IMPLEMENTATION OF NOVEL PIPELINE VLIW ARCHITECTUR IN FPGA

R.SESHASAYANAN * and Dr S.K.SRIVATSA**

* Lecturer, Department of Electronics and Communication, Anna University , Chennai – 25
** Professor, Department of Electronics and Communication, Anna University, Chennai –25

Summary
Technology has seen the development of processor industry right from micro to the latest Nano-technology with Speed and performance being important criteria, not much attention has been given to the power requirement for these integrated Circuits. Present fully synchronous processors have evolved with a Global clock which is supplied throughout the die; this has resulted in unwanted power consumption and dissipation. The other significant problem is supply of this global clock with a low skew through the entire die. Each and every existing synchronous processor cannot be converted to Asynchronous processors due to certain design constraints such as complexity and compatibility. An architecture, which is a hybrid between a fully synchronous and an Asynchronous processor, termed as Globally Asynchronous and Locally Synchronous processor (GALS) is being incorporated in our design. We propose to analyze a Very Long Instruction Word (VLIW) processor which exploits Instruction Level Parallelism (ILP) to increase the speed of execution. Simulation analysis is done on a prototype VLIW processor under GALS multiprocessor environment to reduce power with same performance.

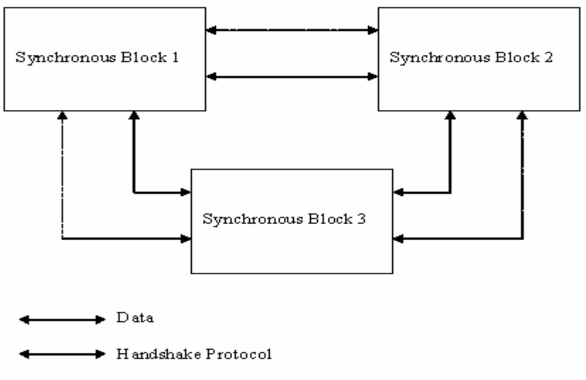
Key words:
Globally Asynchronous Locally Synchronous (GALS), Very Long Instruction Word (VLIW), Field programmable gate array (FPGA) etc

1 Introduction to GALS Design

The trend towards system-on-chip designs leads to chips containing several memories and IP modules which all have different cycle times. In future technologies it will become increasingly difficult to distribute high-speed low-skew clock signals. Therefore, future chips will contain several locally clocked sub modules, which communicate through dedicated glue logic. These heterogeneous systems are called GALS (Globally Asynchronous Locally Synchronous).

The GALS basically consists of a large number of synchronous modules, which are synchronized by a clock locally and communicate asynchronously with other synchronous blocks. As the global clock net gets divided, the constraints of clock skew on the synchronous modules get eased. The GALS methodology is well effective only as long as the overheads due to local clock generation and asynchronous handshaking are kept low compared to the gain achieved from the elimination of global clock. The self-timed approach is efficient, since it does away with the need to time-align the operation of all modules within the framework of a common base clock period. Instead, each module is driven from a local clock generator in its self-timed wrapper [3]. The local generator is made pausable and controlled such as to prevent any timing violations from occurring within the locally synchronous island's data interface. By this method metastability is prevented rather than resolved. No extra latency is required for synchronizers, FIFO. The local clock frequency is chosen to perfectly fit the needs of the particular module. The basic block diagram is shown in figure 1.1

In this paper we try to emulate the importance of high performance pipelined processors under this GALS technology and how efficient they are when it comes to obeying low power norms.



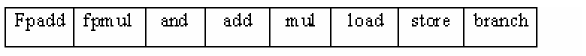
1.1.1 Figure 1 GALS architecture

1.2 VLIW Processor

Very Long Instruction Word (VLIW) is one particular style of processor design that tries to achieve high levels of instruction level parallelism (ILP) by executing long instruction words composed of multiple operations [1,2]. The long instruction word called a

Manuscript received July 5, 2007
Manuscript revised July 25, 2007

MultiOp consists of multiple arithmetic, logic and control operations each of which would probably be an individual operation on a simple RISC processor. The VLIW processor concurrently executes the set of operations within a MultiOp thereby achieving instruction level parallelism. MultiOp consists of many tightly coupled independent operations each of which execute in a small and statically predictable number of cycles. In such a system the task of grouping independent operations into a MultiOp is done by a compiler or binary translator.



Fpadd	fpmul	and	add	mul	load	store	branch

Figure 2 Format of MultiOP

1.3 Conditions for using VLIW Processor

Usage of VLIW processor has a few constraints:

- There must be high degree of ILP in the program. Functional units must be duplicated which posses hardware design difficulties.
- Efficient compiler is required which can predict the path of branch instructions more accurately.

VLIW processor requires duplication of hardware. In this design there are 2 fixed point ALU, 1 fixed point multiplication/division unit, 1 floating point addition/subtraction unit, 1 floating point multiplication/division unit, 1 load unit, 1store unit and 1 branch unit. This hardware duplication facilitates execution of several instructions simultaneously.

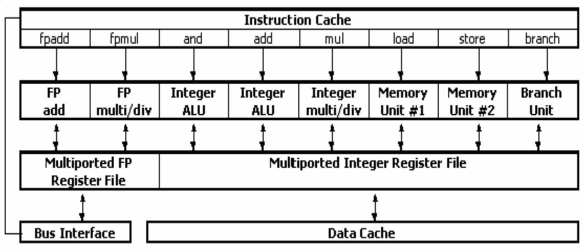
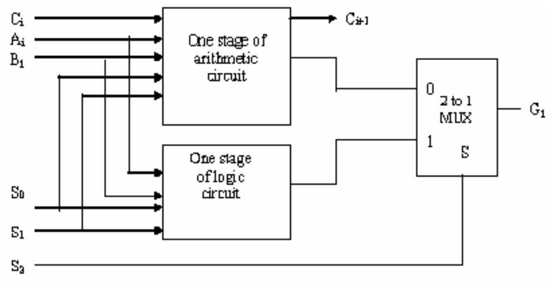


Figure 3 Functional Unit Duplication in VLIW Processor

2. 2 Implementation of functional blocks

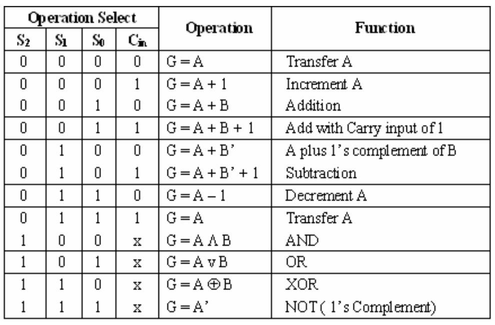
The ALU is capable of handling 16 bits Addition, Subtraction, Increment and Decrement of data simultaneously and it has an input carry bit C_{in} .The outputs are 16-bit sum and carry output bit C_{out}. The logic unit is also capable of handling16 bit data and it performs four basic logical operations AND, OR, NOT and XOR using the two select lines $S_{1,}$ $S_{0.}$ The carry input and output bit are 'don't care' bits for logical operations.

The arithmetic unit and logic unit are differentiated using the Mode select line $S_{2.}$ It takes one bit of inputs $A_{i,}$ B_i and carry C_i and performs the corresponding arithmetic and logic operation based on the operation select lines $S_{1,}$ $S_{0.}$ The output G_i is selected from either arithmetic unit or logical unit output using a 2 to 1 Mux with S_2 as select line. The block diagram and truth table are shown below



2.1.1.1.1.1.1

2.1.1.1.1.1.2 Figure 4 One Bit Operation of ALU



Operation Select				Operation	Function
S_2	S_1	S_0	C_{in}		
0	0	0	0	G = A	Transfer A
0	0	0	1	G = A + 1	Increment A
0	0	1	0	G = A + B	Addition
0	0	1	1	G = A + B + 1	Add with Carry input of 1
0	1	0	0	G = A + B'	A plus 1's complement of B
0	1	0	1	G = A + B' + 1	Subtraction
0	1	1	0	G = A – 1	Decrement A
0	1	1	1	G = A	Transfer A
1	0	0	x	$G = A \wedge B$	AND
1	0	1	x	$G = A \vee B$	OR
1	1	0	x	$G = A \oplus B$	XOR
1	1	1	x	G = A'	NOT (1's Complement)

Table 1 Truth Table of ALU

2.1.1.2

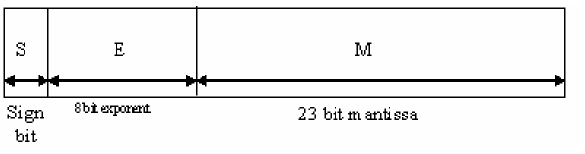
2.1 Multiplication and division unit

The multiplication unit performs both unsigned and signed multiplication. It takes two 8 bit inputs A and B and the output P is 16 bit. The multiplication is done using Modified Extended Booth Algorithm which is preferred to other multiplication algorithm like Repeated Addition and Booth algorithm because of its rapid computation efficiency. Modified Booth's Algorithm needs just 'n/2' steps to arrive at the product of two n bit numbers. The select line is used to choose between unsigned and signed multiplication outputs.

The division unit performs both unsigned and signed division. It takes two inputs 16 bit dividend A and 8 bit divisor B and the outputs are quotient Q 16 bits and remainder R 8 bit. Both the division is done using Non-Restoring algorithm. Non-Restoring Algorithm needs just ‘n’ additions or subtractions. The select line is used to choose between unsigned and signed division outputs.

2.2 Floating point operation

Floating point is represented by IEEE 754 standard which comprises of 23-bit mantissa M, an 8-bit exponent field E and a sign bit S. The mantissa is said to be normalized if the digit to the right of the radix point is not zero, that is, no leading zeros appear in the magnitude part of the number. Hence the magnitude part of the normalized mantissa is always ‘1’. To facilitates to test for zero the floating point exponents should be encoded in excess-K code similar to excess-3 code, where the exponent field E contains an integer that is the desired exponent value plus K (K=127).



2.1.1.2.1.1

2.1.1.2.1.2 Figure 5 IEEE 754 standard

The number N represented by a 32-bit floating point has the following set of interpretations:

1. If $0 < E < 255$, then $N= (-1)^S * 2^{E-127} * (1.M)$
2. If $E = 0$ and $M != 0$, then $N =(-1)^S * 2^{E-126} * (0.M)$
3. If $E = 0$ and $M = 0$, then $N = (-1)^S*0$

For the addition and subtraction of two floating point numbers, their exponents must be made equal before the corresponding mantissa can be added or subtracted. This exponent equalization is done by right shifting the mantissa X_M associated with the smaller exponent X_E a total of Y_E-X_E digits to form the new mantissa $X_M *2^{(X_E - Y_E)}$ which can be combined directly with Y_M. Addition and subtraction has four steps:

1. Compute Y_E-X_E, a fixed point of subtraction.
2. Shift X_M by Y_E-X_E laces to the right to form $X_M*2^{(X_E-Y_E)}$.
3. Compute $X_M *2^{XE-YE} \pm Y_M$, a fixed point addition or subtraction.
4. Normalize the result

Floating point multiplication involves a fixed-point multiplication of the mantissas and a fixed-point addition of their exponents. The multiplication is done using Modified Booth’s Algorithm.

$X * Y = (Xm * Ym) * 2^{Xe+Ye}$ ……...........(1)

Floating point division involves a fixed-point division of the mantissas and a fixed-point subtraction of their exponents. The division is done using Non-Restoring Algorithm.

$X / Y = (Xm / Ym) * 2^{Xe-Ye}$ ....(2)

3 VLIW Format

The VLIW processor designed is a 16-bit processor and it is capable of executing 6 instructions per cycle. Two separate register sets for fixed and floating-point operations are available. Each register set has 64 registers. The fixed-point registers are 16 bit in length and floating-point registers are 32 bit in length. The individual registers are addresses with a six bit address. The processor is designed with Harvard architecture. It has separate code memory and data memory. So, the processor has access to code and data at the same time. This enables execution speedup. The code memory address bus is 16 bit and its data bus is 128 bit. The data memory address bus is 8 bit and its data bus is 16 bit. It has a prefetch queue of size 6. The block diagram is shown in figure 3.1.

The processor executes 6 instructions per cycle. It can perform

- One floating point addition/subtraction
- One floating point multiplication/division
- Two fixed point ALU operation
- One fixed point multiplication/division
- One load/store operation.

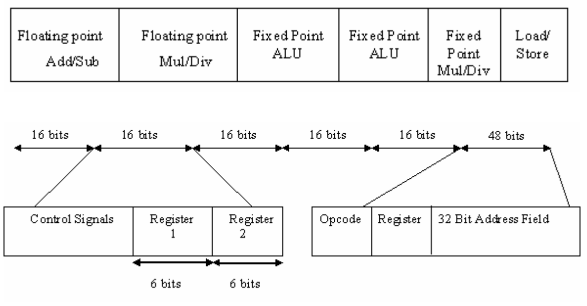


Figure 6 VLIW Format

4 GALS Modules implementation

The VLIW processor designed is split into three GALS modules operating in different frequencies. The GALS modules are

- Memory-Prefetch Queue Module
- Decoder Module
- Functional Unit - Register Module.

4.1 Memory - prefetch module

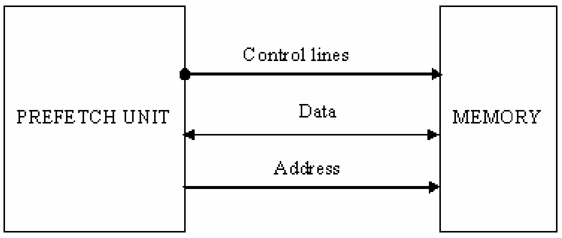


Figure 7 Memory - Prefetch Module

The memory-prefetch module fetches the 128 bit VLIW instructions from the memory and places it in the prefetch queue of size 6. This enables increasing the speed of the processor through pipelining. The module passes the VLIW from the queue to the decoder module. The module operates at 80 MHz.

4.2 Decoder module

The decoder module decodes the VLIW passed by the memory-prefetch module and generates the control signals and register addresses for all the six instructions. The module operates at 60 MHz.

4.3 Functional unit-register module

The functional unit-register module consists of

- Functional units
    - One floating point addition/subtraction unit
    - One floating point multiplication/division unit
    - Two fixed point ALU
    - One fixed point multiplication/division unit
    - One load/store unit
- Register set
    - Fixed-point register set
    - Floating-point register set.

The module uses the controls signals and register addresses from decoder module. It accesses the registers for data and passes them to functional units to compute the result. The result is then stored back into the register. The module is operated at 50 MHz. The block diagram is shown below

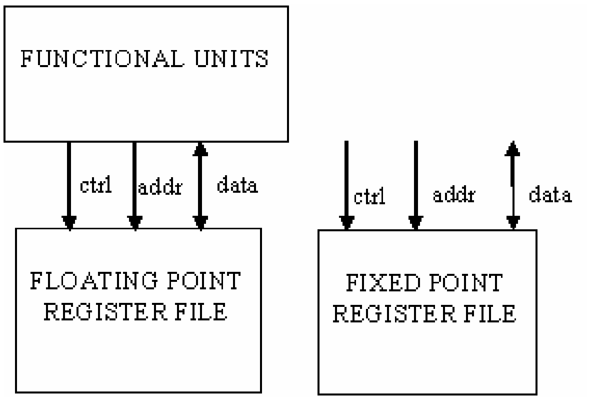


Figure 8 Functional Unit – Register Module

5. Simulation results

The simulation result of VLIW processor is shown below and it can performs 6 instructions at a time.

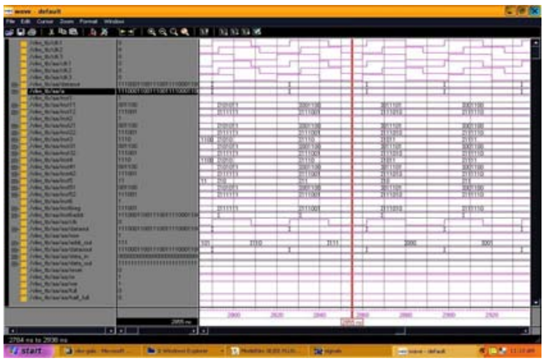


Figure 9 Simulation result of VLIW processor

The VHDL code is synthesized using FPGA Advantage tool and reports are shown below

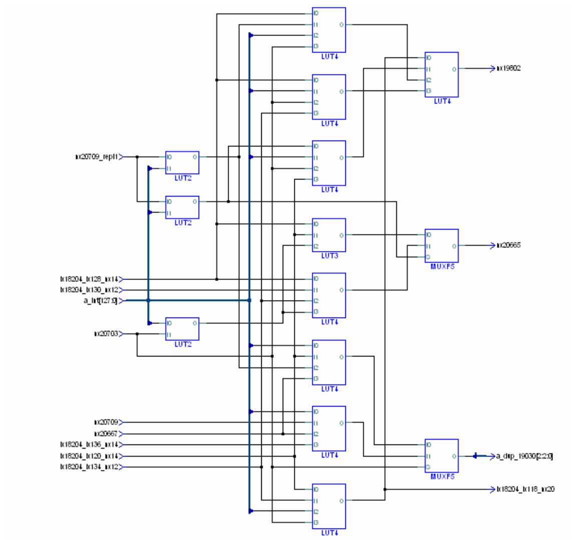


Figure 10 Synthesis report of decoder

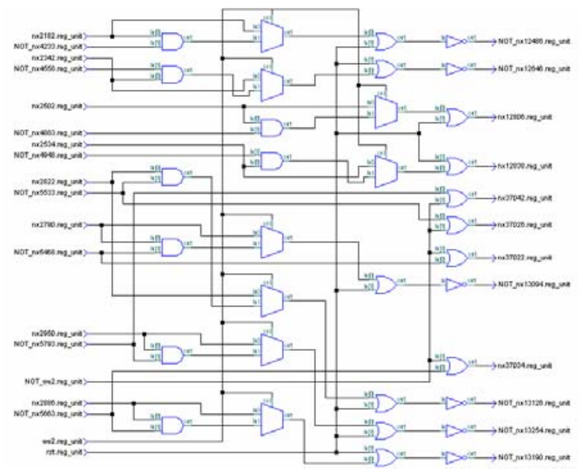


Figure 11 Synthesis report of ALU

The VLIW processor is simulated with global clock and the synthesis report gives 30.2 MHz operation. In this design we found that the minimum clock required is about 50 MHz. There will be 1.5 times increase in the speed of operation and also the circuit is free from the metastability and clock skew.

6 Conclusion and future work

The VLIW processor is implemented using GALS architecture with pipeline and each module is running at different clock frequency. Using this architecture the power is reduced due to local clock and the ALU which is not operational will consume less power due to low clock speed.

The future work of the project is to include Dynamic voltage scaling for further reduction in power and also this architecture can be implemented in ASIC.

REFERENCES

1. Joseph A. Fischer (1983) ‘Very Long Instruction Word Architectures and the ELI-512’,Proceedings of the 10'th Symposium on Computer Architectures, pp. 140-150.

2. Joseph A. Fisher (1998) ‘Very Long Instruction Word Architectures and the ELI-512', 25 Years ISCA: pp. 263-273.

3. John P. Hayes (1998) ‘Computer Architecture and Organization’, McGraw-Hill.

4. Jens Muttersbach (2001) ‘Globally-Asynchronous Locally Synchronous Systems for VLSI Systems’. PhD thesis, ETH, Zurich.

Mr..R. Seshasayanan was born in the year 1958 and received his B.E degree from College of Engineering and M.E. degree from Anna university in the year 1980 and 1983 respectively .He is presently working as Lecturer in the Department of Electronics and Communication, Anna University, and pursuing his Doctoral degree in the field of Embedded system and area of interests are VLSI Design, Reconfigurable architecture and Low power design.