Evolution in Electrical and Electronic Engineering Vol. 4 No. 1 (2023) 589-596

EEEE

Homepage: http://publisher.uthm.edu.my/periodicals/index.php/eeee
e-ISSN : 2756-8458

Hardware Implementation of Hough Transform for Straight Line Detection

Dhivaakar Ravindran[1], **Chessda Uttraphan**[1]*

[1] Faculty of Electrical and Electronic Engineering,
Universiti Tun Hussein Onn Malaysia, Parit Raja, 86400, Johor, MALAYSIA

*Corresponding Author Designation

DOI: https://doi.org/10.30880/eeee.2023.04.01.071
Received 01 February 2023; Accepted 10 April 2023; Available online 30 April 2023

Abstract: The use of line detection algorithms is becoming increasingly important in the field of image processing, particularly in areas that require automation, such as lane detection in self-driving cars and the localization of robots in the field of robotics. The Hough Transform is considered as a common algorithm for line detection, but its computationally intensive nature can lead to poor performance when implemented on general-purposed processor. The proposed work looks at implementing the Hough Transform on a Field Programmable Gate Array (FPGA) instead, as FPGAs holds parallel data processing capabilities, which reduces system latency. The work involves identifying the necessary parameters for the application of Hough Transform and designing the system by utilizing RAM and ROM modules on the Intel Quartus Prime, while taking into account the arithmetic operations required to produce the desired output. The execution time of the Hough Transform on an FPGA was compared to its execution time in the MATLAB software environment, with the result showing that the FPGA's implementation is 94 times faster than the execution of Hough Transform in MATLAB.

Keywords: Field Programmable Gate Array, Hough Transform, Line Detection

1. Introduction

In the current world filled with technological advancements in different sectors such as the medical field, manufacturing, and transportation, it is evident that the automobile sector has improved their production of vehicles in terms of technologically aided features that can be found in newer batches of vehicles [1]. This technological advancement the industry involves the use of line detecting algorithms such as Hough Transform in order to detect the lanes present on the road [2]. Similarly various sectors in the industry require the use of line detecting algorithms such as surveillance and security, medical imaging, and industrial automation whereby the line detection algorithms are used to detect and tract objects on a conveyor belt, guiding robots to pick and place the objects in the correct location [3]. However, the implementation of Hough Transform in general-purposed processors cannot deliver high-performance

Formatted: Line spacing: Multiple 1.08 li

*Corresponding Author: chessda@uthm.edu.my

system because of the serial processing nature in the general-purposed processor [4]. This leads to bottlenecks due to multiple input handlings such as image processing, memory allocation, and mathematical operations [5]. In general-purposed processor, the computation is conducted in a sequential order whereby only one line of code is executed at a time. Meanwhile, the proposed work involves the implementation Hough Transform on a Field Programmable Gate Array (FPGA), which offers parallel processing whereby the circuit on an FPGA will able to compute multiple functions such as handling the processing of images and digital signals in a single clock execution time [6]. Therefore, the application of image processing and recognition on an FPGA is suitable due to its high concurrency and real time image processing capabilities [7].

In this work, the design is simulated in ModelSim, where execution time is compared with the software environment for a fixed parameter of number of pixels.

2. Materials and Methods

2.1 Block Diagram

The system is initially developed based on the mathematical notation of Hough Transform given in Eq. (1), where ρ is the distance from the origin to the closest point on the straight line, (x, y) is the coordinate of the detected edge pixel, and theta is the angle between the x axis and the line connecting the origin with that closest point. The Hough Transform can be implemented in the MATLAB environment with the provided built-in functions that allows users to call the function when the user requires line detection to be executed upon an image. The effectivity of the algorithms in detecting lines in an image can be analyzed by feeding images to the algorithm in MATLAB. Once the suitable parameters are obtained, the operation of Hough Transform can be modified to operate in a hardware level. The hardware level design of the system utilizes multiple RAM and ROM modules to utilize memory storage and lookup tables in the system. The proposed design is illustrated in Figure 1. The targeted hardware is the Cyclone IV on the DE1-SoC board.

$$\rho = x\cos\theta + y\sin\theta \quad \text{(Eq.1)}$$

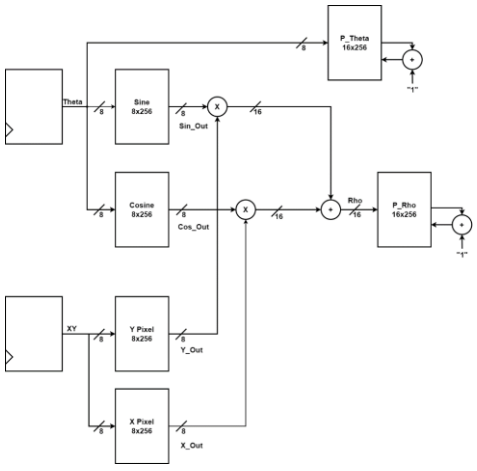


Figure 1: Block diagram of the system

2.2 Methods

The image pixels in the coordinate space (x, y) which are represented as X _Pixel and Y_Pixel are fed into the system respectively. At the same cycle, the value of θ is fed into the lookup table that consists of Read-Only Memory (ROM) modules for Cosine and Sine respectively. The corresponding output from that system is multiplied with X and Y pixels respectively and added together to form ρ. The value of ρ is fed into the Random Access Memory (RAM) as an address and the corresponding data in that address is accumulated. In order to push the values of X and Y pixels into their registers, a counter is required to be implemented to keep count of the pixels that have been pushed whilst ensuring that all pixels are accounted for in the line detecting system. Figure 2 shows details of the proposed system.

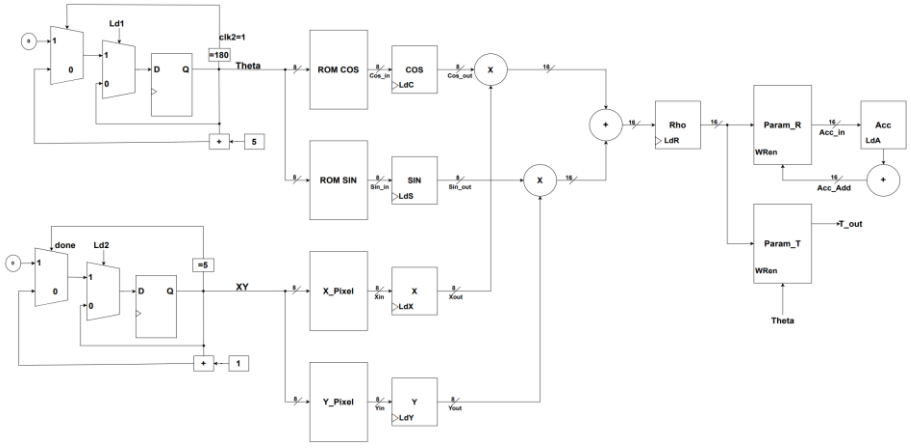


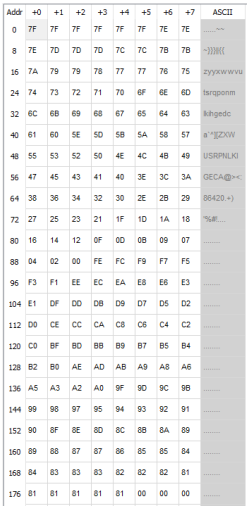
Figure 2: Details implementation of the proposed system

Multiple blocks such as *COS*, *SIN*, *X* and *Y* consists of a 'Ld' signal which represents a load signal. The load signal acts like a gate which allows data to pass through only when the signal is active. This is useful to ensure a synchronized data transfer is present between different components, or to control the flow of data within the system. Other than that, the load signal also ensures that the existing data in the register is not overwritten by a new input data before the operation has been completed. Once all of the number of bits required for each wire has been identified along with the registers and their corresponding input and output pins, the design can be coded via Verilog in Intel Quartus Prime.

2.3 Memory Initialization of Cosine and Sine

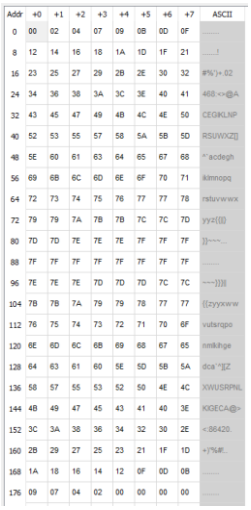
Memory Initialization refers to the process of loading data into a memory component at the start of a design's operation [8]. This can be done by specifying the initial values for each memory location in the design, or by using a memory initialization file. Memory initialization is crucial in order to specify the starting state of the ROM which will store the values for θ and their corresponding output for both sine and cosine. Through the use of the memory initialization file, the memory can be pre-loaded with the data in the form of a lookup table. To use the memory initialization file in Quartus Prime, the file must be created and saved in a specific format. The file must contain one line of text for each memory location, with the address and value separated by a space. The addresses must be listed in ascending order and the values must be specified in hexadecimal format. Once the file has been created, it can be associated with a memory

component in the Quartus Prime project, and the data will be loaded into the memory when the design in compiled. Figure 3 and Figure 4 show the content initialization of the proposed design.



Addr	+0	+1	+2	+3	+4	+5	+6	+7	ASCII
0	7F	7F	7F	7F	7F	7F	7E	7E	……~~
8	7E	7D	7D	7D	7C	7C	7B	7B	~}}}||{{
16	7A	79	79	78	77	77	76	75	zyyxxwwvu
24	74	73	72	71	70	6F	6E	6D	tsrqponm
32	6C	6B	69	68	67	65	64	63	lkihgedc
40	61	60	5E	5D	5B	5A	58	57	a`^][ZXW
48	55	53	52	50	4E	4C	4B	49	USRPNLKI
56	47	45	43	41	40	3E	3C	3A	GECA@><
64	38	36	34	32	30	2E	2B	29	86420.+)
72	27	25	23	21	1F	1D	1A	18	'%#!....
80	16	14	12	0F	0D	0B	09	07	…….
88	04	02	00	FE	FC	F9	F7	F5	…….
96	F3	F1	EE	EC	EA	E8	E6	E3	…….
104	E1	DF	DD	DB	D9	D7	D5	D2	…….
112	D0	CE	CC	CA	C8	C6	C4	C2	…….
120	C0	BF	BD	BB	B9	B7	B5	B4	…….
128	B2	B0	AE	AD	AB	A9	A8	A6	…….
136	A5	A3	A2	A0	9F	9D	9C	9B	…….
144	99	98	97	95	94	93	92	91	…….
152	90	8F	8E	8D	8C	8B	8A	89	…….
160	89	88	87	87	86	85	85	84	…….
168	84	83	83	83	82	82	82	81	…….
176	81	81	81	81	81	00	00	00	…….

Figure 3: Memory initialization of cosine



Addr	+0	+1	+2	+3	+4	+5	+6	+7	ASCII
0	00	02	04	07	09	0B	0D	0F	…….
8	12	14	16	18	1A	1D	1F	21	……!
16	23	25	27	29	2B	2E	30	32	#%')+.02
24	34	36	38	3A	3C	3E	40	41	468:<>@A
32	43	45	47	49	4B	4C	4E	50	CEGIKLNP
40	52	53	55	57	58	5A	5B	5D	RSUWXZ[]
48	5E	60	61	63	64	65	67	68	^`acdegh
56	69	6B	6C	6D	6E	6F	70	71	iklmnopq
64	72	73	74	75	76	77	77	78	rstuvwwx
72	79	79	7A	7B	7B	7C	7C	7D	yyz{{|}
80	7D	7D	7E	7E	7E	7F	7F	7F	}}~~~...
88	7F	7F	7F	7F	7F	7F	7F	7F	…….
96	7E	7E	7E	7D	7D	7D	7C	7C	~~~}}}|
104	7B	7B	7A	79	79	78	77	77	{{zyyxww
112	76	75	74	73	72	71	70	6F	vutsrqpo
120	6E	6D	6C	6B	69	68	67	65	nmlkihge
128	64	63	61	60	5E	5D	5B	5A	dca`^][Z
136	58	57	55	53	52	50	4E	4C	XWUSRPNL
144	4B	49	47	45	43	41	40	3E	KIGECA@>
152	3C	3A	38	36	34	32	30	2E	<:86420.
160	2B	29	27	25	23	21	1F	1D	+)'%#!..
168	1A	18	16	14	12	0F	0D	0B	…….
176	09	07	04	02	00	00	00	00	…….

Figure 4: Memory initialization of sine

2.4 Control Unit Hardware Design

A control unit is a part of the digital system that is responsible for controlling the flow of data and instructions within the system. The control unit generates control signals which indicates the other units in the system on what should be done based on the instructions that is received. In terms of this hardware design, the control unit is required to control the state of the load signals that are present in the registers. In order to determine the proper control flow of the hardware, the control unit is designed to switch between different load signals based on states. In this work, there are a total of eight states in order to execute one complete cycle to obtain the resulting value of ρ and the votes are accumulated at the address. The control sequence is given in Table 1.

Table 1: CS-Table of design

Cycle	RTL code	LdC	LdS	LdX	LdY	LdR	LdA	Wren	Ld1	Ld2
S0	idle;	0	0	0	0	0	0	0	0	0
S1	Theta <- Theta+8'd5 XY<-XY+1;	0	0	0	0	0	0	0	1	1
S2	CosOut<-CosIn; SinOut<-SinIn; XOut<-Xin; YOut<-Yin; Theta <- Theta+5; XY<-XY+1;	1	1	1	1	0	0	0	1	1
S3	Rho<-XOut*CosOut+YOut*SinOut; XY<-XY+d'd1; Theta <- Theta+5;	0	0	0	0	1	0	0	1	1
S4	XY<-XY+d'd1; Theta <- Theta+5;	0	0	0	0	0	0	0	1	1
S5	ACC<=ACC_in; XY<-XY+d'd1; Theta <- Theta+8'd5 ACC_Add=ACC+1;	0	0	0	0	0	1	0	1	1
S6	Param_R<=ACC_Add; XY<-XY+d'd1; Theta <- Theta+8'd5	0	0	0	0	0	0	1	1	1

By correlating to the RTL code, the table can be designed accordingly. For each cycle, the trigger signals that are consisting of 9-bit in total will be represented in the Control Unit of the design. Once the Control Unit has been designed based on the table, the Datapath Unit, Control Unit and the Counter modules can be can be obtained.

3. Results and Discussion

The initial results of execution time for an image were obtained through the software environment of MATLAB. The built-in functions such as tic and toc were implemented in order to obtain the execution time for specific instructions in which they are in between the tic and toc functions respectively. Based on the implementation of a minimal 6 x 6-pixel value image, the elapsed time can be obtained in order to execute Hough Transform as shown in Figure 5.

```
>> ExecutionTIme_StraightLine
Elapsed time is 0.002448 seconds.
```

Figure 5: Elapsed time to execute Hough Transform

In the hardware design, multiple cycles are involved complete the operation of accumulating votes in the address based on the mathematical operation of Hough Transform. In order to analyse the performance of the system, the execution time of the design can be analysed with the time taken for the operation to be completed in a software environment, which in this case is the use of MATLAB. In order to conduct a sustainable analysis, an image with a minimal resolution of 6 x 6 is applied onto the application. The image consists of a pre-edge detected straight line. The pixel value for each location is identified by using the functions available in MATLAB as shown in Figure 6.

```
>> Pixel_Value_Image

x =                     y =

     1                       3
     2                       2
     3                       1
     4                       0
     3                       4
     5                       5
```

Figure 6: Pixel value of image

The pixel values of the 6 x 6 edge detected image can be obtained and substituted into the *mif* file that represents the ROM modules for *X* and *Y* pixels respectively. The *mif* files for *X* and *Y* pixels are shown in Figure 7 and 8, respectively.

Addr	+0	+1	+2	+3	+4	+5	+6	+7	ASCII
0	1	2	3	4	3	5	0	0	.....
8	0	0	0	0	0	0	0	0	.....
16	0	0	0	0	0	0	0	0	.....

Figure 7: ROM of *X* pixel

Addr	+0	+1	+2	+3	+4	+5	+6	+7	ASCII
0	3	2	1	0	4	5	0	0	.....
8	0	0	0	0	0	0	0	0	.....
16	0	0	0	0	0	0	0	0	.....

Figure 8: ROM of *Y* pixel

Once the values of *X* and *Y* pixels are pre-loaded into the ROM modules, the simulation can be executed to observe the operation of Hough Transform onto the pre-loaded pixel values of *X* and *Y* respectively.

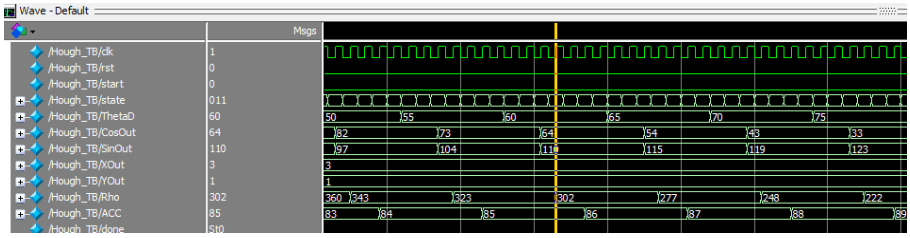


Figure 9: Simulation result of implementing the Hough Transform on the 6x6 pixel image

As shown in Figure 9, values of θ are in the intervals of five units and is declared as *ThetaD* in the testbench. The resulting value of substituting the value of θ into the equations *sin(θ)* and *cos(θ)* can be observed in the parameter *SinOut* and *CosOut* respectively. The result shows the state at which the value of θ is 60. When these values are substituted into the equation, the Sine operation will produce a value of 0.86603 while the Cosine operation will produce a value of 0.5 which are floating numbers. In hardware design, floating numbers are difficult to be processed because they require a more complex representation than integers. Floating point numbers require more sophisticated hardware to manipulate them [9]. Hence, the value is rounded. The round number is then normalized in the range of -127 to 128 (8-bit signed integer). Based on this representation, the value obtained for Sine and Cosine can be obtained as 110d and 64d respectively. By conducting manual mathematical operation, the resulting value that should be obtained in the output *CosOut* and *SinOut* can be verified.

$$CosOut = \cos\left(\frac{60\times\pi}{180}\right) \times 127 \qquad \text{(Eq.2)}$$
$$CosOut \approx 64$$

$$SinOut = \sin\left(\frac{60\times\pi}{180}\right) \times 127 \qquad \text{(Eq.3)}$$
$$SinOut \approx 110$$

With the obtained value of *CosOut* and *SinOut*, values of *CosOut*, *SinOut*, *X_Pixel* and *Y_Pixel* can be substituted into Equation (1). By assuming *X_Pixel* = 3, and *Y_Pixel* =1, Therefore,

$$\rho = 3(64) + 1(110)$$
$$= 302$$

Based on the value obtained, it can be deduced that the same value of ρ can be observed at the output of 'Rho' in Figure 9.

The execution time of the Hough Transform onto a 6 x 6 image in the proposed design is approximately 25.93 µs. For comparison, Hough Transform can be applied in the software environment of MATLAB to the same 6 x 6 image to obtain the time taken to execute the process. With the implementation of 'tic' and 'toc' in MATLAB, the time taken to execute a specific line of code can be obtained [10]. Once the image is loaded into MATLAB, the code can be executed to obtain the execution time of the software. In the MATLAB, it can be observed that the time taken to obtain the results of Hough Transform in is 2448 µs. From the measurement, it can be observed that the execution time is improved by 94 times for hardware implementation as compared to software.

4. Conclusion

In conclusion, a hardware implementation of Hough Transform for straight line detection using an FPGA has the potential to significantly improve the efficiency and convenience of line recognition services in a variety of applications. By implementing the algorithm on an FPGA, it can be considered that the system can offer a better performance spectrum compared to software implementation. The execution time of the FPGA to execute instructions are faster than general purpose processor since FPGA's can operate at a very high speed especially in cases of applications that require fast processing or real-time performance. Other than that, the system can be utilized in real-time processing due to its low power consumption since they consume relatively low power for their operation. With the base line detection algorithm such as Hough Transform involved, the system can be upgraded to process real-time image processing with ease due to their customization feature to meet specific requirements and ensuring that the system is tailored to the needs of the particular line detecting applications.

Acknowledgement

The authors would like to thank the Faculty of Electrical and Electronic Engineering, Universiti Tun Hussein Onn Malaysia for its support.

References

[1] F. Arena, G. Pau, and A. Severino, "An Overview on the Current Status and Future Perspectives of Smart Cars," *Infrastructures*, vol. 5, no. 7. MDPI Multidisciplinary Digital Publishing Institute, Jun. 30, 2020. doi: 10.3390/infrastructures5070053.

[2] Z. Zhang and X. Ma, "Lane Recognition Algorithm Using the Hough Transform Based on Complicated Conditions," *Journal of Computer and Communications*, vol. 07, no. 11, pp. 65–75, 2019, doi: 10.4236/jcc.2019.711005.

[3] A. Shehata, S. Mohammad, M. Ehab Ragab, A. Shehata Hassanein, and M. Sameer, "A Survey on Hough Transform, Theory, Techniques and Applications." [Online]. Available: https://www.researchgate.net/publication/272195556

[4] P. C. J. Otermans, A. Parton, and A. J. Szameitat, "The working memory costs of a central attentional bottleneck in multitasking," *Psychol Res*, vol. 86, no. 6, pp. 1774–1791, Sep. 2022, doi: 10.1007/s00426-021-01615-1.

[5] "FPGA vs. Microcontroller | What is the Difference?" https://www.mclpcb.com/blog/fpga-vs-microcontroller/ (accessed Apr. 09, 2022).

[6] "FPGA vs Microcontroller - Advantages of Using An FPGA." https://duotechservices.com/fpga-vs-microcontroller-advantages-of-using-fpga (accessed Apr. 09, 2022).

[7] C. Chen, "Design of image recognition system based on FPGA," in *2022 7th International Conference on Intelligent Computing and Signal Processing, ICSP 2022*, 2022, pp. 1924–1928. doi: 10.1109/ICSP54964.2022.9778604.

[8] I. Corporation, "Embedded Memory (RAM: 1-PORT, RAM: 2-PORT, ROM: 1-PORT, and ROM: 2-PORT) User Guide; Embedded Memory (RAM: 1-PORT, RAM: 2-PORT, ROM: 1-PORT, and ROM: 2-PORT) User Guide."

[9] G. Mohana Durga and D. Bhavani, "Floating Point Addition, Subtraction and Multiplication on FPGA," *International Journal of Scientific Development and Research*, vol. 3, no. 7, 2018, [Online]. Available: www.ijsdr.org

[10] "Start stopwatch timer - MATLAB tic." https://www.mathworks.com/help/matlab/ref/tic.html (accessed Jan. 29, 2023).