Penerbit Akademia Baru

Journal of Advanced Research Design

Journal homepage:
https://akademiabaru.com/submit/index.php/ard
ISSN: 2289-7984

Dataflow Management (Architecture) with Model-Based Design for SAR Signal Processing on FPGA

Atthawut Tirakitpanitchakorn[1], Nattha Jindapetch[1], Panadda Solod[2], Kiattisak Sengchuai[1], Vasan Jantarachote[1,*]

[1] Department of Electrical Engineering and Biomedical Engineering, Faculty of Engineering, Prince of Songkhla University, Songkhla 90110, Thailand
[2] Faculty of Industrial Education and Technology, Rajamangala University of Technology Srivijaya, Songkhla 90000, Thailand

ARTICLE INFO

Article history:
Received 3 January 2025
Received in revised form 14 February 2025
Accepted 16 June 2025
Available online 25 June 2025

Keywords:
GB-SAR; SLC image; nested loop; Simulink; MBD; HDL coder; FPGA

ABSTRACT

The signal processing in ground-based synthetic aperture radar (GB-SAR) has two main processes. One is the pre-processing data to prepare the data; the other is an algorithm to establish the single-look complex (SLC) image. Normally, the pre-processing data process wastes time recording huge signals for processing and a nested loop inside the process. In this paper, we propose a dataflow management approach for the pre-processing data process to overcome the wasted time on recording data. Against the nested loop, we proposed model-based design (MBD) techniques that support the Hardware Description Language (HDL) coder in Simulink. The proposed method performs well without a nested loop and can be implemented on the Xilinx Zynq Z-7020 Field Programmable Gate Array (FPGA) board, which is a low-cost FPGA. From the result, our proposed method has a high average percentage error in some stages, but the output from the final stage gives a very low average percentage error. The data after our proposed method can give the SLC image, which has the same significance as the SLC image from the original method. Our proposed implementation can perform on the FPGA as well. Therefore, the proposed method using dataflow management with MBD gives data with a low average percentage error and the SLC image similar to the original method.

1. Introduction

Currently, Ground-based synthetic aperture radar (GB-SAR) technology is mainly used for collecting data of interest on Earth, such as measuring soil deformation, biomass monitoring, building infrastructure and even detecting foreign objects at airstrips of the airports [1-4]. It uses radio frequency transmission from the radar system's transmitter to a target object and detects differences in objects through the reflection properties of the object's surface [5]. After the radar system receives the reflected signal. Then the reflected signal is compared with transmitted signal to obtain a signal at baseband frequency called Dechirp signal. Then the Dechirp signal forwards it to the signal

* *Corresponding author*
E-mail address: jvasan@eng.psu.ac.th

https://doi.org/10.37934/ard.135.1.169182

processing part. The signal processing consists of 2 parts: 1. Data management (Pre-processing Data) and 2. Algorithms for processing signal and converting to 2D images or Single Look Complex (SLC) [6].

Pre-processing data is the process of organizing large data into smaller sizes to process the data through algorithms to reduce the time required for processing. This is because the GB-SAR system has a Stop-Wait collection signal operation, which has the disadvantage that there will be unnecessary data for processing through the algorithm during the movement of the transmitter and receiver of the radar system. The algorithm used in the GB-SAR system is an algorithm for managing data and converting it into high-resolution SLC Images [7-10]. The basic operation of the algorithm starts by receiving pre-processed data and doing the Range compression and the Azimuth compression, then improving the quality of the data with the Stolt interpolation [11-16]. Next, the obtained data is plotted as an SLC Image. All the mentioned steps can be done only after the signal has been recorded.

In this research, we have presented data flow management in real-time for pre-processing data using the Model-Based Design (MBD) method on Simulink which is a practical design methodology that can develop rapid prototyping by using MATLAB/Simulink [17-20]. Many applications applied optimization methods such as fixed-point optimization, high-level synthesis optimizations and control data flow graph (CDFG)-based optimizations to improve the performance of lower-cost FPGAs with limited resources [21-24]. A data flow architecture is a high-impact method to improve performance [25-27]. However, there are limitations in that the MathWorks HDL requires that designers follow the coding guidelines [28]. Therefore, there are optimization rooms for code optimization and dataflow management. The proposed MBD is designed to depend on one-by-one data input which can be processed immediately. With our proposed MBD, the recorded signal isn't needed. With the MBD technique, the pre-processing data can be applied to FPGA boards that help with computation for signal processing. In this work, the Xilinx Zybo Z-7020 FPGA board was chosen which is a low-cost FPGA board to prove our MBD can perform on FPGA.

2. Methodology

2.1 Ground-Based Synthetic Aperture Radar (GB-SAR) Signal Processing

In this paper, the Range Migration Algorithm (RMA) algorithm is the algorithm used for GB-SAR signal processing, which will be created through Simulink and MATLAB software. The RMA algorithm design will be optimized to suit the reflected signal recorded from the receiver. The GB-SAR signal processing is divided into three main processes:

i. Data preparation
ii. RMA algorithm computational
iii. Data transformation into images.

Data preparation is the first important step in the GB-SAR signal processing to handle data recorded from the audio recording program. The recorded signals have two parts of information:

i. Pulse signal
ii. Dechirp signal.

Throughout the GB-SAR operation, the pulse signals are recorded in the audio recording program. Pulse signals are used in the data preparation stage to obtain accurate information in the Dechirp signal. Figure 1 shows the waveform of the recorded signal.

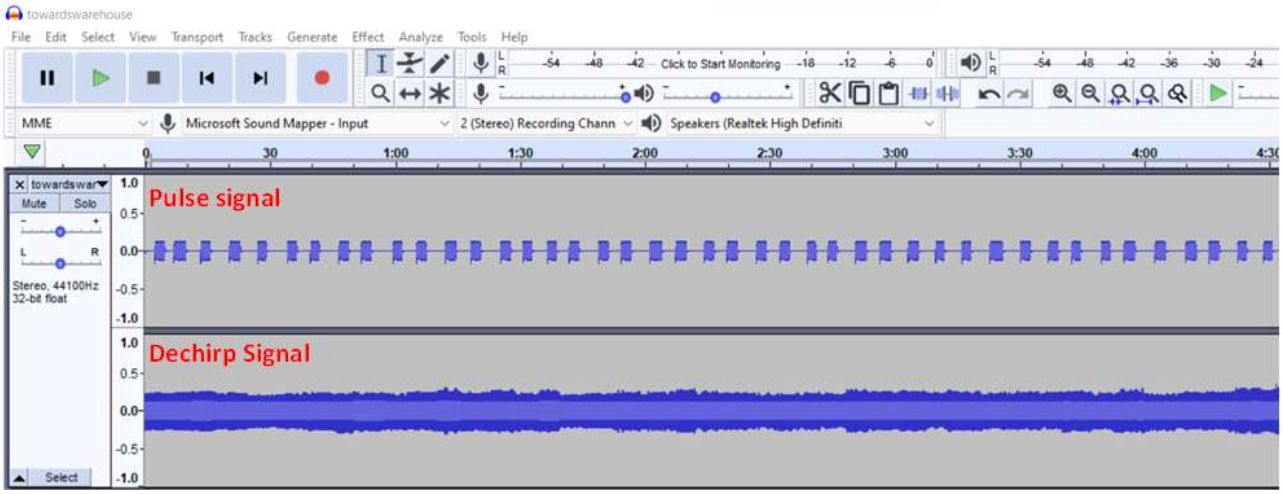


Fig. 1. An example of audio signals recorded with an audio recording program on Audacity [29]

Calculating the RMA algorithm is the second main step of SAR signal processing. The MBD implementation is shown in Figure 2. Input data is raw data converted from audio file to numbers and retrieved through the Simulink block in MATLAB. For splitting the raw data, the reshape block is used to rearrange the data into vector format. Two selector blocks are then used to divide the raw data into two sets:

i. Pulse signal data
ii. Dechirp signal data.

Three MATLAB function blocks are used in the data separation process:

i. The first block is used for separating data by position, which is called the parse by position block.
ii. The second block separates data using the pulse signal, which is called the parse by pulse
iii. The third block makes the data accurate and reliable, which is called the normalization data block.

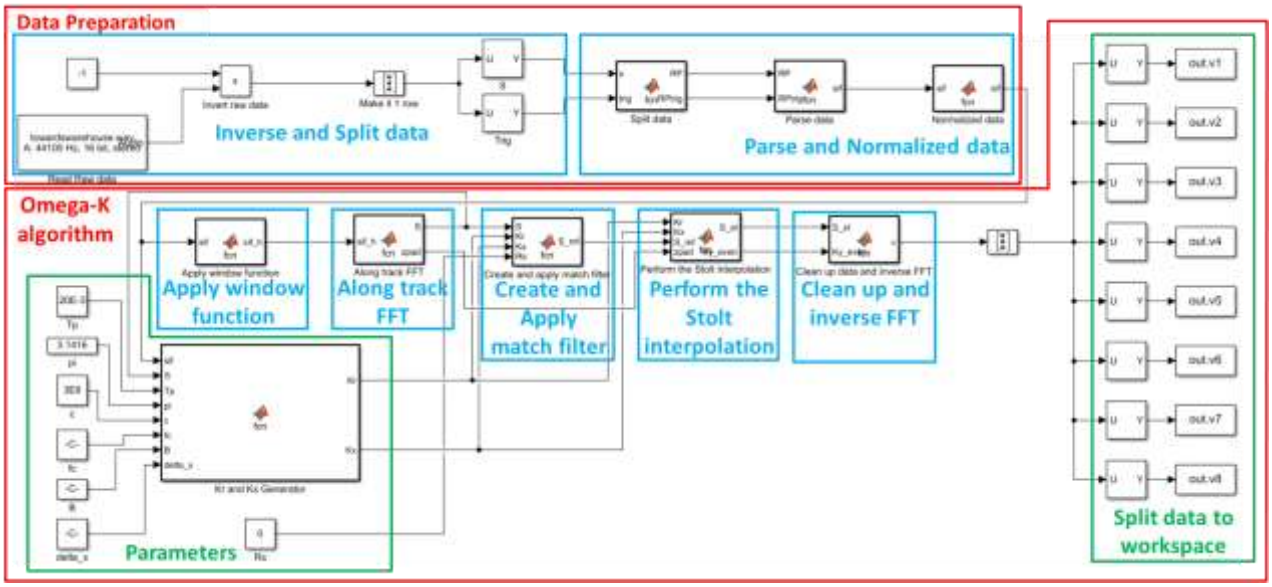


Fig. 2. The RMA algorithm designed using MBD techniques on Simulink

Similarly, calculating RMA also uses the MATLAB function blocks. There are consist of 5 steps inside:

i. Applying the window function for filtering data
ii. Along Track FFT by mapping the total number of data points to a suitable number for Fourier transformation
iii. Creation and applying matched filter to the data
iv. Performing the Stolt interpolation process
v. Cleaning up the data and performing the inverse FFT.

After the mentioned steps, we send the data to an M-file and converting data into images is the process of converting data into an image showing the intensity of the electromagnetic field reflected from an object. Within this step, we use an M-file to create the SLC image.

2.2 Code Optimization Method

We attempt to organize GB-SAR signal processing into stages of signal processing blocks to enable flexible, readily modifiable design. In the original method, we can divide it into 7 blocks, as shown in Figure 3(a). Block S1-S5 is data preprocessing to obtain the clean data for the IFFT and FFT of blocks S6 and S7, respectively. Block S1 (Rp_start) detects the pulse signal to find the recording point. Block S2 writes the signals to the buffer. Blocks S3 and S4 are new pulse synchronization and finding maximum amplitude signals to obtain the best-reflected signals. Block S5 writes the clean signals to another buffer. These blocks are MATLAB codes and some blocks cannot be synthesized to obtain the Hardware Description Language (HDL) codes. This means that such blocks cannot be implemented in FPGAs.

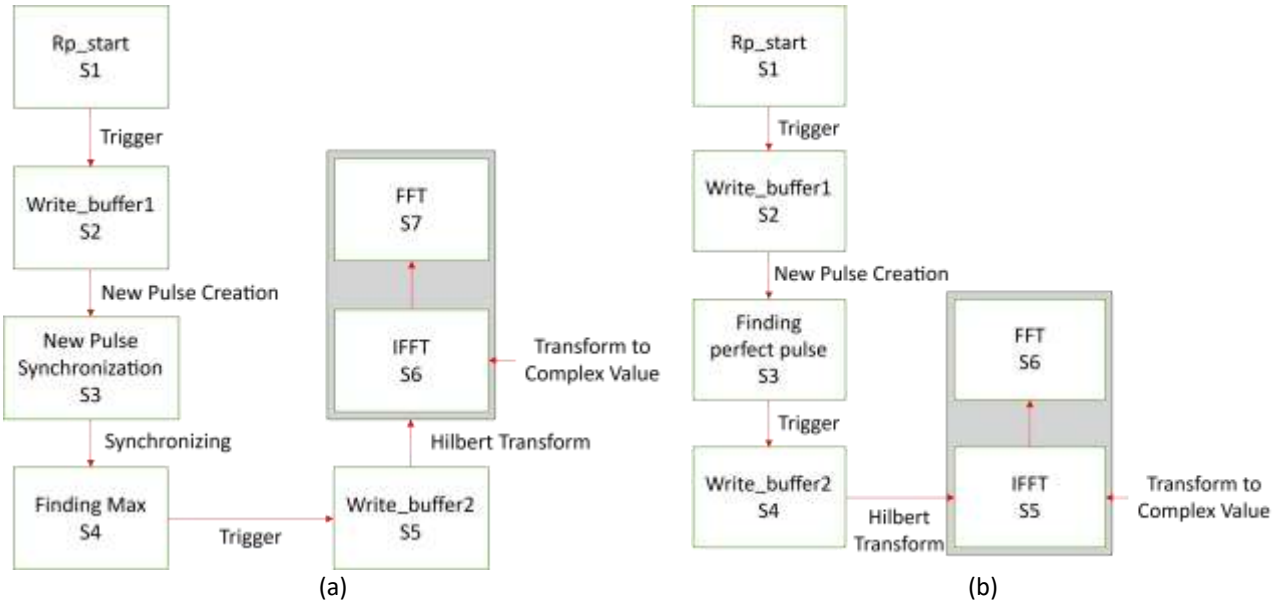


Fig. 3. SAR signal processing blocks: (a) the original method vs (b) the proposed method

There are limitations in Mathworks, where HDL coder cannot stream a loop if there are two or more nested loops at the same hierarchy level within another loop [28]. Moreover, the loop index must be known and increase by 1 on each iteration. Therefore, we optimize MATLAB code to satisfy the coding guides of the MathWorks HDL coder as follows.

According to the HDL coder's coding guides, we optimized the SAR signal processing to six blocks by collapsing blocks S3-S5 into S3-S4, defining the exact variables in S3 and S4 and reducing the redundant divided count in block S5, as shown in Figure 3(b). Table 1 shows the codes comparison between the original method and the proposed method. Now, the optimized code can be synthesized to HDL codes.

Table 1
The code comparison between the original method and the proposed method

The original method	The proposed method

Block S1
rpstart = abs(trig)>mean(abs(trig));
Block S2
count = 0;
Nrp = Trp*FS;
 	
for ii = Nrp+1:size(rpstart,1)-Nrp
  if rpstart(ii) == 1 & sum(rpstart(ii-Nrp:ii-1)) == 0
  count = count + 1;
  RP_org(count,:) = s(ii:ii+Nrp-1);
  RPtrig_org(count,:) = trig(ii:ii+Nrp-1);
  end
end
Block S3
start = (RPtrig_org(jj,:)> thresh);
	Block S1
	rpstart = abs(trig)>mean(abs(trig));
	Block S2
	count = 0;
	Nrp = Trp*FS;
	
	for ii = Nrp+1:size(rpstart,1)-Nrp
	  if rpstart(ii) == 1 & sum(rpstart(ii-Nrp:ii-1)) == 0
	  count = count + 1;
	  RP_pro(count,:) = s(ii:ii+Nrp-1);
	  RPtrig_pro(count,:) = trig(ii:ii+Nrp-1);
	  end
	end
	
	Block S3
	start = (RPtrig_pro(jj,:)> thresh);
	for kk=17:(size(start,2)-2*N)
	  if start(kk)==1&&start(kk-2)==0&&start(kk-3)==0&&start(kk-8)==0&&start(kk-16)==0
	  new_start(kk)=1;
	  end
	end

Block S4
max(RPtrig_org(jj,ii:ii+2*N));
if mean(start(ii-10:ii-2)) == 0 & I== 1
	Block S4
	count = 0;
	for ii = 1765:(size(new_start,2)-2*N)
	  if mean(new_start(ii-1764:ii-1))==0&&new_start(ii)==1
	  count = count + 1;
	  SIF = RP_pro(jj,ii:ii+N-1)'+SIF;
	  RP_syn_pro(jj,:) = RP_pro(jj,ii:ii+N-1);
	  end
	  if count == 1
	  break
	  end
	end

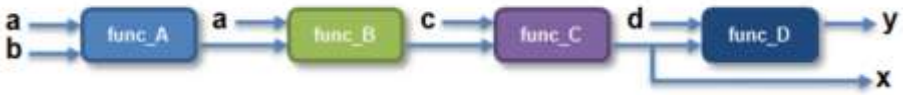
Block S5 count = count + 1; SIF = RP_org(jj,ii:ii+N-1)' + SIF;	Block S5 q = ifft(SIF);
Block S6 q = ifft(SIF/count);	Block S6 sif_pro(jj,:)=fft(q(size(q,1)/2+1:size(q,1)));
Block S7 sif_org(jj,:)=fft(q(size(q,1)/2+1:size(q,1)));	

2.3 Dataflow Accelerators in FPGA

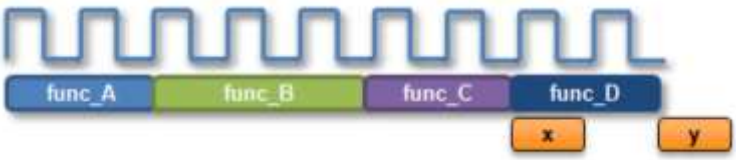
The SAR process is signal processing, which is divided into blocks as shown in Figure 3(b). Each block can be either a function block or an iteration block. Normally, these blocks operate in sequence. The predecessor block should be finished before starting the successor block. Such a behaviour is difficult to achieve real-time processing. Therefore, we apply dataflow accelerators to improve the throughput of the processing.

Figure 4 and 5 show the concepts of dataflow accelerators in FPGA. The dataflow accelerators are based on high-level synthesis techniques such as operation scheduling and pipelining [26,27]. The data dependent between functions is pre-processed to insert a pipeline to increase speed. Figure 4 shows the sequence process of four functions without the dataflow accelerators. Consider Figures 4(a) and 4(b), where there are dataflow relations in sequence functions. Without the concept of dataflow accelerators, Func_B waits for results from func_A. Func_C waits results from func_B. Func_D waits results from func_C. If the functions are scheduled in sequence as shown in Figure 4(c), the latency is nine clock cycles and the throughput is nine clock cycles. When the pipeline is applied, the next iteration of the four functions can start earlier. As a result, the latency is still nine clock cycles, but the throughput is now three clock cycles.

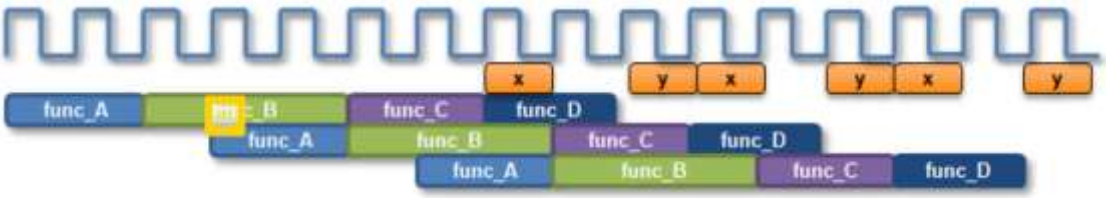
(a) All functions inside the loop for computation



(b) The sequence of computation inside the loop without dataflow accelerators



(c) Without dataflow accelerators, latency is 9 clock cycles and throughput are 9 clock cycles

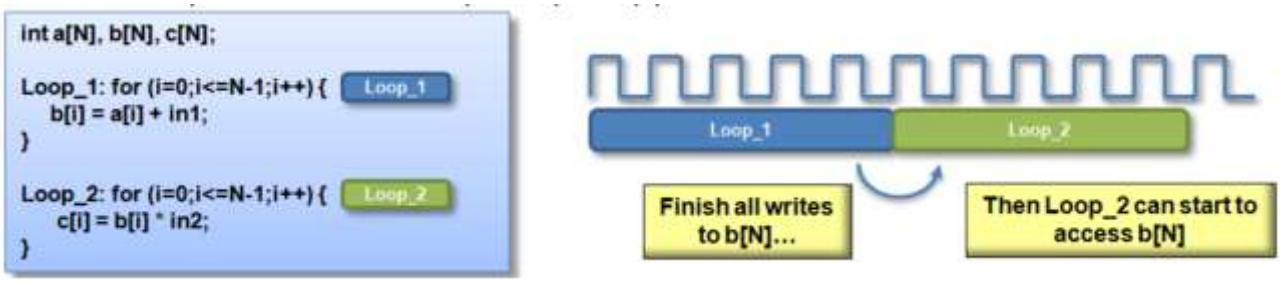


(d) With dataflow accelerators, latency is 9 clock cycles and throughput are 3 clock cycles

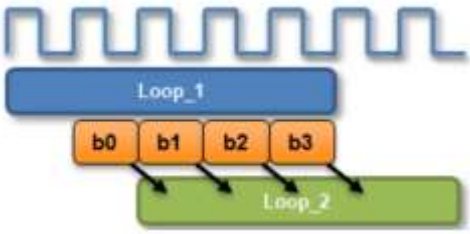
Fig. 4. Dataflow relations in functions [27]

Figure 5 shows the sequence process of four function with the concepts of dataflow accelerators in FPGA. Consider Figure 5(a), where there is a dataflow relation between iteration loops. In each block, iteration loops are repeated according to the number of incoming data samples. Then, the successor block will begin after the predecessor block is completed. Such a sequential process takes a lot of time. In this research, we propose a dataflow architecture to accelerate such data processing. Loop_1 and Loop_2 are consecutive loops. An argument of Loop_2 (c[i]) uses results from Loop_1

(b[i]). In other words, there is a dataflow from Loop_1 to Loop_2. Dataflow pipelining is applied to allow Loop_2 to start when data is ready. As a result, the throughput is improved, as shown in Figure 5(b).



(a) Loop functions and the sequence of computation for each loop without dataflow accelerators

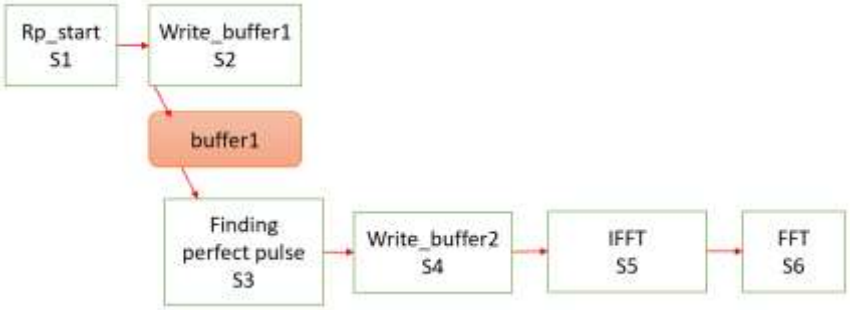


(b) The sequence of computation for each loop with dataflow accelerators

Fig. 5. Dataflow relations in iteration loops [27]

According to the concepts in Figures 4 and 5, the SAR process in Figure 3(b) can be scheduled as Figure 6. Block S2 has a conditional relation with S1. Only when S1 detects the start signal recording condition, block S2 starts recording the signal into the buffer. Blocks S2 and S3 have a dataflow relation. When block S2 starts recording the signal to buffer 1, which is implemented by a dual-port RAM that can read and write at the same time, block S3 can be scheduled to start as soon as block S2 starts recording the signal. Blocks S3 and S4 have the same conditional relation as blocks S1 and S2. Therefore, block S4 can only be started when S3 detects a complete pulse. Blocks S4 and S5 and blocks S5 and S6 are related to dataflow, but IFFT and FTT must wait for the signal samples to reach the size of IFFT and FTT to perform calculations.

(a) Sequential operations



(b) Operations scheduled with dataflow relation

Fig. 6. Dataflow schedule of Figure 3(b)

3. Results

We use MATLAB/Simulink to generate HDL code and simulate the proposed method on a Xilinx Zynq Z-7020 FPGA. We set the dataflow to our MBD for similar real-time processing as 1 by 1 input data. Then use M-file to generate an SLC image. Our experimental results show the sample of RP and RPtrig between the original method (S5) and purposed method (S4). The resources were used in the Register Transfer Level (RTL) code and a Xilinx Zynq Z-7020 FPGA. The critical data path estimation shows a delay of data that occurs inside our proposed method. The comparison of the average percentage error of our proposed method and the original method between M-file Simulink. The SLC image was established by data via our proposed method as well.

3.1 The Sample of RP and RPtrig of the Original Method and Purposed Methods

From the original method, the RP data of S5 is decreased depending on the condition of S3 and S4. The S3 and S4 of the original method depend on the RPtrig data. The sample of RP and RPtrig of the original method is shown in Figure 7(a). On the other hand, the RP data from S4 of the proposed method uses S3 as a trigger signal to buffering data in S4. The S3 of the proposed method generates the perfect pulse that depends on the RPtrig data. The sample of RP and RPtrig of the proposed method is shown in Figure 7(b). The proposed method is to select the RP data that matches the first perfect pulse of S3 and send it to S5 to reduce the number of computations.

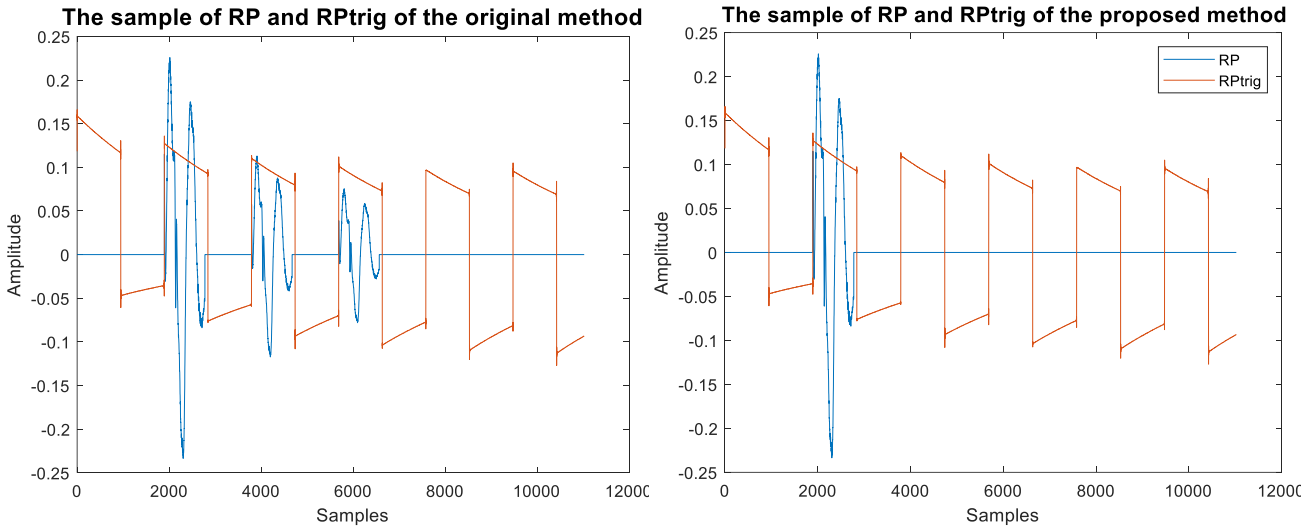


Fig. 7. The sample of RP and RPtrig (a) of the original method and (b) the proposed method

3.2 The Resource Usage Results in RTL Code and FPGA

The resource usage results in the RTL code from the HDL coder tool are shown in Table 2. The first column is a type of resource inside the proposed method which consists of Adder/Subtractors, Registers, Total 1 Bit Registers, RAMs, Multiplexers, I/O Bits and Dynamic Shift Operator. The second column is the total number of resources that were generated for our proposed method.

In addition, Table 3 shows the resource utilization on a Xilinx Zynq Z-7020 FPGA which specifically depends on the RTL code. The resources in RTL code were implemented on the resources inside a Xilinx Zynq Z-7020. Adder/Subtractors and Multiplexers were implemented by LUT that use 7.98% of all LUT in FPGA. Registers and Total 1 Bit Registers were implemented by Flip-Flops (FF) which uses

1.69% of all FF in FPGA. RAMs were implemented by LUTRAM and Block RAM (BRAM) that use 0.05% and 95.36% of all LUTRAM and BRAM in FPGA. I/O Bits were implemented by IO that use 1.60% of all IO in FPGA. Our proposed method must control dataflow by Trigger signal that uses Dynamic Shift operators in the RTL code to generate it. Therefore, the resources for Dynamic Shift operators were implemented by Global Clock Simple Buffer (BUFG) and Mixed-Mode Clock Manager (MMCM) using 9.38% and 25% of all BUFG and MMCM in FPGA.

Table 2
Resource usage results from the RTL code

Resources	Proposed method
Adder/Subtractors	58
Registers	28
Total 1-Bit Registers	337
RAMs	2
Multiplexers	912
I/O Bits	167
Dynamic Shift operators	14

Table 3
Resource utilization on a Xilinx Zynq Z-7020 FPGA

Resources	Available	Proposed method	Utilization (%)
LUT	53200	4247	7.98
FF	106400	1797	1.69
LUTRAM	17400	9	0.05
BRAM	140	133.5	95.36
IO	125	2	1.60
BUFG	32	3	9.38
MMCM	4	1	25.00

3.3 The Critical Path Estimation

Table 4 shows the critical path estimation of the proposed method. The critical path is the propagation delay occurring in each block path within the MBD. The first column shows the order of the block paths from fastest to slowest in the propagation delay in the block path occurring in the critical path. The second column shows the beginning of the propagation delay that occurs in the propagation delay of the block path. The third column shows the propagation delay of each block path in the MBD.

From the results, the proposed method with dataflow accelerators in the FPGA technique is modified by Dual Port RAM System1 and Dual Port RAM System2 which made a real-time dataflow (0 delays) in Dual Port RAM System2 and Abs1_outbuff block paths.

Table 4
Critical path estimation

Id	Propagation (ns)	Delay (ns)	Block Path
1	3.1425	3.1425	Dual Port RAM System1
2	3.1425	0.0000	Dual Port RAM System2
3	3.1425	0.0000	Abs1_outbuff
4	5.3595	2.2170	Compare To Zero_relop_outbuff
5	5.6025	0.2430	Logical Operator2
6	6.2710	0.6685	Logical Operator1
7	6.6880	0.4170	Switch16
8	7.1105	0.4225	Sum8

9	7.9815	0.8710	Relational Operator17
10	8.6555	0.6740	Logical Operator31
11	9.7960	1.1405	AND1
12	10.0390	0.2430	Logical Operator29
13	10.7130	0.6740	Logical Operator30
14	11.1300	0.4170	t1
15	26.9070	15.7770	Sum9_outbuff
16	29.1240	2.2170	Relational Operator16_outbuff
17	29.7980	0.6740	Logical Operator22
18	30.0410	0.2430	Logical Operator32
19	30.6565	0.6155	Logical Operator21
20	31.0735	0.4170	t
21	46.8505	15.7770	Sum11_outbuff
22	62.6275	15.7770	Sum12_outbuff
23	62.7915	0.1640	Unit Delay14

3.4 The Comparison of the Average Percentage Error

After implementing the proposed method on FPGA, we collect the results on each state from the proposed method. The results can be separated into 2 types. The first type is the results from FPGA and the second from Simulink. The results from FPGA consist of data from S1, S2 and S4. The results from Simulink consist of data from S5 and S6. Then we compare the results collected from M-file and Simulink. The results from the m-file are the original method and the proposed method. The results from Simulink are the proposed method and the FPGA board. Then we compare the results by average percentage error of the results between the original method, the proposed method and the FPGA with Eq. (1). The comparison of the average percentage error is shown in Table 5.

$$Average\ Percentage\ Error = \frac{\sum_{i=1}^{N}\left(\frac{|Measured\ Value_i - Original\ Value_i|}{|Original\ Value_i|}\times 100\right)}{N} \quad (1)$$

Table 5
The comparison of the average percentage error

Parameters	M-file	Simulink		
	Original and proposed	Original (M) and proposed	Original (M) and FPGA	Proposed and FPGA
S1	0	16.5004	16.5004	0
S2(RP)	0	0	0	0
S2(RPtrig)	0	0	0	0
S4	0.3531	0.3531	0.3531	0
Real S5	49.9963	49.9963	49.9963	0
ImaginaryS5	0.0028	0.0028	0.0028	0
Real S6	0.1263	0.1263	0.1263	0
ImaginaryS6	0.1301	0.1301	0.1301	0

In the case of M-file, the average percentage error will compare with the original method and the proposed method in the M-file. The average percentage error between the original method and the proposed method of S1 and S2 is 0%. In the S4, the original method is finding the position of the max value of the RPTrig and comparing it with the same position in the pulse of S3 and collecting the results. The proposed method is finding the perfect pulse in S3 and collecting the results. The S5 and S6 in the proposed method is the S6 and S7 in the original method. The average percentage error of S4 in M-file is 0.3531%. We compute the data after S4 with IFFT and FFT that transform the data to complex numbers. The average percentage error of the S5 and S6 should be calculated with the real

part and imaginary part. Ther average percentage error of the S5 is 49.9963% in the real part and 0.0028% in the imaginary part. The average percentage error of the S6 is 0.1263% in the real part and 0.1301% in the imaginary part. So, the average percentage error of S5 in the real part gives the most incorrect results when compared with the original method.

In the case of Simulink, the average percentage error will be compared with the original method in the M-file, the proposed method in Simulink and the FPGA. The average percentage error of S1 between the original method and the proposed method in Simulink is 16.5004%. The results from S2 are dependent on S1. But the average percentage error of S2 is 0% which proves the results from S2 are correct. And the average percentage error of S4, S5 and S6 are same as the average percentage error in M-file. So, our proposed method can give correct results when running in Simulink. Then we implement our proposed method on FPGA and collect the results for comparison between the original method in M-file, the proposed method in Simulink and the FPGA. In the case of the comparison of the average percentage error between the original method in the M-file and the FPGA, all average percentage errors are same as in all average percentage errors between the original method and the proposed method in Simulink. In the last case, we will compare the average percentage error between the proposed method in Simulink and the FPGA. All average percentage errors are 0. Therefore, the proposed method that is implemented on FPGA performs well and can also produce the same results as compared to the proposed method in Simulink.

3.5 The SLC Image after Pre-Processing Data of Original Method and Proposed Method

After pre-processing data with the original method and the proposed method, we use this data to process *via* an algorithm to generate an SLC image. The SLC image after pre-processing data with the original method and the proposed method are shown in Figure 8(a) and Figure 8(b). The cross range is the distance moving of the GB-SAR system. A dark blue colour which is -95 dB colour bar in the cross range is a blind spot. The down range is the distance between objects and GB-SAR system. The dB colour bar indicates the reflected object signal index. The higher value of -75 dB colour bar indicates that there is an object at that location. Therefore, the data after pre-processing data with the proposed method can be used as the data for processing in the RMA algorithm due to the SLC image in Figure 8(b) having the same significance as Figure 8(a).

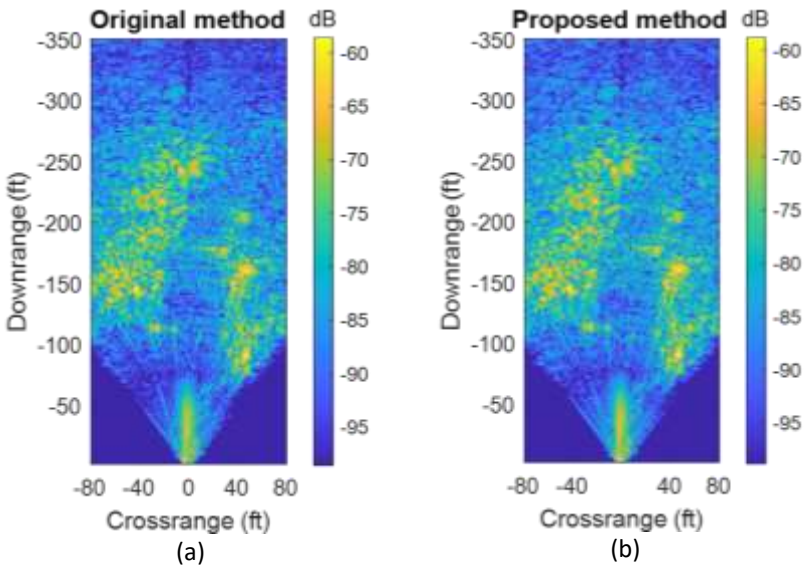


Fig. 8. The SLC image after pre-processing data with (a) the original method (b) the proposed method

4. Conclusions

This paper optimized the pre-processing data step using dataflow management with the MBD technique to improve the data flow from the original method. The original method has a waiting time and nested loop for computation that all made the longest time to generate an SLC image. Our proposed method improves the pre-processing step by reducing the waiting time and nest loops inside. Our proposed method can be implemented on FPGA with the HDL coder in Simulink. The data result after our proposed method can generate an SLC image. The target device for implementation is the Xilinx Zynq Z-7020 FPGA board.

For the optimization process, we proposed to reduce the state of the pre-processing data from 7 states to 6 states by collapsing S3-S4 of the original method into S3 of our proposed method. Because S3 and S4 of the original method are made a nested loop and don't support HDL coder for implementation on FPGA. This paper proposes the new state S3 without a nested loop. In our S3, we find the perfect pulse to select the best-reflected signal. We design the proposed method with the MBD technique which supports Simulink. Inside the proposed method, we select all components that support in HDL coder which can be implemented on FPGA. To prove the proposed method, we apply the proposed method to a M-file, Simulink and the FPGA. Then we compare the correctness of the outputs with the average percentage error between the original state (S1, S2, S5, S6, S7) and the proposed method (S1, S2, S4, S5, S6)

From the experimental results, the average percentage error of S1 is 0% in M-file and 16.5004% in Simulink between the original and proposed methods. By the way, the average percentage error of S2 is 0% which means the output from the proposed method is the same output from the original method. The average percentage error of S4 is 0.3531% in M-file and Simulink between the original and proposed methods but we can accept that value because we use a different method to select the best-reflected signal. The highest average percentage error has come from S5 in the real part but when focused on the final output the average percentage errors of S6 are 0.1263 and 0.1301 in the real and imaginary parts which can be accepted. For the implementation of FPGA, the proposed network on FPGA gives the same average percentage error between the original and proposed methods. The proposed network can perform on FPGA as well because all average percentage errors between the proposed method in Simulink and the FPGA are 0%. To prove our proposed method, we will use the output from S6 of the proposed method and S7 of the original method as data to generate the SLC images via an algorithm. The SLC image from the proposed method has the same significance data as the SLC image from the original method. Therefore, our proposed network can be instead of the original method because it gives a similar SLC image and our proposed method can perform on the Xilinx Zynq Z-7020 FPGA board as well.

In addition, the proposed method can reach faster computation speeds by implementing it on higher-performance devices. The pre-processing data with our proposed method is implemented on FPGA, but the algorithm for GB-SAR isn’t implemented on FPGA which made the speed of computation still slow with the computation between FPGA and PC. Therefore, the computation speed can be fast by integrating the pre-processing data and algorithm in high-performance FPGA.

Acknowledgement
This paper was funded by the Broadcasting and Telecommunications Research and Development Fund for Public Interest (BTFP), the National Broadcasting and Telecommunication Commission (NBTC) under grant number B2-074/1-63 and a grant from Prince of Songkhla University, Hat Yai, Songkhla, Thailand.

References

[1] Iglesias, Ruben, Xavier Fabregas, Albert Aguasca, Jordi J. Mallorqui, Carlos Lopez-Martinez, Josep A. Gili and Jordi Corominas. "Atmospheric phase screen compensation in ground-based SAR with a multiple-regression model over mountainous regions." *IEEE transactions on geoscience and remote sensing* 52, no. 5 (2013): 2436-2449. https://doi.org/10.1109/TGRS.2013.2261077

[2] Moldovan, Adrian-Septimiu, Ştefan-Adrian Toma, Valentin Ioan Poncos, Delia Cosmina Teleagă and Florin Şerban. "Outdoor measurements with ground based C-band synthetic aperture radar." In *2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pp. 3445-3447. IEEE, 2017. https://doi.org/10.1109/IGARSS.2017.8127739

[3] Hu, Cheng, Yunkai Deng, Rui Wang, Weiming Tian and Tao Zeng. "Two-dimensional deformation measurement based on multiple aperture interferometry in GB-SAR." *IEEE Geoscience and Remote Sensing Letters* 14, no. 2 (2016): 208-212. https://doi.org/10.1109/LGRS.2016.2635103

[4] Sato, Motoyuki. "2-D and 3-D near range SAR imaging." In *2017 IEEE Conference on Antenna Measurements & Applications (CAMA)*, pp. 157-160. IEEE, 2017. https://doi.org/10.1109/CAMA.2017.8273387

[5] Zoraya, Alfonso and Rosa Bolaños. "Implementing the LFM-CW MIT Radar at the Ecuadorian Space Institute: Some Results." *Journal of Aerospace Technology and Management* 12 (2020): e0520. https://doi.org/10.5028/jatm.v12.1091

[6] Charvat, Gregory L., Alan J. Fenn and Bradley T. Perry. "The MIT IAP radar course: Build a small radar system capable of sensing range, Doppler and synthetic aperture (SAR) imaging." In *2012 IEEE Radar Conference*, pp. 0138-0144. IEEE, 2012. https://doi.org/10.1109/RADAR.2012.6212126

[7] Li, Hongbin, Weike Feng and Pengcheng Wan. "Parameter-searched 2D orthogonal matching pursuit algorithm for sparse GB-SAR imaging." In *2021 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC)*, pp. 1-6. IEEE, 2021. https://doi.org/10.1109/ICSPCC52875.2021.9564732

[8] Zou, Lilong and Motoyuki Sato. "An efficient and accurate GB-SAR imaging algorithm based on the fractional Fourier transform." *IEEE Transactions on Geoscience and Remote Sensing* 57, no. 11 (2019): 9081-9089. https://doi.org/10.1109/TGRS.2019.2924803

[9] Yang, Yue, Yihe Wan and Qun Wan. "A 2D-PRM-Based Atmospheric Phase Correction Method in GB-SAR Interferometry Application." *IEEE Sensors Letters* 7, no. 6 (2023): 1-4. https://doi.org/10.1109/LSENS.2023.3276784

[10] Feng, Weike, Giovanni Nico and Motoyuki Sato. "GB-SAR interferometry based on dimension-reduced compressive sensing and multiple measurement vectors model." *IEEE Geoscience and Remote Sensing Letters* 16, no. 1 (2018): 70-74. https://doi.org/10.1109/LGRS.2018.2866600

[11] De La Cruz, Luis and Marco A. Milla. "Comparison of GB-SAR imaging algorithms for a landslide monitoring application." In *2020 IEEE XXVII International Conference on Electronics, Electrical Engineering and Computing (INTERCON)*, pp. 1-4. IEEE, 2020. https://doi.org/10.1109/INTERCON50315.2020.9220189

[12] Li, Zhe, Jian Wang and Qing Huo Liu. "Interpolation-free Stolt mapping for SAR imaging." *IEEE Geoscience and Remote Sensing Letters* 11, no. 5 (2013): 926-929. https://doi.org/10.1109/LGRS.2013.2281847

[13] Huai, Yuanyuan, Yi Liang, Jinshan Ding, Mengdao Xing, Letian Zeng and Zhenyu Li. "An inverse extended Omega-K algorithm for SAR raw data simulation with trajectory deviations." *IEEE Geoscience and Remote Sensing Letters* 13, no. 6 (2016): 826-830. https://doi.org/10.1109/LGRS.2016.2548240

[14] Li, Zhongyu, Junjie Wu, Qingying Yi, Yulin Huang and Jianyu Yang. "An Omega-$ k $ Imaging Algorithm for Translational Variant Bistatic SAR Based on Linearization Theory." *IEEE Geoscience and Remote Sensing Letters* 11, no. 3 (2013): 627-631. https://doi.org/10.1109/LGRS.2013.2272755

[15] Wu, Junjie, Zhongyu Li, Yulin Huang, Jianyu Yang and Qing Huo Liu. "A generalized Omega-K algorithm to process translationally variant bistatic-SAR data based on two-dimensional Stolt mapping." *IEEE Transactions on Geoscience and Remote Sensing* 52, no. 10 (2014): 6597-6614. https://doi.org/10.1109/TGRS.2014.2299069

[16] Guo, Shouchang and Xichao Dong. "Modified Omega-K algorithm for ground-based FMCW SAR imaging." In *2016 IEEE 13th International Conference on Signal Processing (ICSP)*, pp. 1647-1650. IEEE, 2016. https://doi.org/10.1109/ICSP.2016.7878107

[17] Hai, Jerry Chan Ting, Ooi Chee Pun and Tan Wooi Haw. "Accelerating video and image processing design for FPGA using HDL coder and simulink." In *2015 IEEE Conference on Sustainable Utilization And Development In Engineering and Technology (CSUDET)*, pp. 1-5. IEEE, 2015. https://doi.org/10.1109/CSUDET.2015.7446221

[18] Titri, Sabrina, Cherif Larbes and Kamal Youcef Toumi. "Rapid prototyping of PVS into FPGA: From model based design to FPGA/ASICs implementation." In *2014 9th International Design and Test Symposium (IDT)*, pp. 162-167. IEEE, 2014. https://doi.org/10.1109/IDT.2014.7038606

[19] Othman, Norliza, Mohamad Hairol Jabbar, Abdul Kadir Mahamad and Farhanahani Mahmud. "Luo Rudy Phase I excitation modeling towards HDL coder implementation for real-time simulation." In *2014 5th International Conference on Intelligent and Advanced Systems (ICIAS)*, pp. 1-6. IEEE, 2014. https://doi.org/10.1109/ICIAS.2014.6869495

[20] Sopajarn, Jerawat, Apidet Booranawong, Surachate Chumpol and Nattha Jindapetch. "Vehicle Following Control via V2V SIMO Communications Using MBD Approach." *IEEE Access* 11 (2023): 124252-124264. https://doi.org/10.1109/ACCESS.2023.3330154

[21] Othman, Norliza, Farhanahani Mahmud, M. Hairol Jabbar and Nur Atiqah Adon. "Cardiac excitation modeling: HDL coder optimization towards FPGA stand-alone implementation." In *2014 IEEE International Conference on Control System, Computing and Engineering (ICCSCE 2014)*, pp. 507-511. IEEE, 2014. https://doi.org/10.1109/ICCSCE.2014.7072771

[22] Besbes, Moez, Salim Hadj Saïd and Faouzi M'Sahli. "FPGA implementation of high gain observer for induction machine using Simulink HDL coder." In *2015 3rd International Conference on Control, Engineering & Information Technology (CEIT)*, pp. 1-6. IEEE, 2015. https://doi.org/10.1109/CEIT.2015.7233064

[23] Solod, Panadda, Nattha Jindapetch, Kiattisak Sengchuai, Apidet Booranawong, Pakpoom Hoyingcharoen, Surachate Chumpol and Masami Ikura. "High Level Synthesis Optimizations of Road Lane Detection Development on Zynq-7000." *Pertanika Journal of Science & Technology* 29, no. 2 (2021). https://doi.org/10.47836/pjst.29.2.01

[24] Chumpol, Surachate, Panadda Solod, Krerkchai Thongnoo and Nattha Jindapetch. "Model-Based Design Optimization using CDFG for Image Processing on FPGA." *ECTI Transactions on Computer and Information Technology (ECTI-CIT)* 17, no. 4 (2023): 479-487. https://doi.org/10.37936/ecti-cit.2023174.252417

[25] Cheng, Shaoyi and John Wawrzynek. "High level synthesis with a dataflow architectural template." *arXiv preprint arXiv:1606.06451* (2016).

[26] Xilinx, A. M. D. "Vitis High-Level Synthesis User Guide (UG1399)." *https://docs. xilinx. com/r/en-US/ug1399-vitis-hls* (2023).

[27] Xilinx. "Improving Performance." (2016). https://slidetodoc.com/improving-performance-this-material-exempt-per-department-of

[28] MathWorks. "Limitations for MATLAB Loop Optimization." *HDL coder user guide*, (2023): pp.8-30, https://www.mathworks.com/help/hdlcoder

[29] Audacity: Free, Open-Source Audio Editing Software. "A yellow and orange waveform between the ears of a set of blue headphones." https://www.audacityteam.org