Chenguang Fan, Muyao Li, Wenzhong Liu and Jingjing Cheng

Preparation of Articles for IEEE TRANSACTIONS and JOURNALS(2021)

***Abstract*—Geophysical inversion in well logging is a core part of data processing. The downhole high-temperature environment is harsh, and the hardware resources of the logging instrument are limited, so the algorithm real-time and inversion accuracy need to be considered. We apply an accelerated projected gradient descend (APGD) algorithm to solve the inverse problem. APGD inversion algorithm acceleration is implemented in FPGA and the pulsating array architecture and distributed structure are used to reduce the operation time of matrix multiplication. To design a multicore processing mechanism in FPGA, we use a multicore architecture based on task migration dynamic thermal management technique, thus balancing the temperature difference between local hotspots on the chip. By comparing the algorithm time consumption under different platforms, this method proves to be more efficient in execution and more accurate, which provides a useful reference for geophysical inversion work in oil logging.**

***Index Terms*—geophysical inversion; accelerated projection gradient descent algorithm; field programmable gate array; dynamic thermal management**

# I. INTRODUCTION

T

he goal of geophysical observations is to determine geological structures from geophysical data [1-2]. Use a simple model to approximate the real geology and try to determine the model parameters from the data, which is called a geophysical inverse problem [3-5]. With the development of theory and the event of small parallel computers, geophysical inversion techniques have become feasible for the oil logging industry [6]. This problem becomes common in the integrated interpretation of geophysical data. Inversion model is usually described as  , where the coefficient matrix  of  is related to the instrument parameters and formation information. The  vector  is the observed data measured by the instrument. The  vector  is the basic physical parameter to be inverted. Suppose that the observed data are linear functions of the basic parameters in the model, the model  describes a linear mapping from  to [7].

The model is an overdetermined equation when , there is no exact solution to the system of equations. The solution obtained by the inversion algorithm is essentially to find the optimal solution of the model parameters in the functional space under the constraints of the observed data, as the dense overdetermined system of linear equations may be solved via the least squares method [8]. The basic parameters are restricted to non-negative values, thus reflecting the real prior formation information, and then the non-negative least squares (NNLS) problem is considered. Numerical methods for solving NNLS problems are developing rapidly. The active set (AS) method [9]was the first NNLS algorithm widely used in inverse problems. Since the method can only handle one constraint per iteration, it is computationally inefficient for large-scale data. The reflective Newton (RN) method[10] is a special case of NNLS inversion for minimization of quadratic functions with upper and lower bounds. The projected gradient descent (PGD) method [11] is a constrained extension of gradient descent, also known as steepest descent. PGD method shows good performance on large-scale problems. The projected quasi-Newton (PQN) method [12] ensures convergence by updating the variable separation and projection of the solution in each iteration. Compared with the PGD method, the complexity per iteration of PQN is higher. Among these methods, the PGD method is a classic method for solving NNLS problems. The algorithm has many advantages, including fast convergence, high accuracy of results, and a clear and simple structure, which makes it suitable for deployment in logging while drilling downhole embedded equipment to complete rapid inversion working.

Driven by the development needs of the logging while drilling tools, we designed a multi-core architecture based on APGD (Accelerated Projected Gradient Descent) for efficient low-power geophysical inversion algorithms where the numerical solution of geophysical inverse problems is realized in the logging while drilling tools.

Section 2 presents the APGD algorithm. Section 3 describes the multicore architecture of the FPGA system, the systolic array technology, the distributed algorithm (DA) and the dynamic thermal management technology based on the built-in temperature sensor to balance the temperature difference between the local hot spots of the processor. Section 4 introduces the experiments and data analysis.

# II. APGD algorithm

It is shown that the complexity of the APGD algorithm:  . The APGD method is less complex and has significant advantages in solving large-scale problems [13].

The NNLS problem is defined：



Given a  matrix  , its rank  , and given a real  vector  , find a real  vector that minimizes the Euclidean length. Influenced by some model-specific physical factors that explicitly enforce non-negativity in its results, this constrained least squares method is called NNLS [14], and its objective function：



This work applies the projected gradient descend algorithm to solve the NNLS problem, set  ,  :





where iteration  can be set as  , and  is the Lipschitz constant of  . The next slide shows  . Expanding (4), we can get:







where  is the number of iterations,  is the result of each iteration, and  is the inversion result when the stop iteration condition is satisfied.

APGD is a combination of Nesterov's accelerated algorithm and projected gradient descent, and makes the algorithm monotonic by adopting adaptive reset parameters. It’s divided into two parts. The main part executes the APGD algorithm. The equation is as follows:







The projected gradient step is (7), the calculation process of Nesterov's acceleration parameter is (8), and the accelerated extrapolation calculation is (9).

Switches the PGD algorithm to update and reset the data and parameters when the residual value increases.



The APGD algorithm integrates PGD and Nesterov's acceleration method, improves the drawbacks of increased residuals, and solves the non-negative constrained least squares problem. The algorithm has fast convergence speed, clear and simple architecture, and easy operations. It is suitable for deployment in embedded devices with limited hardware resources.

# III. FPGA Multicore Processor Architecture

## A. Overall Structure

The FPGA is designed with a multicore processing mechanism, divided into four cores, and a set of APGD inversion algorithms and built-in temperature sensors are deployed in each core. The temperature sensor maps around the algorithm to measure the temperature of each core and compare it with the highest temperature threshold. If the current core temperature is low, the core continues to work, and if the temperature is high, the multicore thermal management system switch task is triggered and the multicore rotates to enable active dynamic thermal management. If the temperature is higher, the multicore thermal management system is triggered to perform task switching and multicore rotation to achieve active dynamic thermal management. Figure 1 shows the overall functional structure of FPGA, which is divided into APGD algorithm unit, temperature measurement Unit and multicore temperature management unit according to the functional requirements.

The APGD algorithm based on FPGA should consider hardware resource consumption and code execution efficiency. This design uses the systolic array architecture to improve the overall execution efficiency of the algorithm, and introduces distributed algorithms, split LUT techniques and parameter simplification operations to reduce the operation time of matrix multiplication. Therefore, the module is divided into three modules, namely the main process module, LUT module and matrix calculation module. The main process module is responsible for the main logical flow of the algorithm, combining matrix variables into addresses for lookup tables and sending the inversion results to the master control system after the algorithm is completed; the LUT module stores the pre-computed partial products of matrix multiplication and the results of the acceleration parameters . The partial products need to be computed subsequently and the results of the acceleration parameters are sent to the APGD module; The matrix calculation module shifts and accumulates the partial products to obtain the matrix multiplication results, and returns them to the APGD module to participate in the algorithm calculation. The system modules are connected by databus and control signals to strictly ensure the execution steps and sequential logic of the algorithm.

The temperature measurement unit is designed as a delay line digital temperature sensor based on a ring oscillator, which is integrated by user-written logic and hardware. It can be laid out anywhere on the chip to achieve temperature measurement for different cores with fewer hardware logic resources.

The multicore temperature management unit receives the temperature value of each core and compares it with the maximum threshold, controls the operation of the APGD algorithm and the temperature sensor. It stops the APGD algorithm of the current core when the temperature is high, switches the other cores to work, and implements a dynamic thermal management technique based on the task migration method to balance the temperature difference between local hot spots.



**Fig. 1.** FPGA overall functional structure diagram.

## B. Systolic Array Design

The systolic array essentially replaces the original hardware design of long broadcast [15], multi-fan-in/fan-out data links with short data links between processing units (PEs) with a certain depth pipeline, where a fixed simple function is computed in each PE and each PE is only connected to other PEs in close proximity. During execution, each PE can selectively read data from its neighboring PEs and store the computed results in its local memory without repeated data entry, and only the boundary PEs of the array can communicate with the outside world [16].

After analysis, there are two equations in the APGD algorithm that can be calculated using the systolic array. This equation is the extrapolation calculation of the APGD algorithm,  is the projection result column vector,  and  are the input column vectors,  is the acceleration parameter, and  denotes the number of current iterations as shown in (11). This equation is the residual calculation as (12), and the 2-norm of  column vectors is calculated. The extraction of square root at the end of the equation is not involved in the pulsating array.





The data in the two equations are one-dimensional vectors, so a one-dimensional linear pulsating array is used [17]. The standard model is shown in Figure 2. It shows that the PEs on the left and right edges are connected to the input or output data. The PEs in the middle only have data transfer with their own PEs adjacent to the left and right, and the intermediate data do not pass through the memory. Generates one data flow every clock cycle until the flow reaches the last PE to output the overall data.



**Fig. 2.** Linear systolic array diagram.

As shown in Figure 3 is the circuit logic diagram of the extrapolation equation, the operation unit consists of subtractor, adder and multiplier. The input is two data, and the output data is  ,  is a fixed value can be pre-loaded into the PE. As shown in Figure 4 is the circuit logic diagram of residual calculation, the operator unit consists of adder and multiplier. The input is two data and the output data is  .



**Fig. 3.** Extrapolation equation circuit logic diagram.



**Fig. 4.** Residual equation circuit logic diagram.

After the PE unit is designed, it is necessary to build a systolic array circuit according to the overall process of the algorithm. Analysis of the algorithm process shows that there is no sequential logical relationship between the variables in the extrapolation equation and the residual equation. In order to improve the efficiency of the algorithm, two sets of systolic arrays are executed in parallel. The logic circuit of the systolic array is shown in Figure 5.

Other processes of the algorithm output the address of the lookup table, obtain the column vector of the matrix multiplication result through the lookup table, and then calculate the two equations through the systolic array. Different vector sizes correspond to different numbers of systolic arrays PE. The last PE outputs the equation result and returns to the main process module for other operations.



**Fig. 5.** APGD systolic array logic circuit.

The systolic array architecture in FPGAs accelerates the main processing modules. Due to the FPGA's ability to process data in parallel, this architecture allows for fast algorithms in embedded devices. Its overall structure is simple, regular and has a high degree of modularity.

## C. Distributed Algorithm Implementation

There are a lot of matrix and column vector multiplications in APGD algorithm. In this study, the distributed algorithm [18] is used to complete the fast computation of multiple cumulative multiplication and addition to improve the computational efficiency. The distributed algorithm obtains the partial product result through the LUT, and then accumulates and shifts the result of the LUT to obtain the algorithm result. A standard cumulative addition operation is (13).



Where  is a constant,  is the input variable, and  is the result of the cumulative multiplicative addition calculation.



where  denotes the b-th bit of  and  is the sign bit of  . Converting standard multiplication and addition to LUT operations and accumulation is (14). The bits of the input variables are used as addresses for LUT mapping, and the mapped values are shifted in the FPGA to achieve the corresponding power-of-two weighting to obtain the output result.

The size of the LUT can be further reduced by partitioning the complete table into multiple partial tables, with the partial tables operating in parallel [19], such that  :



Figure 6 shows the hardware structure block diagram of the partitioned LUT. The input variable  is stitched to form multiple sets of LUT addresses, which are sequentially shifted into the LUT to complete the mapping to obtain the partial product. The partial product is output under the control of the state machine through the accumulator and shift registers to obtain the final result.



**Fig. 6.** Split LUT hardware structure diagram.

In the APGD algorithm, there are three different constant matrices that need to be matrix multiplied, which are  ,  and  , the corresponding column vector cumulative additions are12, 10, 10，so the specific construction of the three-group partitioned LUT is 4×23, 5×22, 5×22 . The acceleration parameters  and  are related to the number of iterations, and the number of iterations is used as the address to construct the LUT mapping. The address is reassigned to zero when the parameters need to be reset. Parameter simplification can omit the operation process of squaring, dividing, and the extraction of square root, which reduces the use of FPGA logic units and improves the algorithm processing speed. In addition, the open root calculation is required in the process of calculating the residuals. After analyzing the residuals in this algorithm, we can replace the calculation of specific results with a simple comparison of sizes.

## C. Built-in Temperature Sensor Design

The on-chip temperature sensor network is an important part of the dynamic thermal management function, and the accuracy, resolution and sampling rate of the temperature sensors can have a significant impact on the management unit's decisions [20]. Within the constraints of hardware resources, the sensors need to be designed to occupy fewer logic units and can be deployed anywhere on the chip to enable temperature measurements on different cores.

This design uses a delay line digital temperature sensor based on a ring oscillator, as shown in Figure 7 shows the design structure of the temperature sensor. The temperature measurement circuit can be retriggered by a ring oscillator, thus continuously output square wave signal for a period of [21-22]. The variable gain time amplifier (VGTA) amplifies  by any multiple of time, outputs the amplified pulse as  , feeds  into the TDC of the binary counter for counting, and finally outputs the accumulated digital code  .



**Fig. 7.** Temperature sensor structure diagram.

The design uses Quartus II (Intel logic design software) to lock the synthesized hardware logic in any area, but cannot specifically set the layout wiring of the device. Therefore, even though the same code logic, the delay line structure after layout wiring will not be exactly the same. And the output digital code will have large differences under the same temperature conditions. A variable time amplifier gain  will be output by a single-point calibration circuit, so that different FPGAs or different regions of the same FPGA output the same digital code at the same temperature. The specific structure is shown in the single-point calibration circuit in Figure 7 above, which requires a set of temperature measurement circuits to measure the digital code  at a specific temperature. We use the temperature and digital code as the reference, compare it with the digital code  to be calibrated. At the same time, use the successive approximation (SAR) method instead of complete search for accelerated calibration. The amplification time gain of the measurement circuit corresponding to the current delay line is obtained by multiple approximations until  is approximately equal to  . The single-point calibration circuit has no practical use after calibrating each measurement circuit to get a different gain, so this circuit does not occupy the resources of on-chip temperature measurement.

The period of  is much shorter than the resolution of the required temperature sensor, so it is more appropriate to use a binary accumulator TDC measurement than a direct measurement using a ps-level high-precision TDC. The structure of the VGTA is shown in Figure 8.  is periodically amplified by a programmable binary subtraction counter and deburred by a D flip-flop.



**Fig. 8.** Variable gain time amplifier structure diagram .

The time domain temperature sensor consists of a ring oscillator, a time amplifier and a TDC, which is synthesized from user-written logic and hardware. The sensor can be laid out at any position on the chip to achieve temperature measurement of local hot spots and single-point calibration for different chips. Determined digital code output at the same temperature to ensure stability. The temperature sensors are maximally dispersed throughout the processor core and connected into a network through the underlying infrastructure. It passes the temperature values to the thermal management unit to achieve the active dynamic thermal management function of the FPGA.

# IV. Experiments and Data Analysis

## A. Comparison of APGD Algorithm Results

We verified the performance of the FPGA-based APGD algorithm with random data, and the APGD algorithm is compared by running on FPGA platform and PC platform. Randomly generate a set of matrices  and . The size of the matrix  is set to 12 × 10 and the size of the  matrix is 12 × 1. The size of the least square solution  is 10 × 1.

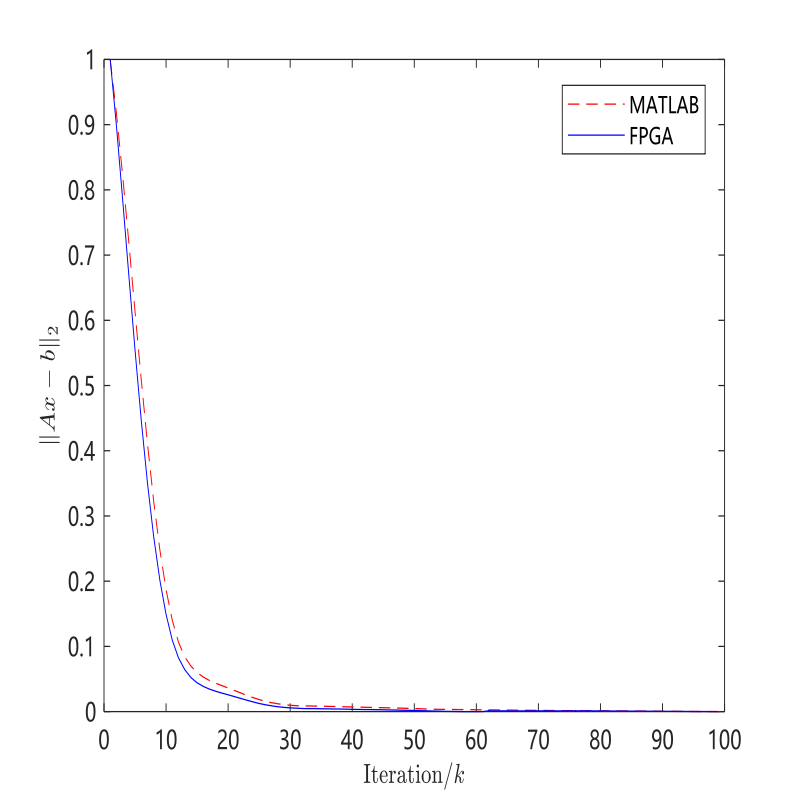
The results of the APGD algorithm under FPGA are shown in Figure 9, where the residual values  and the least square solutions  are taken out at the 60th iteration. At this time, the residual values remain basically unchanged, and the result of the 60th time is selected as the result of APGD algorithm.

The floating-point data in the APGD algorithm is represented in the FPGA using the Q12 fixed-point format, and the result  is basically the same as the APGD algorithm result on the MATLAB after conversion to float32 format. Therefore, the FPGA implementation of the algorithm acceleration architecture is proved to be correct.



**Fig. 9.** FPGA Simulation Results.

Figure 10 shows a comparison of the convergence curves of the residual normalization of the algorithm in MATLAB and FPGA. It can be seen from the figure that the residual attenuation corresponding to the two implementations is basically the same, which verifies the convergence of the APGD algorithm and the feasibility of the FPGA implementation of this algorithm.



**Fig. 10.** PC-side and FPGA residual iteration.

## B. Resource Usage and Processor Efficiency

Table I shows the on-chip IP resource usage and the ratio of total resources in the 10M50SCE144A7G. With a total of 50,000 flip-flop resources and 1638kB of M9K (RAM resources) on-chip. Only 15.97% of the RAM resources are consumed through a distributed algorithm to simplify the partition LUT.

TABLE I

IP Resources and Memory Consumption

|  |  |  |
| --- | --- | --- |
| Resources | IP and RAMs | %FPGA |
| LUTs | 11079 | 22.16 |
| Slice registers | 5560 | 11.32 |
| M9Ks | 262 | 15.97 |

The execution time of this algorithm is tested on PC, DSP and FPGA respectively to compare the algorithm time consumption. The PC-side algorithm experiments were tested on a Win10 system equipped with a 2.40GHz Intel Core i5-10200H processor. Both the DSP and FPGA operate at a 30M frequency system clock. The time consumption of the APGD algorithm is shown in Table 2. In terms of time consumption, DSP>PC>FPGA.

TABLE II

Comparison of FPGA and DSP Algorithm Time Consumption

|  |  |  |  |
| --- | --- | --- | --- |
| Method to realize | Single no  Update（us） | Single  Update（us） | 100  iterations（us） |
| PC（2.40GHz） | 5.11 | 9.83 | 567.76 |
| DSP（30MHz） | 451.33 | 873.43 | 49358.76 |
| FPGA（30MHz） | 1.54 | 3.01 | 171.94 |

Through the above experimental comparison, the implementation of APGD algorithm on FPGA is greater in execution efficiency, which is of great significance and research value. Compared with other existing works, it has a greater advantage in processing efficiency and performance.

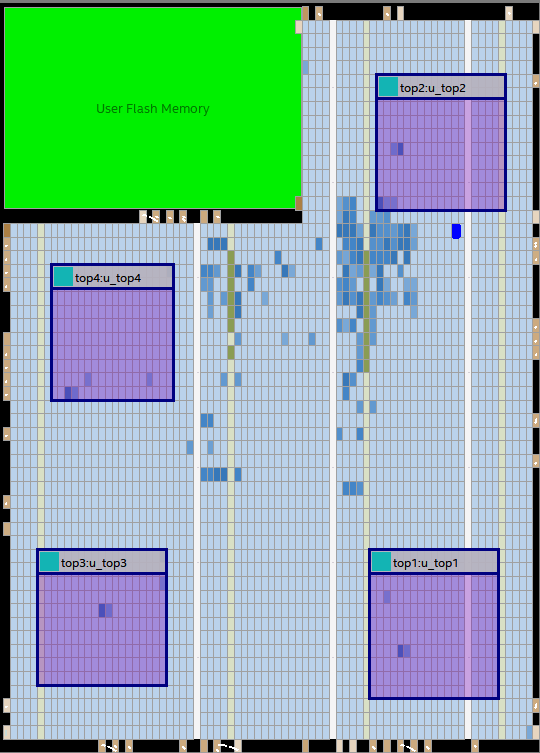
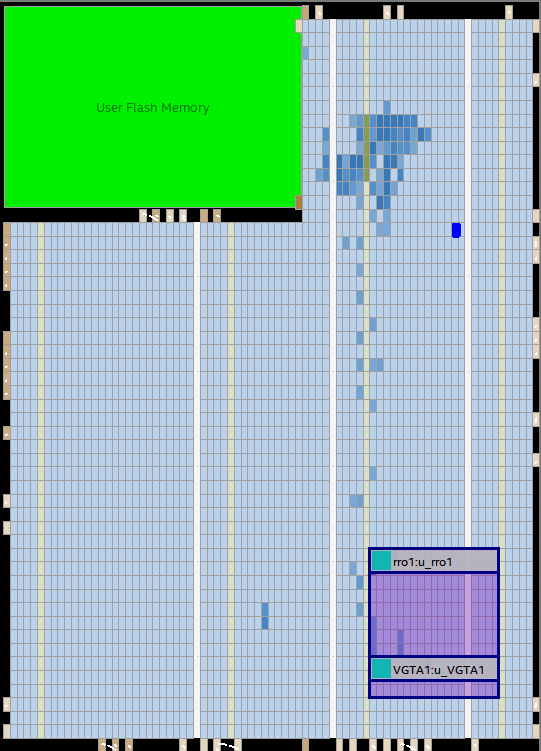
## C. Temperature Experiment

This design builds a set of high-temperature test platform. Figure 11 shows the structure of the high-temperature test platform. It consists of four parts, which are power supply, FPGA circuit board, incubator and PC host computer. The power supply is a DC adjustable power supply, and the 3.3V power supply can meet the circuit work requirements. The FPGA circuit board deploys a built-in temperature sensor to measure the local hot spot temperature, and transmits the digital code result to the PC host computer through the serial port. The model of the incubator is KCW-400, which is a high and low temperature cycle experiment box. The adjustable temperature range is -40~210℃, and the power supply voltage is 380V/50Hz. The PC uses the serial port terminal to receive the temperature digital code result.



**Fig. 11.** High temperature test platform structure diagram.

In order to verify that the FPGA task migration method can balance the temperature between local hot spots, the high-temperature experiment needs to be carried out twice, and the high-temperature experiments are performed and compared for the FPGA single-core execution task and multicore task migration respectively. Figure 12 shows the hardware logic layout of multi-core and single-core. The purple block diagram represents the hardware processing core that completes the layout in the FPGA. Each core area is composed of APGD algorithm and temperature sensor. Use the Logic Lock Regions function in Quartus II to lock the module in the user-defined region to prevent the hardware logic location from changing each time the layout is synthesized.

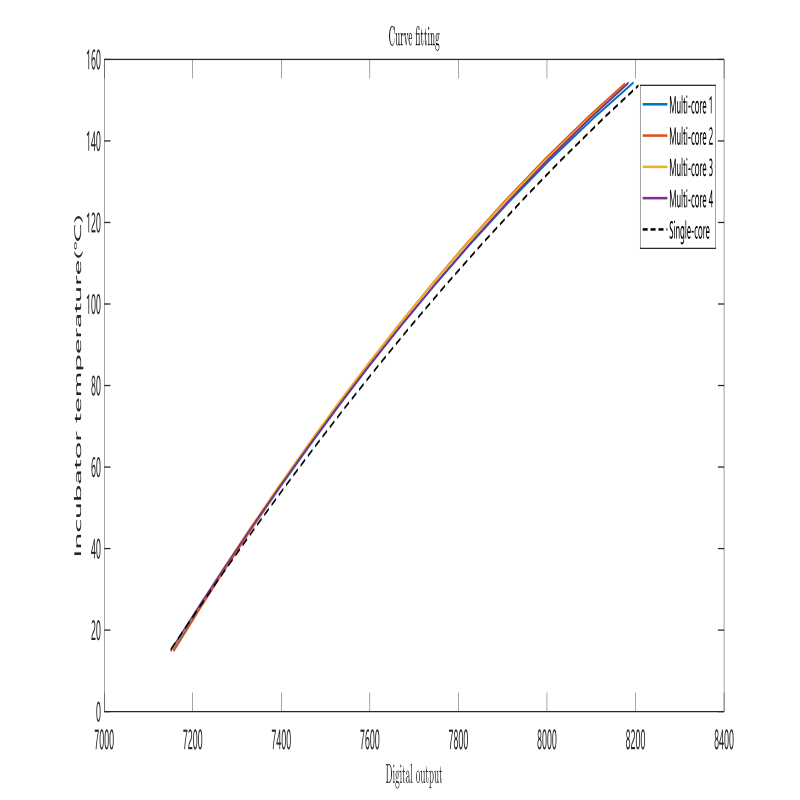
 

(a) (b)

**Fig. 12.** FPGA hardware logic layout.

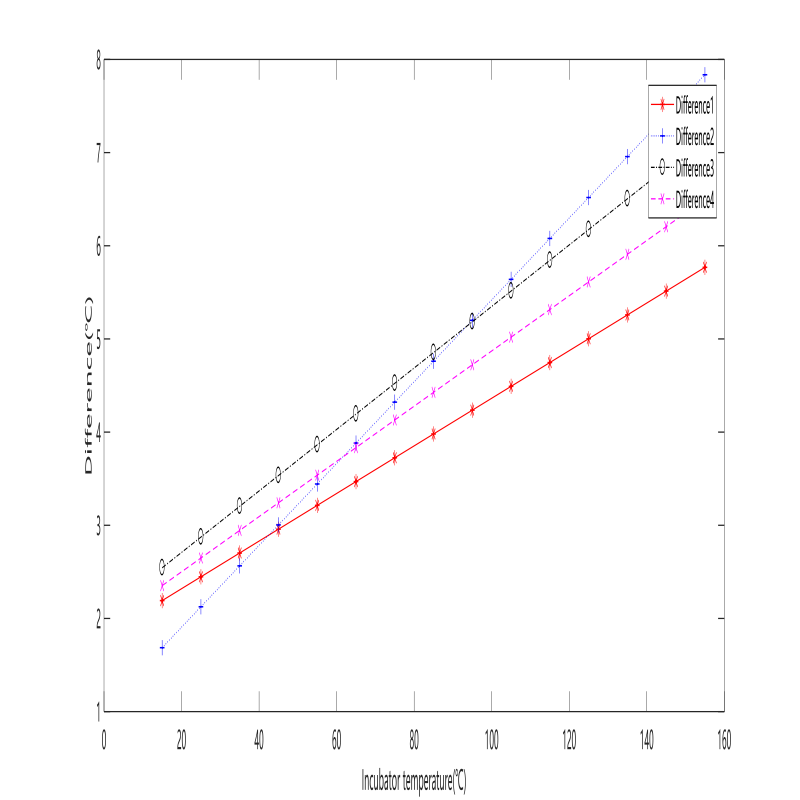
(a)Multi-core logic layout (b)Single-core logic layout

The experimental temperature range is 15~155°C, and the measurement is performed every 10°C. In order to prevent the data fluctuation of the incubator, when the temperature is stable, the measurement is performed at an interval of one minute, for a total of five measurements, and the average value is taken as the digital code at the current temperature. Since the Quartus software does not support an accurate thermal simulation model, the relationship between the temperature value and the digital code needs to be determined by curve fitting, As shown in Figure 13, the x-axis is the digital code, and the y-axis is the temperature value of the incubator. The five curves in the figure are the four curves under the multicore architecture in the first experiment and the curves under the single-core architecture in the second experiment, it can be seen that the digital code is positively correlated with the temperature value.



**Fig. 13.** Secondary fitting curve.

In this experiment, in order to verify the temperature difference between the multi-core and the single-core, the temperature of the single-core is used as the benchmark to obtain the temperature value of each position of the multicore and compare the difference. Figure 14 shows the difference between the single-core temperature value and the multicore temperature value at each location, where the x-axis is the temperature value of the incubator, and the y-axis is the difference value. It can be seen that as the ambient temperature increases, the temperature difference between the two gradually increases. The maximum temperature difference can reach about 5.5~8℃ at 155℃. This shows that the built-in temperature sensor in the multicore architecture measures a lower local temperature on the chip, and its heat dissipation capability is stronger. Therefore, this experiment can verify that the dynamic thermal management technology based on task migration can reduce the local hot spot temperature in the FPGA chip.



**Fig. 14.** Single-core sensor and multicore temperature difference

# V. Conclusion

In this study, we apply the APGD algorithm to solve the geophysical inverse problem in well logging. The algorithm has a fast convergence speed, clear architecture, and easy operations. Based on the actual logging requirements, we construct the coefficient matrix in the inversion model, design the matrix size to be 12×10, and fuse the data and parameters in the APGD algorithm to realize the geophysical inversion function. This algorithm is deployed in FPGA, and the pulsed array and distributed structure are applied to realize the hardware acceleration of the algorithm.

To address the problem of degraded reliability of high temperature downhole chip operation, a multi-core architecture based on task migration dynamic thermal management techniques is deployed to balance the temperature difference between local hot spots on the chip.

The correctness of the FPGA-based APGD algorithm is verified by numerical simulation experiments. The evaluation of the algorithm time consumption under different platforms shows that the method performs more efficiently. The multi-core architecture of FPGA is compared with the single-core architecture in high-temperature experiments, and the results show that the multi-core architecture can reduce 5.5 to 8°C at 155°C ambient temperature. This design provides a useful reference for research work of geophysical inversion in the field of well logging.

References

1. X. Zhang and A. Curtis, “Bayesian Geophysical Inversion Using Invertible Neural Networks”, Journal of Geophysical Research: Solid Earth, vol. 126, no. 7, 2021.
2. A. Melo and Y. Li, “Geology differentiation by applying unsupervised machine learning to multiple independent geophysical inversions”, Geophysical Journal International, vol. 227, no. 3, pp. 2058–2078, 2021.
3. N. B. Christensen\*, “Resolution attributes for geophysical inversion models: Depth of investigation and novel measures”, Near Surface Geophysics, vol. 20, no. 1, pp. 3–15, 2022.
4. Kim Y, Nakata N. Geophysical inversion versus machine learning in inverse problems[J]. The Leading Edge, 2018, 37(12): 894-901.
5. Russell B. Machine learning and geophysical inversion—A numerical study[J]. The Leading Edge, 2019, 38(7): 512-519.
6. J. Giraud, M. Lindsay, V. Ogarko, M. Jessell, R. Martin, and E. Pakyuz-Charrier, “Integration of geoscientific uncertainty into geophysical inversion by means of local gradient regularization”, Solid Earth, vol. 10, no. 1, pp. 193–210, 2019.
7. Y. Luo and R. Duraiswami, “Efficient Parallel Nonnegative Least Squares on Multicore Architectures”, SIAM Journal on Scientific Computing, vol. 33, no. 5, pp. 2848–2863, 2011.
8. A. Nalci, I. Fedorov, M. Al-Shoukairi, T. T. Liu, and B. D. Rao, “Rectified Gaussian Scale Mixtures and the Sparse Non-Negative Least Squares Problem”, IEEE Transactions on Signal Processing, vol. 66, no. 12, pp. 3124–3139, 2018.
9. Lawson C L, Hanson R J. Solving least squares problems[M]. Society for Industrial and Applied Mathematics, 1995.
10. Coleman T F, Li Y. A reflective Newton method for minimizing a quadratic function subject to bounds on some of the variables[J]. SIAM Journal on Optimization, 1996, 6(4): 1040-1058.
11. Bertsekas D P. On the Goldstein-Levitin-Polyak gradient projection method[J]. IEEE Transactions on automatic control, 1976, 21(2): 174-184.
12. Kim D, Sra S, Dhillon I S. Tackling box-constrained optimization via a new projected quasi-Newton approach[J]. SIAM Journal on Scientific Computing, 2010, 32(6): 3548-3563.
13. Fearnley J, Goldberg P W, Hollender A, et al. The complexity of gradient descent: CLS= PPAD∩ PLS[C]//Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing. 2021: 46-59.
14. Ou W, Yuan D, Liu Q, et al. Object tracking based on online representative sample selection via non-negative least square[J]. Multimedia Tools and Applications, 2018, 77(9): 10569-10587.
15. Kumar M. Design of IIR systolic array architecture by using linear mapping technique[J]. International Journal of Computer Applications, 2019, 182(39): 14-19.
16. Mora J, de la Torre E. Accelerating the evolution of a systolic array-based evolvable hardware system[J]. Microprocessors and Microsystems, 2018, 56: 144-156.
17. Zhang J, Zhang W, Luo G, et al. Frequency improvement of systolic array-based CNNs on FPGAs[C]//2019 IEEE International Symposium on Circuits and Systems (ISCAS). IEEE, 2019: 1-4.
18. Hwang S, Han G, Kang S, et al. New distributed arithmetic algorithm for low-power FIR filter implementation[J]. IEEE signal processing letters, 2004, 11(5): 463-466.
19. Kumar R N, Chandran V, Valarmathi R S, et al. Bitstream compression for high speed embedded systems using separated split look up tables (LUTs)[J]. Journal of Computational and Theoretical Nanoscience, 2018, 15(5): 1719-1727.
20. Rahimipour S, Flayyih W N, Kamsani N A, et al. Low-Power, Highly Reliable Dynamic Thermal Management by Exploiting Approximate Computing[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2020, 28(10): 2210-2222.
21. Chen P, Shie M C, Zheng Z Y, et al. A fully digital time-domain smart temperature sensor realized with 140 FPGA logic elements[J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2007, 54(12): 2661-2668.
22. Chen P, Chen S C, Shen Y S, et al. All-Digital Time-Domain Smart Temperature Sensor With an Inter-Batch Inaccuracy of—0.7° C—+ 0.6° C After One-Point Calibration[J]. IEEE Transactions on Circuits and Systems-I-Regular Papers, 2011, 58(5): 913.

