Circuits, Systems, and Signal Processing (2023) 42:724–747
https://doi.org/10.1007/s00034-022-02233-x

Brain Tumor Detection from Brain MRI Using Soft IP Core on FPGA

Nazifa Tabassum[1] · **Sheikh Md. Rabiul Islam**[1] · **Farzana Bulbul**[1]

Received: 4 April 2022 / Revised: 2 November 2022 / Accepted: 2 November 2022 /
Published online: 22 November 2022

Abstract

Field-programmable gate array (FPGA) attempts a proper solution for fulfilling the requirements of high-performance real-time DSP systems. Any IP core based on FPGA has the benefit that it merges flexibility, timing efficiency, and algorithm adaptations from programmable logic with the efficiency provided by the processor placed inside the system. This type of tectonics is a compatible approach for the implementation of real-time biomedical applications. In this work, we have designed and proposed a soft IP core on FPGA that can detect the presence of brain tumors from MRI images with noticeably great performance. This designed brain tumor detection system requires only 6.49 μs to give satisfactory output. It is a low-cost system that can perform more flexibly than the alternate CPU-based approaches for having dynamic reconfigurability. The XILINX VIVADO Integrated Design suite has been used as the software designing platform. This designed IP core can read several brain MRI images and process them in parallel. The average power consumption of this IP core is around 82 mW and the maximum memory space is 30.906 MB. Therefore, this design can be used effectively as a faster, small power and memory-consuming system for clinical usage.

Keywords Brain tumor · Xilinx · VIVADO · FPGA · IP core · MRI

✉ Nazifa Tabassum
nazifa.ece12@ece.kuet.ac.bd

Sheikh Md. Rabiul Islam
robi@ece.kuet.ac.bd

Farzana Bulbul
farzanaece2k15@gmail.com

[1] Department of Electronics and Communication Engineering, Khulna University of Engineering and Technology, Khulna 9203, Bangladesh

1 Introduction

Field-programmable gate arrays (FPGAs) [18] in embedded designs are becoming increasingly popular due to their capability of dynamic reconfigurability even in run-time, thoroughly or to some extent. It has become a very cost-effective [22] and time-efficient solution for embedded and custom hardware designs [15]. The benefit of using FPGAs is that the functionality that the system has, can be modified or updated at a later time. This is the additional functionality available in the FPGAs that can be completely reprogrammed [2, 17] with new logic. Whenever there is a need to modify or change the logical operations within an FPGA without disturbing the whole logic and without obstructing the flow of data and it is required to update the functionality within any specific block, it can be partially reconfigured. Thus, FPGAs are the best choice for implementing real-time signal and image processing operations. The FPGAs are configured traditionally using Hardware Description Languages and Verilog HDL and VHDL (VHSIC) [19] are the basic languages for this purpose, which offer designers to design a different level of abstraction. The advantages of FPGAs have made them more applicable for biomedical applications as biomedical signals and images are to be analyzed for diagnosis purposes within a very short time. Also, these applications of diagnosis should be portable for flexibility in analysis.

Any systems-on-chip (SoC) [10] implemented on FPGAs has the advantage that it merges flexibility with algorithm adaptability from programmed logic with the incredible proficiency of the processor inside the system. This type of design is the best choice for biomedical applications as it has to work on real-time signal and image processing purposes. The advantage of the FPGA approach to designing the designed IP core is to make the system very efficient and faster than the existing methods, also having lower costs and more flexibility than the alternate approaches. An IP (intellectual property) core [23, 25–27] is a set or block of data or logic operations that are used in designing a field-programmable gate array (FPGA) or application-specific integrated circuit (ASIC) for a specific product. Intellectual property (IP) means a set of previously configured logic functions that can be used for implementing custom designs. The Xilinx® company provides a vast collection of IPs that are optimized for designing Xilinx FPGAs.

FPGA-based image processing works have been studied for the last two decades [8, 11, 12]. We want to be introduced an FPGA-based medical diagnosis system that can identify tumor-type abnormalities from MRI images using the logic blocks and analyzing the pixel values. There are many applications of FPGA in biomedical applications. Some work has been done on FPGA for different signal or image processing applications. Several researchers have studied the implementation of filters on FPGAs. The proposed design of the FPGA IP core can be used in biomedical applications in such a way that can remove the human workload with an incredibly fast response time. The main intention is to design an IP core that is a complete system for medical image processing and can detect brain tumors from reading brain MRIs.

In 2006, an FPGA-based real-time image segmentation technique was developed by Dillinger [16] which was used for data processing in medical systems. A medical image compression technique was designed in Xilinx FPGA that used wavelet transforms in 2009 [4]. For de-noising medical images, an FPGA-based architecture was

designed in 2010 by Ahmad et al. [1]. In the same year, an FPGA-based medical image segmentation algorithm was developed by Cinar et al. [14]. This image segmentation technique was based on the k-NN classifier. A medical image data acquisition system was designed and implemented in 2011 which is based on FPGA [30]. In 2014, a simultaneous reconstruction and segmentation technique was developed in FPGA for medical images by Li et al. [21]. They used computed tomography (CT) images for this purpose. A novel VLSI architecture was designed on FPGA by Nelakuditi et al. [24] for real-time image segmentation. They used image filtering and thresholding technique for the overall system design. In the same year, an image enhancement algorithm was developed by Chiuchisan [13]. In 2016, an edge detection algorithm for medical images was implemented on FPGA [7]. All of these researches were conducted for different image processing techniques on FPGAs. Moreover, some research has been conducted to process MRI images on FPGAs. A brain MRI image classification system was designed in 2010 by Binothman et al. [5]. This classification was done by using the support vector machine (SVM). A brain tumor detection system was developed by Thomas and Prasanna Kumar [29] that uses brain MRIs. This system is based on image segmentation and morphological processing. In 2016, a feature extraction algorithm was developed in FPGA that uses MRI images and is based on segmentation and wavelet analysis [9]. In 2018, an MRI gradient generation system was implemented on FPGA [20]. Again, a segmented feature fusion in MRI images was implemented in FPGA that uses a wavelet [31]. An overall system that can work on medical MRI images to detect brain tumors within a very short time remains a big challenge. This work intends to fill this gap and a smart IP core is intended to design for fulfilling the purpose. Also, the accuracy of this kind of IP core has to be very high as they are designed for real-time biomedical systems.

2 Materials and Methods

2.1 Data Collection

For brain tumor detection we required a set of brain MRI images for working on the abnormality detection regarding the presence of a tumor. We collected a set of normal and abnormal brain MRI Images for setting reference values and testing purposes, respectively. In this case, we collected Brain MRI images from GitHub [https://github.com/topics/brain-mri] [26] and then convolved the images with Sobel operator Kernels to get the edge-detected images at the output. The edge-detected images are in binary format as they contain either 0 s or 1 s. This is because we have to work on the binary format to set the logic of the codes to work on this. The pixels of binary format is the main dataset of this proposed system. We collected about 20 images among which 10 were of normal brains and those were used for choosing the reference value for this design. The rest of the images (10 images) contained tumor which was used for testing the system.

2.2 Proposed Methodology

The block diagram of a general MRI image processing on FPGA is shown in Fig. 1. Firstly, the binary forms of the MRI images were taken as input. The Xilinx VIVADO integrated software environment was used for executing Verilog codes [26]. An algorithm was developed to find out brain abnormalities or tumors from MRI images and a Verilog code was designed according to the algorithm. The Verilog code was designed in such a way that could work on digital logic to read out the pixel values. Simulation is a program that allows users to perform virtual processing without actually operating in real time. Again, the synthesis step is started after the simulation step. Both the simulation and the synthesis parts were done by using the Xilinx VIVADO integrated software environment. When the simulation part was ready, the code is executed on Altera DE II FPGA Board via using Quartus® Prime software.

We intend to design logic blocks in Verilog to read MRI Images and find out the specific spatial area or segment the images based on some predefined criteria. In this case, we have used Brain MRI images for brain tumor detection.

Figure 2 shows the block diagram of this proposed work. Firstly, an MRI image is taken and the pixel values from the grayscale image are read for one clock pulse. The grayscale image has pixel values between 0 and 255. We require 8-bit registers to store these pixel values. When all the pixel values are stored in registers, we define two 3×3 masks for horizontal and vertical gradient calculation used in the Sobel Edge detection operator. After edge detection, we get the binary image in which the edges are shown. The pixel values of these edge-detected images are then stored in another register. As the edge-detected images are in binary format, only two values

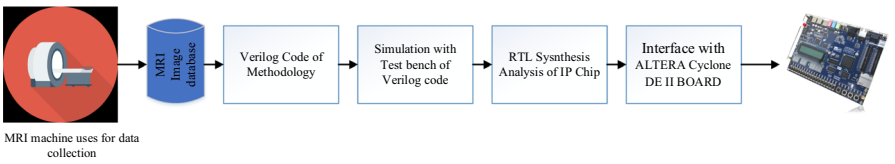


Fig. 1 General block diagram of MRI image processing on FPGA

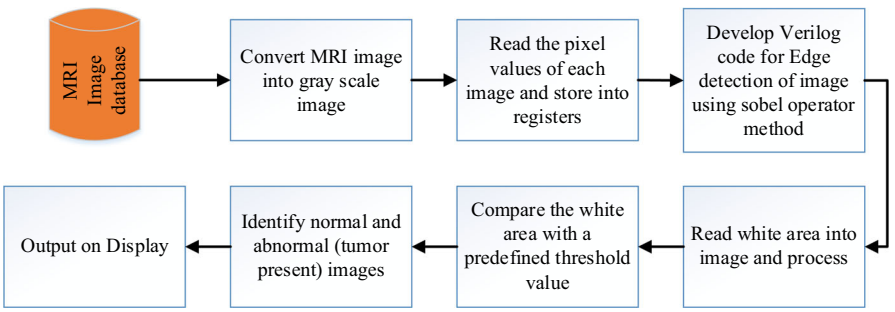


Fig. 2 Block diagram of the proposed methodology for MRI image processing

are stored in the registers (either 0 or 1). These 0's and 1's are then calculated and the percentage is determined. The no. of 1's indicates the white area is detected and the percentage of the white area is calculated. Previously, some normal Brain MRIs were used in the same procedure to define a threshold value. From the detected white area in the normal images, the threshold value is chosen and this is used as the reference value in brain tumor detection. Once the detected white area percentage is compared to the threshold, the brain condition can be identified. The proposed methodology is designed in Verilog codes on Xilinx VIVADO integrated software environment (ISE) Suite. We designed the Verilog code using logic gates and clock pulses and set the threshold value for comparing with pixel values and locating the desired area to be detected. Then from the detected pixel values, the desired result is shown on the display. The designed algorithms of the proposed system were tested on the ISim simulator of Xilinx VIVADO integrated software environment (ISE).

2.3 Edge Detection Using Sobel Operator

The first thing for MRI image processing is to read the grayscale image and store the pixel values. As in grayscale image the pixel values vary between 0 and 255, we store the values in 8-bit registers first. Then convolution [3, 6] with the 3×3 masks is performed on the images. The image pixels are sub-divided into 3×3 size and then convolved with the masks. Figure 3 shows the convolution process of the images. The Sobel edge detection is a gradient-based edge detection method and the Sobel operator is a linear operator. The 0° convolution kernel moved pixel by pixel and then line by line across the image (as shown in Fig. 3) to detect an edge in the x-direction. Similarly, the 90° convolution kernel moved pixel by pixel and then line by line across the image to detect edges in the y-direction. The 0° convolution kernel convolved with input image gives gradient component G_x, 90° convolution kernel convolved with input image gives gradient component G_y.

Where,

$$G_x = (p_7 + 2p_8 + p_9) - (p_1 + 2p_2 + p_3) \tag{1}$$

$$G_y = (p_3 + 2p_6 + p_9) - (p_1 + 2p_4 + p_7). \tag{2}$$

The gradient of a 2D image function $I(x, y)$ for co-ordinates (x, y) is defined as a vector and given by,

$$\nabla I = \begin{pmatrix} G_x \\ G_y \end{pmatrix}. \tag{3}$$

The magnitude of this vector is given by, $\text{mag}(\nabla I) = \sqrt{\left[G_x^2 + G_y^2\right]}$. (4)

For faster computation, Eq. (4) can be approximated as,

$$\text{mag}(\nabla I) \approx |G_x| + \left|G_y\right|. \tag{5}$$

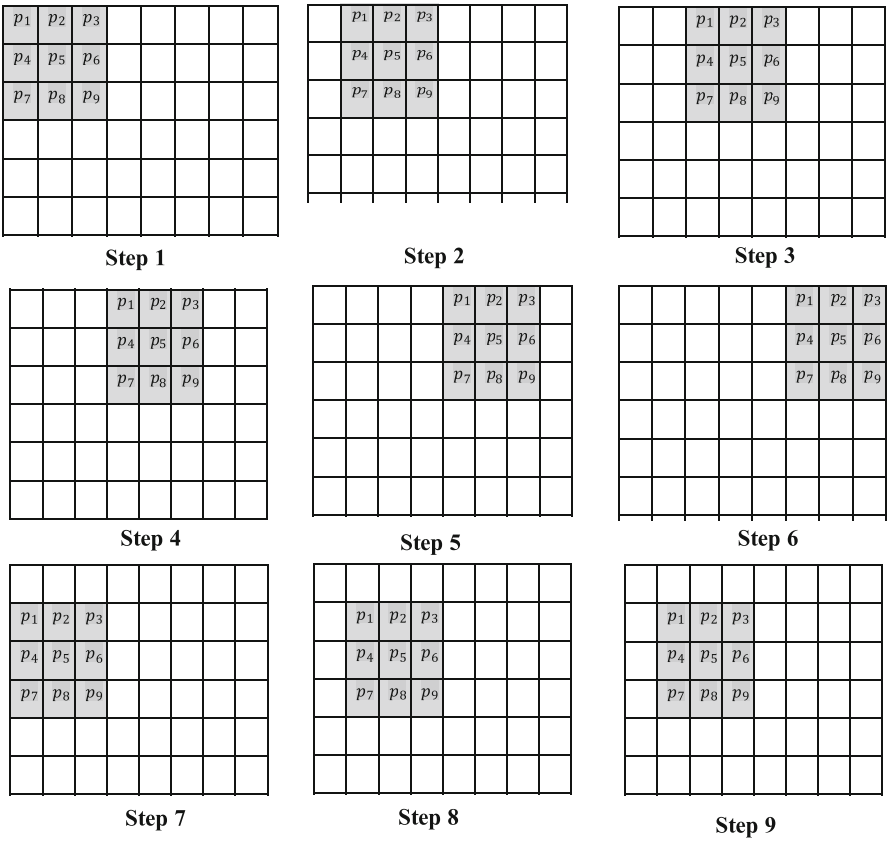


Fig. 3 Convolution technique (step by step) used in edge detection

Here, the window size 3×3 is chosen because the larger the size of the window, the larger number of FIFO generators is required for the accumulation of rows and columns. Also, the number of mathematical calculations increases, and hence the required hardware resources of FPGA are increased.

2.4 Designed System Architecture for Edge Detection

The Xilinx VIVADO integrated software environment (ISE) Design Suite is used for designing the algorithm for edge detection using the Sobel operator. Here, Verilog codes are used to design the algorithm. The MRI grayscale images of size $M \times N$ are taken as input to the system. Then the edge-detected binary image of the same size is obtained at the output. Figure 4 illustrates the system architecture used for edge detection. The 8-bit pixel values of input images are taken as input to the 3×3 pixel generation module. This module contains two FIFO (First In First Out) generators and three shift registers. Sobel edge detection module contains firstly convolution, then

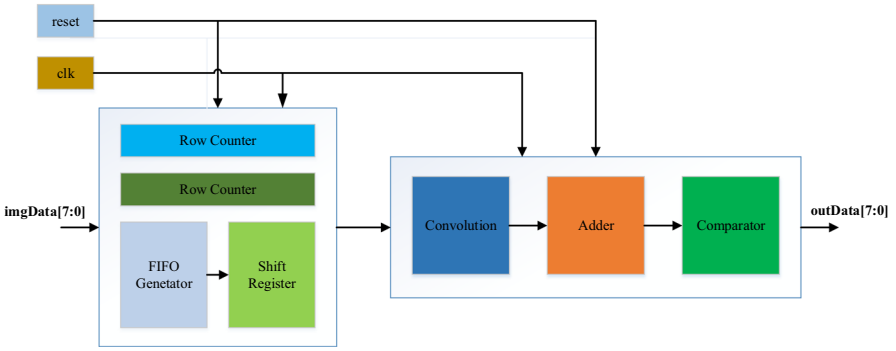


Fig. 4 System architecture for edge detection

addition, and finally threshold comparison units. The modules are synchronized with the input clock so that the pixel values are read and stored at each clock pulses. If the reset button is low, the system goes to a reset position.

Figure 5 demonstrates the memory architecture of the designed system for edge detection. As the used mask size is 3 × 3, we required two FIFO generators as a memory array and three shift registers for the accumulation of input pixel data. The FIFO generator accumulates the number of pixels in one row of the input image. At every positive edge of the clock pulse, the input pixel data is stored, taken out from the FIFO generator, and passed through the shift registers. These 9 pixels are then convolved with the Sobel operator masks and compared with a threshold for edge detection.

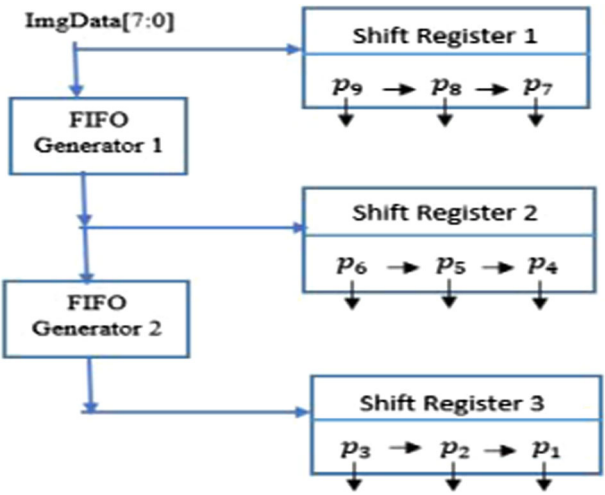


Fig. 5 3 × 3 sized pixel generator

2.5 Pixel Value Calculations

After the edge detection process is completed, the output image is then used as a binary image. The next operation is identifying the 0's and 1's and storing these values into another register. In a binary image, 1's represents the white portion and 0's indicates the black portion. Again, for a normal brain MRI, the detected edges are found out. If there is any abnormality that means any tumor present in the brain, the detected edges (white area) increase. This is used as the baseline to identify the pixel values. Firstly, we look for the white area in the edge-detected image which is considered to be normal by analyzing a data set of normal brain MRIs. After selecting the normal range, we set it as the threshold value for further use. Comparing with the threshold value, if the white portion increases, it indicates that there must be an abnormality (tumor present) in the brain. Hence, in this way, the classification between normal and tumorous brains is performed. The mathematical calculations are described below. Let's consider,

n_n^{total} = The total number of pixels in a single normal brain MRI

n_n^1 = No. of 1's in that normal brain MRI

White_{Th} = Percentage of 1's in total pixels.

For threshold value selection, we have to calculate the white portion percentage up to which level, is considered normal. Thus, the threshold value is selected as follows,

$$\text{White}_{\text{Th}} = \frac{n_n^1}{n_n^{total}} \times 100. \tag{6}$$

Now, this equation is for choosing a threshold for one single image. To make the system more reliable, we used 10 normal brain MRI images so that a more accurate value can be used for threshold selection. The following equation is for choosing the average threshold value;

$$\text{Th} = \frac{1}{N}\sum_{i=1}^{N}(\text{White}_{\text{Th}})_i \tag{7}$$

where N = No. of total normal brain MRIs used to select threshold.

To gain a more accurate value of the threshold, we used a weighted average in which the number of 1's in all the normal images is divided by the total number of pixels in those (in this case, N) images. We define,

$$N^{total} = \sum_{j=1}^{N}\left(n_n^{total}\right)^j \tag{8}$$

$$N^1 = \sum_{k=1}^{N}\left(n_n^1\right)^k \tag{9}$$

where N^{total} = Total no. of pixels in N normal images; N^1 = No. of total 1's in N normal images.

Hence, we find the weighted average of the threshold value as follows;

$$\mathrm{Th_W} = \frac{N^1}{N^{total}} \times 100 = \frac{\sum_{k=1}^{N} \left(n_n^1\right)^k}{\sum_{j=1}^{N} \left(n_n^{total}\right)^j} \times 100. \tag{10}$$

So, Eqs. (7) and (10) are used to find the average and weighted average of the percentages, respectively. As the weighted average is considered to be more accurate for averaging percentages, we choose the latter one for final use. Also, we calculated the differences between these two averages, so that we could choose the maximum rebate value for deciding abnormality or presence of the tumor. We chose the rebate value as follows,

$$R = \frac{\mathrm{Th_W} - \mathrm{Th}}{\mathrm{Th}} \times 100. \tag{11}$$

So, the maximum limit that can be considered as normal case surplus the threshold, is chosen as,

$$T_{\mathrm{Max}} = \mathrm{Th_W} \pm R = \mathrm{Th_W} + R. \tag{12}$$

Again, the same procedure is followed to find the no. of 1's in the test images.
n_t^{total} = The total number of pixels in test brain MRI
n_t^1 = No. of 1's in test brain MRI
$\mathrm{White_{Test}}$ = Percentage of 1's in total pixels.
For threshold value selection, we have to calculate the white portion up to which level, is considered normal. Thus, the threshold value is selected as follows,

$$\mathrm{White_{Test}} = \frac{n_t^1}{n_t^{total}}. \tag{13}$$

Now, T_{Max} is compared to $\mathrm{White_{Test}}$ so that the decision regarding whether there is any tumor present or not can be made as follows;

$$\begin{aligned} &\text{If } \mathrm{White_{Test}} \leq T_{\mathrm{Max}}, \text{ is it considered a normal brain.} \\ &\text{Or, } \mathrm{White_{Test}} > T_{\mathrm{Max}}, \text{ considered as abnormal brain having a tumor.} \end{aligned} \tag{14}$$

2.6 Algorithms and Flowcharts

The overall system includes edge detection and white area detection from the no. of 1's in the detected images. After edge detection using the Sobel operator, the system takes these images as binary input and a quick calculation is performed for detection of the no. of 0's and 1's. Once the no. of 1's is calculated, the percentage of 1's in the total image is calculated as a percentage. The overall calculation is divided into two parts. In the first part, a threshold or reference value is set by using the normal brain

edge-detected images. After that, brain abnormality or presence of a tumor is detected using the no. of 1's in the abnormal brain edge-detected images and comparing this percentage with the threshold value defined previously. Figure 6 shows the flowchart of the first part in which a threshold or reference value is calculated from the pixel values of the normal brain edge-detected images. Firstly, the edge-detected images are taken as input, and depending on the values, the 1's and 0's are segregated. The total number of 1's (which represents the white portion of the image) is calculated and divided by the total number of pixel values in that image. This indicates the white portion of the image which is to be considered normal and used to set the threshold value of the proposed system. The threshold value is stored in registers. This process is continued for all the normal brain edge-detected images so that a final threshold value can be selected for abnormality or tumor detection (Fig. 7).

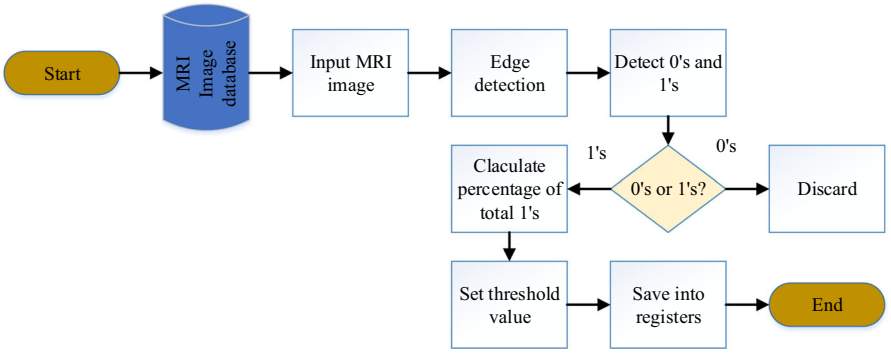


Fig. 6 Flowchart of threshold value calculation from normal brain MRI

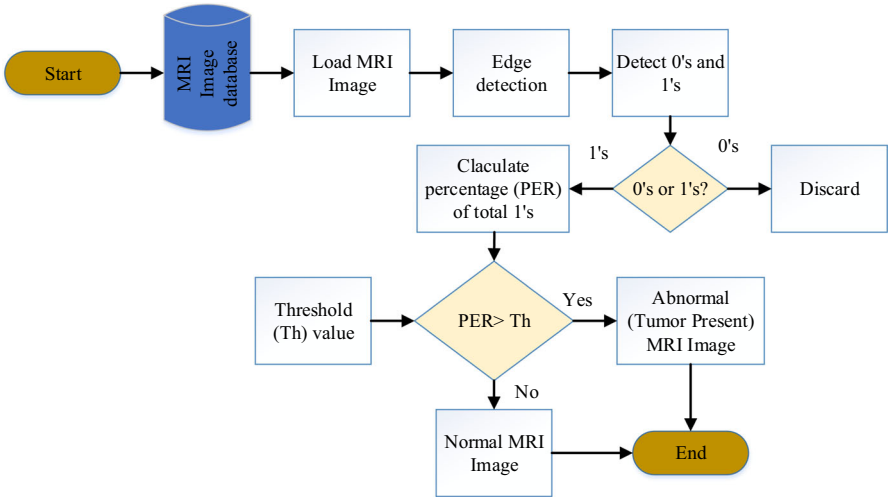


Fig. 7 Flowchart of normal and abnormal (tumor) brain segregation

3 Experimental Results and Discussion

3.1 RTL Diagrams

For MRI image processing and analysis on FPGA, the code was designed in Verilog on Xilinx VIVADO Integrated Software and after completing the simulation and synthesis of the designed system the final RTL diagram of the IP core was analyzed as shown in Fig. 8 internal architecture of IP core. Figures 9 and 10 show the basic RTL diagrams of the system where the images are read by the IP core line by line and pixel by pixel and convolved with the Sobel operator. Firstly, the images are taken into the image control block where the pixel values are read and stored. As the images are in grayscale, the pixel values vary from 0 to 255 and thus 8-bit registers are required to store the pixel values as binary. The input pixel values are passed through the IBUF's (Input Buffers) as shown in Fig. 8. For every *axi_clk* (clock pulses) the data are passed one by one. Then with the help of the FIFO generator and the shift registers the 3×3 block pixels are generated. This is known as *o_pixel_data_valid* from the output of *imageControl* block and inserted into the input of *conv* block as *i_pixel_data_valid.*

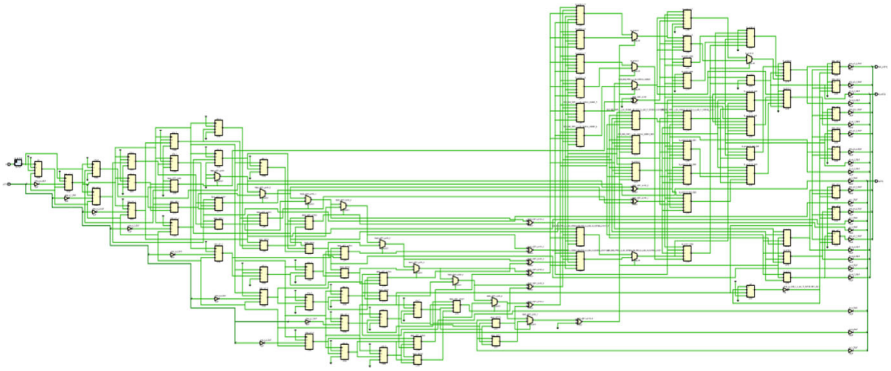


Fig. 8 Internal architecture of the IP core

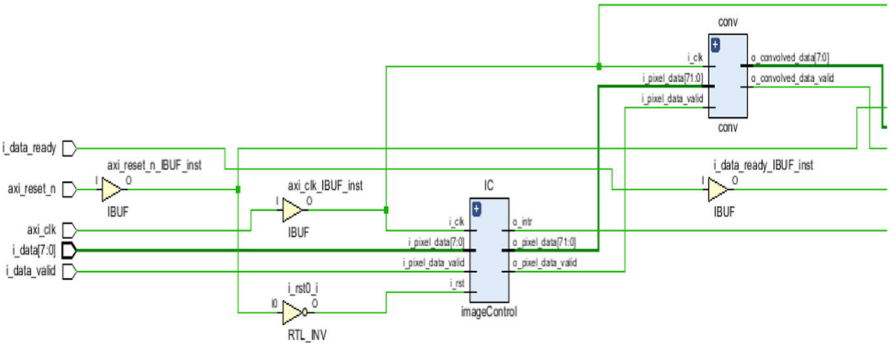


Fig. 9 RTL diagram (part-1) for edge detection of MRI images

The next block is the *conv* (convolution block). In this block, the 0° and 90° Kernels are defined and taken into action. Both the G_x and Gy gradients are calculated from the 9 pixels originated from the 3 × 3 blocks generated in the *imageControl* block. When the convolution result of pixel values is ready, the result is passed through the buffers and known as *o_convolved_data.* The next portion of the RTL block is shown in Fig. 10.

From Fig. 10, it is shown that after convolution the data is passed through the *outputBuffer* block. In this block, there is an adder to perform Eq. (5) to add the gradients for magnitude calculation. After that, the calculated magnitude is then compared to a threshold value for final edge detection. When the edge-detected image is ready, it is a binary image that is used further for pixel value (either 0 or 1) calculation.

By using FOR loops, we separated the no. of 1's in the edge-detected binary image which indicates the white portion in the total image. For every clock pulse, the system takes the pixel values and passes through a loop-up-table (as shown in Fig. 11) for segregating the no. of 1's in the total binary image. After taking the pixel values, the system automatically processes the values in parallel and calculates the white portion of the image so that the abnormal (tumorous) and normal brains can be segregated. The Advanced HDL Synthesis Report (Macro statistics) is shown in Table 1. Also, the power analyzer report is shown in Table 2.

From Table 1, it is noticeable that the proposed method for MRI image processing in Verilog requires very small memory which is a maximum of 30.906 MB. Again four 8-bit registers, two FIFO generators, and three shift register to work as the memory element and store data. The number of total IO Buffers required is 18 and we also

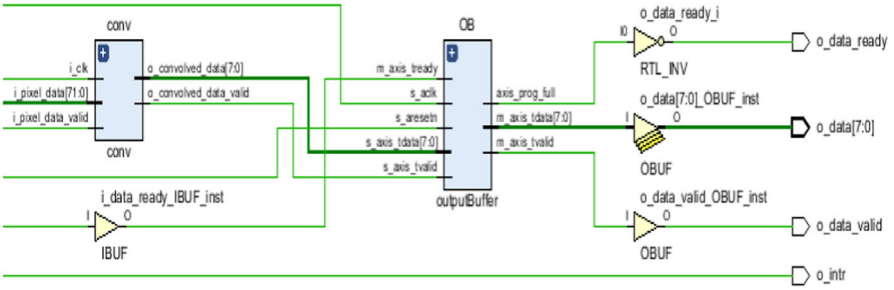


Fig. 10 RTL diagram (part-2) for edge detection of MRI images

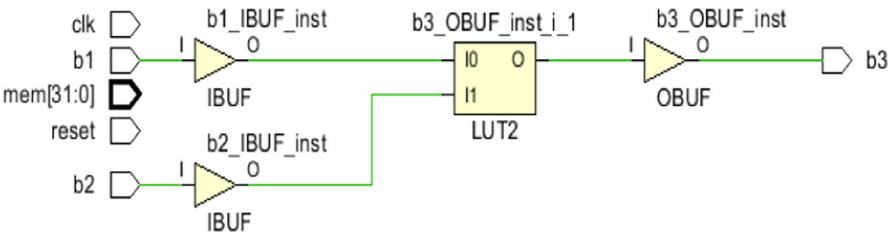


Fig. 11 IBUFs, LUTs, and OBUFs in the system to calculate no. of 1's

Table 1 Advanced HDL synthesis report of brain tumor detection on FPGA

Registers (8 bit)	4
FIFO generator	2
Shift registers	3
Clock buffers	1
IO buffers	18
IBUF	10
OBUF	8
Adder	1
Comparator	4
Maximum memory usage	30.906 MB

Table 2 Power analyzer report of brain tumor detection on FPGA

Core junction temp	25.4 °C
Voltage supply (v_{cc})	1.7 V
Total current	20 mA
Total power	82 mW

need one adder for gradient addition and one comparator for comparison with the threshold. Also, from Table 2, it is clear that the total voltage and current required to run the FPGA board are very small which are 1.7 V and 20 mA, respectively. The total power consumption is 82 mW and core junction temperature is 25.4 °C.

3.2 Timing Diagrams

The timing diagram for the edge detection process is shown in Fig. 12. The *file[31:0]* and *file1[31:0]* are the input and final output files, respectively. The input grayscale

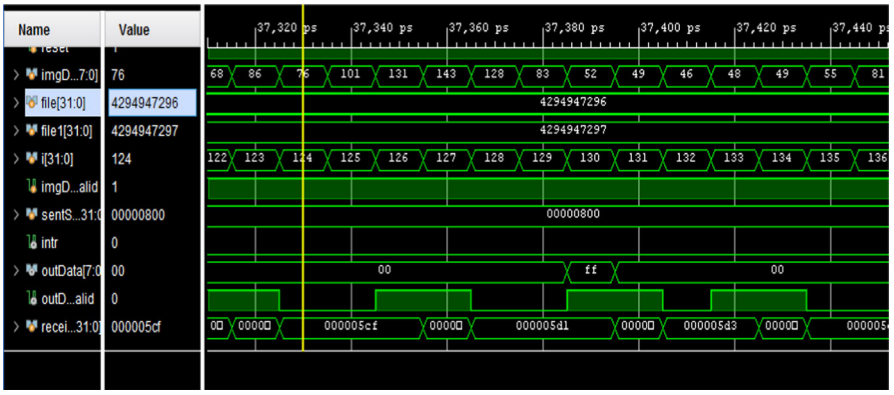


Fig. 12 Timing diagram for edge detection from brain MRI image

images are loaded into a register as *imgData[7:0]*which is shown at the top of Fig. 12. The values are 68, 86, 76, 101, 131, 143, 128, 83, 52, 49……. are the gray levels here which vary from 0 to 255. The *outData_valid* is a single bit parameter that indicates the 3×3 sized pixel values have generated by using three-line buffers. Before getting the final edge-detected image, the gradients (G_x and Gy) are calculated internally and added using an adder. Finally, by using a comparator the comparison with a threshold is performed and then the final edge-detected images are obtained. The edge-detected output image pixel values are stored in *outData[7:0].* As these images are in binary form these output pixel values are either 0 or 255. Here, 'ff' indicates the value 255 as it means 8-bit 1's.

When the edge-detected images are ready, the pixel value calculation part is done. The equations described are then used for the detection of the white area by calculating the threshold first. At first, the ten normal images are used to determine the threshold value. Equations (6)–(12) are used for threshold value calculation. This threshold is used for further normal and abnormal (tumorous) brain segregation. The images were loaded one by one into the system so that the pixel values could be analyzed more specifically. All the pixel values were taken into the register memory. Every pixel value was taken by single clock pulses. This required 32-bit registers to save the pixel data. Figures 12, 13, 14, 15, and 16 show the timing diagrams of 5 normal brain MRI images. The *i[31:0]* stores the total pixel values while the *b[31:0]* stores the no. of detected 1's (Fig. 17).

So, the total no. of pixels is stored in *i[31:0]* and the detected 1's are stored in *b[31:0].* In this way, the other normal MRI image values have been stored and calculated. Finally, a threshold value was determined from their weighted average and

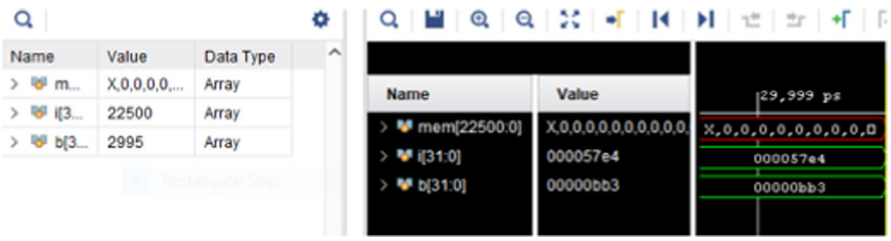


Fig. 13 Timing diagram for detecting no. of 1's in Normal-1

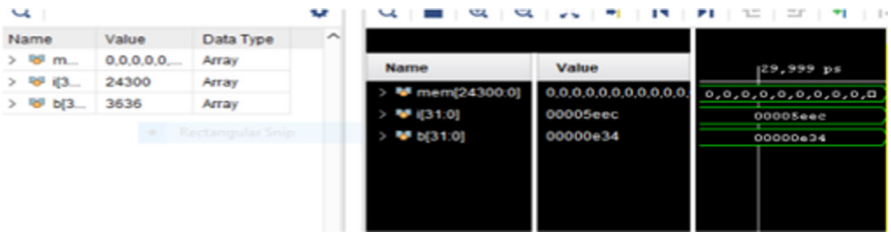


Fig. 14 Timing diagram for detecting no. of 1's in Normal-2

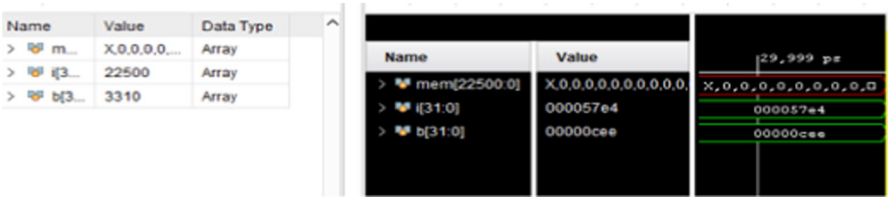


Fig. 15 Timing diagram for detecting no. of 1's in Normal-3

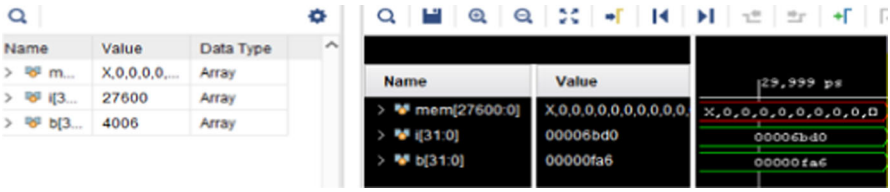


Fig. 16 Timing diagram for detecting no. of 1's in Normal-4

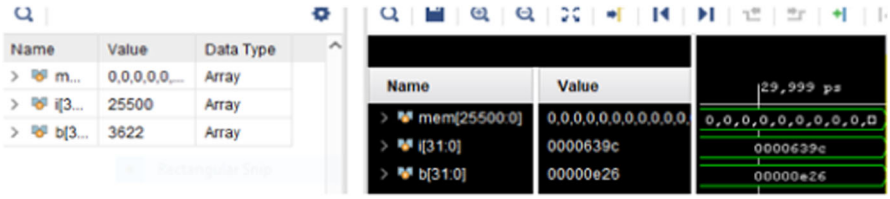


Fig. 17 Timing diagram for detecting no. of 1's in Normal-5

was used as a reference value for further comparison purposes. Next, ten more brain images were used in the same way. First, the edge detection, then the pixel value calculation were performed. When the no. of 1's in these images were obtained, its percentage was calculated and finally compared with the previously selected threshold value. Figures 18, 19, 20, 21, and 22 show the timing diagrams of 5 test images which were edge-detected first and then loaded for detecting no. of 1's. From the timing diagram, it is noticed that the total pixel value was stored in *i[31:0]* and no. of 1's were detected in *b[31:0]* as before.

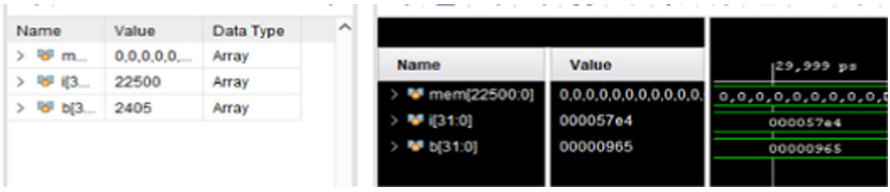


Fig. 18 Timing diagram for detecting no. of 1's in Test-1

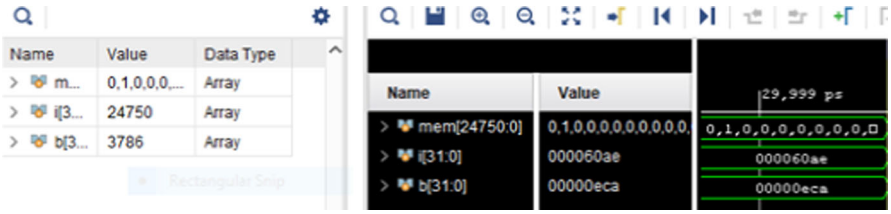


Fig. 19 Timing diagram for detecting no. of 1's in Test-2

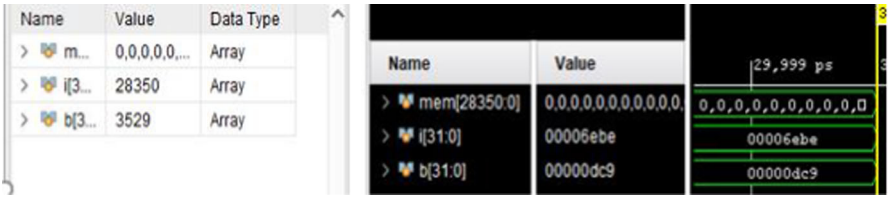


Fig. 20 Timing diagram for detecting no. of 1's in Test-3

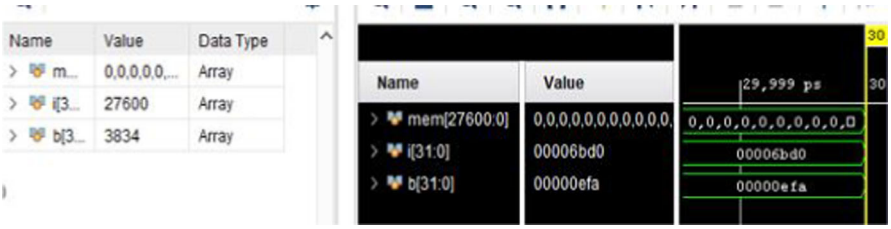


Fig. 21 Timing diagram for detecting no. of 1's in Test-4

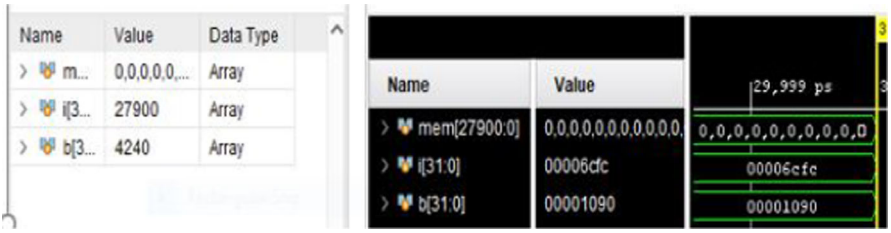


Fig. 22 Timing diagram for detecting no. of 1's in Test-5

Thus, after the calculations and comparison with the threshold value, the normal and abnormal (tumorous) brains were segregated.

3.3 Result Analysis

We have used a total of 20 images for testing the system. Among them, some were normal brain MRIs (10 images) so that we could find the threshold value to use as a reference in abnormal (tumorous) and normal image segregation. Once the threshold value was found, then the segregation of normal and abnormal (tumorous) brain images was done. By comparing the white pixel percentage with the threshold value, the decision was made for identifying normal and abnormal images. Figure 23 shows the original normal brain MRI images that were used to calculate the threshold. The main goal was to find out the normal range of the white portion in the normal brain MRIs and use it as a threshold. After loading the images, the edge detection using the Sobel operator was performed in Verilog. Figure 24 shows the edge-detected images of those five MRI images that are in Fig. 23. When the edge-detected images were ready, the pixel value calculation and the white area identification by calculating the no. of 1's in the register is done. From the detected white area in the total 10 images, the final threshold value is calculated.

After obtaining the edge-detected binary images the calculation of finding no. of 1's and 0's to calculate the desired threshold was performed. Equations (8), (9), and (10) calculate the threshold values. The basic task was to calculate the total number of 1's in the binary images which ultimately led to the detection of the white area in those images. Table 3 shows the total and detected where area pixels and the corresponding thresholds for the used 10 images. In this table, we have used the normal average using Eqs. (6) and (7).

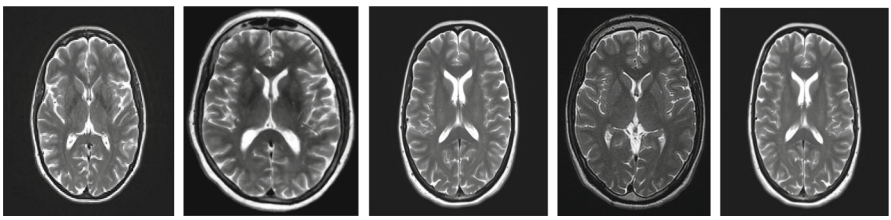


Fig. 23 Normal brain MRI's to detect the threshold value

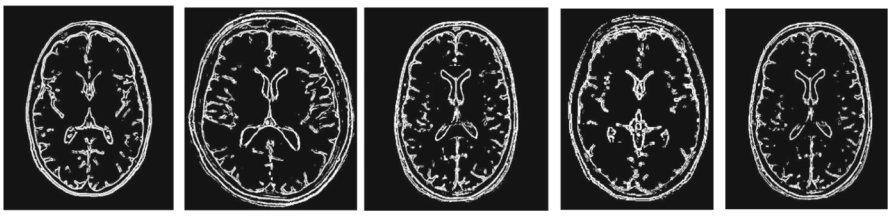


Fig. 24 Edge-detected images to detect the threshold value

Table 3 Total pixels and detected 1's in the normal brain images

Normal images	No. of total pixels ($n_n{}^{\text{total}}$)	No. of 1's ($n_n{}^{1}$)	Percentage (%),White_{Th}
Normal-1	150*150 = 22,500	2995	13.311
Normal-2	150*162 = 24,300	3636	14.963
Normal-3	150*150 = 22,500	3310	14.711
Normal-4	150*184 = 27,600	4006	14.514
Normal-5	150*170 = 25,500	3622	14.203
Normal-6	150*172 = 25,800	3541	13.725
Normal-7	150*162 = 24,300	3598	14.806
Normal-8	150*160 = 24,000	3310	13.792
Normal-9	150*172 = 25,800	3498	13.558
Normal-10	150*180 = 27,000	3890	14.203

From the table data, we calculated the normal average of these percentages, and the calculation is given below;

$$\text{Th} = \frac{1}{10}\sum_{i=1}^{10}(\text{White}_{\text{Th}})_i = \frac{1}{10}(13.311\% + 14.963\% + \cdots + 14.203\%) = 14.179\%.$$

Next, using Eqs. (8), (9), and (10) we get the weighted average as follows;

$$N^{total} = \sum_{j=1}^{10}\left(n_n^{total}\right)^j = (22500 + 24300 + 22500 + \cdots + 27000) = 2, 49, 300$$

$$N^1 = \sum_{k=1}^{10}\left(n_n^1\right)^k = (2995 + 3636 + 3310 + \cdots + 3890) = 35406$$

where N^{total} = Total no. of pixels in N normal images.

And N^1 = No. of total 1's in N normal images.

Hence, we find the weighted average of the threshold value as follows;

$$\text{Th}_{\text{W}} = \frac{N^1}{N^{\text{total}}} \times 100 = \frac{35406}{2, 49, 300} \times 100 = 14.202\%.$$

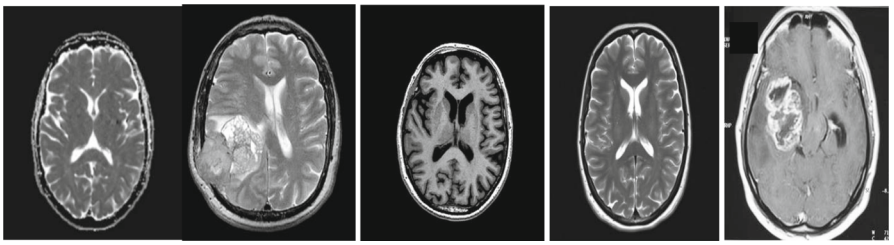


Fig. 25 Brain MRI's for normal and abnormal (tumorous) brain segregation

Then, using Eqs. (11) and (12) we find the rebate value and the maximum threshold limit as follows;

$$R = \frac{\text{Th}_\text{W} - \text{Th}}{\text{Th}} \times 100 = \frac{14.202 - 14.179}{14.179} \times 100 = 0.1622\%.$$

$$T_\text{Max} = \text{Th}_\text{W} \pm R = \text{Th}_\text{W} + R = (14.202 + 0.1622)\% = 14.3642\%.$$

This T_Max is the desired threshold value for segregating further images.

Now, the same procedure is followed to segregate the normal and tumorous brain images. 10 images were taken as test images. Firstly, their edges were detected by using the Sobel operator as before, and then the no. of 1's were found to calculate the white area. Figure 25 shows the first five images that were used to test the system. Figure 25 shows the corresponding edge-detected images.

Now, Eqs. (13) and (14) are used for segregating normal and abnormal (tumorous) brain images. The same procedure is followed to find the no. of 1's in the test images.

n_t^{total} = The total number of pixels in test brain MRI

n_t^1 = No. of 1's in test brain MRI

$\text{White}_\text{Test} = \frac{n_t^1}{n_t^{total}}$ = Percentage of 1's in total pixels.

Now, T_Max is compared to White_Test, so that the decision regarding segregation between normal and abnormal brains can be made as follows:

If $\text{White}_\text{Test} \leq T_\text{Max}$, is it considered a normal brain.

Or, $\text{White}_\text{Test} > T_\text{Max}$, considered as an abnormal (tumorous) brain.

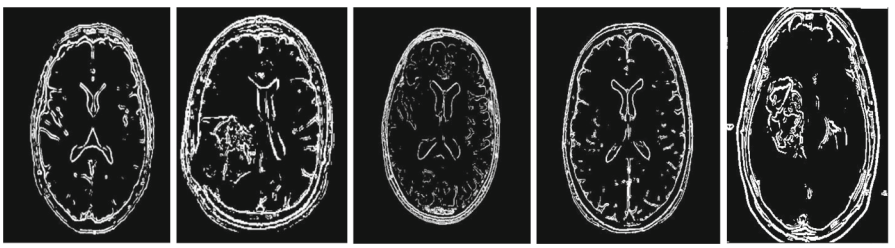
Table 4 shows the result for ten images that are used for testing the performance of the proposed system. When the calculation of White_Test is done, then it is compared to the previously selected threshold T_Max. After the comparison is done, the calculated result is then matched with the actual result. Table 4 shows that 9 among ten images were successfully segregated as a normal or tumorous brain (Fig. 26).

3.4 Comparison with Relevant Works

Table 5 shows the comparison between similar works for Image processing on FPGAs. This table shows that the proposed designed IP core on FPGA for brain MRI image

Table 4 Total pixels and detected pixels for normal and abnormal brain segregation

Name	Total pixels, $n_t{}^{total}$	Detected 1's, $n_t{}^{1}$	Percentage, $\text{White}_{\text{Test}}$ (%)	Calculated result	Actual result	Tumor detected?
Test-1	150*150 = 22,500	2405	10.689	Normal	Normal	YES
Test-2	150*165 = 24,750	3786	15.297	Abnormal	Abnormal	YES
Test-3	150*189 = 28,350	3529	12.448	Normal	Normal	YES
Test-4	150*184 = 27,600	3834	13.891	Normal	Normal	YES
Test-5	150*186 = 27,900	4240	15.197	Abnormal	Abnormal	YES
Test-6	150*170 = 25,500	3808	14.933	Abnormal	Abnormal	YES
Test-7	150*168 = 25,200	3902	15.484	Abnormal	Abnormal	YES
Test-8	150*172 = 25,800	3712	14.387	Abnormal	Normal	NO
Test-9	150*162 = 24,300	3834	13.185	Normal	Normal	YES
Test-10	150*184 = 27,600	4047	14.663	Abnormal	Normal	YES



a For Test 1 **b** For Test 2 **c** For Test 3 **d** For Test 4 **e** For Test 5

Fig. 26 Edge-detected images for normal and tumorous brain segregation

processing and segregating normal and abnormal (tumorous) images has outperformed the relevant researches where the processing time is incredibly fast which is 6.49 μs per image. The power consumption is only 82 mW and the mask size is 3*3, which is better than others. Also, we required only 32 memory elements where others required higher ones.

Table 5 Comparison between similar works for brain MRI image processing on FPGA

Publication	Research motive	Processing on FPGA	Remarks
[16]	Real-time Image Segmentation on FPGA	Clock Frequency: 66 MHz Processing time: 2 s	Year: 2006 FPGA: Virtex-2
[21]	Medical Computed Tomography (CT) image segmentation on FPGA	Processing time: 441 s Power consumption: 4.8 W	Year: 2014 FPGA: Virtex-7
[24]	Real-time Medical Image Segmentation on FPGA	Power consumption: 5mW Memory elements: 198	Year: 2015 FPGA: Virtex-6
[7]	Edge detection algorithm on FPGA for medical image	Clock Frequency: 110 MHz Processing time: 1 s for 30 frames Mask size: 2*1 and 3*1	Year: 2016 FPGA: Spartan-6
This work	Brain MRI image segmentation and tumor detection	Clock Frequency: 125 MHz Processing time: 6.49 μs Power consumption: 82 mW Mask size: 3*3 Memory elements: 32	Year: 2021 FPGA: Only software simulation

4 Conclusions

This work investigates the performance of FPGAs in biomedical image processing. The IP core can load MRI image pixel data and process the image as required. The main intention was to detect brain abnormality from MRI images. Both the normal and abnormal brain MRIs were used and a threshold value was chosen from the normal MRI's to be used as a reference value. Medical MRI images were chosen for the image processing purpose for the medical diagnostic system. Brain MRI images were chosen for segregating normal and tumorous brains on FPGA. The basic operation was Edge detection by using the Sobel operator and then identifying the white area in those images. Finally, a comparison with a predefined threshold was done and normal and tumorous brain images were segregated and the presence of tumor was detected. The results can be shown in 6.49 μs which is very fast. The main advantage of the proposed system is the reconfigurability and portability of the entire IP core. If there is a user demand to change logic operations the IP core can be reprogrammed at any time. Also, it requires only 5–10 s to configure the FPGA board. The average power consumption of the IP core is around 82 mW, the maximum memory space is 30.906 MB for brain MRI image processing. Hence, the designed IP core is of very low power as well as low memory consumption.

5 Future Works

This work can be enhanced and cope with a larger dataset to apply to the patient monitoring systems. It will be interesting to see how this proposed system is used as a simple and fast way to diagnose a patient's monitoring system. Also, this designed

system can be used to diagnose Medical MRI in finding other types of brain abnormalities such as brain hemorrhage, when this model would be designed for a larger amount of MRI dataset and machine learning algorithm will be applied on it.

Acknowledgements The authors wish to thank Department of Electronics and Communication Engineering, Khulna University of Engineering and Technology, Bangladesh staffs and all professors of this department for supporting equipment and other facilities. This work was not supported in part by a grant.

Funding No third party or organization provided funds for this research so far.

Data Availability Brain MRI images database used from GitHub [https://github.com/topics/brain-mri].

Declarations

Conflict of interest The authors declare no conflict of interest regarding this research work.

References

1. A. Ahmad, A. Amira, H. Rabah, Y. Berviller, FPGA-based architectures of finite radon transform for medical image de-noising, in *2010 IEEE Asia Pacific Conference on Circuits and Systems* (2010), pp. 20–23. https://doi.org/10.1109/APCCAS.2010.5774903
2. A. Alkamil, D.G. Perera, Efficient FPGA-based reconfigurable accelerators for SIMON cryptographic algorithm on embedded platforms, in *2019 International Conference on ReConFigurable Computing and FPGAs (ReConFig)* (2019), pp. 1–8. https://doi.org/10.1109/ReConFig48160.2019.8994803
3. H.M. Amjad, J. Niu, K. Hu, N. Akram, L. Besnard, Verilog code generation scheme from signal language, in *2019 16th International Bhurban Conference on Applied Sciences and Technology (IBCAST)* (2019), pp. 457–462. https://doi.org/10.1109/IBCAST.2019.8667266
4. H. Bessalah, F. Alim-Ferhat, H. Salhi, S. Seddiki, M. Issad, O. Kerdjidj, On line wavelets transform on a xilinx FPGA circuit to medical images compression, in *2009 International Conference on Digital Image Processing* (2009), pp. 8–12. https://doi.org/10.1109/ICDIP.2009.89
5. M.F. Binothman, N. Abdullah, N.A.B.A. Rusli, An overview of MRI brain classification using FPGA implementation, in *2010 IEEE Symposium on Industrial Electronics and Applications (ISIEA)* (2010), pp. 623–628. https://doi.org/10.1109/ISIEA.2010.5679389
6. B. Bipin, J.J. Nair, Image convolution optimization using sparse matrix vector multiplication technique, in *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)* (2016), pp. 1453–1457. https://doi.org/10.1109/ICACCI.2016.7732252
7. I. Bouganssa, M. Sbihi, M. Zaim, Implementation on a FPGA of edge detection algorithm in medical image and tumors characterization, in *2016 5th International Conference on Multimedia Computing and Systems (ICMCS)* (2016), pp. 59–64. https://doi.org/10.1109/ICMCS.2016
8. S.C. Chan, H.O. Ngai, K.L. Ho, A programmable image processing system using FPGA, in *Proceedings of IEEE International Symposium on Circuits and Systems—ISCAS'94*, vol. 2 (1994), pp. 125–1282. https://doi.org/10.1109/ISCAS.1994.408921
9. B. Chandra, M. Sharma, Segmentation's based feature extraction of MRI images using wavelet and implementation on FPGA, in *2016 International Conference on Automatic Control and Dynamic Optimization Techniques (ICACDOT)* (2016), pp. 135–141. https://doi.org/10.1109/ICACDOT.2016.7877566
10. X. Chen, X. Wang, Y. Liu, Z. Liu, J. An, FPGA verification of radar signal processing based on SoC, in *2019 IEEE International Conference on Signal, Information and Data Processing (ICSIDP)* (2019), pp. 1–4. https://doi.org/10.1109/ICSIDP47821.2019.9172898
11. I. Chiuchisan, Implementation of medical image processing algorithm on reconfigurable hardware, in *2013 E-Health and Bioengineering Conference (EHB)* (2013), pp. 1–4. https://doi.org/10.1109/EHB.2013.6707298

12. I. Chiuchisan, A new FPGA-based real-time configurable system for medical image processing, in *2013 E-Health and Bioengineering Conference (EHB)* (2013), pp. 1–4. https://doi.org/10.1109/EHB.2013.6707301
13. I. Chiuchisan, An approach to the verilog-based system for medical image enhancement. *2015 E-Health and Bioengineering Conference (EHB)* (2015), pp 1–4. https://doi.org/10.1109/EHB.2015.7391461
14. S. Cinar, M.N. Kurnaz, Segmentation of medical images by using kNN classifier on field programmable logic array (FPGA), in *National Conference on Electrical, Electronics and Computer Engineering* (2010), pp. 516–520
15. Y. Cui, C. Wang, Y. Chen, Z. Wei, M. Chen, W. Liu, Dynamic reconfigurable pufs based on FPGA, in *2019 IEEE International Workshop on Signal Processing Systems (SiPS)* (2019), pp. 79–84. https://doi.org/10.1109/SiPS47522.2019.902044
16. P. Dillinger, J.F. Vogelbruch, J. Leinen, S. Suslov, R. Patzak, H. Winkler, K. Schwan, FPGA based real-time image segmentation for medical systems and data processing, in *14th IEEE-NPSS Real Time Conference, 2005* (2005), p. 5. https://doi.org/10.1109/RTC.2005.1547401
17. E. Giordano, F.D. Marco, G. Pravadelli, A model-based design flow for dynamic partial reconfigurable FPGAs, in *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)* (2019), pp. 3099–3103. https://doi.org/10.1109/SMC.2019.8914616
18. M.C. Herbordt, T. VanCourt, Y. Gu, B. Sukhwani, A. Conti, J. Model, D. DiSabello, Achieving high performance with FPGA-based computing. Computer **40**(3), 50–57 (2007). https://doi.org/10.1109/MC.2007.79
19. IEEE draft standard for VHDL language reference manual. IEEE P1076/D13, July 2019, pp. 1–796 (2019)
20. K. Khandagle, S.S. Rathod, Implementation of MRI gradient generation system and controller on field programmable gate array (FPGA), in *2018 International Conference on Communication Information and Computing Technology (ICCICT)* (2018), pp. 1–4. https://doi.org/10.1109/ICCICT.2018.8325876
21. P. Li, T. Page, G. Luo, W. Zhang, P. Wang, P. Zhang, P. Maass, M. Jiang, J. Cong, FPGA acceleration for simultaneous medical image reconstruction and segmentation, in *2014 IEEE 22nd Annual International Symposium on Field-Programmable Custom Computing Machines* (2014), pp. 172–172. https://doi.org/10.1109/FCCM.2014.54
22. H. Lin, C. Hsueh, COS: a configurable os for embedded SoC systems, in *12th IEEE International Conference on Embedded and RealTime Computing Systems and Applications (RTCSA'06)* (2006), pp. 242–245. https://doi.org/10.1109/RTCSA.2006.24
23. S. Mahmood, J. Rettkowski, A. Shallufa, M. H¨ubner, D. Göhringer, IP core identification in FPGA configuration files using machine learning techniques, in *2019 IEEE 9th International Conference on Consumer Electronics (ICCE-Berlin)* (2019), pp. 103–108. https://doi.org/10.1109/ICCE-Berlin47944.2019.8966236
24. U.R. Nelakuditi, T.S.R. Bharadwaja, N. Bhagiradh, Novel VLSI architecture for real time medical image segmentation, in *2015 2nd International Conference on Electronics and Communication Systems (ICECS)* (2015), pp. 1084–1088. https://doi.org/10.1109/ECS.2015.7124748
25. X. Pang, D. Yu, J. Li, Y. Guo, Design and application of IP core in SoC technology, in *2010 Third International Symposium on Information Science and Engineering* (2010), pp. 71–74. https://doi.org/10.1109/ISISE.2010.94
26. R. Ranjbarzadeh, A.B. Kasgari, S.J. Ghoushchi, S. Anari, M. Naseri, M. Bendechache, Brain tumor segmentation based on deep learning and an attention mechanism using MRI multi-modalities brain images. Sci. Rep. **11**(1), 10930 (2021). https://doi.org/10.1038/s41598-021-90428-8
27. R. Rekha, K.P. Menon, FPGA implementation of exponential function using cordic IP core for extended input range, in *2018 3rd IEEE International Conference on Recent Trends in Electronics, Information Communication Technology (RTEICT)* (2018), pp. 597–600. https://doi.org/10.1109/RTEICT42901.2018.9012611
28. K. Smelyakov, M. Shupyliuk, V. Martovytskyi, D. Tovchyrechko, O. Pono marenko, Efficiency of image convolution, in *2019 IEEE 8th International Conference on Advanced Optoelectronics and Lasers (CAOL)* (2019), pp. 578–583. https://doi.org/10.1109/CAOL46282.2019.9019450
29. H.M.W. Thomas, S.C. Prasanna Kumar, Detection of a brain tumor using segmentation and morphological operatorsfrom MRI scan with FPGA, in *2015 International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT)* (2015), pp. 728–731. https://doi.org/10.1109/ICATCCT.2015.7456979

30. C. Xuesen, H. Liguo, D. Jinbo, Design and implementation of FPGA base diagnosis of medical image data acquisition equipment, in *IEEE 2011 10th International Conference on Electronic Measurement Instruments*, vol. 3 (2011), pp. 51–55. https://doi.org/10.1109/ICEMI.2011.6037853
31. S.S. Yadav, V. Mittal, FPGA implementation of segmented feature fusion in MRI images using wavelet. *2019 International Conference on Communication and Electronics Systems ICCES)* (2019), pp. 1946–1951. https://doi.org/10.1109/ICCES45898.2019.9002032