Article

High Flexibility Hybrid Architecture Real-Time Simulation Platform Based on Field-Programmable Gate Array (FPGA)

Ruyun Cheng , Li Yao, Xinyang Yan *, Bingda Zhang and Zhao Jin

The Key Laboratory of Smart Grid of Ministry of Education, Tianjin University, Tianjin 300072, China; 2020234020@tju.edu.cn (R.C.); yaoli@tju.edu.cn (L.Y.); bdzhang@tju.edu.cn (B.Z.); 1017234013@tju.edu.cn (Z.J.)
* Correspondence: yanxinyang@tju.edu.cn; Tel.: +86-188-6187-1051

Abstract: With the expansion of system scale and the reduction in simulation step size, the design of a power system real-time simulation platform faces many difficulties. The interactive operation of real-time simulation presents the characteristics of phased and centralized. This paper proposes selecting the appropriate simulation method for each sub-network according to the system operation requirements, and the sub-network simulation method can be changed with the change in system operation requirements in the simulation process. In order to change the sub-network simulation method in the simulation process, a high flexibility hybrid architecture real-time simulation platform based on FPGA was designed. The main body of the architecture runs in the high control mode of instruction flow and uses instruction flexibility to realize the requirement of changing methods. The algorithm modularity architecture is used as an auxiliary architecture to reduce the instruction cost and increase the computing power. Finally, the hybrid architecture real-time simulation platform was implemented in the Xilinx VC709 board (Xilinx corporation, San Jose, CA, USA), and the verification results show that under the same system scale, the hybrid architecture simulation platform combined with simulation method changing realizes shorter simulation step and complex interactive operation.

Citation: Cheng, R.; Yao, L.; Yan, X.; Zhang, B.; Jin, Z. High Flexibility Hybrid Architecture Real-Time Simulation Platform Based on Field-Programmable Gate Array (FPGA). *Energies* **2021**, *14*, 6041. <https://doi.org/10.3390/en14196041>

Academic Editor: Robert Griño

Received: 30 July 2021

Accepted: 20 September 2021

Published: 23 September 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

Keywords: real-time simulation; field-programmable gate array (FPGA); simulation method and data change; high flexibility hybrid architecture

1. Introduction

Climate change has become an important topic of global environmental issues, and climate risk indices are increasing [1]. China has proposed a dual carbon target to reach the peak of carbon dioxide emissions by 2030 and achieve carbon neutralization by 2060. The power system is the hub of the energy chain and plays an important role in the emission chain [2]. In order to realize the optimal allocation of energy resources and achieve a dual carbon target, a large-scale AC/DC hybrid power system is the development trend in the future [3,4]. Real-time simulation plays an important role in verifying the control method of hybrid power systems, ensuring the safe operation of devices and developing new power electronic equipment. With the expansion of the power grid interconnection scale and the investment of new power electronic equipment, the simulation step size becomes smaller, and the simulation calculation burden becomes larger [5,6], which puts forward higher requirements for the performance of real-time simulation platform.

With the continuous emergence of high-performance computing devices, there are many design schemes for real-time simulation platforms. Parallel computer architecture is the mainstream of commercial real-time simulation platform, which is implemented by multi-core processors [7], or PC clusters [8]. The underlying hardware of these platforms is still serial execution; in order to achieve large-scale system simulation, a large number of underlying hardware needs to be set, which brings cost and communication problems [9]. Therefore, some underlying hardware with parallel computing capabilities, such as GPU and FPGA, are used as auxiliary hardware to form heterogeneous architecture

and undertake part of the computing tasks [10–12]. Heterogeneous architecture platforms are implemented mainly by network decoupling, and each processor is responsible for computing a certain number of subsystems [13]. Coarse-grained parallel techniques, time synchronization and data communication between processors become the main problems of these simulation platforms; it is difficult to make full use of processor performance and has the problem of simulation accuracy. FPGA is a fully configurable device with distributed memory structure and pipeline structure, which can design specific architecture for the application. These advantages make FPGA the main hardware to participate in the real-time simulation of power systems.

The design architecture of electromagnetic transient real-time simulation platform based on FPGA can be divided into two kinds: algorithm modularity architecture (AMA) [14–16] and instruction flow driven architecture [17–19]. AMA establishes dedicated modules according to the algorithm form, and the modules are fixedly connected by the algorithm process in order. Under the guidance of the global control module, the start control of the special module is realized. AMA has the advantages of simple control and high computing efficiency. However, it is difficult to deal with some processes with branches and complex calculations; in addition, dedicated modules are often idle in the calculation process, which cannot make full use of the resources on the FPGA chip. The feature of instruction flow-driven architecture is to design a high reusability calculation unit based on the simulation method, and the data address and the operation type of the calculation unit are given by instruction to realize calculation. Instruction flow-driven architecture has strong flexibility and high utilization of computing units. However, the bandwidth requirement of instructions is too high, which makes it difficult to read instructions from external memory. Storing instructions on FPGA consumes a lot of memory resources, which limits the scale of the simulation.

In order to simulate a larger system scale and improve the performance of a real-time simulation platform, this paper designs a highly flexible hybrid architecture real-time simulation platform. Section 2 introduces the SSN simulation algorithm, analyzes the characteristics of real-time simulation operation and proposes to change the simulation method and data in the simulation process to expand the simulation scale. Section 3 illustrates the limitations of changing methods and data in the simulation process of the existing architecture and gives the design method of hybrid architecture. Section 4 verifies the effectiveness of the hybrid architecture simulation platform.

2. Analysis of Real-Time Simulation Characteristics

2.1. SSN Method

The simulation method with high parallelism and low computation is beneficial to expand the scale of real-time simulation. The SSN method selects some nodes to divide the system into multiple sub-networks [20]. The state-space method is used to solve the sub-networks, and the node equation is used to solve the system nodes. SSN method improves the computing parallelism of the system and balances the amount of computation between state-space and node equation, which shows its applicability for real-time simulation. The basic form of sub-network is shown in Figure 1.

Figure 1. Schematic diagram of sub-network.

The port voltage $u(t)$ is taken as the input variable vector of sub-network, and the port current $i(t)$ is taken as the output variable vector of sub-network; the state-space equation is written for the sub-network and discretized by backward Euler method as follows [20]:

$$x(t) = A_k x(t - \Delta t) + B_k w(t) + E_k u(t) \quad (1)$$

$$i(t) = C_k x(t - \Delta t) + D_k w(t) + F_k u(t) \quad (2)$$

where $x(t)$ and $x(t - \Delta t)$ are the state variable vector at the current moment and previous moment. $w(t)$ are the internal independent current source vector of the sub-network at the current moment. $A_k, B_k, C_k, D_k, E_k, F_k$ are the coefficient matrix, the value of k is related to the running state of the sub-network. The automatic formulation methods for (1) and (2) can be found in [21]. Equation (1) is the state variable update formula, Equation (2) is the Norton equivalent expression of sub-network, and the first two terms on the right side of the equation are combined to become the internal injection source of the sub-network port $i_s(t)$:

$$i_s(t) = C_k x(t - \Delta t) + D_k w(t) \quad (3)$$

After the Norton equivalent circuit of each sub-network is calculated by Equation (3), the node voltage equation of the system can be constructed [20]:

$$G_{ex} \cdot u_{ex}(t) = i_{ex}(t) \quad (4)$$

where G_{ex} is the system equivalent conductance matrix, $u_{ex}(t)$ is the system node voltage vector, $i_{ex}(t)$ is the injection current vector of the system node. Equation (4) is still sparse and can be solved by the node elimination method, then update state variable of sub-networks by Equation (1) to complete a simulation step calculation.

In the case of the pre-stored $A_k, B_k, C_k, D_k, E_k, F_k$ coefficient matrix, SSN has the advantages of less computation, simple process and high parallelism. However, in the real-time simulation, due to the requirements of hardware in the loop (HIL) or operator training, there are many interactive operations in the simulation system, such as fault settings and parameter settings. The variable network structure and parameters make the memory requirement of the coefficient matrix increase rapidly, which limits the simulation scale of SSN in a real-time simulation platform.

2.2. Variable Detailed Sub-Network

Components that can interact exist in almost every sub-network of the system, but these interactions do not occur at the same time. In the process of real-time simulation, the interactive operation of the system presents the characteristics of phased and centralized. During a continuous period of time, when performing fault tests and parameter settings on several sub-networks, other sub-networks hardly carry out any interactive operations. Therefore, at a certain simulation moment, it can be considered that the system has only a few sub-networks that can interact, which are called detailed sub-network, and other sub-networks cannot interact, which are called simple sub-network. The system that was originally a detailed sub-network representation turns into a system with a mixed representation of detailed sub-networks and simple sub-networks. By changing the detailed sub-network in the simulation process, each sub-network of the system can still realize interactive operation. This process is called a variable detailed sub-network.

After introducing the concepts of detailed sub-networks and simple sub-networks, SSN is also improved accordingly. Detailed sub-networks can set fault, load and operation status of the equipment in order to avoid consuming a lot of memory. The node analysis method, which is suitable for variable network structure and parameters, is used to solve the sub-network; the simple sub-network only has some state changes in switching elements and nonlinear elements, the memory requirement of the state-space method is acceptable. The state-space method is still used to solve the sub-network. It should be noted that the number of detailed subnetworks is limited to a certain simulation moment

and accounts for a small proportion in the total number of sub-network, so the improved SSN method retains the characteristics of less calculation burden, balances the operation requirements and memory requirements and can simulate a larger scale system.

For the real-time simulation platform, the application difficulty lies in the variable detail sub-network. When the detailed sub-network is changed, the involved sub-network needs to change the simulation method, as shown in Figure 2. Changing the simulation method also brings the corresponding data preparation requirements, the state variable of two methods can be inherited, but the coefficients need to be changed. Therefore, the real-time simulation platform should be able to change the sub-network simulation method and coefficients during the simulation process.

Figure 2. Framework of variable detail sub-network SSN method.

3. Hybrid Architecture Based on FPGA

3.1. Hybrid Architecture Design Analysis

Changing the simulation method can be considered as the branch selection of the simulation program. The dedicated calculation units in the AMA are fixedly connected. The flexibility of the architecture is poor, and it is difficult to implement branch selection in the simulation process. Instruction flow-driven architecture works under instruction control; changing the simulation method can be realized by replacing instructions, which is easy to implement. In the simulation process, instruction RAM is read at any time. In order to avoid the impact of the instruction replacement process on the operation of the simulation platform, the ping-pong operation is used to complete the instruction replacement. The simulation method replacement process of instruction flow-driven architecture is shown in Figure 3. However, the ping-pong operation needs to consume twice as much memory, which is

unacceptable for instruction flow-driven architecture with high memory consumption. Therefore, it is necessary to reduce the memory cost of instructions.

Figure 3. Instruction replacement process of instruction flow-driven architecture.

After the compiling software compiles the simulation method into instructions and downloads them to FPGA, the instruction driven architecture reads the instructions and decodes them after the start of the simulation step, and sends them to the high reusability calculation unit to complete the expression operation until the end of the simulation step, then continue with the same operation for the next simulation step. The fine-grained instructions are at the expression level, and it is convenient for discovering the parallelism of the algorithm and improving flexibility. However, the conversion of all arithmetic expressions into corresponding implementation instructions brings high instruction memory consumption. If there is a correlation between some arithmetic expressions, use the correlation to design a dedicated calculation unit to achieve expression autonomous calculation, instruction only controls the startup of the dedicated computing unit so that the instruction cost can be reduced. This process is the introduction of AMA. The introduction of AMA needs to meet the following conditions:

- The correlation arithmetic expressions shall have the same operation type to simplify the design of special computing units;
- The correlation operation expression occupies a certain proportion in the simulation method to avoid reducing the utilization of resources due to the idle of dedicated calculation units after the introduction of AMA.

Correlation arithmetic expressions that meet the above two conditions can be designed with algorithmic modular architecture. Then, AMA is used as the auxiliary computing unit, which is introduced into the instruction flow-driven architecture to form a hybrid architecture. The hybrid architecture retains the flexibility of the instruction-driven architecture and uses AMA to be responsible for part of the calculation to reduce instruction cost and increase computing power so that variable detail sub-network SSN method can be applied.

3.2. Method Task Division and Analysis

The first problem to be solved in a hybrid architecture is to find the correlation arithmetic expression that meets the introduction conditions of AMA. By observing Equations (1) and (3), it can be found that due to the simplification of the solution process by the state space method, the Norton equivalent formula and the state variable update formula of the simple sub-network are in the form of matrix-vector multiplication. Since most of the sub-networks in the system are simple sub-networks, Equations (1) and (3) account for a certain proportion in the simulation algorithm. Therefore, AMA can be introduced.

To analyze the dependence of Formula (1) and Formula (3) on other calculation formulas in the method, Equation (1) is divided into two parts:

$$x_{\text{int}}(t) = A_k x(t - \Delta t) + B_k w(t) \quad (5)$$

$$x(t) = x_{\text{int}}(t) + E_k u(t) \quad (6)$$

Equation (5) does not depend on the solution value of system node voltage. Combined with the framework in Figure 2, the variable detail sub-network SSN method can be divided into six tasks, and its dependencies are shown in Figure 4.

Figure 4. Variable detail sub-network SSN method task division and analysis.

The node analysis method used in tasks 1 and 3 is strongly related to the model and process, and there are many types of expressions and complex data addressing; Task 2 is solved by the node elimination method, the data are dependent on the calculation process, and the control is complicated. These tasks are suitable for instructions control to achieve high resource utilization and complex calculations. From the observation in Figure 4, it can be found that tasks 1, 2, 3 and tasks 4, 5, 6 can be calculated in parallel, respectively, which means that the idle rate of computing units in the hybrid architecture is low. Therefore, the introduced AMA can be designed through tasks 4, 5 and 6.

3.3. Algorithm Modularization Architecture Design

The algorithm modularization architecture aims to complete the autonomous operation of tasks 4, 5 and 6 after receiving the start instruction. Before designing the startup instructions and hardware modules of the architecture, it is necessary to determine the autonomous computing process. Unify the task form as follows:

$$\begin{cases} i_s(t) = [C_k & D_k] \begin{bmatrix} x(t - \Delta t) \\ w(t) \end{bmatrix} \\ x_{\text{int}}(t) = [A_k & B_k] \begin{bmatrix} x(t - \Delta t) \\ w(t) \end{bmatrix} \\ x(t) = [E_k & x_{\text{int}}(t)] \begin{bmatrix} u(t) \\ 1 \end{bmatrix} \end{cases} \quad (7)$$

The three types of calculation tasks can be expressed in the form of matrix-vector multiplication $R = H_k \times s$. The purpose of a unified task form is to establish the same autonomous computing process. The autonomous operation process of defining matrix-vector multiplication is shown in Figure 5. By decomposing the matrix-vector multiplication into several row operation subtasks, the row operation subtasks are highly parallel. In order to speed up the matrix-vector multiplication solution, multiple row operations can be solved in parallel. However, when the number of parallel row operations is insufficient at the end stage, the invalid operations need to be supplemented. The matrix dimension is generally a multiple of three due to the three-phase system. In order to reduce the number of invalid operations, three row operations are selected for parallel calculation.

Figure 5. Autonomous operation process of matrix-vector multiplication.

According to the autonomous operation process, the data of the $m \times n$ coefficient matrix of H_k show the law of sequential reading. The multiplied vector s is read sequentially within row operations and circularly between row operations. If the data in Equation (7) can be stored in sequence according to the operation flow of Figure 5, the data search in the autonomous operation of the matrix-vector multiplication task can be realized by providing several first addresses at startup.

The data storage and addressing process of Equation (7) is shown in Figure 6a. The coefficient matrix is only used inside the algorithm modularization architecture, so it is stored in the local RAM, and because of the parallel calculation of three row operations, local RAM is set as vector memory, read and write in SIMD mode. Most multiplied vectors need to communicate with the outside, so they are stored in the shared RAM. The multiplied vector of each row operation is the same, shared RAM is set as scalar memory. The autonomous operation of tasks $i_s(t)$ and $x_{int}(t)$ can be completed by providing coefficient matrix first address MA, multiplication vector first address MB, output first address MY, row operation length m and number n as shown in Figure 6b,c. The coefficient of $x(t)$ task cannot be continuously stored in the local RAM because the address of x_{int} is fixed in the local RAM while the value address of the E_k coefficient matrix changes with the sub-network operating state k , an additional x_{int} address MX is required to read the coefficient matrix as shown in Figure 6d.

Figure 6. Data storage design of AMA. (a) Data storage order of AMA; (b) Data addressing requirements and process of $i_s(t)$ task; (c) Data addressing requirements and process of $x_{int}(t)$ task; (d) Data addressing requirements and process of $x(t)$ task.

From the above analysis process, the AMA is designed, as shown in Figure 7. The OP flag in the startup instruction is used to distinguish whether MX is used for coefficient matrix addressing. The address generation unit generates a data address in cooperation with the row and column counter, and the read–write controller completes the data read–write operation of memory and the calculation unit port according to the address. The numbers in the figure are pipeline stages, the data in the floating-point arithmetic unit are double floating-point numbers, and the accumulation channel converts the double floating-point number into a fixed-point number to reduce the accumulation pipeline delay. In order to reduce the loss of accuracy, the fixed-point number is as wide as possible, and a 140 (40.100) bit fixed-point number is used in this paper.

Figure 7. Algorithm modularization architecture.

3.4. Hybrid Architecture Design

After introducing the AMA into the instruction flow-driven architecture, the hybrid architecture is shown in Figure 8. Multiple PE and AMA can be designed on FPGA, and the number is limited by FPGA resources.

Figure 8. Hybrid architecture based on FPGA.

3.4.1. PE Structure

PE is a high reusability calculation unit, which needs to implement all formula types in tasks 1, 2 and 3. Statistical analysis is carried out for these task formula types. The formulas with high frequency and their descriptions are shown in Table 1.

Table 1. Task formula type and description.

Formula Type	Formula Description
$Y = \sum A$	Injection current source calculation. Formation of system node voltage equation.
$Y = A \times B + C$	Historical current source calculation, back-substitution calculate the node voltage, branch current update.
$Y = A \times B/C + D$	Node elimination calculation.
$Y = A + B$	Branch voltage calculation.

According to the formula type in Table 1, the PE structure is designed as shown in Figure 9. The numbers in the figure are pipeline stages, and A, B, C and D are calculation input channels; the Y channel completes the output of the main formula. The delay from each input channel to the Y channel makes the pipeline stages output from the Y channel the same under different data flow directions so as to ensure that there will be no conflict in the pipeline and cause flow interruption. The Z channel completes special operations and data transmission. Special operations include logarithmic operations, exponential operations and other operations with fewer occurrences and long pipeline length.

Figure 9. PE structure.

PE instruction can be divided into two parts, the data address instruction of the port and control instruction. The port address instruction is sent to the read–write controller; The control instruction completes the operation type selection of the arithmetic unit.

3.4.2. Ping-Pong Operation

The instruction storage is divided into two parts, the instruction storage of PE and the start instruction storage of AMA. The depth of the instruction RAM of the PE is large, which is related to the simulation step length Δt and the FPGA operating frequency f . The storage depth is 10,000 when the simulation step is 50 μs , and the operating frequency is 200 Mhz.

AMA instruction RAM only stores the startup instruction, the transmission frequency of the start command is low, and its storage depth is set to 512 according to the minimum depth of the on-chip RAM. This also reflects the reduction in instruction costs. The coefficient memory is divided into two parts, PE stores the detailed sub-network coefficients, and AMA stores the simpler sub-network coefficients. When changing the detailed sub-network, the instruction RAM and PE Coe RAM perform a ping-pong operation at the end of the simulation step; AMA local RAM has a low proportion of changing coefficients, so the coefficients of all sub-networks can be stored to avoid ping-pong operation. When the parameter setting operation of the detailed sub-network occurs, the coefficient modification is also completed through the ping-pong operation of the PE Coe RAM. At the same time, the parameter modification detailed sub-network also changes the corresponding coefficients in the AMA through the download channel so that the correct coefficients can still be obtained when the sub-network becomes a simple sub-network.

3.4.3. Indexing Unit Design

Most of the data addresses in the instruction memory correspond to the position of the data in the memory and can be directly addressed by the address. However, some data will change during operation, such as the calculation coefficient of nonlinear components. The piecewise linearization strategy is adopted in this paper so that the coefficients will change according to the network state during operation. The detailed sub-network shows the change in the difference in the coefficient of the node analysis method. In a simple sub-network, the value of the entire coefficient matrix will change. These coefficients are stored in their respective coefficient RAM. The instructions in the instruction memory should be indexed according to the network operation state to find the correct coefficient address. This process is an indirect addressing process.

The instructions requiring indirect addressing are the port address of PE and the MA coefficient matrix address instructions of AMA. The indirect addressing circuit is shown in Figure 10. Store the indirect addressing coefficients at a fixed offset, and the influence word is the current state of the component. Indirect addressing can be completed by providing the first address, decoding method and influence word. These parameters are stored in the index guidance RAM; when indirect addressing is required according to the address range of the port instruction, the port address is taken as the index value of the index guidance RAM, and the parameter is taken out to complete the indexing. The index guidance RAM of PE performs ping-pong operation together with the Coe RAM of PE.

Figure 10. Hybrid architecture indexing unit.

4. Validation

4.1. Architecture Design Scale

The Xilinx vc709 development board is selected to build a hybrid architecture real-time simulation platform. The FPGA chip used is xc7vx690t-2ffg1761. The chip contains 433,200 Slice LUTs, 3600 DSP chips and 147,036 K dual-port Bram. Taking 200 MHz as the timing constraint, in order to compare the differences between different structures, the PE

in Figure 9 is constructed in Architecture 1 to form an instruction flow driven architecture without instruction ping-pong, Architecture 2 implements instruction ping-pong based on the instruction flow driven architecture composed of PE and the Hybrid Architecture are designed, respectively. The maximum design scale and resource consumption are shown in Table 2. The statistics of parallel computing power are based on the number of input ports of PE and AMA.

Table 2. Architecture design scale and resource consumption.

Resource Usage	Architecture 1	Architecture 2	Hybrid Architecture
Number of calculation units	11 PE	5 PE	5 PE and 5AMA
Parallel computing power	44	20	35
LUT consumption	51.38%	28.84%	35.62%
DSP consumption	37.27%	16.94%	20.2%
Bram consumption/(Instruction RAM)	57.34%/(33.67%)	57.72%/(30.61%)	59.08%/(31.97%)

As shown in Table 2, the utilization rate of Bram resources under the maximum scale of the three architectures is close. Bram resources consume a lot of wiring resources. Under the timing constraint of 200 MHz working frequency, the wiring constraint is high, and the Bram with a high utilization rate limits the scale of architecture design. Architecture 1 has stronger parallel computing power, but the method cannot be switched during the simulation process. At the same time, it can be found that a large amount of architecture Bram consumption is instruction RAM, resulting in Architecture 2 limiting the design scale and reducing the parallel computing power after the instruction RAM ping-pong operation. By introducing AMA with low instruction RAM cost, the hybrid architecture can switch the method and improve the parallel computing power.

4.2. Method Validation and Architecture Validation

The four-machine AC/DC hybrid system shown in Figure 11 is selected as the simulation example. The rated AC voltage of the AC bus on the rectifier side is 345 kV, and the rated AC voltage of the AC bus on the inverter side is 230 kV. There are damping filter devices and capacitive reactive power compensation equipment on the converter buses on both sides. The structure and parameters of the DC part are designed according to CIGRE DC transmission system standard [22,23], and a 12 pulse thyristor converter is adopted for the rectifier and inverter. The rectifier side adopts constant current control, and the inverter side adopts constant current control, low-voltage current-limiting control and arc extinguishing angle control. Faults can be set everywhere in the system, such as line short circuit fault, thyristor fault setting, etc.

Figure 11. The four-machine AC/DC hybrid system.

The parameters of the equipment in the AC system are shown in Table 3.

Table 3. The parameters of the equipment in the AC system.

The Parameters of Line				
Line	R (Ω/km)	L (mH/km)	C ($\mu\text{F}/\text{km}$)	Length (km)
L1	0.0502	1.0335	0.000227	10
L2	0.0502	1.0335	0.000227	15
L3	0.0502	1.0335	0.000227	10
L4	0.0502	1.0335	0.000227	15
The Parameters of Transformer				
transformer	$S_N(\text{MVA})$	$U_k\%$	$I_0\%$	$\Delta P_0(\text{kW})$
parameter	610	14	0.1	1279.1
The Parameters of Synchronous Generator				
generator	S_N	X_d	X'_d	X''_d
parameter	600MVA	2.27 p.u.	0.3 p.u.	0.22 p.u.
generator	T'_{d0}	T''_{d0}	T'_{q0}	T''_{q0}
parameter	4.3 s	0.05 s	0.85 s	0.07 s
				X_δ
				0.134 p.u.
				R_a
				0.002 p.u.

This paper does not discuss the division of sub-networks and follows the basic principle of division: the sub-network should be as large as possible, and the storage capacity should be within a reasonable range when solving with the state-space method. Synchronous generator involves coordinate transformation and does not participate in sub-network division. The sub-network is divided, as shown in Figure 11. Table 4 shows the number of expressions and coefficient storage of each sub-networks under detailed sub-networks and simple sub-networks.

Table 4. The number of expressions and coefficient storage of each sub-networks.

Sub-Network	Number of Expressions			Coefficient Storage		
	Detailed Sub-Network (Node Analysis Method)	Detailed Sub-Network (State Space Method)	Simple Sub-Network (State-Space Method)	Detailed Sub-Network (Node Analysis Method)	Detailed Sub-Network (State Space Method)	Simple Sub-Network (State-Space Method)
1#	3214	1242	1242	148	$2^2 \times 2562$ ¹	2562
2#	2835	432	432	64	$2^6 \times 4916$	4916
3#	2486	398	398	61	$2^6 \times 3686$	3686
4#	1247	673	673	40	$2^2 \times 1228$	1228
5#	2835	432	432	64	$2^6 \times 4916$	4916
6#	2486	398	398	61	$2^6 \times 3686$	3686
7#	3214	1242	1242	148	$2^2 \times 2562$	2562
8#	4652	1856	1856	236	$2^4 \times 5148$	5148
9#	4652	1856	1856	236	$2^4 \times 5148$	5148

¹ The exponential bit of 2 represents the number of faults that can be set in the subnetwork.

It can be seen from Table 4 that the state-space method adopted by the sub-network has less computation; the simple subnetwork greatly reduces the number of faults to be stored in the state space equation. Therefore, the variable detail sub-network SSN method can speed up the network solution process and reduce the storage demand.

Architecture 1 adopts the node analysis method for the whole system, and Architecture 2 and hybrid architecture adopt variable detail sub-network SSN method. When selecting 8# subnetworks as detailed sub-network and other subnetworks as simple sub-networks, the completion time of the simulation example in the three architectures is shown in Table 5.

Table 5. Simulation completion time in three architectures.

Architecture 1	Architecture 2	Hybrid Architecture
17.14 μ s	21.8 μ s	13.82 μ s

Although Architecture 2 reduces the amount of computation through the variable detail sub-network SSN method, the decline of parallel computing power increases the simulation time. By introducing AMA, the hybrid architecture increases the computing power and shares some computing tasks, which can reduce the simulation completion time.

4.3. Simulation Results

In order to verify the accuracy of hybrid architecture real-time simulation platform based on FPGA, the simulation results are analyzed and compared with PSCAD simulation results.

Set 7# as a detailed sub-network at the beginning of the simulation. At $t = 0.2$ s, set the three-phase metallic grounding fault of the converter bus at the inverter side and clear the fault after 0.2 S. Figure 12 shows the waveform of AC current at the inverter side in the FPGA and PSCAD. At $t = 5$ s, set 5# as detailed sub-network and change 7# to a simple subnetwork. When $t = 5.2$ s, set the thyristor trigger pulse of the inverter side converter to be lost, and clear the fault after lasting for 0.2 S. Figure 13 shows the waveform of AC current at the inverter side in the FPGA and PSCAD. The correctness of the simulation platform designed in this paper can be verified by Figures 12 and 13. The main source of the error of the simulation platform is the accuracy loss in the process of AMA floating-point number conversion.

Figure 12. Waveform of AC current at inverter side in the FPGA and PSCAD.

Figure 13. Waveform of AC current at inverter side in the FPGA and PSCAD.

5. Conclusions

This paper proposed variable detail sub-network SSN method, the method changing in the simulation process not only balances the amount of calculation and storage but also can complete the complex test process in the real-time simulation process. In order to satisfy the flexibility of the method, a real-time simulation platform of hybrid architecture based on FPGA was designed. Compared with instruction flow-driven architecture, it improves the computing power while maintaining flexibility. The simulation platform has better adaptability to simulation systems with large-scale and complex test requirements. However, for a simulation system with low test requirements, it is difficult to fully utilize the performance of the simulation platform.

Observing the instruction cost consumption of hybrid architecture shows that PE instruction storage still occupies a lot of Bram resources, which limits the scale of architecture that can be designed in FPGA. The reduction in PE instruction storage can be studied from the instruction similarity. For example, the solution process of the node analysis method for the same equipment is the same, and the solution process of system node voltage equation can be fine-grained decomposed, instructions can be compressed through instruction similarity, which will be carried out in future research work.

Author Contributions: Conceptualization, R.C. and X.Y.; data curation, Z.J.; formal analysis, R.C., L.Y. and X.Y.; methodology, R.C. and X.Y.; project administration, L.Y. and B.Z.; resources, R.C. and X.Y.; software, R.C., X.Y. and Z.J.; supervision, L.Y. and B.Z.; validation, R.C., X.Y. and Z.J.; visualization, R.C.; writing—original draft, X.Y.; writing—review and editing, R.C., L.Y. and X.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: Thanks for the experimental site provided by the Key Laboratory of the Ministry of Education for Smart Grid of Tianjin University.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

Abbreviations used in the article and their corresponding meanings:

FPGA	Field-Programmable Gate Array.
AMA	Algorithm Modularity Architecture.
SSN	State Space Nodal.
HIL	Hardware-in-the-Loop.
SIMD	Single Instruction Multiple Data.
PE	Processing Element.

References

- Lin, B.; Xu, M. Regional differences on CO₂ emission efficiency in metallurgical industry of China. *Energy Policy* **2018**, *120*, 302–311.
- Wen, Y.; Cai, B.; Xue, Y. Assessment of Power System Low-carbon Transition Pathways Based on China's Energy Revolution Strategy. *Energy Procedia* **2018**, *152*, 1039–1044.
- Liu, H.B.; Bian, D.; Sun, L.; Yun, Z.J.; Li, Y. Research on Electromechanical-Electromagnetic Transient Hybrid Simulation of AC/DC Hybrid System. *Power Syst. Prot. Control* **2019**, *47*, 39–47.
- Li, Y.L.; Zhang, X.; Li, Y.J.; Chen, Z.J.; Wu, M.Q. Current Situation and Challenges of Simulation Technology for AC/DC Hybrid Power Grid. *Electr. Power Constr.* **2015**, *36*, 1–8.
- Li, W.; Bélanger, J. An Equivalent Circuit Method for Modelling and Simulation of Modular Multilevel Converters in Real-Time HIL Test Bench. *IEEE Trans. Power Deliv.* **2016**, *31*, 2401–2409.
- Estrada, L.; Vázquez, N.; Vaquero, J.; de Castro, Á.; Arau, J. Real-Time Hardware in the Loop Simulation Methodology for Power Converters Using LabVIEW FPGA. *Energies* **2020**, *13*, 373.
- Tomim, M.A.; Martí, J.R.; De Rybel, T.; Wang, L.; Yao, M. MATE network tearing techniques for multiprocessor solution of large power system networks. In Proceedings of the IEEE PES General Meeting, Minneapolis, MN, USA, 25–29 July 2010; pp. 1–6.
- Hollman, J.A.; Martí, J.R. Real time network simulation with PC-cluster. *IEEE Trans. Power Syst.* **2003**, *18*, 563–569.
- Li, P.; Wang, Z.Y.; Wang, C.S. Design of Parallel Architecture for Multi-FPGA Based Real-time Simulator of Active Distribution Network. *Autom. Electr. Power Syst.* **2019**, *43*, 174–182.
- Zhou, Z.; Dinavahi, V. Parallel massive-thread electromagnetic transient simulation on GPU. *IEEE Trans. Power Deliv.* **2014**, *29*, 1045–1053.
- Ould-Bachir, T.; Saad, H.; Denetiere, S.; Mahseredjian, J. CPU/FPGA-Based Real-Time Simulation of a Two-Terminal MMC-HVDC System. *IEEE Trans. Power Deliv.* **2017**, *32*, 647–655.
- Saad, H.; Ould-Bachir, T.; Mahseredjian, J.; Dufour, C.; Denetiere, S.; Nguefeu, S. Real-Time Simulation of MMCs Using CPU and FPGA. *IEEE Trans. Power Electron.* **2015**, *30*, 259–267.
- Song, Y.; Chen, Y.; Huang, S.; Xu, Y.; Yu, Z.; Xue, W. Efficient GPU-Based Electromagnetic Transient Simulation for Power Systems With Thread-Oriented Transformation and Automatic Code Generation. *IEEE Access* **2018**, *6*, 25724–25736.
- Chen, Y.; Dinavahi, V. FPGA-Based Real-Time EMTP. *IEEE Trans. Power Deliv.* **2009**, *24*, 892–902.
- Wang, C.S.; Ding, C.D.; Li, P.; Yu, H. Real-time transient simulation for distribution systems based on FPGA, Part I: Module realization. *Proc. Chin. Soc. Electr. Eng.* **2014**, *34*, 161–167.
- Wang, C.S.; Ding, C.D.; Li, P.; Yu, H. Real-time transient simulation for distribution systems based on FPGA, Part II: System architecture and algorithm verification. *Proc. Chin. Soc. Electr. Eng.* **2014**, *34*, 628–634.
- Zhang, B.D.; Fu, S.W.; Jin, Z.; Hu, R.Z. A Novel FPGA Based Real-Time Simulator for Micro-Grids. *Energies* **2017**, *8*, 1239–1255.
- Zhang, B.D.; Hu, R.Z.; Tu, S.J.; Zhang, J.; Jin, X.L.; Guan, Y.; Zhu, J.J. Modeling of Power System Simulation Based on FRTDS. *Energies* **2018**, *11*, 2749–2766.
- Zhang, B.D.; Wang, Y.; Tu, S.J.; Jin, Z. FPGA-Based Real-Time Digital Solver for Electro-Mechanical Transient Simulation. *Energies* **2018**, *11*, 2650–2669.
- Dufour, C.; Mahseredjian, J.; Bélanger, J. A Combined State-Space Nodal Method for the Simulation of Power System Transients. *IEEE Trans. Power Deliv.* **2011**, *26*, 928–935.
- Wasynczuk, O.; Sudhoff, S.D. Automated state model generation algorithm for power circuits and systems. *IEEE Trans Power Syst.* **1996**, *11*, 1951–1956.

-
22. Atighechi, H.; Chiniforoosh, S.; Jatskevich, J. Dynamic Average-Value Modeling of CIGRE HVDC Benchmark System. *IEEE Trans. Power Deliv.* **2014**, *29*, 2046–2054.
 23. Kwon, D.; Moon, H.; Kim, R. Modeling of CIGRE benchmark HVDC system using PSS/E compared with PSCAD. In Proceedings of the 2015 5th International Youth Conference on Energy (IYCE), Pisa, Italy, 27–30 May 2015; pp. 1–8.