FPGA-based Real Time Embedded Hough Transform Architecture for circles detection

CHUQUIMIA Orlando[1], PINNA Andrea[1], MARSALA Christophe[1], DRAY Xavier[2], BERTRAND Granado[1]

Abstract— **Hough Transform is a widely used shape-based algorithm for object detection and localization [6], this technique can be generalized to parametric curves as circles. For a real time execution and embedded integration, several optimizations are necessary due to the large memory and computational requirements. This paper presents an efficient real-time pipelined architecture with a FPGA implementation of our Hough Transform for multi-circles detection. The computation of center candidates was improved. A three stages pipeline architecture was designed in order to reduce the processing latency and cadence. The architecture has been integrated into a Xilinx Zynq-7000 XC7Z020 containing a FPGA Artix-7. The global system uses 78.5 BRAMs, 153 DSP slices, 21638 LUTs. Our global system can support a maximum clock frequency of 128.89 MHz. We validate our architecture using a 125MHz clock frequency and we obtain a latency of 33.214 ms and an interval between two images of 16,607 ms for a 1920x1080 pixels image. According to our results, our architecture offer a throughput more than 4 times better than the faster state of the art architecture.**

I. INTRODUCTION

Most of 95%[4] of colorectal cancers begin as a growth on the inner lining of the colon or rectum called as polyp.

To reduce the incidence of Colorectal cancer, authors in [14], [17] proposed a new paradigm of Wireless Capsule Endoscopy [7] that can automaticaly recognize polyp in situ.

They have proposed a specific processing chain embedded in a System on Chip integrated inside a capsule.

In this chain they use the Hough transform to detect circles in HD images as Regions candidates to contain a polyp. It is a widely used technique, for object localization since 1962 [6], that can be generalized to parametric curves as circles [2]. Once regions are selected they use a learning algorithm to decide if there is a polyp in the region.

In this article, we have studied only the Hough transform part to determine if it can be embedded in real time, this can be a first step in order to determine if it can be embedded in the next generation of capsule that will embed a HD images. We focused on it because we have measured its execution on an embedded processor, an ARM Cortex A9, using OpenCV library and running at 667 MHz and see that it takes 998.260 ms for an image of size 1920x1080. This execution time is too high, 25 times higher than the expected goal that is to proccess images with the same video quality of an endoscope, that acquire a 1920x1080 pixel image every 40 ms. This result excludes an optimized software embedded implementation and obliges us to a custom digital hardware implementation in FPGA.

[1] LIP6, CNRS UMR 7606, Sorbonne Université, Paris, France.
[2]APHP, Hôpital Saint-Antoine, Sorbonne Université, Paris, France.

II. STATE OF THE ART

A state of the art has been realized to analyse the implementation of the Hough transform in a FPGA considering timing contraints. Here, we present the highlights found, a survey of Hough transform methods can be read in [2] and [13]. We can notice some facts:

1) all the FPGA implementations [1], [3], [19], [16], [8], [11] use embedded internal memories or BRAMs (Block RAMs),
2) all the works consider only the acceleration of the voting process of the Hough transform. In this process the goal is to accumulate intersection points in the Hough parameter space, corresponding to the number of possible circles. Then a vote is done to find the local maximum that are considered as real circle. This process is memory and time consuming, two different approaches emerged, the first approach uses the original Hough transform algorithm were the parametric equations of a circle are used to find the center and the radius of the circles [1], [3], [16], [8], [11] and the second approach uses a modified version of Hough transform called One dimensional Hough transform algorithm [19].

We describe below the works related to both approaches.

A. Original Hough transform Implementations

For the original Hough transform implementation, in [3], authors demonstrate that using an external memory to store the voting procedure is limited due to the data transfer bandwidth. In this implementation an efficient internal memory structure is considered for the voting process, where the size of the Hough space is reduced. Computations are distributed in mathematical units, each unit has access to its own memory module with a double buffer technique to avoid external memory, this allows multiple parallel read/write operations at the same time.

In [1], authors use a CORDIC algorithm to implement the Hough transform. Specificity of this work is that it detects only one circle owning of the target application, an iris detection. The same approach is used in [11], where a FPGA-based hardware accelerator for iris localization is introduced.

In [16], authors adopt the *scanline-based ball detection* algorithm for the edge detection stage and edge-flag algorithm for the voting process.

In [8], authors propose a Hough transform algorithm combined with a graph clustering algorithm for FPGA-based multi-circle detection.

B. 1D Hough transform algorithm Implementation

Goneid et al. introduce a modified version of Hough transform dedicated to detect multi-circles [5] in 1997. This method, called 1D Hough Transform multi-circles detection can successfully extract non-overlapping circles and ellipses in binary images, even in the presence of random noise. This method is easy to implement since each of the object's parameters is accumulated in its own one-dimensional parameter space. Zhou and al. [19] implement it on a FPGA.

This modified version of Hough Transform algorithm can be represented by the simplified pseudocode in Algorithm 1 and can be summarized as follow:

1) From a contour image, first, to create a histogram of the middle of the segments constituted by the points of contours. At first, for the horizontal segments, we create histogram V_x (steps [1-7] of algorithm 1 and in a second time for the vertical segments, we create histogram V_y (Steps [8-14] of algorithm 1. We obtain the center candidates (i, j) from combination of local maximums of line and column histograms.
2) We build a 3D histogram V_r of the distances between all the point of contour and the center candidates. Steps [15-24] of algorithm 1, the more voted distance r become the radius of the center candidate (i, j);
3) If the value $V_r(i, j, r)$ is greater than $f * 4\sqrt{2}r$ then $||(x, y) - (i, j)|| = r$ becomes a circle. Steps [20-22] of algorithm 1. f can be adapted as the sensitivity threshold to detect circles.

In [19], authors propose an architecture based on FPGA implementation of algorithm 1. They choose a 9-bit integer words for the data in histogram V_x and histogram V_y and a 17-bit integer words for the histogram V_r. They use multiple BRAMs to implement these histograms by 100 voting modules in parallel. The number of x-coordinate or y-coordinates of center candidates is set to 10, therefore, 100 center candidates are constructed. Finally the radius is coded with 13-bit integer.

C. Analysis of the state of the art

The implementations performance are shown in table I.

As we can see in table I, there is no FPGA work using larger image sizes as 1920x1080. In table I we have extrapolated for each referenced work, using a simple size factor, the latency for a 1920x1080 image size. We can see that none of these state of the art method reach a latency less than 40 ms for this image size. Then it is necessary to design a new digital hardware architecture to accelerate the Hough transform computation.

Polyps can be show as protrusions and detected using the local curvature of the edge-image searching circular or elliptical shapes [9], [12]. Based on analyses of polyps images from colon examination [15], it was observed that polyps do not always have a regular circular or elliptical shape, it depend of the noisy level, quality and resolution of the image. In addition, in [19] the key technique for accelerating Goneid algorithm is an efficient usage of DSP slices and block RAMs. Based on these constatations we have choose to investigate Goneid algorithm to realize an optimized version that can compute Hough Transform in less than 40 ms for a 1080x1920 image size.

Algorithm 1 Original 1D Hough Transform algorithm

Require: edge image I of $[W, H]$ size
1: **for** each row i from 1 to H **do**
2: **for** each edge $I(i, j)$ with j from 1 to $W - 1$ **do**
3: **for** each edge $I(i, k)$ with k from $j + 1$ to W **do**
4: $V_x(\frac{j+k}{2}) = V_x(\frac{j+k}{2}) + 1$
5: **end for**
6: **end for**
7: **end for**
8: **for** each column i from 1 to W **do**
9: **for** each edge $I(j, i)$ with j from 1 to $H - 1$ **do**
10: **for** each edge $I(k, i)$ with k from $j + 1$ to H **do**
11: $V_y(\frac{j+k}{2}) = V_y(\frac{j+k}{2}) + 1$
12: **end for**
13: **end for**
14: **end for**
15: **for** each local maximum $V_x(i)$ in V_x **do**
16: **for** each local maximum $V_y(j)$ in V_y **do**
17: **for all** edge $I(k, m)$ in image **do**
18: $V_{r(i,j,||I(k,m)-(i,j)||)} = V_{r(i,j,||I(k,m)-(i,j)||)} + 1$
19: **end for**
20: **if** maximum value of $V_r(i, j, r) > f * (4\sqrt{2}r)$ **then**
21: $||(x, y) - (i, j)|| = r$ is a circle
22: **end if**
23: **end for**
24: **end for**

III. PROPOSED METHOD

Our FPGA based architecture for a real-time implementation of the 1D Hough transform multi-circles detection is focused on the acceleration of the construction of histograms Vx and Vy, [1 -7] and [8-14] respectively to the algorithm 1. Our solution significantly reduces the use of memory and the latency execution. First, we propose an equivalent algorithm, algorithm 2, which gives the same results, that is to say the same x-coordinate and y-coordinate histograms as in steps [1 -7] and [8-14] of the algorithm 1.

Algorithm 2 is obtained first by rewriting the steps [1-7] of the algorithm 1 as shown in algorithm 3, taking into account all the points of the image instead of just taking the contour points.

In second, we change the order of the for-loops and obtain a new formulation of the steps [1-7] shown in algorithm 4.

Finally, we can rewrite the i-loop as a sum, and rewrite the steps [1-7] as proposed in steps [1-5] of the algorithm 2.

TABLE I

STATE OF THE ART OF HOUGH TRANSFORM HARDWARE IMPLEMENTATION.

Authors	FPGA	Resources used	Image resolution [pixel]	Latency [ms] / frequency [MHz]	Throughput [Mpixel/s]	Latency [ms] extrapoled for 1920x1080 resolution
Ferhat et al, 2012 [1]	Virtex 2	578 slices, 8 BRAM	one circle	- / 66.9	-	-
Elhossini et al, 2012 [3]	Virtex 4	256Kb of BRAM	800x600	33.33 / -	14.40	144
Zhou et al, 2014 [19]	Virtex-7	5562Kb of BRAM, 398 DSP48	400x400	5.338 / 181.812	29.974	69.18
Seo et al, 2015 [16]	Kintex 7	internal memory	640x480	10 / 250	30.720	67.5
Irwansyah et al, 2015 [8]	Virtex 4	512Kb of BRAM	1024x1024 scaled by factor 2 (512x512)	7.8125 / 135	33.554	61.8
Kumar et al, 2017 [11]	Artix 7	52 BRAMs of 36Kb	320x240	3.46 / 203	22.197	93.42

Algorithm 2 Our Modified 1D Hough Transform multi-circles detection algorithm to compute histograms Vx and Vy

for each column j from 1 to $W-1$ **do**
  for each column k from $j+1$ to W **do**
    $V_x(\frac{j+k}{2}) = V_x(\frac{j+k}{2}) + \sum_{i=1}^{H} I(i,j) * I(i,k)$
  end for
end for
for each row j from 1 to $H-1$ **do**
  for each row k from $j+1$ to H **do**
    $V_y(\frac{j+k}{2}) = V_y(\frac{j+k}{2}) + \sum_{i=1}^{W} I(j,i) * I(k,i)$
  end for
end for
for each local maximum $V_x(i)$ in Vx **do**
  for each local maximum $V_y(j)$ in Vy **do**
    for all edge $I(k,m)$ in image **do**
      $V_{r(i,j,||I(k,m)-(i,j)||)} = V_{r(i,j,||I(k,m)-(i,j)||)} + 1$
    end for
    if max value of $V_r(i,j,r) > f * (4\sqrt{2}r)$ **then**
      $||(x,y)-(i,j)|| = r$ is a circle
    end if
  end for
end for

Algorithm 3 1D Hough Transform multi-circles detection algorithm detail modification: optimization of the edge point

for each row i from 1 to H **do**
  for each point $I(i,j)$ with j from 1 to $W-1$ **do**
    for each point $I(i,k)$ with k from $j+1$ to W **do**
      $V_x(\frac{j+k}{2}) = V_x(\frac{j+k}{2}) + I(i,j) * I(i,k)$
    end for
  end for
end for

Algorithm 4 1D Hough Transform multi-circles detection algorithm detail modification: change the order of the for-loops

for each column j from 1 to $W-1$ **do**
  for each column k from $j+1$ to W **do**
    for each row i from 1 to H **do**
      $V_x(\frac{j+k}{2}) = V_x(\frac{j+k}{2}) + I(i,j) * I(i,k)$
    end for
  end for
end for

The advantage of our histogram construction process show in algorithm 2 is that we obtain the accumulation value for each coordinate of the x-coordinate each two j-loops, that means that we obtain an x-coordinate accumulation value every two columns read and a y-coordinate accumulation value every two rows read. It has a singnificant impact in latency computation and resources consommation as it is explained in section IV-A.

We have validated our algorithm on image of closed contour as show in figure 1. In this image f correspond to the sensitivity threshold visible in step [16] of our algorithm 2. A green circle indicates that we have localized a closed contour.

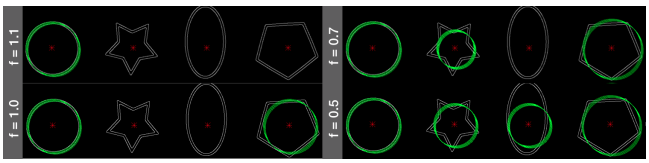

Fig. 1. Algorithm validation with different value of the sensitivity threshold f

In the next section we describe our digital hardware architecture that implement our algorithm.

IV. CHT ARCHITECTURE

In figure 2, we introduce our architecture to implement the algorithm 2. The overall architecture is composed by five modules described below:

- **x-coordinate and y-coordinate computation modules**, these modules compute N_c x-coordinates and N_c y-coordinates. They produce ${N_c}^2$ center candidates from combination of each x-coordinates and y-coordinates;
- **radius computation**, this module builds for each center candidate a histogram using Euclidean distance between this center candidate and each edge point. Once the histogram is built, this module assigns the most accumulated Euclidean distance as the radius. This module selects as a real circle the center candidates and radius where the accumulation value is $> f * 4\sqrt{2}r$;
- **Registers module**, this module register the N_c x-coordinates and the N_c y-coordinates of centers candidates and the ${N_c}^2$ circles in parallel.

In the next sections we describe each module of our architecture.

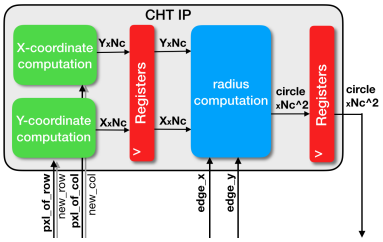

Fig. 2. Our digital architecture.

A. *x-coordinate and y-coordinate computation module*

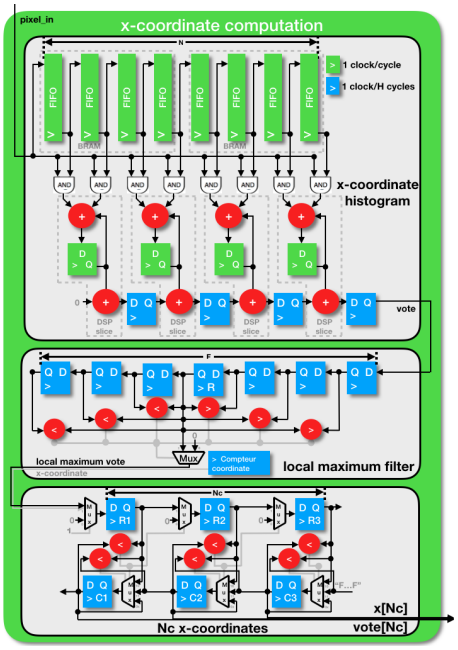

Fig. 3. x-coordinate computation architecture.

This module is fully pipelined and shown in figure 3. It computes N_c x-coordinates for a maximum circle diameter D_m, $D_m > 1$. in the first stage, for an image of size WxH, each column of H bits is shifted by a FIFO memory. We use here one 36Kb Block RAM that can shift $\frac{36K}{H}$ columns at the same time and a total of $\frac{D_m * H}{36K}$ BRAMs of 36 Kb are used in parallel to compare one input column with D_m others columns. $\frac{D_m}{2}$ DSP48 slices are used to add the votes between the input column and every two columns. The total votes are accumulated in the *blue* registers to be added each H cycles (when the input column is totally read). With this architecture, we compute a x-coordinate histogram value every column read.

In the second stage, the vote values are filtered in order to find the local maximum. A sliding window of size F, that corresponds to minimum Euclidian-distance between two centers candidates, is used to compare a vote value with the $\frac{F}{2}$ previous values and with the $\frac{F}{2}$ posterior values. The output is the vote corresponding to the largest histogram value for a x-coordinate. If there are two circles with centers separated by an Euclidean-distance less than F pixels in the image, only the circle with more edge points will be detected.

Finally, in the third stage, all the local maximum histogram values are stored in registers R_i and C_i following the next rules:

- if $R_i > C_i$ then the register $R_{i+1} = R_1$ else $R_{i+1} = 0$;
- if $R_i > C_{i+1}$ *and* $R_i > C_i$ then the register $C_i = C_{i+1}$ else if $R_i > C_{i+1}$ *and* $R_i < C_i$ then the register $C_i = R_i$ else $C_i = C_i$.

Hence, the larger values will be gradually transferred to the right side through the registers C_i. This process is executed until all the columns in the image are read. Finally the largest local maximum histogram values are stored in C_i and their respective coordinates correspond to the N_c x-coordinates of the circles candidates. Using a similar architecture, N_c y-coordinates are calculated.

For an image of WxH size, $\frac{D_m(W+H)}{36K}$ BRAMs of 36Kbits and D_m DSP48 slices are necessary to calculate the x-and y-coordinates in $W * H + 2 + \frac{D_m}{2} + \frac{F}{2} + N_c$ cycles.

Each one x- and y-coordinates are combined to obtain ${N_c}^2$ center candidates.

B. *Radius computation module*

In figure 4, we present our fully pipelined module proposed for the radius computation. In this module, in the first stage, an Euclidean-distance histogram is built in parallel for each center candidate. This stage computes the Euclidean-distance between one center candidate and all edge points. Each add and substract computations are performed in 2 cycles, the multiplication computation is performed in 3 cycles. Each operation is executed using one DSP slice. We use the Xilinx CORDIC IP core [18] to compute the square root of the Euclidean-distance with a 16 bits integer number in 8 cycles. To vote the Euclidean-distance we use a memory, we propose the architecture illustrated in figure 5, in this architecture each memory is implemented in one BRAM of 18Kbits that enables a simultaneous read and write in one cycle and avoid the collisions.

Once all edge points of the image are read, in the second stage, for each center candidate, all the values of the memory are read in order to find the most voted Euclidean-distance in $\frac{N}{2}$ cycles. This Euclidean-distance r becomes the radius of this center candidate. We compare this radius r to a threshold of $4\sqrt{2}r$ [10] to determine if it corresponds to a true circle. It is possible to modify this threshold in order to make the verification more sensitive.

Each radius that corresponds to a true circle and the corresponding center become inputs to the shift registers.

In total, $4N_c + {N_c}^2$ DSP slices and $\frac{{N_c}^2}{2}$ BRAMs of 36 Kbits are necessay to perform the radius voting process for ${N_c}^2$ center candidates and obtain their radius in $W * H + 18 + \frac{D_m}{2}$ cycles.

V. SoC IMPLEMENTATION

To validate our architecture, we prototype it on a SoC-based system, the Digilent ZedBoard Zynq-7000 ARM/FPGA XC7Z020 SoC Development Board. Zynq is

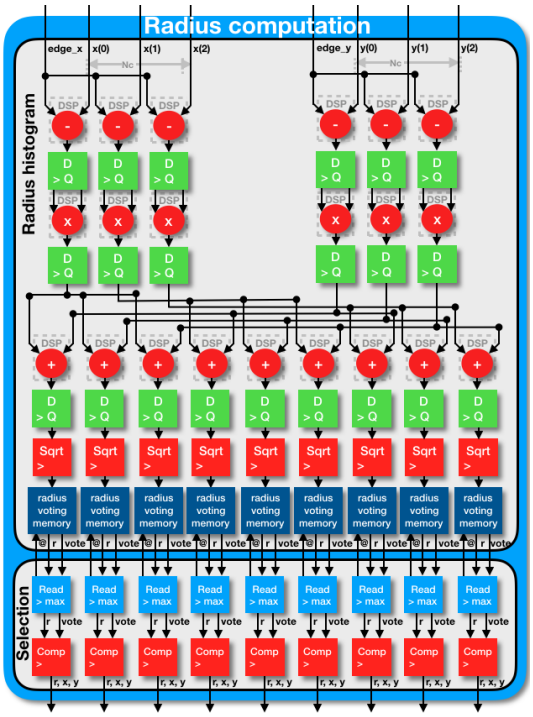

Fig. 4. Radius computation module architecture.

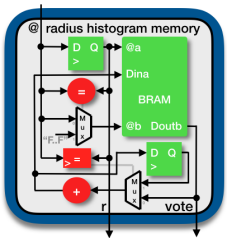

Fig. 5. Proposed radius histogram memory architecture.

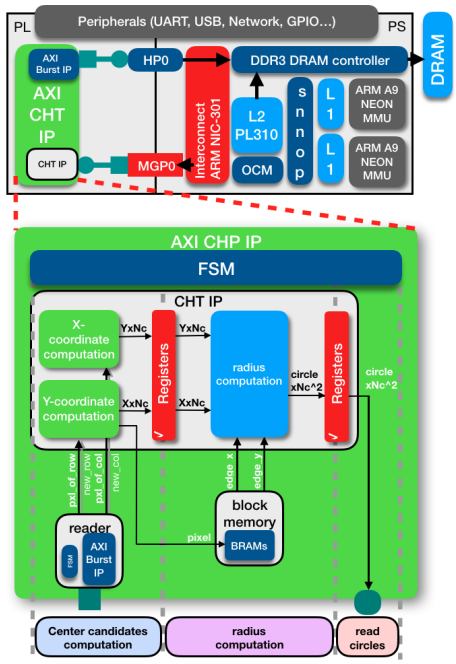

Fig. 6. System diagram of our AXI CHT IP core integrated on a Zynq architecture.

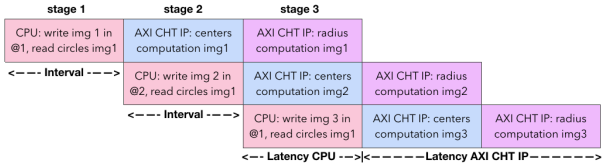

Fig. 7. System Pipeline

not the final platform, as it cannot be integrated inside a capsule, we plan to integrate it in a new Artix7 from Xilinx, compatible with a capsule, that was not available at the time we make the experiments. Our first goal was to validate our Hough Transform IP and measure its execution time on a real SoC not to far to the final device.

In figure 6, we illustrate the integration of our architecture, the Circle Hough Transform (CHT) IP in this SoC.

We have realised a pipeline of three operations that are: first, write an image into the DRAM memory and read computed circles, second, center computation and third, radius computation. The pipeline execution in the global system is shown in figure 7.

As we can see, we use two address in DRAM $(@1, @2)$ in order to read and write an image at the same time. The CHT IP uses one HP master AXI ports to read images to compute centers candidates and calculate the radius in parallel.

To store the image in a block memory we use BRAMs of 36 Kb that can shift $\frac{36K}{W}$ rows at the same time, as we already shift D_m rows in y-coordinate computation module, we need shift only $H - N$ rows so we need add $\frac{(H-D_m)*W}{36K}$ BRAMs of 36 Kb.

The center candidate coordiantes computation stage take $W*H+2+\frac{D_m}{2}+\frac{F}{2}+Nc$ cycles and the radius computation stage take $W * H + 18 + \frac{D_m}{2}$ cycles. The total AXI CHP IP latency to computate the circles will take 2 times the center candidate coordinates computation because is the bigest latency stage in the pipeline.

So, to implement our proposed Hough Transform architecture we need $\frac{(H-D_m)*W}{36K} + \frac{D_m(W+H)}{36K} + \frac{{N_c}^2}{2}$ BRAMs of 36Ks and $D_m + 4N_c + {N_c}^2$ DSP slices in order to detect ${N_c}^2$ circles with a maximum circle diameter of D_m and separated at least F pixels in an image of WxH size with $2(W * H + 2 + \frac{D_m}{2} + \frac{F}{2} + N_c)$ cycles of latency and $W * H + 2 + \frac{D_m}{2} + \frac{F}{2} + N_c$ cycles of interval between two images. As we see, the latency and interval is directly linked to the image size. The bottleneck increasing the image resolution is limited by the number of BRAM that we have. The maximum circle diameter D_m and the number of circles to detect ${N_c}^2$ is limited by the number of BRAM and DSP48 that we have.

We have realised an implementation to detect ${N_c}^2 = 25$ circles with a maximum circle diameter of $D_m = 108$

and a sliding window $F = 40$. We consider an image of size 1920x1080. Detail of the implementation can be see in table II.

TABLE II

AXI CHT IP CORE IMPLEMENTATION RESULTS.

Architecture	CHT	AXI CHT IP core	Full System
LUT	20637	20770	21638
LUTRAM	275	278	373
FF	18629	18697	19706
BRAM (36Kb)	78.5	78.5	78.5
DSP	153	153	153
Freq. maximum[MHz]	149.16	147.99	128.89
Latency[ms]	27.805	28.025	32.178
Interval[ms]	13.902	14.012	16.089
Throughput[Mpixel/s]	149.154	147.984	128.89
fps[frame/s]	72	71	62

TABLE III

COMPARISON WITH THE STATE OF THE ART.

Our method	Artix7 with a frequency of 128.89MHz				
	Estimated and simulated				Implemented
Resolution -[pixel]	320 x240	400 x400	512 x512	640 x480	1920 x1080
BRAM of 36Kb	17.5	20.5	22.5	28.5	78.5
DSP48 slices	153	153	153	153	153
Latency [ms]	1.19	2.48	4.07	4.77	32.18
Interval [ms]	0.60	1.24	2.03	2.38	16.08
fps [frames]	1676	805	491	419	62
Throughput -[Mpixel/s]	128.8	128.9	128.9	128.9	128.9

As we can see in table II our Hough Transform can work with a maximum frequency of 149.16 MHz alone and 128.885 MHz in the global system. That is due to the distance between the furthest DSP slice and the AXI AMBA interconnection. Our digital architecture can process an image of 1920x1080 pixels in less thant 40 ms as it is expected and give a 62 fps throughput.

With table III, we can compare our Hough Transform architecture with the state of the art in table I. We calculate the processing time, BRAM and DSP of our architecture to realize a fair comparison with the same sizes of image. As we notice, our architecture is the better and offer a throughput 4 times better than the faster state of the art architecture.

VI. CONCLUSIONS

In this paper we proposed a efficient real-time pipelined architecture Hough Transform for multi-circles detection to help to localise polyps in gastrointestinal tract images. In the our architecture, an efficient method is implemented to significantly reduce internal use of memory and reduce time execution. Our architecture supports a maximum clock frequency of 149.16 MHz alone and 128.885 MHz in a global system to detect until 25 circles with a maximum circle diameter of 108 pixels. Our design has been validated on a Xilinx Zynq-7000 XC7Z020 using 78.5 BRAMs, 153 DSP slices, 21638 LUTs with a 125MHz clock. It obtains a latency of 33.214 ms and an interval between two images of 16,607 ms for a 1920x1080 pixels image. This architecture can process 62 images per second, and it offers a throughput 4 times better than the faster state of the art architecture.

REFERENCES

[1] ALIM, F. F.-T., MESSAOUDI, K., SEDDIKI, S., AND KERDJIDJ, O. Modified circular hough transform using fpga. In Microelectronics (ICM), 2012 24th International Conference on (2012), IEEE, pp. 1–4.

[2] ANTOLOVIC, D. Review of the hough transform method, with an implementation of the fast hough variant for line detection. Department of Computer Science, Indiana University (2008).

[3] ELHOSSINI, A., AND MOUSSA, M. Memory efficient fpga implementation of hough transform for line and circle detection. In Electrical & Computer Engineering (CCECE), 2012 25th IEEE Canadian Conference on (2012), IEEE, pp. 1–5.

[4] FERLAY, J., SOERJOMATARAM, I., DIKSHIT, R., ESER, S., MATHERS, C., REBELO, M., PARKIN, D. M., FORMAN, D., AND BRAY, F. Cancer incidence and mortality worldwide: sources, methods and major patterns in globocan 2012. International journal of cancer 136, 5 (2015), E359–E386.

[5] GONEID, A., EL-GINDI, S., AND SEWISY, A. A method for the hough transform detection of circles and ellipses using a 1-dimensional array. In Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on (1997), vol. 4, IEEE, pp. 3154–3157.

[6] HOUGH, P. V. Method and means for recognizing complex patterns, Dec. 18 1962. US Patent 3,069,654.

[7] IDDAN, G., MERON, G., GLUKHOVSKY, A., AND SWAIN, P. Wireless capsule endoscopy. Nature 405, 6785 (2000), 417.

[8] IRWANSYAH, A., IBRAHEEM, O. W., HAGEMEYER, J., PORRMANN, M., AND RÜCKERT, U. Fpga-based circular hough transform with graph clustering for vision-based multi-robot tracking. In ReConFigurable Computing and FPGAs (ReConFig), 2015 International Conference on (2015), IEEE, pp. 1–8.

[9] KARARGYRIS, A., AND BOURBAKIS, N. Detection of small bowel polyps and ulcers in wireless capsule endoscopy videos. IEEE transactions on Biomedical Engineering 58, 10 (2011), 2777–2786.

[10] KULPA, Z. On the properties of discrete circles, rings, and disks. Computer graphics and image processing 10, 4 (1979), 348–365.

[11] KUMAR, V., ASATI, A., AND GUPTA, A. Hardware accelerators for iris localization. Journal of Signal Processing Systems (2017), 1–17.

[12] MAMONOV, A. V., FIGUEIREDO, I. N., FIGUEIREDO, P. N., AND TSAI, Y.-H. R. Automated polyp detection in colon capsule endoscopy. IEEE transactions on medical imaging 33, 7 (2014), 1488–1502.

[13] MUKHOPADHYAY, P., AND CHAUDHURI, B. B. A survey of hough transform. Pattern Recognition 48, 3 (2015), 993–1010.

[14] ORLANDO, C., ANDREA, P., XAVIER, D., AND GRANADO, B. Polyps recognition using fuzzy trees. In Biomedical & Health Informatics (BHI), 2017 IEEE EMBS International Conference on (2017), IEEE, pp. 9–12.

[15] ROMAIN, O., HISTACE, A., SILVA, J., AYOUB, J., GRANADO, B., PINNA, A., DRAY, X., AND MARTEAU, P. Towards a multimodal wireless video capsule for detection of colonic polyps as prevention of colorectal cancer. In Bioinformatics and Bioengineering (BIBE), 2013 IEEE 13th International Conference on (2013), IEEE, pp. 1–6.

[16] SEO, S.-W., AND KIM, M. Efficient architecture for circle detection using hough transform. In Information and Communication Technology Convergence (ICTC), 2015 International Conference on (2015), IEEE, pp. 570–572.

[17] SILVA, J., HISTACE, A., ROMAIN, O., DRAY, X., AND GRANADO, B. Toward embedded detection of polyps in wce images for early diagnosis of colorectal cancer. International Journal of Computer Assisted Radiology and Surgery 9, 2 (2014), 283–293.

[18] WALTHER, J. S. A unified algorithm for elementary functions. In Proceedings of the May 18-20, 1971, spring joint computer conference (1971), ACM, pp. 379–385.

[19] ZHOU, X., ITO, Y., AND NAKANO, K. An efficient implementation of the one-dimensional hough transform algorithm for circle detection on the fpga. In Computing and Networking (CANDAR), 2014 Second International Symposium on (2014), IEEE, pp. 447–452.