table can be extracted

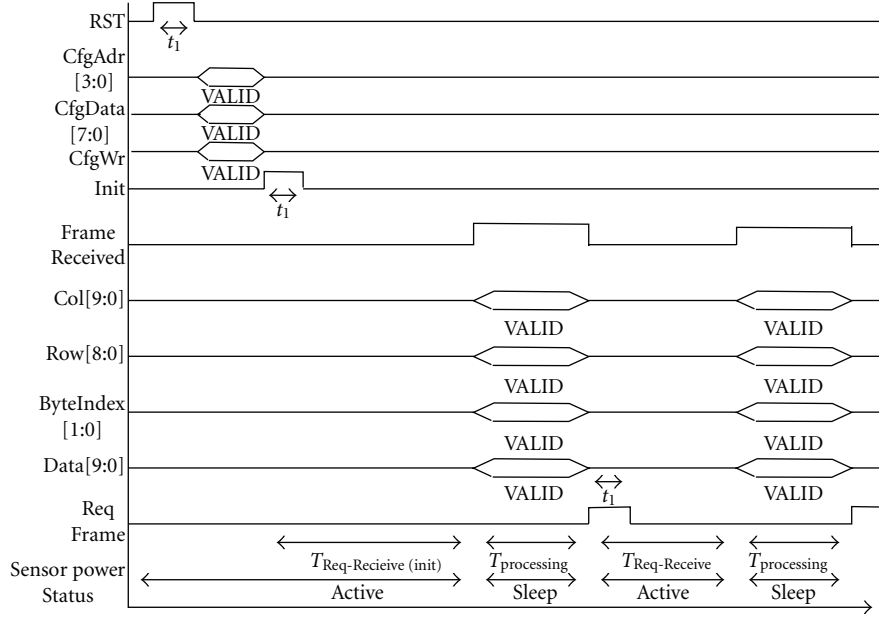


FIGURE 4: Operational timing diagram of the iBRIDGE.

for other image sensors from the respective datasheets. The timing diagram of the overall operation of the iBRIDGE is shown in Figure 4 (here, $t_1 = 1.25$ microseconds to 2.5 microseconds, i.e., equal to at least one I2C_CLK).

The iBRIDGE is also compatible with the HD image sensors that use the parallel DVP interface. The procedure to configure the iBRIDGE is similar to that of the SD sensors as discussed above; however, a full-precision *Data(9:0)* is used to access the pixel data. The following sections describe briefly the internal architecture of the iBRIDGE.

5.3. Sensor Control. This block is used to configure and control different modes of the image sensor. After the *Init* signal is received, it generates the *RESET* signal for the image sensor and then waits for 2000 *EXTCLK* cycle, which is required for the image sensor to accept I2C commands for the first time. After the wait period, it sends commands to the image sensor using the I2C interface block to configure it to the required frame size and colour. The command frames are made by taking the information provided in the configuration register as shown in Table 1. After a wake up command is sent, the image sensor starts to produce image data. After a complete frame is received in the bridge's memory, the controller sends a sleep mode command to the sensor to reduce the power consumption. When a *ReqFrame* signal is received, it sends the wake up command and the next image data starts to store in the memory. The process is implemented in a finite state machine (FSM) structure as shown in Figure 5.

5.4. I2C Interface. This module is used to generate the I2C protocol bits in single master mode [37]. This protocol allows communication of data between I2C devices over two wires. It sends information serially using one line for data (SDA)

and one for clock (SCL). For our application, the iBRIDGE acts as master and the image sensor acts as the slave device. Only the required subset of the I2C protocol is implemented to reduce the overall logic usage.

5.5. Clock Generator. The Clock Generator generates the clock signal at the *EXTCLK* pin, which must be fed to the image sensor. A parallel resonant crystal oscillator can be implemented to generate the clock [38]. An 800 KHz clock signal, called the I2C_CLK, is also required for the I2C Interface and the Sensor Control modules. The clock signal can be generated by dividing the *EXTCLK* using a mod-*n* counter. The I2C Interface module generates clock at SCL pin having half of the I2C_CLK frequency. A simplified diagram of this block is shown in Figure 6.

5.6. Memory Addressing and Control. This module manages the data pins for the image sensor interface and generates address and control signals for the Memory block of the iBRIDGE. It implements a 19-bit up counter and is connected with the address bus of the memory. The *DOUT(7:0)* is directly connected with the data bus of the memory. When *VD* and *HD* are both high, valid image data comes at the *DOUT(7:0)* bus. In the valid data state, at each negative edge event of *DCLK*, the address up-counter is incremented. At each positive edge event of *DCLK*, *WR'* signal for the memory is generated. After a complete frame is received, the address up-counter is cleared and *FrameReceived* signal is asserted high. The simplified diagram of the module is shown in Figure 7.

5.7. Random Access Memory (RAM). A single port random access memory module is used to store a frame. Depending upon the application's requirement, a different memory size

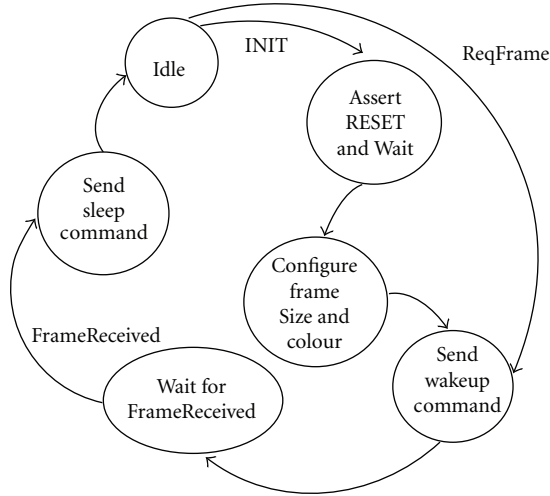


FIGURE 5: FSM in the sensor control block.

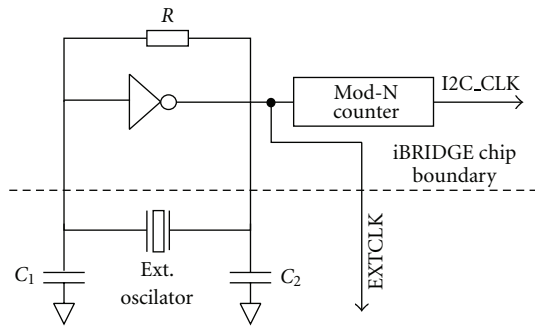


FIGURE 6: Clock generator module.

can be chosen. In the iBRIDGE, one multiplexer for address bus and two tristate buffer for data-bus are used for proper writing in and reading from the memory.

5.8. Read Address Generator. The Read Address Generator takes the *Col* (9:0), *Row* (8:0) and *ByteIndex* (1:0) as inputs and generates the physical memory address from column and row position of the frame. To access a pixel value at column *C* where, $(0 \leq C \leq W-1)$ and at row *R* where $(0 \leq R \leq H-1)$, the physical memory address is calculated using (6). Here, *W* is the image width and *H* is the image height. *Bytes_per_pixel* is taken from the configuration register as shown in Table 1. If the *Bytes_per_pixel* is more than one, the other consecutive bytes can be accessed by placing the offset on *ByteIndex* bus. Figure 8 shows the internal structure of this block:

$$\text{Adr} = \text{Bytes_per_Pixel} \times C + (\text{Bytes_per_pixel} \times W \times R) + \text{ByteIndex}. \quad (6)$$

6. Performance Evaluation

The proposed iBRIDGE design has been modelled in VHDL and simulated for functional verification. As a proof of

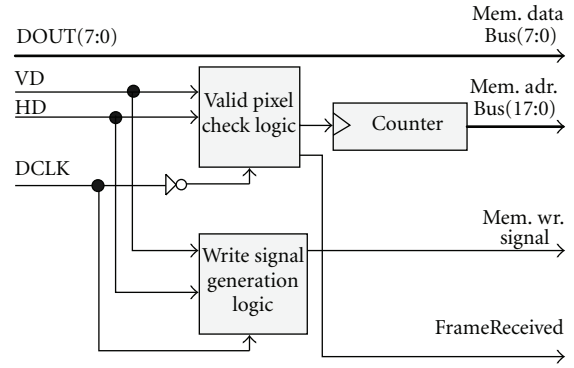


FIGURE 7: Memory addressing and control module.

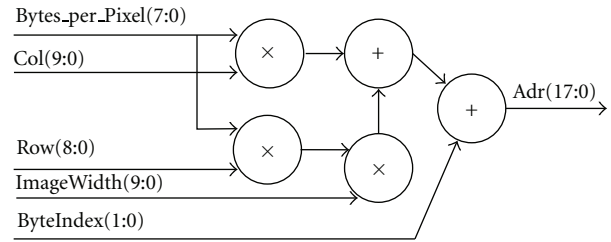


FIGURE 8: Read address generator module.

concept, as well as to evaluate the performance of the design in real-world hardware, the iBRIDGE has been synthesized in Altera DE2 board's FPGA [39]. Several FPGA pins are connected with different on-board components, such as 512 KB of SRAM, clock generators, and 40 general purpose input/output (GPIO) ports. The internal modules of the iBRIDGE, except the RAM, have been synthesized onto the Cyclone II FPGA. It occupies 433 logic elements (LE), 270 registers, and 6 embedded 9-bit multiplier elements. The iBRIDGE's RAM module is connected with the 512 KB SRAM of the DE2 board. The on-board clock generator is used as the clock input for the bridge. The image sensor interface and the image processor interface of iBRIDGE are assigned with different GPIO ports. A commercial image sensor (TCM8230MD) from Toshiba has been used as the image sensor interface where a commercial MCU (ATmega644) from Atmel serves as the image processor interface. The MCU is then connected to a personal computer (PC) using COM port. A level converter IC (MAX232) was used to generate the appropriate logic levels to communicate with the PC. A software is written in MS Visual Basic to display the captured images. The block diagram of the overall experimental setup is shown in Figure 9. The actual setup is shown in Figure 10. In this setup, the microcontroller is set to run at 1 MHz—it shows that the image pixels can still be fully and randomly accessed while running at such a slower rate.

The graphic user interface (GUI) is shown in Figure 11. Here, the user may choose point, rectangle, circle, or full image as the desired region-of-interest. For instance, when the "rectangle" is chosen, the user randomly chooses the top-left and bottom-right coordinates of an image segment using

TABLE 3: Synthesis results on Xilinx FPGA.

Xilinx FPGA device	Area utilization		Max freq. of DCLK (MHz)
	Registers (% utilization)	Logic cells (% utilization)	
Virtex2p, XC2VP2FG256	305 (32%)	484 (51%)	248.0
Virtex4, XC4VLX15SF363	292 (31%)	501 (53%)	200.3
Spartan3, XC3S50TQ144	299 (31%)	492 (51%)	142.6
Virtex E, XCV50ECS144	296 (16%)	1162 (66%)	149.5
Virtex5, XC5VLX330	285 (37%)	368 (48%)	224.7

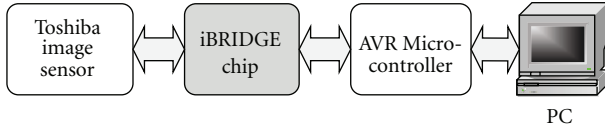


FIGURE 9: Block diagram of the experimental setup for verification.

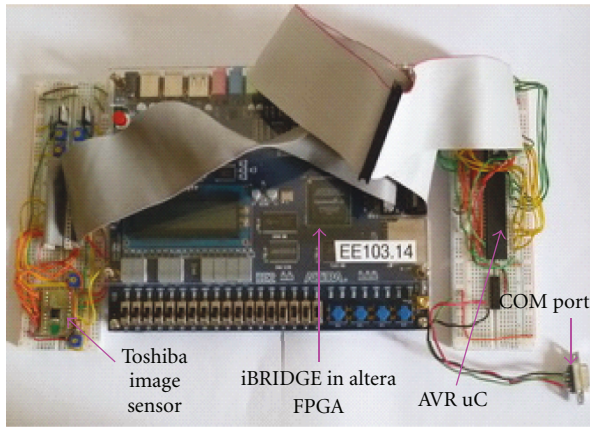


FIGURE 10: Photograph of the actual experimental setup for verification.

the mouse pointer. The software then sends each column (C) and row (R) positions inside the chosen rectangle to the MCU through the PC's COM port. The MCU then places the position values at the row and column buses and reads the corresponding pixel data through the data bus.

Figures 12(a) and 12(b)–12(d) show a full image and randomly accessed images, respectively, captured by the MCU using the proposed iBRIDGE. It is also possible to access the pixel data in other shapes such as ellipse, pentagon, and hexagon. In that case, the GUI needs to be updated with the corresponding geometric equations. As shown in Figure 12, the image pixel thus can be accessed in a random fashion using the iBRIDGE. The demonstration is shown here using the setup shown in Figure 10 and a software GUI; however, similar access is possible in real time using a hardware-coded MCU at the desired speed, which make the iBRIDGE very useful in embedded system applications such as, pattern recognition, robotic vision, bio-medical imaging, image processing, and tracking system.

The iBRIDGE hardware is synthesized using Synopsys's Synplify Pro [40] for different Xilinx FPGA devices. The

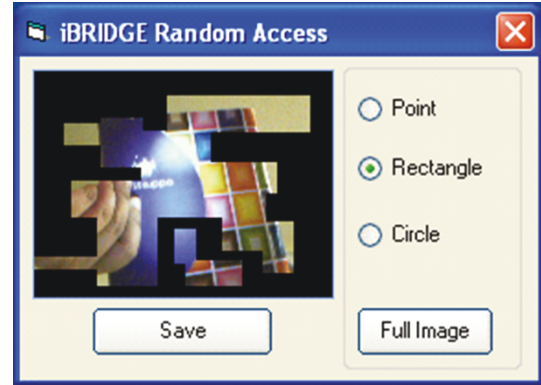


FIGURE 11: A screen-shot of the GUI.

synthesis results are shown in Table 3. It should be carefully noted that these results give us a preassessment of the resource utilization of the iBRIDGE chip when implemented in FPGA. The design is however intended to be used in an ASIC platform.

The iBRIDGE is later implemented using Artisan 0.18 μm CMOS technology. The synthesis results are shown in Table 4. A crystal oscillator pad is placed inside the chip to connect an external crystal. The chip layout (without the memory block) is shown in Figure 13. The design consumes 13.8 mW of power when running at 10 MHz with a 3.0 V supply.

In order to show the significance of the proposed iBRIDGE, we present two sets of comparisons. In Table 5, we compare the synthesized hardware data of the iBRIDGE with other image sensors. The first set of sensors are “application-specific” and do not support random access of the image pixels. The second sets of sensors are of general type and support random access, but the image pixel arrays are dedicated with fixed resolution. While comparing with other sensors, we need to remember that the iBRIDGE does not contain any dedicated image sensor, rather facilitates the interfacing of image sensor with image processor, and enables random access. In that sense, the proposed iBRIDGE can be connected to any “general-purpose” DVP-based image sensors of “any resolutions”—this is a key advantage. As an example, in Table 5, we also present the result when the iBRIDGE is interfaced with an advanced OmniVision HD image sensor (OV2710). With such setup, the performance of the iBRIDGE is noticeably better compared to all sensors in terms of pixel array, silicon area, data rate, and power

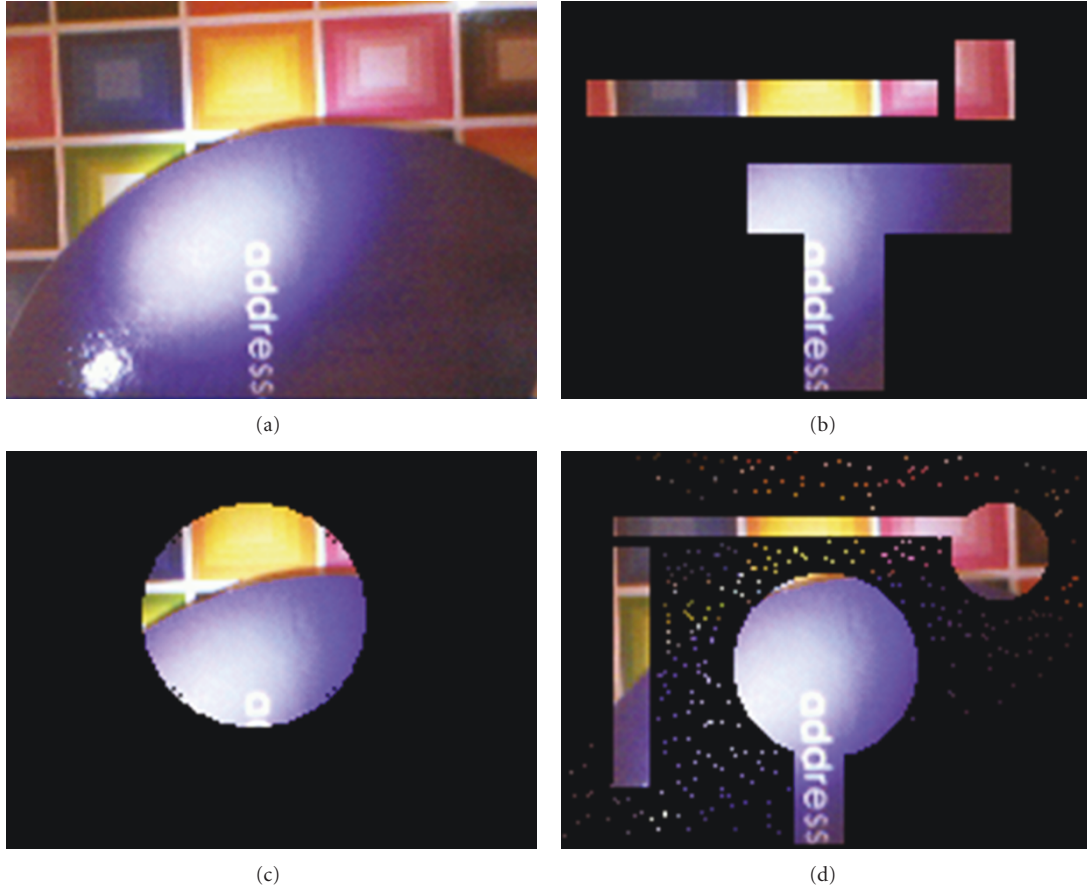


FIGURE 12: Captured image: (a) full image; (b)–(d) randomly accessed pixel image using the iBRIDGE.

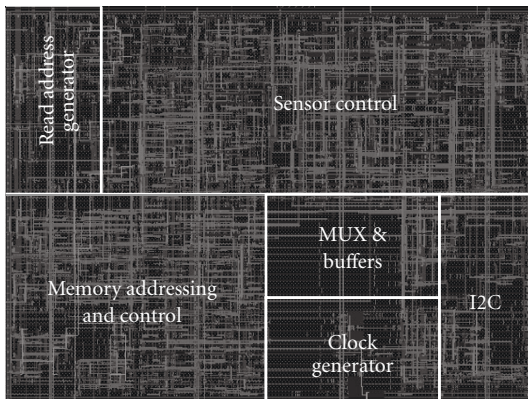


FIGURE 13: Chip layout of the iBRIDGE core.

consumption. Note that, in Table 5, the die area (i.e., core area plus the I/O pads) is used for the iBRIDGE.

In Table 6, we present the performance of the iBRIDGE when interfaced with both SD (TCM8230MD) and HD (OV2710) image sensors. It can be seen that, with a very little increase in hardware (i.e., 1.96 mm^2) and power consumption (i.e., 13.8 mW), any DVP-compatible commercial image sensor can be converted to a high-speed

TABLE 4: Synthesis results in ASIC.

Inputs/Outputs	36/17
Technology	0.18 μm CMOS
Die dimension ($W \times H$)	$1.4 \text{ mm} \times 1.4 \text{ mm}$
Core dimension ($W \times H$)	$0.4 \text{ mm} \times 0.3 \text{ mm}$
Number of cells	1,446
Number of gates	4,735
Max DCLK frequency	254 MHz
Core power consumption	13.8 mW @ 3.0 V

randomly accessible image sensor. Given the data rate, that is 254 megasamples/sec and the equations in Section 4.1, the iBRIDGE supports 333 fps for the VGA (640×480) and 56 fps for the full HD (1920×1080) resolution. It is worth noticing from Tables 5 and 6 that this data rate supported by the iBRIDGE is much higher than other image sensors for same frame resolution.

To show the advantages of the proposed iBRIDGE, in Table 7 we compare the performance of a low-performance MCU interfaced with iBRIDGE with high-performance MCUs. The comparison is based on two scenarios: one where a high-speed image sensor is connected with a high-performance MCU, and another where the same sensor is

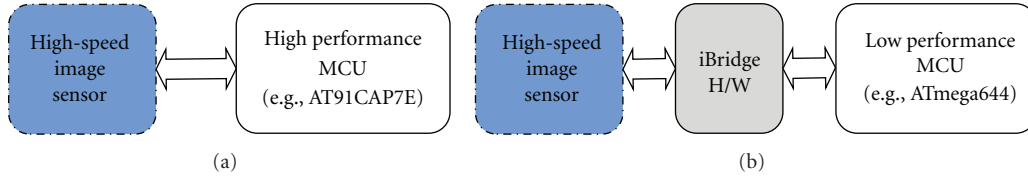


FIGURE 14: Interfacing with image sensor: (a) without iBRIDGE and (b) with iBRIDGE.

TABLE 5: Hardware comparisons with other sensors.

Design	Process	Pixel array (resolution)	Size	Chip area (mm ²)	Data rate	Power (mW)	Random access?	Application type
Zhang et al. [13]	0.18 um	—	2.95×2.18	6.43	2 fps	3.6 @1.8 v	N	S (Wireless endoscopy)
Nishikawa et al. [17]	0.25 um	256×256	10×5	50.0	3,000 fps	—	N	S (Cosine transform)
Lin et al. [18]	0.5 um	33×25	—	—	—	—	N	S (Lossy wavelet)
Yoon et al. [25]	0.35 um	352×288	3.55×2.4	8.52	30 fps	20 @3.3 v	N	S (Mobile communication)
Elouardi et al. [26]	0.6 um	16×16	—	10.0	—	30	N	S (Retina based)
Ji and Abshire [28]	0.18 um	256×256	3×3	9.0	—	—	N	S (Low light)
Takayanagi et al. [29]	0.25 um	3840×2160	19.7×19.1	376.27	60 fps	597@3.3 v	N	S (UDTV)
Teman et al. [27]	0.18 um	64×64	—	—	100 fps	2	N	S (Tracking)
Oi et al. [9]	0.8 um	128×128	5×5	25.0	60 fps	6.8 @3.3 v	Y	S (3D viewing)
Yadid-Pecht et al. [6]	3.0 um	80×80	7.9×9.2	72.68	—	—	Y	G
Scheffer et al. [7]	0.5 um	2048×2048	16.3×16.5	265.69	—	<100	Y	G
Decker et al. [8]	0.8 um	256×256	—	—	390 fps	52	Y	G
Chapinal et al. [10]	0.7 um	128×128	—	16.0	—	6.5 @5 v	Y	G
Dierickx [12]	0.5 um	2048×2048	16×16	256	8 fps	—	Y	G
Proposed iBridge (without sensor)	0.18 um	Any size	1.4×1.4	1.96	254 mega-samples/sec	13.8 @3 v	Y	G
iBridge with OV HD sensor (OV2710) [32]	0.18 um	Full HD 1920×1080	—	45.74	30 fps	363.8 @3 v	Y	G

“—”: not reported; “fps”: frames per sec; “Y”: Yes; “N”: No; “G”: General purpose; “S”: Application specific.

TABLE 6: Performance advantage of iBridge with commercial image sensors.

Design		Pixel array	Area (mm ²)	Data rate	Power (mW)	Random access?
Toshiba SD (TCM8230MD) [33]	Without iBRIDGE	VGA 640×480	36.00	30 fps	60.0	NO
	With iBRIDGE	VGA 640×480	37.96	30 fps ¹	73.8	YES
OmniVision HD sensor (OV2710) [32]	Without iBRIDGE	Full HD 1920×1080	43.78	30 fps	350.0	NO
	With iBRIDGE	Full HD 1920×1080	45.74	30 fps ²	363.8	YES

¹30 fps is the maximum rate supported by the Toshiba sensor. The iBRIDGE, however, can support 333 fps for the VGA resolution.

²30 fps is the maximum rate supported by the Omnivision HD sensor. The iBRIDGE, however, can support 56 fps for the full HD resolution.

TABLE 7: Performance advantage of iBRIDGE with high-performance MCU.

	High-performance MCU		Low-performance MCU + iBRIDGE
	AT91CAP7E	AT91SAM7S512	ATmega644
Cost of programmer	Costly		Cheap
DIP packaging	Not available (one needs an adaptor to mount them on white boards; requires circuit design on PCB)		Available (easily mounted on white boards)
Firmware development (program complexity)	Relatively difficult to program; more control registers to configure; longer development time		Simpler to program; less configuration registers; shorter development time
Resource utilization	Low (many advanced features such as six-layer advanced high-speed bus (AHB), peripheral DMA controller, USB 2.0 full-speed device, and FPGA Interface may not be used for simple imaging application)	Medium (as some advanced features such as full-speed USB 2.0 and Real-time Timer (RTT) may not be used for simple imaging application)	High (the features are simple and may be adequate for simple imaging application)
Power consumption	High (since large MCU is running at high clock speed at all times)		Low (since the small MCU is running at low speed at all times, however, the tiny size iBRIDGE is running at a higher speed)
Image sensor configuration	Complex		Simple
Memory capacity	Fixed (160 KB)	Fixed (64 KB)	Variable (the memory capacity can be varied depending on the application)
Real-time random access of pixels	Complex		Simple (Row-Column addressable)
Power saving mode	Manual		Automated
I ² C protocol	Needs to be configured		Already configured
Maximum speed (at which image sensor can be interfaced)	80 MHz	55 MHz	254 MHz
Types of image resolution supported	SubQCIF, QQVGA, QVGA	SubQCIF, QQVGA	Any resolution (SubQCIF, QQVGA, QVGA, VGA, Full HD, UHDTV, etc.)

connected with a low-performance MCU via the iBRIDGE. The scenarios are shown in Figure 14. It should be noted that, as stated in Section 1, the low-performance MCU cannot be directly interfaced with high-speed image sensors. From Table 7, it can be seen that the iBRIDGE enables simple and quick interfacing of low-performance MCU with high-speed sensors. It also helps to shorten the design/development cycle time and facilitates rapid system level prototyping. Thus the design objectives presented in Section 3 are fully met.

7. Conclusion

In this work, the design of a bridge architecture, named as iBRIDGE, is proposed to overcome the speed gap between commercially available CMOS image sensors and microcontrollers. The iBRIDGE can be configured to work

with any DVP-based SD and/or HD image sensor. By using the proposed bridge, a slow and low-power microcontroller (or image processor) with little memory capacity can communicate with high-speed image sensors to capture images of large size. The pixel data can also be accessed in a random access fashion through a parallel memory access interface at a desired speed. The control and status registers provide a comprehensive control of the image sensor. The I²C communication protocol is built into the iBRIDGE core. The design is power-efficient as the iBRIDGE forces the image sensor to sleep mode when in data-access mode. An integrated clock generator provides the necessary clock signals eliminating the need for external clock source. When implemented using CMOS 0.18 μ m Artisan library cells, the design costs 4,735 gates and 0.12 mm² silicon area. The synthesis results show that the iBRIDGE supports a data rate up to 254 MHz and suitable for rapid prototyping in different high-speed and low-power embedded system applications.

Acknowledgments

The authors would like to acknowledge the Natural Science and Engineering Research Council of Canada (NSERC) for its support to this research work. The authors are also indebted to the Canadian Microelectronics Corporation (CMC) for providing the hardware and software infrastructure used in the development of this design.

References

- [1] J. Uthus and O. Strom, "MCU architectures for compute-intensive embedded applications," Atmel White Paper, 2005.
- [2] L. Fanucci, S. Saponara, and A. Morello, "Power optimization of an 8051-compliant IP microcontroller," *IEICE Transactions on Electronics*, vol. E88-C, no. 4, pp. 597–600, 2005.
- [3] S. Saponara, L. Fanucci, and P. Terreni, "Architectural-level power optimization of microcontroller cores in embedded systems," *IEEE Transactions on Industrial Electronics*, vol. 54, no. 1, pp. 680–683, 2007.
- [4] A. Holberg and A. Saetre, "Innovative techniques for extremely low power consumption with 8-bit microcontrollers," Atmel White Paper, 2006.
- [5] T. Wilmshurst, *Designing Embedded Systems with PIC Microcontrollers: Principles and Applications*, Newnes, 2007.
- [6] O. Yadid-Pecht, R. Ginosar, and Y. Shacham-Diamand, "A random access photodiode array for intelligent image capture," *IEEE Transactions on Electron Devices*, vol. 38, no. 8, pp. 1772–1780, 1991.
- [7] D. Scheffer, B. Dierickx, and G. Meynants, "Random addressable 2048 x 2048 active pixel image sensor," *IEEE Transactions on Electron Devices*, vol. 44, no. 10, pp. 1716–1720, 1997.
- [8] S. Decker, R. D. McGrath, K. Brehmer, and C. G. Sodini, "A 256 x 256 CMOS imaging array with wide dynamic range pixels and column-parallel digital output," *IEEE Journal of Solid-State Circuits*, vol. 33, no. 12, pp. 2081–2090, 1998.
- [9] R. Oi, T. Hamamoto, and K. Aizawa, "Pixel based random access image sensor array for real-time IBR," in *Proceedings of the 1st IEEE International Conference on Sensors (Sensors '02)*, vol. 1, pp. 134–139, June 2002.
- [10] G. Chapinal, S. A. Bota, M. Moreno, J. Palacin, and A. Herms, "A 128 x 128 CMOS image sensor with analog memory for synchronous image capture," *IEEE Sensors Journal*, vol. 2, no. 2, pp. 120–127, 2002.
- [11] R. L. Baer and V. K. Srikantam, "Imaging parallel interface RAM," U. S. Patent no. 20070024713, July, 2005.
- [12] B. Dierickx, D. Scheffer, G. Meynants, W. Ogiers, and J. Vlummens, "Random addressable active pixel image sensors," in *Advanced Focal Plane Arrays and Electronic Cameras*, vol. 2950 of *Proceedings of SPIE*, pp. 2–7, Berlin, Germany, October 1996.
- [13] M. Zhang, A. Bermak, X. Li, and Z. Wang, "A low power CMOS image sensor design for wireless endoscopy capsule," in *Proceedings of IEEE-BIOCAS Biomedical Circuits and Systems Conference (BIOCAS '08)*, pp. 397–400, November 2008.
- [14] D. O. Faigel and D. Cave, *Capsule Endoscopy*, Saunders Elsevier, 2008.
- [15] J. H. Kim, S. G. So, and K. S. Joo, "Development and performances of a dental digital radiographic system using a high resolution CCD image sensor," in *Proceedings of IEEE Nuclear Science Symposium Conference Record*, vol. 3, pp. 1649–1653, October 2004.
- [16] Y. Cao, F. Tang, A. Bermak, and T. M. Le, "A smart CMOS image sensor with on-chip hot pixel correcting readout circuit for biomedical applications," in *Proceedings of the 5th IEEE International Symposium on Electronic Design, Test and Applications (DELTA '10)*, pp. 103–107, January 2010.
- [17] Y. Nishikawa, S. Kawahito, M. Furuta, and T. Tamura, "A high-speed CMOS image sensor with on-chip parallel image compression circuits," in *Proceedings of IEEE Custom Integrated Circuits Conference (CICC '07)*, pp. 833–836, September 2007.
- [18] Z. Lin, M. W. Hoffman, W. D. Leon, N. Schemm, and S. Balkir, "A CMOS front-end for a lossy image compression sensor," in *Proceedings of the 7th IEEE International Symposium on Circuits and Systems (ISCAS '07)*, pp. 2838–2841, May 2007.
- [19] K. Takita and S. Hirose, "Development of image sensor unit for rescue devices," in *Proceedings of the IEEE International Conference on Robotics and Biomimetics (ROBIO '05)*, pp. 687–692, July 2005.
- [20] K. Umeda, "A compact range image sensor suitable for robots," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 3, pp. 3167–3172, May 2004.
- [21] H. J. Park, K. B. Kim, J. H. Kim, and S. Kim, "A novel motion detection pointing device Using a binary CMOS image sensor," in *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS '07)*, pp. 837–840, May 2007.
- [22] Y. Oike, M. Ikeda, and K. Asada, "A CMOS image sensor for high-speed active range finding using column-parallel time-domain ADC and position encoder," *IEEE Transactions on Electron Devices*, vol. 50, no. 1, pp. 152–158, 2003.
- [23] S. Zhan, T. Kurihara, and S. Ando, "Facial authentication system based on real-time 3D facial imaging by using correlation image sensor," in *Proceedings of the 6th International Conference on Intelligent Systems Design and Applications (ISDA '06)*, vol. 2, pp. 396–400, October 2006.
- [24] F. C. Kao, C. Y. Huang, Z. H. Ji, and C. W. Liu, "The design of intelligent image sensor applied to mobile surveillance system," in *Proceedings of IEEE Intelligent Vehicles Symposium (IV '07)*, pp. 264–269, June 2007.
- [25] K. Yoon, C. Kim, B. Lee, and D. Lee, "Single-chip CMOS image sensor for mobile applications," *IEEE Journal of Solid-State Circuits*, vol. 37, no. 12, pp. 1839–1845, 2002.
- [26] A. Elouardi, S. Bouaziz, A. Dupret, L. Lacassagne, J. O. Klein, and R. Reynaud, "A smart sensor for image processing: towards a system on chip," in *Proceedings of International Symposium on Industrial Electronics (ISIE '06)*, vol. 4, pp. 2857–2862, July 2006.
- [27] A. Teman, S. Fisher, L. Sudakov, A. Fish, and O. Yadid-Pecht, "Autonomous CMOS image sensor for real time target detection and tracking," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '08)*, pp. 2138–2141, May 2008.
- [28] H. Ji and P. A. Abshire, "A CMOS image sensor for low light applications," in *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS '06)*, pp. 1651–1654, May 2006.
- [29] I. Takayanagi, M. Shirakawa, K. Mitani et al., "A 1.25-inch 60-frames/s 8.3-m-pixel digital-output CMOS image sensor," *IEEE Journal of Solid-State Circuits*, vol. 40, no. 11, pp. 2305–2313, 2005.
- [30] T. Lenart and V. Öwall, "Xstream—a hardware accelerator for digital holographic imaging," in *Proceedings of the 12th IEEE International Conference on Electronics, Circuits and Systems (ICECS '05)*, Gammarth, Tunisia, December 2005.
- [31] "Aptina MT9V011 Image Sensor," September 2010, <http://www.apptina.com/>.

- [32] “OmniVisoin OVM7690 CameraCube,” September 2010, <http://www.ovt.com/>.
- [33] Toshiba, “TCM8230MD Image Sensor,” September 2010, <http://www.sparkfun.com/>.
- [34] “National LM9618 Image Sensor,” September 2010, <http://www.datasheetcatalog.org/>.
- [35] “Kodak KAC-9630 Image Sensor,” September 2010, <http://www.kodak.com/>.
- [36] “Pixelplus PO6030 Image Sensor,” September 2010, <http://www.pixelplus.com/>.
- [37] “The I2C Bus Specification,” September 2010, <http://www.nxp.com/>.
- [38] R. Wagner, “UART Crystal Oscillator Design Guide,” Data Communications Application Note, March 2000, <http://www.exar.com/>.
- [39] “Altera DE2 Board,” September 2010, <http://www.altera.com/>.
- [40] “Synopsys,” September 2010, <http://www.synopsys.com/>.