ISSN 1392 - 1215 ELEKTRONIKA IR ELEKTROTECHNIKA. 2005. Nr. 3(59)

T 125 AUTOMATIZAVIMAS, ROBOTOTECHNIKA

Hardware Emulation of Large Scale Boolean Equations Systems

S. M. Hyduke
Aldec Inc., 2260 Corporate Circle Henderson, NV 89074 USA, phone: (702) 990-4400
V.I. Hahanov
AD Department, Kharkov national University of Radio Electronics,
Ukraine, Kharkov,61166, Lenin ave., 14 , phone: (380)57- 70-21-326,e-mail:hahanov@kture.kharkov.ua
O.V. Melnikova
AD Department, Kharkov national University of Radio Electronics,
Ukraine, Kharkov,61166, Lenin ave., 14 , phone: (380)57- 70-21-326,e-mail:hahanov@kture.kharkov.ua
I.V. Hahanova
AD Department, Kharkov national University of Radio Electronics,
Ukraine, Kharkov,61166, Lenin ave., 14 , phone: (380)57- 70-21-326,e-mail:hahanov@kture.kharkov.ua

Introduction

The PRUS (Programmable Unlimited Systems) technology is the newest device architecture for quick, efficient and inexpensive digital circuit implementation. The PRUS device architecture is based on massively parallel processors that communicate with each other at high speeds without any hardware handshake [1].

PRUS is a vast processor network that processes blocks of logic functions and provides for exchange of intermediate results between all processors. Since PRUS needs only 48 bits of memory to emulate a dual input gate, a 256 Mbytes RAM or ROM memory allows emulating designs well over 20 million ASIC gates. Flip-flops require from 48 to 64 bits of RAM memory and can be freely intermixed with gates in any circuit combination, providing for total design freedom.

Since basic network processors can be built with less than 90 gates each, PRUS devices with hundreds of such processors can be made even with today's silicon technology. Because of its memory efficiency in emulating gates and flip-flops, PRUS will be the lowest cost control device for industrial, medical, security, military and other applications. It is particularly well suited for signal processing, routers, pattern recognition and encrypting devices.

Actuality

There are several technologies to process Boolean equations containing million lines or equivalent gates [2,3].

1. Personal computer or workstation based on Intel microprocessor can be used. Each equation will be processed sequentially using software approach, because only one processor exists although high-performance. The cost and time consumptions of this solution are very high.

2. Specialized parallel processor based on PLD. In this case high level of parallelism processing of equations compensates relatively low (compared to CPU) clock speed. This reprogrammable solution is absolute winner in performance. But significant drawback is absence of flexibility, which typical for software approaches. Moreover, implementation into PLD is high cost if the volume of future distribution will be tens of thousands chips.

3. The third solution is closely related with integration of CPU, PLD and ASIC advantages, such as [4]:

1) Flexibility of Boolean equations programming, which allows on-the-fly editing specification in form of source code.

2) Minimal possible instruction set, which allows simple circuit solutions.

3) Parallelization of Boolean equations processing, because of PLD ideology, but with CPU elements, which means having lots of interconnected single-bit processors with simple instruction set for parallel programming.

4) Multiprocessor implementation into ASIC, which allows maximum clock rate and minimal cost per chip for large manufacturing volumes (more 10 000), low power consumption.

5) Pipelining of Boolean equations processing is exclusive property, which typical to multiprocessor systems, where pipelined processing is one of the main operation mode besides parallel and sequential ones.

Thus necessity in multiprocessor for Boolean equations processing with properties mentioned before, conditioned by current microelectronics market trends. What does it mean? In the nearest future actual and complex computational problems will be solved by specialized micro- and multiprocessor. Customization index is growing on the microelectronics market. When scope of universal computers begins to exceed needs of

current market sector, business competition shifts accents from increasing overall performance to properties improvements. End-user is ready to pay additional money for that. This is not only convenience and reliability, but high-performance solution of complex problem, custom properties, power saving, autonomy, miniaturization, flexibility and programmability.

PRUS Architecture

Structure of PRUS Matrix. PRUS is a system of many simple logic processors (sequencers) connected in matrix of various sizes: 4x4, 8x8, 16x16, etc. The processors can be connected "in chain", as "honey-comb" network and in other combinations (Fig.1). Some asymmetrical connectivity among the processors was also tried for improved program memory utilization.

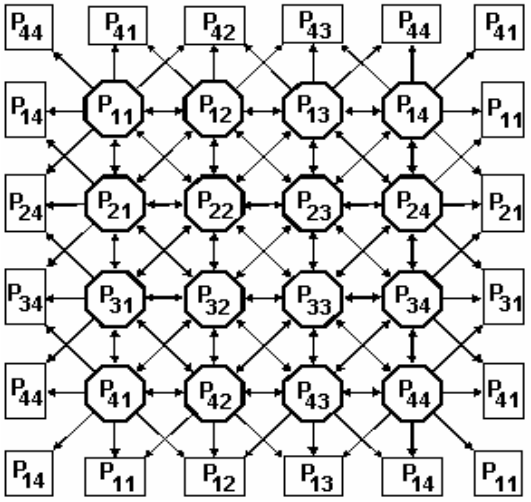

Fig. 1. PRUS Matrix

Sequencers are logic processors that do not have jump, branch or similar instructions that change the order of instruction execution. How a set of synchronized sequencers connected in a network can be used for quick and efficient processing of system level designs is described below.

Since the PRUS technology is based on mathematical algorithms, they provide quick and automatic allocation of design sections to processors in the network. As a result, the designer knows up front how fast his or her design will run in PRUS. No manual tweaking is needed for improving memory utilization or device performance. Each sequencer can be interacting with its eight neighboring sequencers to exchange data.

This allows splitting logic equations between neighboring sequencers and provides for handling larger designs and greater design flexibility.

Each sequencer has its own Program Memory for storing binary instructions executing the assigned logic equations and internal RAM Data Memory for storing the results of logic equation processing. During device programming each Program Memory is loaded with its own set of machine instructions representing the associated logic equations. Since all sequencers and the associated program memories are driven preferably by a single common address register, they operate and execute logic equations synchronously.

To speed design execution, multiple sequencers are used for parallel processing of logic equations. The PRUS Matrix is comprised of a set of sequencers, having Program Memories driven by a common Address Counter for the entire processor network.

The Structure of a Sequencer. Sequencers emulate Boolean expressions. All sequencers in a network operate synchronously on streams of single-bit input logic data and perform logical "and", "or" and "xor" operations. The resulting emulation data are saved in local Data Memory. In order to synchronize the entire system, each PRUS sequencer in a network has its Program Memory addressed by the same common Address Counter (Fig. 3).

Program Memory stores instructions for the entire simulation cycle. It operates under control of input address lines and produces instruction words, shown by red color. The instruction word controls operation of all blocks in a sequencer. Program Memories of all sequencers can be combined into a single memory addressed by the same Address Counter.

Data Memory is used for storing data produced by execution of logic equations by the sequencer. Typically, the lower 10 to 14 lower bits of the instruction word serve as Data Memory address, addressing from 1024 to 16,384 RAM locations.

InMux – Input Multiplexer is used for feeding external data into the sequencer. It can select the data from one of eight neighbor processors (NP) or from one of eight input bits when Mux_En signal is active. Otherwise, it selects data from the internal Data Memory.

Single Bit Processor (SBP) – performs two concurrent AND and XOR logic operations on true or inverse input values. They are performed by means of AND Block and XOR Block flip-flops. The SBP is a non-Van Neumann processor because it processes first all input data and after all "read input data" operations have been completed, an "output" instruction selects output either from the AND Block or XOR Block. In place of AND or XOR flip-flops an OR operator can also be used.

SBP Mux –Single Bit Processor Multiplexer selects either the AND Block or XOR Block data to the SBP output. It can select ANDT (operation AND-True), ANDI (operation AND-Not), XOR block output or data from Input Multiplexer for fast transmission to the neighboring sequencers as NP signal.

Output Buffer – provides data to neighboring processors via the NP signal line. It holds data for several clocks so that it can be "read" by the neighboring processors.

Output Decoder – it holds data till is reset by another PRUS instruction,

Instruction Decoder – decodes special instructions, such as NOOP or "no operation", END of working program and others, listed in section Instructions for PRUS Martix. It produces control signals for Output Decoder, Data Memory, Input Multiplexer and Single Bit Processor. The Instruction Decoder also produces the END_INSTR signal, which controls the Program Memory Address Counter (see Fig.2). When active, the END_INSTR resets the Address Counter.

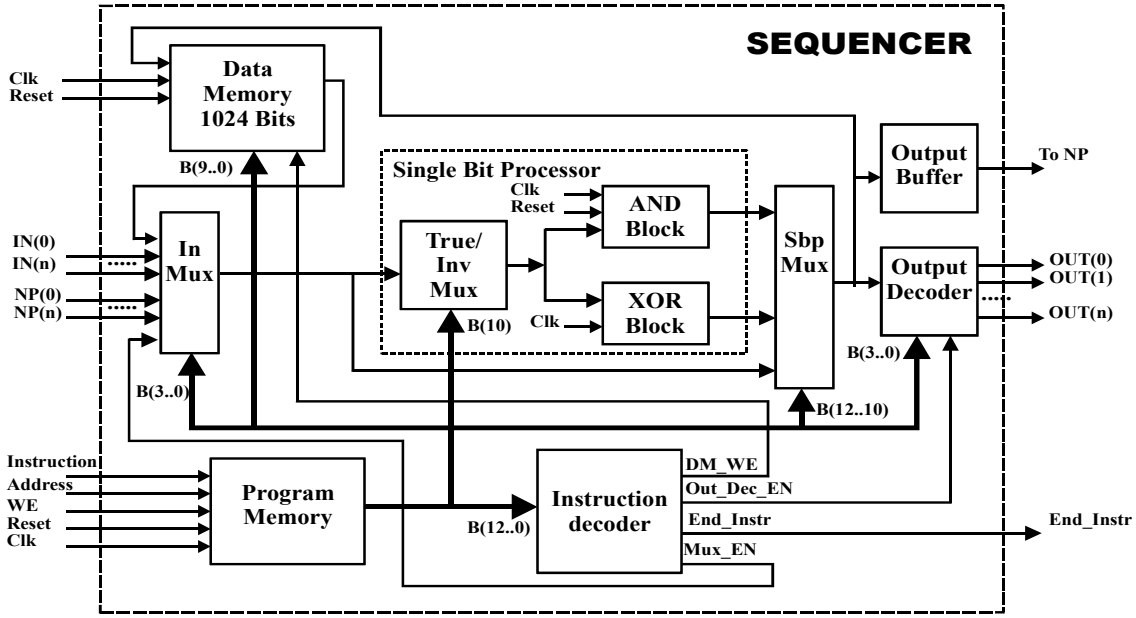

Fig. 2. General structure of one sequencer

The Structure of the Single Bit Processor

The single bit processor (SBP) is at the heart of sequencer operations. It performs the basic logical operations such as "and", "not" and "or". Fig. 3 shows embodiment of a single bit processor that facilitates reverse Polish notation (RPN) machine instruction execution. The processor is performing in parallel the AND and OR logic operations on all input signals. Those concurrent operations are executed by the *AND Block* and *OR Block*, comprised of one logic element and one-bit register.

Since de Morgan theorem allows conversion of AND operations into OR operations and vice versa, the concurrent presence of both registers AND and OR is not mandatory.

For example, the register OR can be replaced by a block performing the XOR logic operation (Fig. 3).

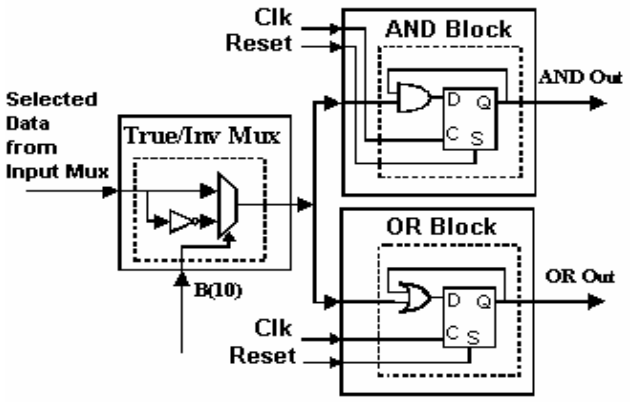

Fig. 3. Structure of Single Bit Processor with AND and OR blocks

In both versions (Fig. 3 and Fig.4) the AND register is set to logical '1'at the beginning of each logic equation execution, and it will be reset by the first logical'0'. The OR register (Fig.3) is set to logical '0' at the beginning of logic equation processing and will permanently be set to '1' by the first logical '1' that appears on its input. Register XOR (Fig.4) doesn't require setting to any value caused by its combinatorial operation. Initializing of registers is invoked by having '1' in the highest bit position of instruction word.

True/Inv Multiplexer selects either 'True' or 'Not' input signal value and passes it to the SBP processor. This selector is under control of the second highest bit of the instruction word. The True/Inv Multiplexer is fed input data by the Input Multiplexer that selects data either from input signal lines (IN), one of the neighboring processors (NP) or from internal Data Memory.

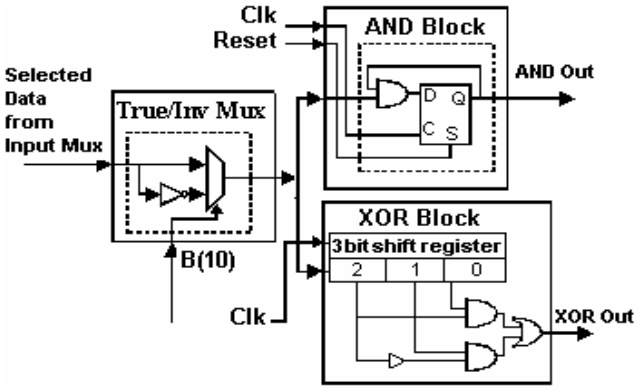

Fig. 4. Structure of Single Bit Processor with AND and XOR blocks

PRUS Design Flow

PRUS Design Flow consists of three main parts:

1) Converting the design into set of Boolean equations;

2) Distribution of Boolean equations between the processors;

3) Emulation of design in hardware.

PRUS can work with designs inputted in the form of schematics, VHDL and Verilog design files [5,6,7].
Designs can be represented by HDL code, tables, FSM flowcharts and functional statements. Both RTL and Behavioral designs can be supported.

First, hardware description language files are converted into an equivalent set of Boolean logic equations. Next, these equations are mathematically optimized, converted into binary instruction code using the RPN method and distributed between sequencers according to an algorithm that simplifies communication between design sections located in different sequencers. For optimum utilization of silicon resources, approximately the same number of instructions is assigned to each sequencer.

Because of the synchronous operation of sequencers, the compiler that distributes logic equations between them can calculate in advance at what time the result of each logic operation will be provided on the selected sequencer output. The compiler can then make such arrangement of instructions in the interconnected neighboring sequencer that it will be ready to read this output as it occurs and without any 'hand-shake' control signals that are typically used to facilitate communication between processors.

The design tools for development of devices under PRUS are simple and operating about two orders of magnitude faster than the tools for designing devices employing the current silicon architectures. Specifically, only the functional simulation of the design will be needed. Since the number of instructions in a program memory and the operating clock speed determine the sampling rate of the logic circuit located within the sequencer, there is no need for timing analysis. This sampling rate is the maximum response time of the sequencer to the external signals. By lowering the number of instructions in a sequencer and increasing its clock speed, this sampling rate can be increased accordingly.

PRUS Emulation in Hardware

A model of a programmable device operating as a network of 64 processors (PRUS for PLD), downloaded into HES3-3000 (AllTech) with FPGA Virtex II and 256 Mb RAM has been designed and tested. It emulated correctly tens of digital designs. While the Design Compiler (Synopsys) was spending the same amount of time compiling designs for FPGAs and PRUS, the distribution of logic equations (40000 gates) in PRUS took less than 10 minutes. Also, the entire PRUS design flow was running from a script, resulting in flawless operation of the RTL design on a network of processors. On the other hand, the place and route of the same design in FPGA took several hours and required expert knowledge of the tools and design process.

The entire processor network has been designed for hardware and emulation code efficiency. It is expected that over 90% of the chip area will be used by highly structured memory, resulting in efficient use of silicon. Also, because of small number of registers in the device (2 flip-flops per processor), there will be low power dissipation even when emulating large systems. Test stimulus is fed to the system from Active-HDL through PLI interface.

Conclusions

The main results of PRUS technology are scientific novelty and practical application. The scientific novelty is defined by following points:

1. The new high-performance PRUS technology for parallel processing of Boolean equations has been proposed.

2. The architecture of multiprocessor, where every single-bit processor is connected with eight neighbors, has been developed.

3. The structure of single-bit processor has been represented. It performs AND, OR, NOT, XOR operations. The structure of sequencer has been represented also. The main parts of it are: single-bit processor, program memory, data memory, control circuit.

4. The model of calculations has been proposed. The logic equations are distributed uniformly between processors using dichotomy method.

The practical application of PRUS technology shows the advantages of its appearing on EDA market in comparison with existing technologies:

1. The present technologies also require cumbersome and labor-intensive critical path timing analysis of the routed designs. Some large FPGA designs need to be rerouted over forty times before the desired operational speed is achieved. PRUS methodology eliminates entirely the timing analysis. The first compilation is the only one designer will ever need.

2. The current device architectures that use deep sub-micron silicon technologies require complex analysis of cells and their connections to determine the overall design performance. New physical phenomena are playing larger role at higher cell densities, making the layout analysis continuously more complex. It is becoming now quite apparent that some form of incremental compilation will be necessary for the layout of high-density deep sub-micron devices. However, such incremental silicon compilation will require a substantial human involvement, which will slow even more the design process. It is thus another object of PRUS to eliminate any manual physical layout operations.

3. Efficient testing of complex devices requires placing additional boundary-scan circuits on the silicon. This makes designing a complex operation and lowers the effective utilization of silicon area. However, since there is no other good way to test the silicon, this process is widely applied to ASIC devices in excess of 100,000 gates. It is yet another object of PRUS to provide for effective device testing without any additional boundary scan or similar circuits.

4. Due to a random nature of cell utilization, large areas of the silicon are set aside to facilitate connections between cells in gate arrays, CPLDs and FPGAs. This lowers the effective utilization of the silicon. PRUS is based on highly regular memory architecture and does not require design dependent interconnect areas, thus improving the silicon utilization.

5. The CPLD and FPGA reprogrammable devices put severe restrictions on the connections between gates and flip-flops. The number of used gates and flip-flops, even in the largest FPGA devices, is thus strictly limited. The PRUS technology on the other hand allows total freedom in emulation of either gates and/or flip-flops in RAM. With millions of flip-flops available in PRUS, it is thus particularly well suited for sequential circuit applications.

6. The current technologies dissipate large amount of heat because circuits operate in parallel. This limits the design size that can be placed on silicon. PRUS uses a parallel-serial circuit operation, which lowers power dissipation and allows considerably higher circuit densities. Moreover, since the basic PRUS processor is made with only two flip-flops and less than ninety cells, such processor is dissipating a small amount of energy.

7. The current silicon compilers are based more on art than strict mathematical algorithms. As a result, designers must manually tweak some of the circuits for better performance or improved area utilization. This requires high level of expertise, constant employee education and trial and error approach for best results. Since PRUS is based on mathematical algorithms, it provides fully automated design environment, which eliminates manual tweaking of designs and lowers the level of expertise demanded from the designer.

8. Since the current design tool technology is sensitive to the physical phenomena in silicon, designers continuously need to buy newer and more advanced software. PRUS isolates the designer from changes in the silicon technologies so that one tool set will be able to handle all future silicon process enhancements.

References

1. **Hyduke S.** *U.S. Patent 6,578,133*, June 10, 2003.
2. **Baneres D., Cortadella J., Kishinevsky M.** A Recursive Paradigm to Solve Boolean Relations // Proceedings of Design Automation Conference, P. 416 - 421.
3. **Richard J.** Discrete Mathematics, Prentice Hall 2001, 621 p.
4. **Voros N.S., Sanchez L., Alonso A., Birbas A.N., Birbas M., Jerraya A.** Hardware-Software Co-Design of Complex Embedded Systems. Design Automation for Embedded Systems.– Boston: Kluwer Academic Publishers. – 2003. – P.5-34.
5. Active-HDL User's Guid. Second Edition. – Copyright. – Aldec Inc. – 2003. – 213 p.
6. **Palnitkar S.** Verilog HDL. A Guide to digital design and synthesis. Sunsoft Press. A prentice Hall Title, 2002. – 396 p.
7. **Roth C.H.Jr.** Digital Systems Design UsingVHDL. PWS Publishing Company, 20 Parkl Plaza, Noston, MA 02116 ISBN. – 470 p.

Pateikta spaudai 2005 01 15

S.M. Hyduke, V.I. Hahanov, O.V. Melnikova, I.V. Hahanova. Didelių Bulio lygčių sistemų aparatūrinė emuliacija // Elektronika ir elektrotechnika. – Kaunas: Technologija, 2005. – Nr.3(59). – P.9–13.

Siūloma naši Bulio lygčių sprendimo technologija pagrįsta sferinio vieno bito multiprocesoriaus PRUS (Programmable Unlimited Systems), realizuoto ASIC kristale panaudojimu. Jis leidžia atlikti lygiagretų, nuoseklų ir konvejerinį Bulio lygčių, užrašytų operacijų AND, OR, NOT, XOR pagrindu, apdorojimą. Multiprocesorius yra ekonomiškas – lygčių sistemos iš 20 milijonų ventilių apdorojimui užtenka tik 256 Mb operatyviosios atminties. Il. 4, bibl. 7 (anglų kalba; santraukos lietuvių, anglų ir rusų k.).

S.M. Hyduke, V.I. Hahanov, O.V. Melnikova, I.V. Hahanova. Hardware Emulation of Large Scale Boolean Equations Systems // Electronic and Electrical Engineering. – Kaunas: Technologija, 2005. – Nr. 3(59). – P. 9–13.

This paper offers high-performance technology for processing Boolean equations, based on Compiler Synchronized Parallel-processor Network-based Logic Device PRUS (Programmable Unlimited Systems) - single-bit spherical multiprocessor, which implemented into ASIC. This technology allows to perform parallel, sequential and pipelined Boolean equations processing using AND, OR, NOT, XOR operations. Multiprocessor is very efficient in hardware implementation – e.g. 256MB RAM is enough for processing Boolean equations containing 20 millions gates. Ill. 4, bibl. 7 (in English, summaries in Lithuanian, English, Russian).

S.M. Hyduke, В.И. Хаханов, О.В. Мельникова, И.В. Хаханова. Аппаратная эмуляция систем Булевых уравнений большой размерности // Электроника и электротехника. – Каунас: Технология, 2005. – № 3(59). – P. 9–13.

Предлагается высокопроизводительная технология решения булевых уравнений, основанная на использовании сферического однобитного мультипроцессора PRUS (Programmable Unlimited Systems), реализуемого на кристалле ASIC. Он позволяет осуществлять параллельную, последовательную и конвейерную обработку булевых уравнений, записанных в базисе операций AND, OR, NOT, XOR. Мультипроцессор экономичен в аппаратурном исполнении – для обработки системы уравнений, насчитывающей 20 миллионов вентилей, необходимо иметь всего 256 Мбайт оперативной памяти. Ил. 4, библ. 7 (на английском языке; рефераты на литовском, английском и русском яз.).