Adaptive loop pipeline control mechanism for Coarse-Grained Reconfigurable Block Cipher Array

Tongzhou Qu*, Zibin Dai, Chen Lin and Anqi Yin
*The Institute of Information Science and Technology, Zhengzhou 450000, China
*Corresponding author’s e-mail: dubito@non-you.com

Abstract. Coarse-grained Reconfigurable Array (CGRA) is a new processor architecture suitable for Block Cipher calculation. However, it has the problems of complex control mechanism and poor support for loop pipeline mapping. Based on the CGRA architecture designed for Block Cipher algorithms, this paper proposes a scenario jump model which can describe the process of loop pipeline running. Based on this model, a control structure supporting Block Cipher loop pipeline mapping was designed on the process platform CGCLA (Coarse Grain Cipher Logic Array), which could automatically complete the control functions described by virtual instructions in the model driven by the configuration flow to realize the adaptive control. Compared with similar process structures, CGCLA platform has more complete control functions. The mapping results of AES algorithm show that the proposed technique can achieve 2.15 times performance improvement and reduce the amount of algorithm configuration information by about 25%.

1. Introduction
With the continuous growth of communication requirement in the fields of information safety, it is most important for efficient cipher processor to heighten overall communication performance. In view of the continuous updating of cryptographic algorithm and the actual demand of multi algorithm encryption, reconfigurable cipher processor has wide application scenario and become one of the research hotspots. Coarse-grained reconfigurable array is new framework based on configuration information driven, has rich computations and interconnect resources and can fully develop the parallelism of application processing. Currently, it has been regarded as one of the best choices of hardware implementation of Block Cipher algorithm.

At present, there are many study about high efficiency and flexible array structure. Many documents consider loop pipeline capacity of array processor in designing. Michael Lomonaco put forward reconfigurable cipher processor CRYPTARRAY[1], which collected Processing Element of $4 \times 4$ scope, and setting up a shared storage area for inter pipeline stages storage to heighten performance, but it cannot schedule the keys in the process of pipeline and do not realize the whole cipher function. Daniele Fronte[2] and others design an array architecture “Celator” coupled ARM processors, which can implement most of the Block Ciphers under the control of microinstruction in a pipelining way. But the array resources are not made full use of in the process of instruction prefetching, decoding, and writing back. Guo Yansong[3] and others present an array Block Cipher processor “BCORE”, which integrate 16 x 32 reconfigurable units and support deep pipeline. The array adopt three level control mechanism of the configuration controller, array controller and unit controller, supporting dynamic reconfiguration in a unit level, but hierarchical control path organization limits throughput improvement. In the above
structure, there is a problem that the support for loop pipeline mapping of Block Ciphers is poor, and the corresponding control mechanism is complicated.

Current research on array architecture is mainly focus on developing the resource and performance advantages of CGRA’s data path, ignoring the important influence of its control structure on the overall efficiency. This paper establishes CGRBA (Coarse-Grained Reconfigurable Block Cipher Array, CGRBA) scenario-jump model and designs a set of control instruction sets. Based on the models, we have designed one kind of Reconfigurable Control Unit (RCU), it can enhance the adaptive capability of array pipeline to loop operating characteristics of Block Cipher. The experiment maps more than 20 kinds of Block Cipher Algorithms such as AES and analyses performance on the CGCLA processing platform. The result proves that the proposed structure has good algorithm adaptability and achieve certain performance improvement.

2. CGRBA scenario jumping model

2.1. Model establishment

Block Cipher algorithm is a typical computing intensive application, which has three parts of executing process: pre-processing, round function and tail processing. It realizes data encryption to satisfy the safe requirement of application through the loop iterative of round function. When Block Cipher are implemented on the array structure. CGRAs are usually organized into a certain depth pipeline by registers to develop the parallel processing capability of Block Cipher. Based on the loop pipelining feature, this paper abstracts the important time points during the array execution, then decompose the work process of CGRBA into different scenarios, according to the division of time points. The definition of scenario shown as below table 1.

| definition | time points description |
|------------|-------------------------|
| $s_0$      | At initial scenario, the array does not work |
| $s_1$      | From the array starting work to the pipeline is filled |
| $s_2$      | From the pipeline is filled to all data entering the loop |
| $s_3$      | From all data entering the round loop to the first set of data leaving the round loop |
| $s_4$      | From the first set of data away from the round loop to the first set of data being output |
| $s_5$      | From the first set of data being output to the last set of data being output |

To realize the self-jump of scenarios in Figure 1, this paper establishes a mathematical model describing the transformation relationship between scenarios:

$$F = (S, T, E, I, a)$$ (1)

$S$ is a collection of all CGRBA’s scenarios:

$$S = \{s_0, s_1, s_2, s_3, s_4, s_5\}$$ (2)

Collection $T$ includes the jump path $t$ of all scenarios in $F$; $E$ is a set of conditional $e$ that triggers the transformation of each scenario; Define function $a$ in $F$

$$a : T \rightarrow E : \{a(t) = e, t = (s_i, s_j) \mid t \in T; e \in E; s_i, s_j \in S\}$$ (3)

Mapping $a$ allocates to each jumping path $t$ and belonging to its own touching condition $e$, under the scenario $s_i$ when condition $e$ happens, path $t$ is activated, scenario skips from $s_i$ to $s_j$. Compared with computing tasks in the common field, the calculation of Block Cipher tasks has certain determinacy with its known algorithm structure, and users can make sure the arithmetic status of CGRBA. When array deal with the cipher application, array scenario jumps in accordance with fixed order. Table 2 defines controlling parameter of producing scenario jump conditions.
Table 2. CGRBA controlling parameter definition

| parameter | description                  |
|-----------|------------------------------|
| $j_1$     | pre-processing clock overhead|
| $n_1$     | CGRA’s pipeline depth        |
| $n_2$     | the number of CGRA’s loop iterative |
| $j_2$     | tail processing clock overhead |

Figure 1 shows the scenario how to jump as below. When array receives the signal of starting to calculate, condition $e_1$ happens and enter the scenario $s_1$:

$$a(t_0)=e_1, \quad t_0=(s_0, s_1)$$  \hspace{1cm} (4)

After $n_1$ clock circle, array pipeline is filled, condition $e_2$ happens and the scenario from $s_1$ to $s_2$ ;

$$a(t_1)=e_2, \quad t_1=(s_1, s_2)$$  \hspace{1cm} (5)

After $j_1$ clock circle, the last group of data complete the pre-processing, and all data enter the loop of round function, condition $e_3$ happens and the scenario from $s_2$ to $s_3$;

$$a(t_2)=e_3, \quad t_2=(s_2, s_3)$$  \hspace{1cm} (6)

After that, executing $n_1$ group data is running the $n_2$ circle in the circulation body, and after the clock circulation of $(n_2-1) \times n_1$, the first group of data leaves circulation, condition $e_4$ happens and the array scenario from $s_3$ to $s_4$;

$$a(t_3)=e_4, \quad t_3=(s_3, s_4)$$  \hspace{1cm} (7)

After $j_1$ clock circle, the last group of data end post treatment output, condition $e_1$ happens and the scenario skip to $s_5$;

$$a(t_4)=e_1, \quad t_4=(s_4, s_5)$$  \hspace{1cm} (8)

The following is a discussion of the special Block Cipher structure, if the pre-processing part is not included in the algorithm composition ($j_1=0$), after array pipeline is filled, it skips from scenario $s_1$ to $s_3$;

$$a(t_5)=e_1, \quad t_3=(s_1, s_3)$$  \hspace{1cm} (9)

If algorithm composition don’t include post treatment parts, the data is out of the loop and is directly output and the array directly skip from scenario $s_3$ to $s_5$;

$$a(t_6)=e_5, \quad t_5=(s_3, s_5)$$  \hspace{1cm} (10)

If the loop body on the array is only executed once ($n_2=0$), after all data enter the round function, the first data will leave the circulation, the array directly skip from $s_2$ to $s_4$;

$$a(t_7)=e_4, \quad t_7=(s_2, s_4)$$  \hspace{1cm} (11)

When output is done, array is on the scenario $s_5$, depending on whether the array continues to work, the scenario jumps back to $s_0$ or $s_1$.

$$a(t_8)=e_0, \quad t_8=(s_5, s_0)$$  \hspace{1cm} (12)
$$a(t_9)=e_1, \quad t_9=(s_5, s_1)$$  \hspace{1cm} (13)

Figure 1. Scenario skipping sketch
2.2. Loop pipelining scheduling

In view of the scenario division of the CGRBA running process, this paper defines the basic operation of control instruction set I to describe the execution of loop pipelining. In different scenarios, CGRBA executes relates controlling instructions to perform the basic operations required for the loop pipeline scheduling.

Loop input change instruction (IC): The input of the loop body of Block Cipher calculation is uncertain, which may come from the outside of the loop body or the output of its last iteration. IC is used to change the input data source of the loop pipeline; Loop output change instruction (OC): The loop output of Block Cipher calculation is uncertain, and may be fed back to the input of the loop body or output as the calculation result; the OC instruction outputs the data in the loop pipeline, and the execution of the IC and OC instructions is shown as Figure 2 (a), (b); Data input instruction (ID): Array read external memory operation; Data output instruction (OD): Array write external memory operation; Key read instruction (KR): Read the key from the internal memory of the array; Key write instruction (KW): Write the key to the internal memory of the array; Change Key Address instruction (KC): Change the read and write address of the internal memory of the array; Configuration reuse instruction (C): When the block cipher loop unfolds the mapping, all unfolded round function operators have exactly the same operation and data flow. The configuration reuse instruction can realize the configuration information storage and repeated configuration at one time. Figure 2(c) shows a schematic diagram of the instruction C.

The above virtual instructions do not consider specific hardware implementation methods, execution efficiency, and resource constraints, the scenario jumping model F abstracts the execution scenario and jump conditions of Block Cipher on CGRA. Through the user-defined control instruction set I in the model, we describe the scheduling operations that arrays need to perform in different scenarios. The control instructions as above do not take into account the specific hardware implementation methods, execution efficiency, and resource constraints, only to express the scheduling operations that need to be implemented in the whole CGRA scenario jumping process.

2.3. Model analysis

The model F abstracts the execution scenario and jump condition of Block Cipher on CGRA, and the custom virtual instruction set I describes the specific operation of CGRBA loop pipeline scheduling. From the above model, the following conclusions are drawn:

(1) The performance improvement of Block Cipher processing is optimized for the maximum degree of the loop line, and the CGRBA’s loop control should be separated from the configuration control, and the individual loop controller is used to schedule the loop pipeline.

(2) Controller design should consider the implementation of the virtual instruction in different scenarios in the model, and make full use of the feature of configuration flow on CGRA, maximizing
the execution efficiency of the virtual instruction, and optimizing the interaction between the controller and the array data path.

3. Adapted loop control

3.1. Reconfigurable cipher code processing platform CGCLA

The overall architecture of coarse-grained reconfigurable cipher code processing array CGCLA is shown in Figure 3. It is mainly composed of reconfigurable computing subsystem, configuration subsystem and control subsystem. The computational granularity of the reconfigurable arithmetic subsystem is 32 bits, containing a 4x4 function unit FU, which interacts with the external input and output FIFO. Each FU contains three four-directional interconnection nodes CB and a reconfigurable cipher processing unit PE, CB and PE are connected through the 8x8 bidirectional Mesh interconnection network. The two register file RKB, which has a depth of 64, is used to store keys.

![Figure 3 CGCLA overall architecture](image)

The PE contains 4 types of cipher code operation units: the NF unit, the AL unit, the LG unit, and the BP unit, respectively calculating the nonlinear operations, logical operations, arithmetic operations, bit extraction and shift operations in cipher code operations, each unit supporting the post exclusive or function; The data interaction in PE realizes through the Crossbar network; a branch transmission network can be set up in the processing unit, which can be configured flexibly according to the application requirements. The configuration subsystem contains 4 configuration pages to store the configuration information needed for the array to run. It can switch flexibly under the control of configuration controller to achieve dynamic partial reconfiguration, and granularity of reconstruction is FU at the minimum. Since this paper focuses on how to optimize the controller design of CGRA, it doesn’t introduce the calculation and configuration subsystem of the array in detail.

3.2. Reconfigurable controller

According to conclusion 1 in the model analysis, the reconfigurable controller designed in this paper separated the array loop control from configuration control, so that the loop scheduling of CGCLA was decoupling from the configuration control. The loop control was no longer completed solely by the configuration switch. Thus reducing the clock overhead introduced by the configuration storage and configuration switch. The reconfigurable controller is divided into configuration controller (CCU) and loop controller (LCU). According to the control function. The configuration controller is responsible for the distribution and switching of configuration information, and the loop controller schedules CGRBA’s pipeline based on scenario jump. Reconfigurable controller’s features are as follows:
(1) Dynamic partial reconfigurable: In order to improve the flexibility of CGCLA, the minimum granularity controlled by CCU is a single PE. The configuration storage of CGCLA is divided into 4 pages, and the page address can be changed arbitrarily under the control of CCU to realize dynamic partial reconfigurable. Multiple CCUs are connected through static routing, supporting configuration address broadcast to specific PE; By selecting local configuration address or router broadcast configuration address, two configuration control modes of “one to one” or “one to many” are realized. On the premise of following the data correlation constraint, users can develop the parallelism between the configuration phase and the computation phase. Configuration time of CGCLA is hidden within the computation time to further improve the resource utilization rate of CGCLA.

(2) Centralized loop pipeline control: the loop controller is set in centralized way to control the data access and loop entry/exit judgment. LCU runs in accordance with scenario jump model F, and issues control instructions to different units of CGCLA according to the functions described in the virtual control instructions. In this paper, configuration flow driven control is adopted to optimize the clock overhead caused by instruction decoding. After the configuration information injecting, the LCU controls the loop pipeline running in an adaptive manner and automatically exits the loop.

3.3. Loop controller RCU

The reconfigurable control subsystem is divided into configuration controller and loop controller. Configuration controller is responsible for distribution and switching of configuration information and can dynamically reconfigure CGCLA during operation. The loop controller RCU is based on the scenario jumping model proposed above, and the auxiliary configuration controller implements array control to judge the operation scenario of CGCLA and execute corresponding control instructions automatically. It is clear that the configuration controller itself can achieve configuration switching on its own, and can also configure switch according to the control instructions issued by RCU.

![Figure 4. RCU hardware structure](image)

The RCU structure shown as in Figure 4 (a) includes RCU configuration register, configuration parsing unit, parameter register, condition generator, scenario jump engine, end judgment logic, interconnect control module, access control module and storage control module. The functions of each module are as follows: the RCU configuration register receives configuration information from the configuration controller and sends it to the configuration resolution unit. The configuration parsing unit is configured for each module; the controlling parameters J1, J2, N1 and N2 are stored in the parameter register. The conditional generator triggers the scenario jumping condition according to the control parameters by clock drive. The scenario jump engine is started by the external input START signal and is controlled by the conditional generator to jump the scenario; the scenario jump engine runs according to the scenario skipping model F, and sends out different control instructions under different scenario of F. After each loop operation is finished, the execution of instruction F is done by the end judgment logic,
and the control scenario jumps back to S0. Configure switch instruction C commands configure controller to configure switch, and three control modules execute the rest instructions separately.

3.4. Controlling model design
RCU contains three types of control modules to control the data access, loop control and key scheduling of CGCLA according to the scenario's jump.

The access control module realizes the management of data interaction between the array and the external input and output FIFO, receiving the control instructions ID and OD from the scenario jump engine. And judge whether making external FIFO possible according to the input and output FIFO feedback read and write signal. When the module receives the ID instruction and the input FIFO is not empty, the read request is sent to the input FIFO; When the module receives the OD instruction and the output FIFO is not full, it sends a writing request to the input FIFO, otherwise, the stop generator is sent to the conditional producer to keep the scenario unchanged.

The interconnect control module receives the control instructions IC and OC from the scenario jump engine, and can change the input and output connection relationship of the PE according to the instruction, while receiving the instruction, it also receives the 16 bit control word \(p_{15}p_{14}...p_0\) from the scenario jump engine. Each bit corresponds to one of the PE in the array. If \(p_i=1\), the relationship between the corresponding PE is changed. As shown in Figure 5 (a), When the module receives the control instruction IC, the partial interconnection structure of \(p_0=1\) and PE0 changes, the feedback result from the previous iteration becomes the input of the first row of PE. When the module receives the control instruction OC, as shown in Figure 5 (b), at this time, the partial interconnect structure of the \(p_3=1\), PE3 is changed, and the result of the PE is no longer feedback but direct output. Through the interconnection control module, CGCLA can adapt to realize the loop iteration and jump out of the Block Cipher function.

![Configuration reuse instruction C](image)

Figure 5. CGCLA design for virtual instruction

The configuration reuse module realizes the function of configuring the reuse instruction C. This paper proposes a configuration reuse strategy for PE in the same column in CGCLA, which supports one-time storage of configuration information and concurrent multi-path configuration. As shown in Figure 5(b), the scenario jump engine can drive the configuration reuse module to issue a 16-bit control word \(C[15],..., C[k],..., C[0]\), when \(C[k]=1\), the configuration input source for PE[k] is changed from the local configuration memory to the configuration bus. When the configuration information used by some PE is same, the configuration information that can be reused is loaded into the configuration bus, and the source of PE configuration page is selected according to the 16-bit control word. After the
transfer path—the local memory-configuration bus-local memory, multiple PEs can share the same configuration input. Configuration reuse strategy virtualizes many times configuration resources in limited memory space. And develops the reusability of configuration information when the Block Cipher round function rolls out.

Figure 6. (a)Hardware structure of storage control module (b) state transition of storage control module

The storage control module receives the key loading instruction K from the scenario jump engine, In Block Cipher algorithm, the frequency of key usage is proportional to the number of execution cycles. Accordingly, an RKB address counter with controllable counting frequency as shown in Figure 6(a) is designed to realize dynamic key scheduling. The output port of the address counter is also connected to comparing unit 1, 2 and RKB (as read and write addresses) to reset the counter in a specified clock cycle and set the value of the presupposition comparing unit by configuring information; as in Figure 6(b), the running state of RKB is divided into three segments: 1, no start count; 2, count to default of the comparing unit 1; 3, count to the default of the comparing unit 2. The cipher code instruction K also gives the Sel value of the selection signal, to decide whether to select const1 or const2 in different states as counting cycles of counters. The values of the 3 comparing unit are obtained by the parameter register. Const1 corresponds to J1 as the initialization stage before the array enters the round function; Const2 corresponds to N2, which represents the number of cycles performed by the round function; Const3 corresponds to N1, which determines the number of clock cycles consumed by the execution of a round function. The storage control module realizes the dynamic scheduling of the key usage of the Block Cipher by transforming the six scenarios of the array into three states.

4. Algorithm adaptation and performance analysis

4.1. Algorithm adaptation
This paper uses EDA tools to synthesize the 4x4 scale CGCLA structure RTL level code in the 55nm technology. Adapting to 20 kinds of typical Block Cipher algorithms such as AES, DES, IDEA and so on the CGCLA platform, the results of which show that the CGCLA platform algorithm has a good adaptability.

Table 3 shows the control mechanism comparison between CGCLA and similar processing structures. According to the table, except for BCORE[3] and S-RCCPA[5], other structures do not support distributed configuration switching, and all CGRA architectures can accelerate block cipher processing in the way of loop pipeline. However, CGCLA supports loop pipeline mapping of block cipher, realizes configuration reuse and dynamic key scheduling, and has the advantages of simple mapping method and less configuration switching times compared with other architectures.
Table 3. Comparison of control functions between CGCLA platform and other structures

| Controller function | Cryptarray [1] | Cleator [2] | BCORE [3] | COBRA [4] | S-RCCPA [5] | This Paper |
|---------------------|---------------|-------------|-----------|-----------|------------|------------|
| Distributed config. switching | ×             | ×           | √         | ×         | √          | √          |
| Loop pipeline map.   | √             | √           | √         | √         | √          | √          |
| Adaptive Loop       | ×             | ×           | √         | ×         | ×          | √          |
| Control reuse       | ×             | ×           | ×         | ×         | ×          | √          |
| Dynamic key         | ×             | ×           | ×         | ×         | ×          | √          |

Note: √ indicates that the control function is supported by the corresponding processor, and × indicates that it is not supported.

In order to explore the effect of circle controller RCU on the overall performance of CGCLA, this paper compares the performance of the unused RCU array processor (processor A) with the RCU (processor B) array platform on the prerequisite of not calculating the injection time. Apart from the controller, the two processors use the same operation and interconnection units. The experiment result is shown as table 4 that processor B saves 50% configuration pages when dealing with different cryptographic algorithms in same process and similar processor area. This mainly attributes to the RCU’s ability to adaptively implement loop entry and exit, thereby saving a configuration page for CGCLA. At the same time, because RCU can effectively develop the flow depth of the array mapped Block Ciphers, the processor B improves the performance of 3.33~9.28 times compared to the processor A.

Table 4. Comparison result of two kinds of CGCLA platform

| processor | A       | B       |
|-----------|---------|---------|
| Technology| 55nm    | 55nm    |
| Frequency | 150MHz  | 150MHz  |
| Area      | 50mm²   | 50mm²   |
| Algorithm | throughput (Mbps) | configuration page | throughput (Mbps) | configuration page |
| AES       | 1600Mbps | 2        | 5358Mbps | 1        |
| SMS4      | 564Mbps  | 2        | 5237Mbps | 1        |
| IDEA      | 960Mbps  | 3        | 3290Mbps | 2        |
| DES       | 872Mbps  | 2        | 2908Mbps | 1        |

5. In conclusion
This paper explores the circulating flow control mechanism of Block Cipher based on coarse-grained reconfigurable array platform of CGCLA, and put forward one kind of CGRA scenario skipping model and a set of control instruction. A kind of loop controller RCU is designed based on its model to separate the loop control from configuration control and can realize adaptive circulating flow control of CGCLA. The proposed scheme is verified experimentally and its result shows that RCU can improve the performance of Block Cipher while reducing the number of array switching. Compared with other Block Cipher processing platform, it has outstanding advantage. The next step is to consider to further reduce the power consumption of the array by optimizing the configuration control structure, and seek a comprehensive balance in the area efficiency, power consumption and control efficiency of cipher processing.
References

[1] Lomonaco M J. (2004) CRYPTARRAY: A scalable and reconfigurable architecture for cryptographic applications. https://stars.library.ucf.edu/cgi/viewcontent.cgi?article=1140&context=etd.

[2] Fronte D. (2010) Celator: Design and development of a reconfigurable cryptographic co-processor. LAP Lambert Academic Publishing, Saarbrücken, Germany.

[3] GUO Yansong and LIU Leibo. (2015) A block cipher oriented coarse-grained reconfigurable array and AES algorithm mapping. MICROELECTRONICS & COMPUTER, 32:1-5.

[4] Elbirt A J and Paar C. (2005) An instruction-level distributed processor for symmetric-key cryptography [J]. IEEE Transactions on Parallel & Distributed Systems, 16:468-480.

[5] CHEN Tao, LUO Xingguo, LI Xiaonan, et al. (2014) An Architecture of Stream Based Reconfigurable Clustered Block Cipher Processing Array[J]. Journal of Electronics & Information Technology, 36:3027-3034.

[6] Du Y, Li W, Dai Z, et al. (2020) PVHArray: An Energy-Efficient Reconfigurable Cryptographic Logic Array with Intelligent Mapping. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 28:1302-1315.