ISSN – 2348-2397

APPROVED UGC

SHODH SARITA

Vol. 7, Issue 27(III) July – Sept 2020

AN INTERNATIONAL BILINGUAL PEER REVIEWED REFEREED RESEARCH

DESIGN OF ASIP MULTISTAGE PARALLELISM ON RISC PIPELINING VLIW ARCHITECTURE

Mood Venkanna[1], Dr Rameshwar Rao[2] and Prof P. Chandra Sekhar[3]

[1] Research scholar, Dept. of Electronics and communication Engineering, UCE, Osmania University, Hyderabad, Telangana.

[2]Retired Professor Dept. of Electronics and communication Engineering, UCE, Osmania University, Hyderabad, Telangana.

[3]Professor Dept. of Electronics and communication Engineering, UCE, Osmania University, Hyderabad, Telangana.

ABSTRACT

The integrated processor is application-specific and is used in different fields for faster processing. ASIPs are more common compared to ASICs. Various techniques are used for the ASIP architecture. In this article ASIP is based on the architecture of the pipeline. In efficient pipeline architecture, the execution time is proposed. The designed pipeline architecture is supervised by Clock Gating technology. The Arm system architecture is carried out and is tested in the core processor using the VLIW architecture. The test is carried out via VHDL. The performance of the proposed technology is shown in the results section along with the RTL logic diagram. Design has also concentrated on the design on ALUs, CUs, and CPUs Memory etc. With Four stage pipelined Multi stage RISC with parallel Multiplier which is also more suitable for 8086 Micro Processor.

Keywords: Embedded processor, ASIP, Pipeline architecture, ARM-ISA, ICORE, VLIW, VHDL.

I. INTRODUCTION

An embedded device makes today's daily lives healthier and more efficient. Such a modern device needs to take critical problems like low-energy concentration into account. A highly efficient system can lead to more energy supply and thus heat the system causing design problems. The reduction of power is the primary design criteria that challenge today's researchers among the different questions of an embedded device. Each embedded system's main module is [1–4]: General Purpose Processors (GPPs), Application Specific Integrated Circuits (ASICs) Application Specific Instruction set Processors (ASIPs)

An ASIP has been produced with common characteristics for a particular application. It is an intermediate

solution that provides greater versatility from a single-purpose, compared to a single-purpose processor. It also gives better performance, power and size than a processor for general purposes. During the ASIP design review we found that the multi-issue pipeline design with vector processing systems was one of the big candidates for solving ASIP problems. The selected core was a VLIW processor (Very Long Instruction Word), in which the vector operations were taken into account in order to adjust the energy supplied for better functions [4-8].

In connection with high-performance processor architecture, both super-scalar processors as well as VLIW were designed [7, 8]. This technique examines the parallelism of the ILP or instruction point, with the goal of providing more IPCs or instructions per cycle, but with different scheduling instructions. The superscalar processors are able to receive multiple instructions within the same cycle, which can detect ILP and provide parallel execution contributing to greater power consumption and device size. The VLIW processors also have additional ILP detection hardware or timeline instructions as these are software-analyzed instructions that take the command memory a priori. VLIW processors are more applicable to embedded systems needing better performance due to the advantages of parallel processing using simple hardware.

This method aims to produce a VLIW ASIP using a clock gating device to reduce flip flops' power consumption. This depends on the signals obtained from the conditions of execution of the registers. In order to imitate pipelines, we automatically create minimum conditions of efficiency in the generation of the ASIP, in order to reduce power consumption in a broad data path corresponding to the proposed VLIW ASIP.

VLIW ASIPs or scalar ASIPs need to follow few specific limitations. The following: field demand, energy consumption and output computation. The design space requires experimentation in order to determine the best parameters for the ASIP architecture [8, 9]. A built-in processor evolution mechanism with great features that can consume little power in one chip is needed. This must be implemented with a reduced and low delay heterogeneous device. For the implementation of the functional unit, the ASIP specification is made either integrated with the chip or implementable on the peripheral devices in question.

A reduction in the area and the correct delay are necessary for the conventional ASIP design. A more powerful ASIP with low power must therefore be built to reduce the energy usage. Different power management methods have been implemented, including the clock gates, the RTL stage of transmission and operating isolation [10, 11].A low power but high performance VLWI ASPE is attempted with the clock gate because there are so several VLIW ASIPs in this research

Register of the pipeline. Reasonable standards need to be met. In addition to that the power, a minimum execution requirement is to be extracted. But, extracting it takes longer and is likely to make errors in a complicated data route. These problems are solved automatically with low power VLIW ASIP. Through using this approach, we produce gating signals to which the registry functions of pipelines.

Researchers in this area have made important progress in developing ASIPs. Most of the work has been based on providing special guidelines for increasing efficiency and lowering costs [7-12]. Parallel architecture has recently been active in the creation of ASIP. It was suggested that a VLIW ASIP should have a distributed register structure [13]. Jacome et al . suggested in [14, 15] exploring the design space for data paths at VLIW ASIP. Kathail et al. supported the VLIW design flow of the processor that enables hardware to be not programmed [16]. In [17] the customizable multiprocessors were suggested, while in [18, 19] the authors differ in the design of the ASIP pipeline number.

II. METHODOLOGY

There are a variety of sequences in the one-pipe method of the programming process. To explore the design area, the number of the pipeline is chosen. Then, as exploration progresses, the minimum 2-pipe structure and the number iteratives to 4-pipelines are increased. The instructions are stored simultaneously in a separate pipeline. Guidance is given for each sequence in the correct pipeline in order to receive the instructions from a suitable program sequence. As in every pipe, the number of pipes is lower than in any other subset of the ISA target records. Pipeline Stage RISC- All ALUs, CUs, CPUs, memory have been built here. This works more than 50MHz in the Max Clock frequency. This is a modern development as it has a high-clock frequency architecture and a pipeline. Muti-Stage Parallel RISC – Here we have designed all the blocks as built in Pipeline Stage RISC with the design is four-stage and parallel with an 8-bit input and an 8-bit output port and parallel multiplier. This works more than 300MHz with Max Clock frequency. It will be fresh, as it is planned with the use of parallel multipliers in multi-stage architecture for such a high frequency. It is 8086 MP-compatible RISC- Here all registry blocks (PC, PSW, A, B), bank registry and ULA has been built, as well as GPIO blocks, Memory Data and Power. It can function more than 50MHz with the Max Clock frequency. This is unique as it is compatible with the 8086 MP set of instructions.

The proposed ASIP architecture has been given in Fig. 1. It is a RISC-like design having with four stages. These are (a) pipelines (b) Instruction Fetch (c) Instruction Decode (d) Instruction Execute and Memory Access/Write Back.

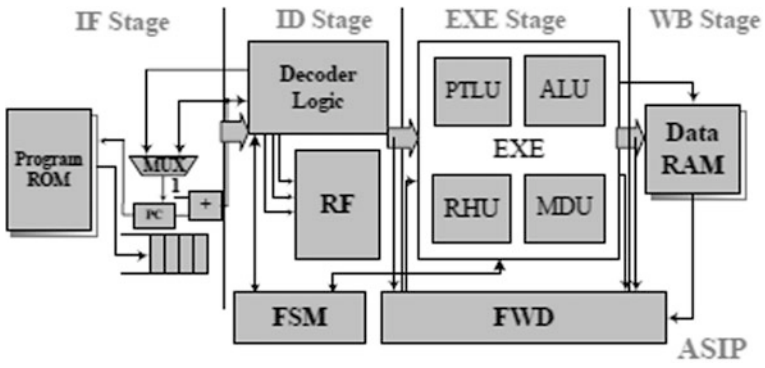

Fig. 1 Proposed pipelined architecture

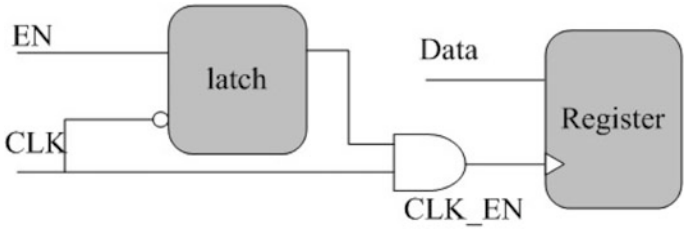

Fig. 2 The clock-gating structure

A. Instruction Pipelining

The balance between ASIPs between the Singles Processors (SPPs) and the GPPs. The research designs an ASIP that relies on the CORDIC algorithm in order to optimize its data path for planned operations such as the built-in control and digital signaling applications. For this purposes, Xilinx Cor- portions, Inc's common IDE or Xilinx Integrated Development Environment. The pipeline splits the execution of the instruction into different, basic cycles. This leads to successive duplication of the executed instructions in an instruction stream. In RISC processors, the classic pipeline architecture includes five phases. Those are: (a) fetch (b) fetch(c) fetch (d) memory access(e) write back (WB) command instruction. Such function overlaps the execution of the first command in a command stream with the second one and therefore the third command will be fetched. For subsequent steps, the pipeline registers are used to record the intermediate results [1–8].

This work has seven stages in the pipeline structure. The EX stage performs integer arithmetic and logic instructions with a single multiplier process integer. ASIP data paths are implemented by the usual functional system which works parallel to the EX stage of the database processor. The following parts define this problem and its potential solutions.

Multi-cycle non-pipelined special instructions in ASIPs pose risks due to the inconsistency between the multi-cycle ISE and other special instructions that follow. A pipeline stall can be used to remove structural hazards and/or data hazards using compiler NOPs or by hardware interlocks. This offers lower efficiency, however, so that most processors use data transmission so that data dangers are removed.

B. VLIW

In VLIWs, ample parallelism is required in a coding segment in order to fill the operating slots to keep many, separate functional units occupied. In a single but larger loop[17] the identification of the parallelism involves the unrolling of loops and the preparation of code. There are two static planning strategies in each compiler, such as local and worldwide feet. Although local planning relies on the parallelism of code block analysis, global scheduling is both complex and costly across divisions [17–20].

Because of the large code size and operational limitations of the lockstep, VLIWs are facing technical problems. When unrolling loops multiply with the original loop core, the size of the code increases. To counter this problem, word encoding is used to compress and extend the main memory during processor decoding. In addition, the overall instruction word can be decreased by exchanging the instructions in an immediate field corresponding to the functional units [17]. In VLIWs, a room for smart improvements is always available since each saved bit is transferred to a smaller instructions memory.

The VLIW lock-step could lead to unnecessary stalls if there is no risk detection hardware. In this case, all devices are moving concurrently from one point to the other, because there is a stand in each device which also causes a stand in other devices. Consequently, codes are malfunctioning if many devices have the same memory or record. When functional units operate asynchronously with the instructions and can be done using a hardware danger detection unit or the regular scheduling [17], then VLIW functional units can be autonomous.

VLIW has a variety of heterogeneous and separate running slots to run concurrently if the compiler is sufficiently capable. It is also modular, suitable for ASIPs as usable units for better use application can be added or removed. A single dedicated scalar lane for computation is available for the operation with ASIPs. To perform intensively complex computation new instructions and functional units can be added.

C. Generation of ASIP

Different scalar ASIPs and VLIW ASIPs have automatically been suggested for the development of ASIPs[7-18]. The strategies include an ADL or MOD, which includes design parameters such as services that are to be used, port data, data flows, etc. The application of ASIP MODs helps to automatically create the required data trajectory and to insert the pipeline information. The pipeline registry logic is given as

$$enp = stallstagep$$

Where p refers to every list of pipelines. Phasep is the number of the pipeline of which p is a part. Stalln suggests an interlocking state of the pipeline that can occur due to multi-cycle operation or structural threat. If enp is valid, the next step of the pipeline is facilitated by data storage. If enp is wrong, stagep will be stopped. The standard VLIW ASIP with its limited ASIP area Eq is intended for high speed. Can be more simplified.
For ASIP pipeline registries, it is easy to use the control signal to gate the clock. In this situation, clock signals are given to all pipelines should the pipelines not be stopped. Memory Configuration Clock Development Mechanism In addition to its instructions, the time taken to transfer instructions and data between the CPU and memory components is often highly influenced by machine performance. The clock mechanism is designed to perform the operation of ting memory. The average cycle is designed to calculate device outputs to achieve the clock cycle required for machine instructions. The clock signal has various characteristics including clock time, clock pulses, edge and edge of trailing. The operation of the clock depends on the conduct of clock components with the planning approaches of the memory architecture. Following parameters such as time settings, time hold and delay of propagation will examine clocking effect.
The ASIP is more robust with higher performance and less power usage in this embedded device. The designer implements both the processor and the memory architecture in the ASIP framework, depending on the applications you like. In the ASIP framework, the solution provides improved performance and low design complexity with retrofitted compiler and compiler technology. The design of the processor consists of various design steps, such as cost, strength, size and innovations in real-time. For specific high-performance embedded computing systems, two types of microprocessor RISC and CISC are commonly used. A pipeline unit operated by DMA circuits is given in the processor. In the processor arithmetic pipeline and instructions pipeline there are two types of pipelines widely used. The generalized instructional process overlapping fetch, decode and execute phases in an instruction pipeline controlled with a stream of instruction. Long memory in the pipeline actually plays a dominant role. Specific pipeline mechanisms are used according to their output by specific technology providers such as ARM, Intel and Motorola [20, 21].Gloria et al . [22] pointed to a few important requirements for the design of ASIP, such as device recognition and fast-to-use functions, often used hardware, etc. (Figure 2).This section has been submitted with theoretical review of VLIW instructions, SIMD instructions and fusions.

Whether TCLK / N is processing the data in the parallel modules or the DMUX and MUX units should be fitted with data delay units. It is possible to concurrently process.

Steps of ASIP Synthesis Gloria et al . [22] have few major ASIP design criteria.
• Evaluate architectural options and use application actions to start the design.
• Hardware functionalities for most of the operations used to speed up the application are important to define And implement.
In the ASIP synthesis, mainly 5 steps are listed below but other methods in this field have been proposed.
1. Application Analysis: it is necessary to analyze ASIP design inputs, test data, and specific design constraints In order to support the hardware synthesis and generation of instruction settings. Application analysis: Every Code is written and evaluated both statically and dynamically in a high-level language. This data analyzed Must be saved for further analysis in an suitable intermediate format.
2. Architectural design discovery Space: Based on design constraints, for a particular application, a potential Collection of architectures are identified and their output measured based on power consumption, hardware Costs, etc.
3. Instruction Set Generation: The instruction set should be developed on the basis of a specific application or Selected architecture which supports the code and hardware synthesis process.
4. Code Synthesis: The synthetization code is generated for the intended application by using a generator of Retarget able code or compiler.
5. Synthesis of Hardware: The ASIP design framework and software set for instruction is used to synthesize Hardware using VHDL / VERLOG resources.

Methodology for ICORE Development

The ICORE architecture is made up of assembler programs and VHDL hardware for the interfaces of the synthesized kernel. In this, arithmetical and logical instructions, program flow control and data movement can be used as an instruction set. It serves as a subset of a DSP instruction that can not contain particular instructions like bit checking, splitting, rounding, standardization and loop operations. ICORE uses blocking registrations or blocking gates linked to a specific bus to prevent the transmission of unwanted values to functional units. In order to decrease the power consumption of ICORE's very wide ROM, the word width of the ROM in an ISA configuration must be reduced to a minimum [23,24].

III. RESULTDISCUSSION

The goal is to develop the separate scalar architecture as well as the vector lanes to improve performance. The instruction must be multiplied or more instructions per clock cycle, similar to VLIW, to increase the vector width or vector lane parallelism.

The RTL hardware description language, such as VHDL , is an attempt to define the processor in structured form such that a network list with synthesis to position and route can be extracted, and hold concept discovery iterations close to impossible. The problem can be solved via machine languages that allow the definition of the processor to increase its abstraction level and to increase the speed of design exploration.

Pipelining increases the processor performance because several operations occur in subsequent clock cycles even though singing fetch / decoding is performed within the first clock cycle.

The pipeline is shown to be susceptible to different risks. For example, when the address is created and executed, various pipelines may access the same memory locations. Therefore, the pipeline register fetch / decode instructions are based on the registers.

The Table 1 Shows the comparison between the existing and proposed methods, the existing method [25] have designed a pipelined RISC architecture. The test stand for the outcome of the simulation is shown in Figs along with the RTL logic diagram and design Summary. For each case, in figures 3 and 4.

Table 1

S.NO.	METHODS	TIME(ns)
1	Method in [25]	17.458
2	Proposed Method	11.144

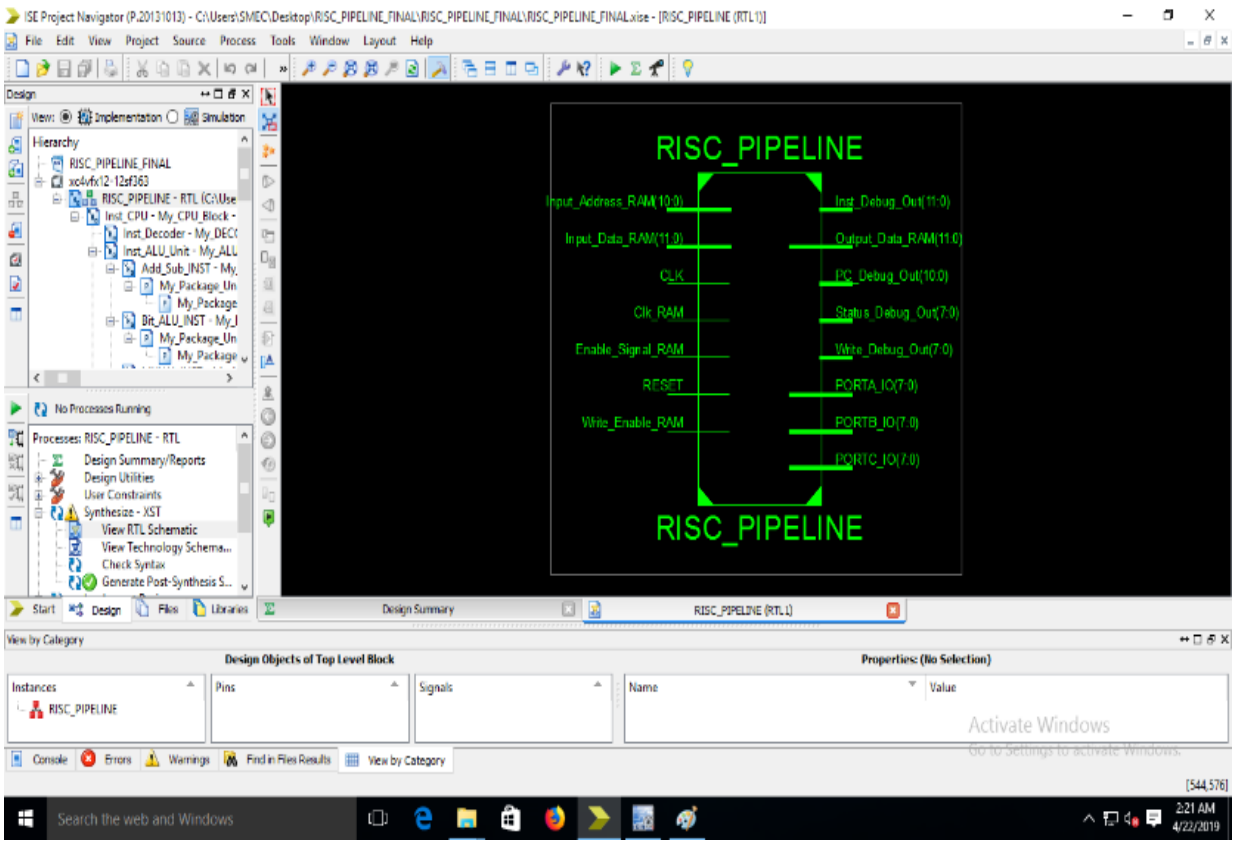

Figure 3 RISC Pipeline RTL Diagram

RISC_PIPELINE Project Status (11/28/2017 - 21:29:52)			
Project File:	RISC_PIPELINE_FINAL.xise	**Parser Errors:**	No Errors
Module Name:	RISC_PIPELINE	**Implementation State:**	Synthesized
Target Device:	xc4vfx12-12sf363	**• Errors:**	No Errors
Product Version:	ISE 14.7	**• Warnings:**	401 Warnings (0 new)
Design Goal:	Balanced	**• Routing Results:**	
Design Strategy:	Xilinx Default (unlocked)	**• Timing Constraints:**	
Environment:	System Settings	**• Final Timing Score:**	

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	264	5472	4%
Number of Slice Flip Flops	188	10944	1%
Number of 4 input LUTs	416	10944	3%
Number of bonded IOBs	64	240	26%
Number of FIFO16/RAMB16s	2	36	5%
Number of GCLKs	2	32	6%

Figure 4 Design Summary

IV. CONCLUSION

We analyzed the design of the ASIP pipeline in this paper. The primary processing steps have been established and the hardware solution factors investigated. We found that the hardware solution must also choose the correct clock gating technique for specific pipeline architecture. Two design approaches have been introduced in this paper: (1) ASIP solution (2). We found from the further analysis that the parallel nature of the changed ASIP was unable to produce any additional output by adjusting the clocking method to the processing pipe. The efficiency and versatility were improved by this attempt. The proposed solution was synthesized with two different vector widths following the completion of functionality variations of the core on two stages, cycle-accurate instruction set, and RTL simulation. When you compare time component of the previous approach are that, while still providing performance space in the time budget, the smaller vector width version provides a strong performance-to-area ratio.

RISC Pipeline Phase- Here are builds all ALUs, CUs, CPUs and the memory. The frequency of Max Clock operates more than 50MHz. It is a modern system, with architecture and a pipeline in high-clock frequency. Multi-Stage Parallel RISC – Here we have planned all the blocks built into the four-stage pipeline Stage RISC with an 8-bit input, an 8-bit output port and a simultaneous multiplier. With Max Clock frequency, it works more than 300 MHz. It will be cool as expected for such a high frequency using parallel multipliers in multi-stage architecture. It's 8086 MP-compatible, RISC- All login blocks, including PC, PSW, A, B, ULA, bank registry and GPIO blocks, memory files, and power have been generated here. The Max Clock frequency can run at more than 50MHz. This is special, since the 8086 MP instruction set is compatible.

References

1. Mood Venkanna, "An Efficient Design of ASIP Using Pipelining Architecture", Advances in intelligent systems and computing book series (ASIP volume 846 Springer) ,PP 117-128,2018
2. K. Hwang, "Advanced parallel processing with supercomputer architectures", Proceedings of theIEEE,vol.75 ,no.10,pp.1348–1379,2017.
3. FrankVahid,Tony Givargis,"EmbeddedSystemDesign"pp.9–11.
4. M. Johnson, Superscalar Microprocessor Design, Prentice-Hall, Inc.,2011.
5. K. Hwang, and Z. Xu, "Scalable parallel computers for real-time signal processing", IEEE signalprocessingmagazine,vol.13,no.4,pp.50–66,2016.
6. K. Hwang, and Y. H. Cheng, "Partitioned matrix algorithms for VLSI arithmetic systems",IEEETransactionsonComputers,vo.100,no.12,pp.1215–1224,2012.
7. J. A. Fisher, "Very long instruction word architectures and the ELI-512", vol. 11, no. 3, pp. 140–150, ACM,2018.
8. M. F. Jacome, G. de Veciana, and V. Lapinskii, "Exploring performance tradeoffsfor clustered VLIW ASIPs", In Proceedings of the 2000 IEEE/ACM international conference on Computer-aided design, pp. 504–510,2010.
9. B. Middha, A. Gangwar, A. Kumar, M. Balakrishnan, and P. Ienne, "A Trimaran based frameworkforexploringthedesignspaceofVLIWASIPswithcoarsegrainfunctionalunits",

InProceedingsofthe15thinternationalsymposiumonSystemSynthesis,pp.2–7,2012.
10. F. Sun, S. Ravi, A. Raghunathan, and N. K. Jha, “Synthesis of custom processors based on extensible platforms”, In Proceedings of the 2002 IEEE/ACM international conference on Computer-aided design, pp. 641–648,2012.
11. P. Brisk, A. Kaplan, R. Kastner, and M. Sarrafzadeh, “Instruction generation and regularity extractionforreconfigurableprocessors”,InProceedingsofthe2002internationalconference onCompilers,architecture,andsynthesisforembeddedsystems,pp.262–269,2012.
12. D. Goodwin and D. Petkov, “International conference on compilers, architecture and synthesis for embedded systems”, In Proceedings of the international conference onCompilers, architecture and synthesis for embedded systems, pp 137–147, ACM Press New York,2013.
13. R.Kastner,S.Ogrenci-Memik,E.Bozorgzadeh,andM.Sar-rafzadeh,“Instructiongeneration forhybridreconfigurablesystems”InICCAD,2011.
14. M. Jacome, G. de Veciana, and C. Akturan, “Resource constrained dataflow retiming heuristics for vliwasips”, In Proceedings of the seventh CODES, pp. 12–16, ACM Press, 1999.
15. M. F. Jacome, G. de Veciana, and V. Lapinskii, “Exploring performance tradeoffsfor clusteredvliwasips”,InProceedingsoftheICCAD,pp.504–510,IEEEPress,2010.
16. V. Kathail, shailAditya, R. Schreiber, B. R. Rau, D. C. Cron-quist, and M. Sivaraman,“Pico: Automaticallydesigningcustomcomputers”,InComputer,2012.
17. F. Sun, N. Jha, S. Ravi, and A. Raghunathan, “Synthesis of application-specific heteroge-neousmultiprocessorarchitecturesusingextensibleprocessors”,InProceedingsofRealTime and Embedded Technology and Applications Symposium, pp. 551–556. IEEE Computer Society,2015.
18. S. Radhakrishnan, H. Guo, and S. Parameswara, “n-pipe: Application specific heterogeneousmult-pipeline processor design”, In The Workshop on Application Specific Processors. IEEE Computer Society,2005.
19. ARMLimited,“ARMArchitectureReferenceManual”,2012.
20. M,. Venkanna, R. Rao, P. Chandra Sekhar, “Application of ASIP in Embedded Design with Optimized Clock Management”, ICITKM Conference, Newdelhi,2017.
21. T. Lang, E. Musoll, and J. Cortadella, “Individual flip-flops with gated clocks for low power datapaths,”IEEE Trans. Circuits Syst. II, vol. 44, no. 6, pp. 507–516, June 2017.
22. A. D. Gloria, P. Faraboschi, “An evaluation system for application specific architectures”, Proceedings of the 23rd Annual Workshop and Symposium on Microprogramming and Microarchitecture. (Micro 23), pp. 80–89,2010.
23. T, S. Bitterlich, and H. Meyr, “Increasing the Power Efficiency of Application Specific Instruction Set Processors Using Datapath Optimization”, IEEE Workshop on Signal Processing, Lafayette, Louisiana, USA, Oct.2010.
24. T. Glokler, and S. Bitterlich, “Power efficient semi-automatic instruction encoding for application specific instruction set processors”, In Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP’01), IEEE International Conference, vol. 2, pp. 1169–1172, 2011.
25. Mr. S. P. Ritpurkar, Prof. M. N. Thakare, Prof. G. D. Korde, “Design and Simulation of 32-Bit RISC Architecture Based on MIPS using VHDL”, International Conference on Advanced Computing and Communication Systems (ICACCS -2015), Jan. 05 – 07, 2015, Coimbatore, INDIA, IEEE