Dynamic Configurable Floating-Point FFT Pipelines and Hybrid-Mode CORDIC on FPGA

Jie ZHOU  Yazhuo DONG  Yong DOU  Yuanwu LEI
Department of Computer Science, National University of Defense Technology
Changsha, Hunan, P.R.China
{zhoujie,dongyazhuo,yongdou,yuanwulei}@nudt.edu.cn

ABSTRACT
Floating-point Fast Fourier Transform (FFT) processor and COordinate Rotation Digital Computer (CORDIC) element play important roles in communication and radar applications. But even with the rapid development of large-scale integrated circuit, it is usually impractical to implement these floating-point computations on FPGA, as they will consume a large amount of chip resources. In this paper, a compact SAR processor, composed of four 1D FFT-PEs (processing elements) and a CORDIC co-processor, is implemented on FPGA. In particular, a dynamic configurable pipeline is used in FFT-PE to reduce the area consumption through reusing floating-point units. And the 32-bit floating-point hybrid-mode CORDIC co-processor is implemented to generate compensation factors and compute transcendental functions in SAR image visualization phase. Experimental results show that our SAR processor performs well both in area and latency. It consumes about 40% of LUTs and DSPs, and about 48% of memory bits on a StratixII FPGA, while the 32-bit floating-point hybrid-mode CORDIC co-processor uses only about 2.6% LUTs and Registers of Virtex5 and achieves a frequency of 217MHz. For the largest SAR image, 4K*4K in our experiment, it takes 1232.6ms to transform the SAR raw data into a visible image of 256 grey levels.

1. INTRODUCTION
FFT and CORDIC processors, as two key building blocks, are widely used in wireless and multimedia communication applications, such as OFDM [3], MIMO [7], CDMA [11] [1]. It is also employed in radar applications, as SAR data processing is a representative instance [2] [4]. In these systems, computations of floating-point, rather than fix-point, should be developed to improve the calculation precision. FPGA has become a promising option for accelerating critical applications in fields of wireless and multimedia communication in the past few decades. Previously, FPGAs have excelled in fixed-point computation, but were unable to effectively perform floating-point computations due to their limited size [6]. However, with the rapid development of large-scale integrated circuit technology, FPGA chips have gradually been used to accelerate scientific applications, which involve plenty of floating-point operations. Consequently, there are lots of studies on implementation of floating-point units on FPGA, such as [10] [8] [5] [12]. Recent researches [14] [13] also show that more than forty 64-bit IEEE floating-point arithmetic units can be implemented in a single FPGA chip. However, it is challenging to implement large floating-point applications, such as massive parallel SAR system, on FPGA, as they will consume large amount of chip resources [6].

In this paper, we design a compact floating-point SAR processor on FPGA with small area consumption and low latency. As the primary component, FFT-PE is implemented with a dynamic configurable pipeline, which results in a dramatic area reduction by reusing some floating-point units. In addition, a 32-bit floating-point hybrid-mode CORDIC co-processor is implemented to generate compensation factors (employing sin/cos) and compute transcendental functions in SAR image visualization phase. Furthermore, a ping-pong SDRAM structure is used to promote the peak bandwidth of external memory.

Experiments on Virtex5 show that our SAR processor, with four FFT-PEs, consumes about 40% of LUTs and DSPs, and about 48% of memory bits on a StratixII FPGA, while the 32-bit floating-point hybrid-mode CORDIC co-processor uses only about 2.6% LUTs and Registers of a Virtex 5 FPGA and achieves a frequency of 217MHz. For the largest SAR image, 4K*4K in our experiment, it takes 1232.6ms to transform the SAR raw data into a visible image of 256 grey levels.

The remainder of this paper is organized as follows. Section 2 gives the architecture of our SAR processor and its computational core FFT-PE. In section 3, the dynamic configurable pipeline of FFT-PE is elaborated and a hybrid-mode CORDIC co-processor is presented. Experimental results are given in section 4. And Section 5 concludes the work.

2. FPGA SAR PROCESSOR STRUCTURE
3. ARITHMETIC UNITS REUSING AND HYBRID-MODE FLOATING-POINT CORDIC CO-PROCESSOR

Figure 1: The structure of FPGA SAR processor.

CS algorithm [9], one of the most mature and frequently used SAR imaging algorithms, consists of multiple FFT/IFFT operations along the range/azimuth directions and factor compensations. In implementation of the SAR processor, we partition the CS algorithm into five phases: (1) Azimuth FFT and CS factor multiplication; (2) Range FFT and compression; (3) Range IFFT, azimuth filter, and phase residual; (4) Azimuth IFFT; (5) SAR image visualization.

Figure 1 shows the hardware architecture of the SAR processor. The SAR raw data, intermediate data and final results are stored in SDRAM 1 and SDRAM 2. And the initial data of compensational factor calculation are store in RAM F, which is the internal memory of FPGA. Main Controller is used to harmonize the processor running orderly. SDRAM Controller 1 and Controller 2 give the Host and computational modules of SAR processor access to SDRAM. And most computations, such as FFT/IFFT and factor compensation, are performed in FFT-PEs. Finally, the Image Generator visualize the SAR image into 256 gray levels.

FFT-PE, the most important calculation element in SAR processor, shown in figure 2, is to accomplish a series of 32-bit floating-point operations, such as FFT/IFFT, compensation factors generation, and factor compensation operations. The core computational unit is Complex Arithmetic Pipeline, which is organized as dynamic configurable pipeline, and a detailed depiction is presented in section 3. FFT-PE is equipped with a ping-pong memory structure, RAM1 and RAM2, served as input/output buffer to load new data and store the results in parallel. Each has two 64-bit ports, through which two complex numbers can be accessed simultaneously. Intermediate results can also be kept in double RAMs. To get the sequential results of FFT/IFFT, the BitRevOrder unit reorders the raw data before storing them into RAM1 or RAM2. All compensation factors, including CS factor, migration correction factor and azimuth factor, are generated in Compensation Factor Pipeline. And the twiddle factors used in FFT/IFFT are stored in Twiddle Factor ROM. In addition, all modules are assisted by the FFT-PE Controller to accomplish tasks assigned by Main Controller orderly.

Large SAR system requires a great deal of floating-point operations, especially floating-point additions and multiplications. Compared with fixed-point units, floating-point units consume larger number of FPGA resources, and thus measures must be taken to reduce the area consumption. With this in mind, we manage to reuse floating-point units in our SAR processor.

In addition, we design and implement a 32-bit floating-point hybrid-mode CORDIC co-processor to accomplish the transcendental functions, such as log, sqrt, and so on, in phases of compensation factors generation and SAR image visualization.

3.1 Floating-Point Units Reusing
The butterfly calculation in FFT/IFFT and factor compensation operation are not executed simultaneously, so we organize Complex Arithmetic Pipeline as dynamic configurable pipeline to reuse floating-point addition and multiplication units through the following two approaches.

1. FFT/IFFT and factor compensation
As shown in figure 3, paths of (1), (2) and (3) perform the butterfly calculation in FFT/IFFT. Notably, path (2), the unit of floating-point complex multiplication, is reused to implement the factor compensation operation, so that four floating-point multiplication units and two floating-point addition units are spared.

2. FFT and IFFT
Complex IFFT algorithm can be performed in the following phases: 1) Conjugate the raw data; 2) FFT calculation; 3) Conjugate and divide N (the dimension of IFFT). That is to say, IFFT can be implemented by adding some logic ele-

Figure 2: The structure of FFT-PE.

Figure 3: The pipeline of Complex Arithmetic.
ments to FFT circuit. To simplify the controlling logic, we process range IFFT and azimuth IFFT in different ways.

(a) Range IFFT, azimuth filter and phase residual

During the processing of the range IFFT, azimuth filter and phase residual, the raw input data are firstly conjugated in BitRevOrder unit and then FFT is performed. And the operation of dividing N is done during compensation factor calculating. Equation (1) is complex multiplication operation. And equation (2) shows the combination operation of conjugation and complex multiplication.

\[(a + ib) \ast (c + id) = (ac - bd) + i(ad + bc) \quad (1)\]

\[(a - ib) \ast (c + id) = (ac + bd) + i(ad - bc) \quad (2)\]

Compare equation (1) with (2), conjugation and complex multiplication can be merge into one operation though changing the addition/subtraction flag of floating-point addition units in complex multiplication element as shown in figure 3.

(b) Azimuth IFFT

Just as the first two steps of range IFFT, azimuth filter and phase residual, the raw input data are conjugated in BitRevOrder and then FFT is performed. Dividing N and conjugating operation is completed by the Division and Conjugate unit. The design is simplified by only subtracting a value of logN from the exponential part of floating-point number instead of a 32-bit floating-point division unit because integer N is a power of 2.

3.2 Hybrid-Mode Floating-Point CORDIC Co-processor

To generate the compensation factors and visualize the SAR image into bmp format, we implement a hybrid-mode 32-bit ANSI/IEEE Std754 floating-point CORDIC co-processor on FPGA. This is a multifunctional computing co-processor, which can accomplish the computations of lots of floating-point transcendental functions, such as trigonometric function, natural exponential and natural logarithm, etc.

The basic CORDIC iteration equations are [13]:

\[
\begin{align*}
X_{i+1} & = K_i(X_i - m\sigma_i2^{-S(m,i)}Y_i) \\
Y_{i+1} & = K_i(Y_i + \sigma_i2^{-S(m,i)}X_i) \\
Z_{i+1} & = Z_i - \sigma_i\alpha_{m,i} \\
\sigma_i & \in \{-1, 1\}, i = 0, 1, \ldots, n - 1
\end{align*}
\]

where the coordinate parameter m defines the coordinate system (circular, linear, and hyperbolic coordinate for m equal to 1, 0, and -1 respectively). The rotation angle is \(\alpha_{m,i} = m^{-1/2}\tan^{-1}(m^{1/2}/2^{m})\) and the scaling factor is \(K_m = \prod_{i} K_i = \prod_{i} \sqrt{1 + m2^{-2S(m,i)}}\). The rotation sequence \(S(m,i)\) is defined as follows:

\[
S(m,i) = \begin{cases} 
0, 1, 2, 3, 4, 5, 6 \ldots n, & m = 1, m = 0 \\
1, 2, 3, 4, 5, \ldots, 13, 13, 14, \ldots, 40, 40, 41, & m = -1
\end{cases}
\]

As shown in figure 4(a), the hybrid-mode CORDIC co-processor includes three modules: pre-processor, CORDIC processor and post processor. Pre-processor transforms the inputs of floating-point format into fixed-point format and expands...
the convergence range into all 32-bit ANSI/IEEE Std754 floating-point data. All the computation of CORDIC functions is accomplished in the CORDIC processor. And the post-processor transforms the output from fixed-point format into standardized ANSI/IEEE Std 754 floating-point format. Sub-figure (b) in figure 4 is the pipeline of data path X and Y, and (c) is Z’s. Sub-figure (d) in figure 4 and figure 5 show the pipelines of post-processor and the pre-processor respectively. In more details, the stages are arranged as follows:

S1: Converting the floating-point format of inputs into fixed-point format.

S2-S3: Scale factor operation and angle mapping in argument reduction.

S4-S5: Calculation mapped angle value in argument reduction.

S6: Selecting the inputs of the CORDIC process module according to the mode selecting sign Mode_sel and the coordinate selecting sign CO_sel.

S7-S14: The first part of CORDIC process phase. Each stage of these pipelines accomplishes an iteration of CORDIC.

S15-S31: The second part of CORDIC process phase. Each stage of these pipelines also accomplishes an iteration of CORDIC. The dissimilarity from S7-S14 is that each stage only needs to calculate in X and Y data path, while the rotation direction is recoded to compute the final value Z25 in Z data path.

S32: Finishing the calculation of corresponding functions.

S33: Counting leading zeros.

S34: Normalization of outputs.

4. EXPERIMENTS

We have implemented a floating-point SAR processor on FPGA. All modules in experiments are coded in Verilog, and synthesized with QuartusII 6.0 and Xilinx ISE 9.1. Moreover, we tested the performance of FPGA SAR processor in experiment 3.

The first experiment presents the resources usage and clock frequency of SAR processor and its main modules synthesized with Altera QuartusII 6.0 using FPGA of StratixII EP2S180F120C5. As shown in table 1, SAR processor with four FFT-PEs occupies 40% ALUTs and 50% memory bits and reaches the clock frequency of 124MHz. The requirement of memory bits is increasing with SAR scale, therefore the internal memory resources of FPGA will be one of constrained factors to the scalability of FPGA SAR processing.

In experiment 2, we synthesized the hybrid-mode floating-point CORDIC co-processor in FPGAs of StratixII and Virtex5 respectively and the results are summarized in Table 2. As shown in Table 4, the 32-bit floating-point hybrid-mode CORDIC co-processor occupies a little area of FPGA, and reaches the clock frequency of 195MHz and 217MHz on StratixII and Virtex5. The co-processor can meet the requirement of SAR imaging and moreover can be used to accelerate the computation of scientific.

Experiment 3 gives the performance and speedup of different sizes of SAR images with different FFT-PE configurations as shown in table 3. Regarding the performance, we test their true processing time. And the speedup is relative value to one FFT-PE. It takes 1232.6ms to transform the SAR raw data to visible image with 256 grey levels and achieves the

<table>
<thead>
<tr>
<th>Table 1: FPGA Resource Usage.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resource</td>
</tr>
<tr>
<td>Controller</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>(8.3%)</td>
</tr>
<tr>
<td>Memory bit</td>
</tr>
<tr>
<td>(9.8%)</td>
</tr>
<tr>
<td>DSP Block</td>
</tr>
<tr>
<td>(6%)</td>
</tr>
<tr>
<td>Clock (MHz)</td>
</tr>
</tbody>
</table>

Figure 5: Pre-Process.
speedup of 3.61. Furthermore, the speedup is increasing with the scale of SAR data matrix. Since our SAR processor is more suitable for large SAR imaging.

5. CONCLUSION
This paper presented the design and implementation of FPGA SAR processor as an instance to show two key floating-point building blocks - FFT and CORDIC - applying in the fields of radar and communication. And we chiefly introduced the dynamic configurable pipeline of FFT processing element and pipelines of hybrid-mode floating-point CORDIC co-processor. Experimental results showed that our FPGA SAR processor only takes 1232.6ms to transform the SAR raw data into a visible image with 256 grey levels. Synthetic SAR processor as an instance to show two key floating-point building blocks - FFT and CORDIC - applying in the fields of radar and communication. And we chiefly introduced the dynamic configurable pipeline of FFT processing element and pipelines of hybrid-mode floating-point CORDIC co-processor. Experimental results showed that our FPGA SAR processor only takes 1232.6ms to transform the SAR raw data into a visible image with 256 grey levels. Synthetic results also showed that 32-bit floating-point hybrid-mode CORDIC co-processor only occupies about 2.6% LUT and Reg of Virtex5 FPGA and reaches the clock frequency of 217MHz and moreover can be used to accelerate the scientific computations.

6. ACKNOWLEDGEMENT
This work is supported by NSFC of China under the grant NO. 60633050 and 60621003.

7. REFERENCES

| Table 2: Synthesis results of 32-bit floating-point hybrid-mode CORDIC co-processors. |
|-------------------------------|-----------|-----------|-----------|-----------|-----------|
| FPGA  | Pre-Process | CORDIC-Process | Post-Process | Total | Frequency (MHz) |
|       | LUT | Reg | LUT | Reg | LUT | Reg | LUT | Reg |
| StratixII | 2135 | 2244 | 2776 | 2522 | 1008 | 558 | 6469(4.5%) | 5372(3.7%) | 195.08 |
| Virtex5 | 2039 | 2057 | 1947 | 2531 | 1211 | 569 | 5412(2.6%) | 5130(2.5%) | 217.24 |

<p>| Table 3: Performance results and speedup. |
|-------------------------------|-------------------|</p>
<table>
<thead>
<tr>
<th>PE</th>
<th>Run Time(ms) / Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Num</td>
<td>64*64</td>
</tr>
<tr>
<td>1</td>
<td>1.19/1</td>
</tr>
<tr>
<td>2</td>
<td>0.68/1.75</td>
</tr>
<tr>
<td>4</td>
<td>0.53/2.24</td>
</tr>
</tbody>
</table>