

A high-speed video processing and display system

Mustafa Dagtekin, Stephen C. DeMarco, Rajeev Ramanath, Wesley E. Snyder

North Carolina State University
Department of Electrical and Computer Engineering
Raleigh, NC 27695

{mdagtek, scdemarc, rramana, wes} @ eos.ncsu.edu

ABSTRACT

A video processing and display system for performing high speed geometrical image transformations has been designed. It involves looking up the video image by using a pointer memory. The system supports any video format which does not exceed the clock rate that the system supports. It also is capable of changing the brightness and colormap of the image through hardware.

Keywords: Indicial Mapping, Frame Buffer, Pointer Memory, FPGA, Lookup-Table, Video Processing

1. INTRODUCTON

Many hardware-based video processing systems are currently commercially available. These systems usually acquire the video signal using fast hardware and process it using software. The communication between the video system and the computer has such high bandwidth that it requires a very high speed bus. For example, PCI bus, which is usually runs at 33 MHz, yields 133 MBits/sec when all of the 32-bits are utilized. Universal Serial Bus (USB) supports as much as 12 Mbits/sec data transfer rates. Even if these ports are perfectly utilized, they can only support, roughly, up to 30-40 MHz video rates. The traditional systems not only utilize these ports for video processing, but also require a very high speed computer to run the software involved. Going beyond these rates requires compression, which also complicates the software and hardware involved and constrained by the speed of the computer.

In this project, a video processing and display system has been designed which implements arbitrary geometrical transformations and does not require the interface bus fast enough to support video rates. It uses the idea of "indicial mapping", which is proposed by DeMarco et. al. [1] The video signals are converted to digital representation by using an analog to digital converter and stored in a memory (frame buffer). The image is displayed by reading out of the frame buffer, using a digital-to-analog converter to produce an analog signal, and adding the required synchronization overhead. A geometrical transformation, such as rotation or translation, can be done by reading arbitrary locations of this memory since these transformations do not involve with contents of the pixels but rather with their coordinates. This is implemented using another memory unit to hold "indices" of this memory. This index memory is read out sequentially.

A video signal consists of the image data and some synchronization overhead. The video standards differ only by timing issues of the video signal, such as active video length, the percentage of the synchronization overhead etc. and the voltage levels of the video signals. By using a configurable system which is capable of changing these timings, it is possible to support any video standard which uses an analog output. The voltage levels can be handled by a pre-amplifier, which is already present in most display units. Our design uses an FPGA module to implement the timing control. A host computer downloads the timing parameters which are utilized by the FPGA.

This paper is organized as follows; Section 2 provides a background about the indicial mapping approach, video systems and the FPGA technology. Section 3 explains the architecture and the operation of the system. Section 4 briefly describes the software involved. Sectoin 5 concludes the paper and Section 6 gives some ideas about a possible future work.

2. BACKGROUND

In a conventional raster scan image display system, the frames for the video signal are stored in a memory sequentially and displayed in the same fashion. The electron beam sweeps both the camera and the monitor from left to right and top to bottom, and it goes to next line after finishing with one line. In this system, another method is used to display frames. As seen in Figure 1, a second memory which contains the addresses of the pixels in the frame buffers is used. This allows us to process the video in real-time. The additional memory, “pointer memory”, is scanned in a raster-scan fashion and the data output of this memory provides the address input to the frame buffer. The pointer memory is written by the user, hence provides the capability to display the frame contents in an arbitrary fashion. In the illustration below, Figure 1, we refer to addresses as ordered pairs. That is, the pair (3,0) denotes the pixel at row 3, column 0 of the image.

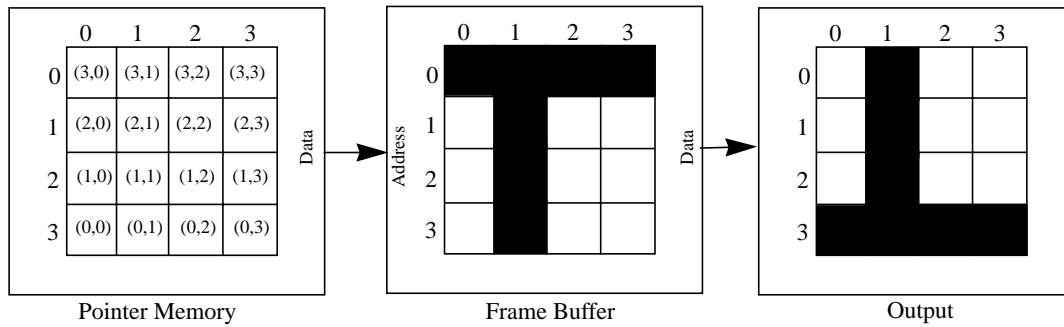


Figure 1: Illustration of Indicial Mapping

Transformation of the image data is performed during the process of display. The transformation, a rotation in this example, is performed by setting a particular pattern into the contents of the pointer memory. Consider for this example, pixel (0,0). In corresponding location (0,0) in pointer memory, is stored the index pair (3,0) is passed as an address to the frame buffer, and the brightness at (3,0) (white) is read from the memory and passed to the display. The values shown in the example of Figure 1 correspond to the mapping of all the pixels in the 4x4 example image, in order to implement a rotation of 180 degrees.

A video signal includes an active video period and a blanking period. In the original vacuum-tube-based television cameras, deflection plates swept the electron beam from left-to-right and top-to-bottom. When the beam reached the right-hand side of the face plate of the tube, it had to be switched off for a brief period of time to allow the deflection plates to be charged to different values. This “electron-beam turned off” time was denoted as “blinking”. In order to comply with standards established for electron beam devices (and most TV monitors are still e-beam devices), the blanking period is still part of the video signal. A typical video signal is shown in Figure 2. The active video period is the information of a frame or a field and each of these is separated by blanking period. A blanking period includes a sync tip which is used to synchronize the display unit with the video signal. There are two types of blanking occur in a video system: *horizontal* and *vertical blanking*. Horizontal blanking determines the end of a line and the vertical blanking determines the end of a frame. The timing of these periods determine the video standard. For example, NTSC standards uses a total of 63.5 ms per line, 14% of which is the blanking period. That standard uses 525 total lines for each frame, 480 lines for active video signal and the rest is for blanking. There are a number of other standards like NTSC, including PAL, SECAM or VGA standards which simply differ from NTSC by the timings mentioned and the voltage levels as mentioned earlier. Our system is capable of handling any video standard, provided the PC board can support the clock speed required by that standard. This is done by using an EPLD/FPGA device.

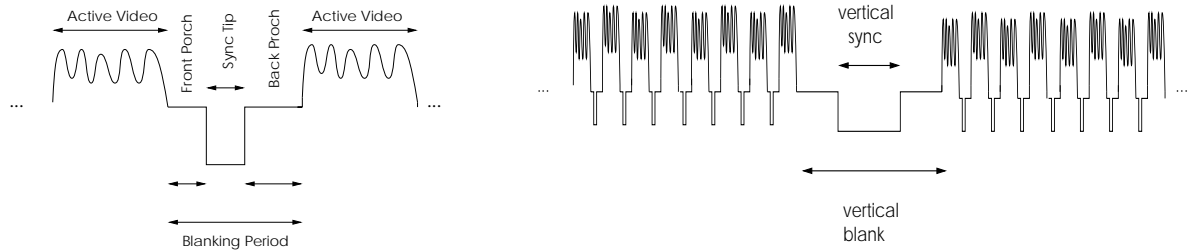


Figure 2: Horizontal and vertical blanking

Field-Programmable Gate Arrays (FPGAs) and Complex Programmable Logic Devices (CPLDs) are electronic chips that have a large number of uncommitted transistors and a configuration engine. Such a device is programmed using a hardware description language and a FPGA synthesizer program. Based on the configuration, the pins may be determined as input, output or bidirectional and the behavior of the system is configured. The unit, once programmed, may implement essentially any logic device/system, including combinational and sequential logic [2]. Some of them use an external EEPROM for configuration, some have an internal EEPROM and some of them are SRAM-based. The SRAM-based units require reprogramming every time the system is restarted.

3. SYSTEM ARCHITECTURE

The top level block diagram of the system is shown in Figure 3. This diagram illustrates the overall architecture, but omits some details. The system consists of several memory units, a PCI interface unit, an analog-to-digital converter, a digital-to-analog converter, an FPGA unit, several flip-flops and multiplexers. The system simply captures every frame that comes from the camera, processes it and sends to the display. The user interaction is done through a PCI interface and that is done in a very small amount of time.

In order to use and manipulate the video data that is supplied from the camera, the analog video must be converted to digital data. This is done by a 10-bit analog-to-digital converter. The digitized video is stored in memory units called “frame buffers”. In order to consider the operation as a real time operation, the video signal supplied from the camera has to be captured, and the processed video has to be displayed simultaneously. A single memory module cannot simultaneously read one location while writing another, therefore, one memory cannot be used to both store and display video concurrently. For this reason, two frame buffers are used in our design. Video information from the A/D converter is written into video RAM (VRAM) 1, while the contents of VRAM 2 are displayed. The functions of the two VRAMs are swapped each frame. Thus, the video signal is displayed with a one frame delay. The video signal only needs to be sampled during active video. During the blanking interval, the information sampled by the ADC is not stored. In our design, the video is sampled at every rising edge of the clock.

Frame buffers are written using sequential addresses generated by the address generator module, but are read using the contents of the pointer memory. The pointer memory module is also addressed by address generator module. Thus, the address generator module supplies the same address for the pointer memory and the frame buffer that is being written. A case is illustrated in Figure 4. In this illustration, VRAM #1 is used in write mode while the other video RAM is used in read mode. At the end of the frame cycle these buffers are swapped.

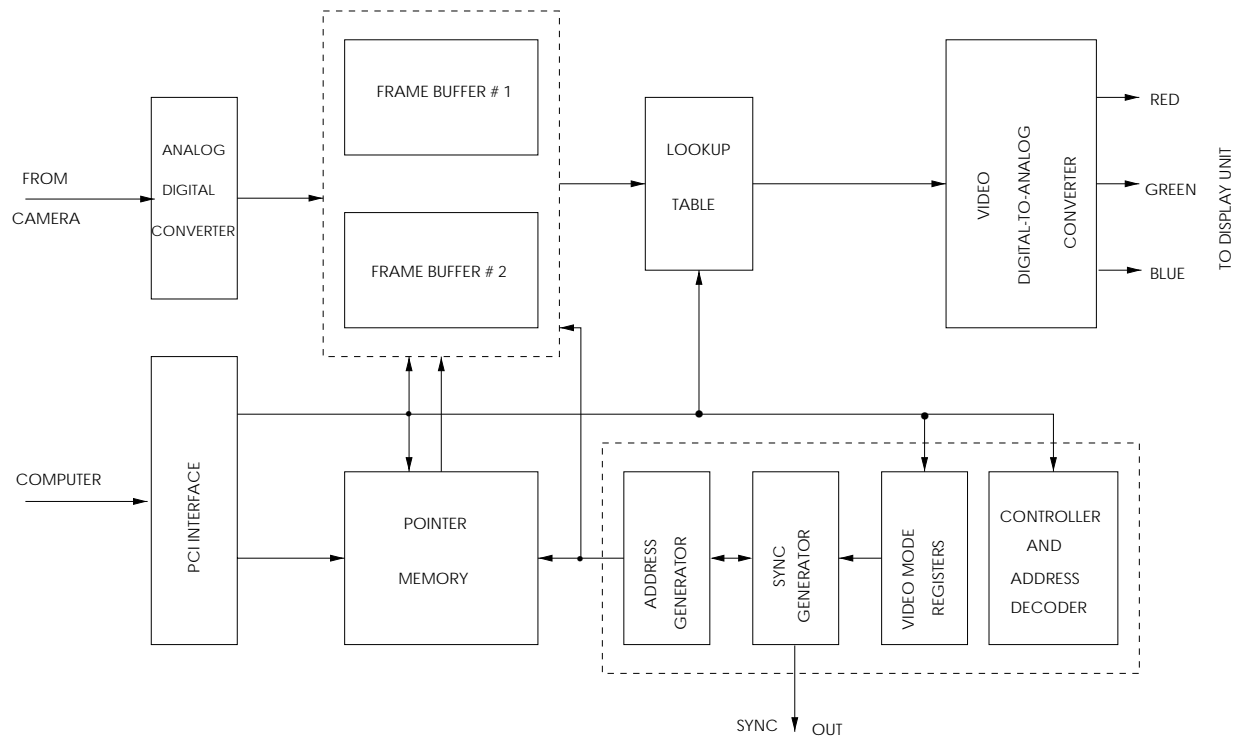


Figure 3: System architecture

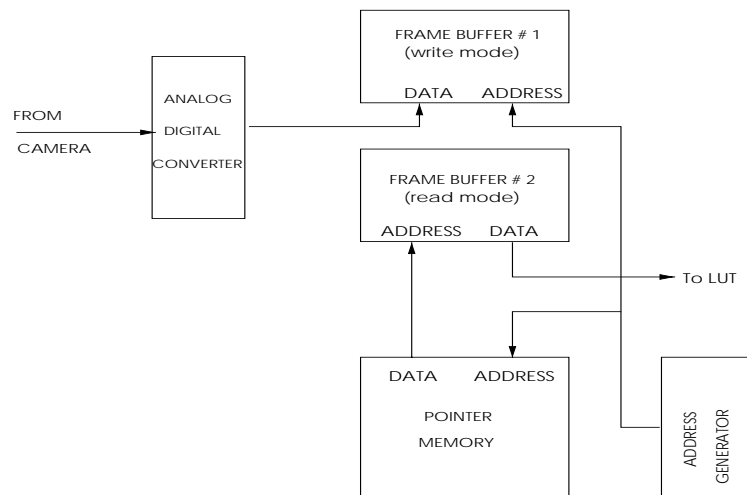


Figure 4: Normal operation mode

The Look-Up Table (LUT) consists of a sequence of brightness values, each mapping (not necessarily uniquely) to a brightness scale from 0 through 255. This scale depends upon the capabilities of the display device. In our case, brightness values are

mapped from 0 through 1023, giving us a complete range of 1024 levels. There are three lookup tables, each corresponding to the desired intensity at a specific color (amount of red, amount of green, and amount of blue)

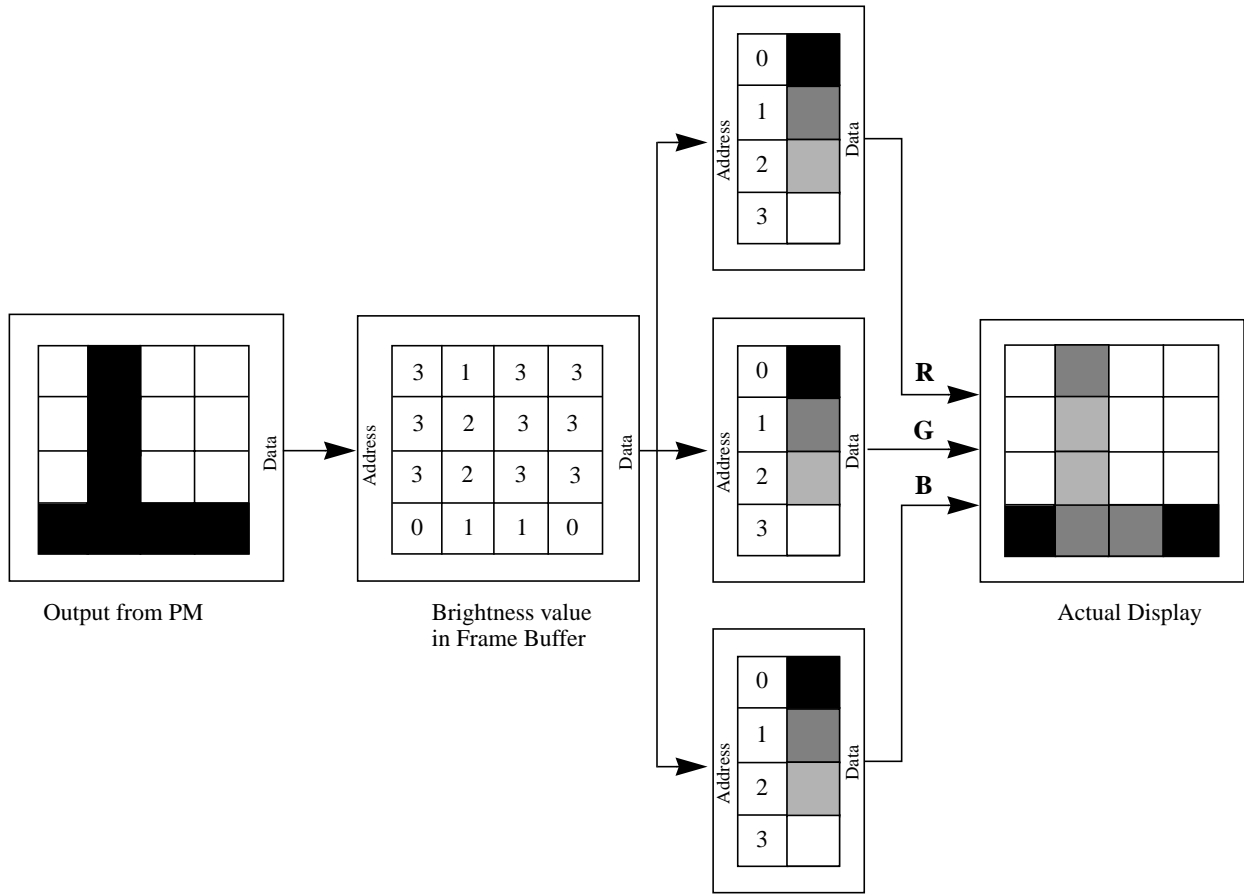


Figure 5: Illustration of the Lookup Tables

When data is read out from the frame-buffer, it is “looked-up” in the LUT and sent to the display device. In the illustration in Figure 5, four gray-scale levels have been defined in the LUT - this gives us a dynamic range of four brightness values which the system is capable of displaying.

Expanding this concept to three independent colors in the spectrum, Red, Green and Blue, each value in the frame buffer (though bounded between 0 and 1023) can be mapped to colors in the visible spectrum which can be obtained as a result of combining the red, green and blue components. This essentially gives us a complete range of 1024^3 colors to choose from. This combination of the three basic colors which depict the colors contained in an image is called a *colormap*. The 10-bit output of the frame buffers are address inputs to the LUT and the LUT puts out 8-bits green, 8-bits red and 8-bits blue video output.

The board sends out intensity values to the Look-up Table (LUT) which sends an output correspondingly. Each intensity input (on a 0 to 1023 scale) results in an output from the LUT of a 24 bit number which contains three fields of 8 bit colors, one in each color (Red, Green and Blue). The software corresponding to this device has in its memory, stored colormaps, such as GrayScale, Inverted GrayScale, Random, Bronson, Heated Spectrum and HotMetal. It also permits the user to define a custom colormap and load it into the LUT. Since the colormap can be modified as desired, functions like level-shifting and windowing

can be implemented with reasonable ease. This aids in contrast enhancement and improving visual appearance of the display.

Digital-to-analog conversion is done by a video DAC. It accepts the RGB data from the output of the lookup table as well as synchronization signals generated by the FPGA and supplies the analog video output of the system. The video signals are run out to BNC connectors to be displayed by an appropriate monitor.

The controller and the address decoder are implemented in the FPGA. Together, these modules handle the PCI accesses and operation modes. The controller module operates the system when it is not in the PCI access mode. These modes are supported by the controller:

- *Normal mode*: This mode of operation causes one frame buffer to be written with the video input data while the other one is being read, and its output passed to the D/A to be displayed.
- *Freeze frame mode*: This mode is used to freeze a picture. In order to do this, one of the frame buffers is not allowed to be written and whilst is being read continuously.
- *Blank mode*: This mode is used when the video signal is in blanking period. The controller disables writing to and reading out of memories during this time.

The address decoder module is used when PCI access is granted. Based on some address bits, the PCI bus is allowed to read/write the memory modules and video mode registers. Since the host computer can write the VRAMs via the PCI bus, the computer can send an image to the VRAM, thus functioning as a graphical display unit.

The address generator is implemented in the FPGA module. The address generator generates sequential addresses for accessing the pointer memory, and (during write cycle) for addressing one of the frame buffers. The generation of the addresses is done by using a counter. The address generator also coordinates with the sync generator to stop counting during blanking cycles.

The sync generator is also implemented in the FPGA. It generates “blank” and “sync” signals which are the synchronization signals for the system. These signals are used to append blanking intervals into the analog video data, a function provided by the video DAC.

The reconfigurability of the system allows the system to support any video mode. The video mode registers are implemented in the FPGA to hold the values to determine the video mode. These registers contain such information as the number of pixels on a line, the length in clock pulses of the front porch (the time between the onset of blanking and the onset of sync), the sync pulse itself, and the back porch, (the time after the sync pulse before the end of blanking). The sync generator uses the values in these registers to generate the sync and blank signals.

In our system the user access is handled by a PCI interface. The user can update the pointer memory contents, lookup table contents, video modes and read from/write to the frame buffers. The ability to combine freeze frame with VRAM read provides the use of the board as a video data acquisition unit. Writing to the pointer memory is an important of this system, because the geometric video processing is based on the content of this memory. Changing the lookup table contents will determine the brightness and colormap. Video modes, including all the sync generator timing parameters, are also written through PCI bus. Finally, the system is capable of displaying static images and the data for these images is supplied by the PCI interface. This is done by putting the system into freeze frame mode and writing a static image into one of the frame buffers.

4. SOFTWARE

Considerable device-driver software was required to use and control the device. Software was written to load the VRAMs, compute timings, and provide all the other functions described above. A user-friendly graphical user interface was written which provides all the capabilities of the board under real-time control. The GUI includes software for computing what the pointer memory contents need to be in order to accomplish rotation, translation, and zoom. Although the board is capable of any geometric transformation of the data, only these functions have been implemented in the current implementation.

All three functions may be implemented as a single matrix multiplication. For example, in order to compute the operation, “rotate by q degrees, then translate by $[dx,dy]$ ”, one first augments the position vector $[x,y]$ by appending a 1, producing $[x,y,1]$. Then the rotation and translation are accomplished by multiplying the point $[x,y,1]$ by a 3×3 matrix as illustrated

below:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & dx \\ -\sin\theta & \cos\theta & dy \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

The mixture of zoom, translation, and rotation in the field of view is accomplished by using a similar coordinate transformation matrix, hidden from the user. The GUI user simply sees four sliders designating how much zoom, rotation, and translation (in x and in y) are desired.

5. CONCLUSIONS

In this project, a video display system that is capable of real-time geometrical image processing has been implemented. The system uses the “indicial mapping” idea, which involves using indices for the image to be displayed. The pointer memory, which is used to hold the indices, is scanned sequentially allowing the frames to be displayed in an arbitrary way. The system is also capable of changing the brightness values and the colormaps utilizing another memory, a brightness/color lookup table. Also, this system is capable of handling any video format which does not require a pixel clock that the circuit board cannot support, and the system is designed to support any pixel clock up to 65 MHz. The current implementation of the NTSC video standard only requires a 12MHz clock.

The system is implemented into a FPGA-based PCB board and configured using NTSC standard. Control and application software has been written and tested.

6. FUTURE WORK

Using FPGAs is not a new approach in video and image processing hardware. But normally these modules are used for design changes and they are not meant to be user-accessible. In order to give user a lot options for a video processing hardware, the system must be reprogrammable. In our system we demonstrated this by providing user selectable video modes. But this can be extended by exploiting the fundamental characteristics of the FPGA modules: *reconfigurability*. That is, we have the capability and software to implement new operation on the board, by downloading new and different code into the FPGA.

Another issue is that our system is only capable of doing geometrical image transformations. That is, we can take a pixel from anywhere in the image and put it anywhere else, but we cannot do neighborhood operations in which a single output pixel depends on more than one input pixel. But there are new FPGA/CPLD modules in the industry which are very dense and fast. For example, Altera Corporation’s Apex device has more than 2 million gates and can be as fast as 150 MHz by a very careful design [4]. This may be exploited to design new systems which are also capable of other image processing issues such as blurring, denoising, segmentation, filtering etc.

REFERENCES

1. S. Demarco, W. E. Snyder, G. L. Bilbro, “Indicial Mapping - A technique for performing real-time, arbitrarily complex, spatial transformation of images”, Unpublished
2. D. H. Ballard, C. M. Brown, *Computer Vision*, Prentice Hall, New Jersey, 1982
3. Altera Corporation, Altera MAX7000 Data Book, 1997
4. Altera Corporation, APEX 20K Programmable Logic Device Family Data Book, 1999
5. J. V. Oldfield, R.C. Dorf, *Field Programmable Gate Arrays*, Wiley-Interscience, New York, 1995
6. A. C. Luther, A. F. Inglis, *Video engineering*, McGraw-Hill, New York, 1999.