Mantik Journal, 8 (3) (2024), ISSN 2685-4236 (Online)

Published by: Institute of Computer Science (IOCS)

Jurnal Mantik

Journal homepage:

SPArc-subset general-purpose microprocessor design and implementation in field programmable gate array

Karel Octavianus Bachri[1*], Justin Alexander[2], Edbert Osmond[3], Enny Widawati[4], Maria Angela Kartawidjaja[5]

[1,4]Graduate School of Electrical Engineering

[2,3]School of Electrical Engineering

[1,2,3,4]Faculty of Engineering, Atma Jaya Catholic University of Indonesia

A R T I C L E I N F O

Article history:

Received Sep 10, 2024
Revised Oct 08, 2024
Accepted Nov 4, 2024

Keywords:

FPGA;
Microprocessor;
RISC;
SPArc Subset;
VLSI.

ABSTRACT

This paper focuses on designing a 32-bit Reduced Instruction Set Computer (RISC) Scalable Processor Architecture (SPArc) subset gener al purposes processor. The design process covers instructions selection, Register Transfer Notation (RTN) design, Datapath Design, and Control Unit Design. Basic integer instruction was selected. Datapath was designed along with the RTN with a five-stage pipeline with direct connection between stages. This design was then validated using functionality simulation test and implemented in Field Programmable Gate Array (FPGA). The performance of the processor was measured using thermal report, power report, and time report. Functional test shows that the Processor can execute instructions as designed, which are Arithmetic/Shift/Logic, Control Transfers, and Memory access instructions. It was validated with the register content and signal generated in each stage. The design was also successfully implemented in FPGA with the maximum clock of 58 MHz as the synthesis report shows. Thermal report shows the thermal properties of the design, which shown the acceptable thermal margin of 56.9 ºC and junction temperature of 28.1 ºC. The power report shows the low power consumption of 0.266 W, which consists of dynamic power of 0.173 W and static power of 0.093 W. This work enables further development and to be used as master processor in System on Chip design of special purpose processors like cognitive processors.

Corresponding Author:

Karel Octavianus Bachri,
Faculty of Engineering, Atma Jaya Catholic University of Indonesia,
Jalan Jenderal Sudirman Kav. 51, Jakarta, Indonesia.
Email: karel.bachri@atmajaya.ac.id

1. INTRODUCTION

There are two types of computer architecture, Complex Instruction Set Computer (CISC) and Reduced Instruction Set Computer (RISC). CISC focuses on reducing the number of instructions in a single task, and thus, the complexity of a single instruction increases (Hennessy & Patterson, 2019). Recent development of CISC are the processor of Personal Computers (Rotem et al., 2022; Vera, 2020). Another type of CISC is also developed toward efficiency and performance (Naffziger et al., 2020; Pomianowski, 2021; Schöne et al., 2021). On the other hand, RISC focuses in simplifying each single instruction, and

thus, reduces the complexity of the processor (Lindholm et al., 2014) and the architectures are application-oriented design (Yue & Mehta, 2023).

Advanced RISC Machine (ARM), and Scalable Processor Architecture (SPArc) are two of the most common used processor types. SPArc Focuses on high performance and Multi-processor Architecture (Fengler et al., 2021; Hidri et al., 2019; Sakamoto et al., 2002; Waterman & Asanov i´c, 2017), on the other hand, ARMs focus on power efficiency, and thus, suitable for hand-held devices. Recent advancement in Digital Technology requires ARM architcture that support power efficiency (Hemanthkumar et al., 2022) and considerable performance (Jyotsna et al., 2022).

The design of 32-bit processors plays a crucial intermediate position in the trend of microprocessor design. It capable of executing 32 bit data and commonly used in various applications (Kulshreshtha et al., 2021; Soundari et al., 2021), while maintaining balance with the less power it needs compared to the 64-bit or above (Tang et al., 2018). The recent development of SPArc is combination of multiple processors in a System on Chip (SoC)(Hepola et al., 2024; Malatesta et al., 2023). And the improvement of the architecture using few pipeline stages (Phangestu et al., 2022; Xiang Lim & G, 2019; Zhang et al., 2021). The main focus of this work is to design an Integer single Processor subset 32-bit SPArc.

2. RESEARCH METHOD

To accomplished the research objective, this work is conducted through the following steps. The first step is Instruction Selection, Register Transfer Notation (RTN) Design, Datapath Design, Control Unit design, Integration, Functionality Test, and Implementation Test (Frühauf et al., 2024; Savadatti et al., 2024).

Performance of microprocessors are shown in the synthesis report, which comprises of thermal report, time report, and power report. Common complete RISC processors consume 10 W to 24 W, depending on the processor type and states. The temperature of common complete RISC processors varies from 42ºC to 60ºC at idle and can reach 85ºC at peak (Suárez et al., 2023). In this work, we focus on the subset of SPArc architecture since the development will be as the controller.

Microprocessors in general comprises of Fetch, Decode, Execute, Memory Access, and Write Back. Instructions are fetched from Read-Only Memory (ROM) and are stored in the Instruction Register. At the following cycle, it is decoded to determine the type of instruction to be executed, the source of operands, and the location to store the result of the operation. The source operands are from register, or immediate. The next cycle is excecute. The Decode Stage gives signals to the Execute Stage to run the operation. After the Execute Stage, the operand will access memory if Load/Store instructions take place, otherwise, no operation will be performed in the memory stage. The last stage of microprocessor is Write Back. At this stage, the result of the instruction is stored in the register (Meyer-Baese, 2021; Waterman & Asanovi´c, 2017). There are several types of SPARC 32 bit Instructions, Memory-access, Arithmetic/Logic/Shift, and Branch, as shown in Table 1.

Table 1. SPArc Instructions Detail

No.	Instruction Category	Instruction	Code	Code	Instruction	Instruction Category	No.
1	Arith.	Add	ADD	SLL	Shift Left Logical	Shift	10
2		Subtract	SUB	SRL	Shift Right Logical		11
3		Unsigned Int. Mul.	UMUL	SRA	Shift Right Arith.		12
4	Logic	And	AND	LD	Load Word	Mem. Access	13
5		And Not	ANDN	ST	Store Word		14
6		Inclusive Or	OR	Call	Call	Ctrl. Transfer	15
7		Inclusive Or Not	ORN	BE	Branch Equal		16
8		Exclusive Or	XOR	BCS	Branch Carry Set		17

9	Exclusive Or Not	XNOR	BN	Branch Negative	18
			BVS	Branch Overflow	19

SPArc Instruction has three main Format. All formats use the first two bits as general operation encoding (op). Format 1 is CALL instruction, the first two bits are op, while the rest 30 bits are used as 30-bit displacement (disp30). Format 2 is used for two types of instructions; they are BRANCH and SETHI Instructions. Format 3 is used for the remaining instructions, which are Arithmetic/Logic/Shift and memory access.

Table 2 shows description of op used in each format. Format 1 is used for CALL instruction and op is set to "01". Format 2 is used for Branch ans SETHI and op, is set to"00". Format 3 is used for the remaining instructions. for memory instruction, op is set to "11", while for Arithmetic/shift/logic and the remaining instructions, op is set to "10".

Table 2. SPArc Instruction operation encoding.

Format	op	Instructions
1	01	CALL
2	00	Branch, SETHI
3	11	Memory Instruction
3	10	Arithmetic, Logical, Shift, and remaining

Table 3. SPArc Instruction Branch and SETHI.

Op2	Instructions
000	Unimplemented
001	Unimplemented
010	Bicc
011	Unimplemented
100	SETHI
101	Unimplemented
110	FBfcc
111	CBccc

Table 3 shows the op2 for Branch and SETHI. The value op2 "000", "001", "011", and "101" are not used in this version of SPArc. The op2 "010" is used for Branch on integer condition codes. The op2 "100" is used for SETHI. The op2 "110" is used for Branch on Floating Point Condition Codes, and op2 "111" is used for Branch on Coprocessor Condition Codes.

The value op3 varies for every Format 3 instructions as shown in

Table 4. op3 for Memory Access Instructions

op3 [3:0]	op3 [5:4] 0	1	2	3	op3 [5:4] 0	1	2	3	op3 [3:0]
0	LD	LDA	LDF	LDC	-	-	-	-	8
1	LDUB	LDUBA	LDFSR	LDCSR	LDSB	LDSBA	-	-	9
2	LDUH	LDUHA	-	-	LDSH	LDSHA	-	-	A
3	LDD	LDDA	LDDF	LDDC	-	-	-	-	B
4	ST	STA	STF	STC	-	-	-	-	C
5	STB	STBA	STFSR	STCSR	LDSTUB	LDSTUBA	-	-	D
6	STH	STHA	STDFQ	STDCQ	-	-	-	-	E
7	STD	STDA	STDF	STDC	SWAP	SWAPA	-	-	F

Table 5. op3 for Arithmetic/Logic/Shift Instructions

op3 [3:0]	op3 [5:4] 0	1	2	3	0	1	2	3	op3 [3:0]
0	ADD	ADDcc	TADcc	WRASR WRY	ADDX	ADDXcc	RDASR RDY STBAR RDPSR	JMPL	8
1	AND	ANDcc	TSUBcc	WRPSR	-	-	-	RETT	9
2	OR	Orcc	TADccTV	WRWIM	UMUL	UMULcc	RDWIM	Ticc	A
3	XOR	XORcc	TSUBccTV	WRTBR	SMUL	SMULcc	RDTBR	FLUSH	B
4	SUB	SUBcc	MULScc	FpoP1	SUBX	SUBXcc	-	SAVE	C
5	ANDN	ANDNcc	SLL	FpoP2	-	-	-	RSTR	D
6	ORN	ORNcc	SRL	CpoP1	UDIV	UDIVcc	-	-	E
7	XNOR	XNORcc	SRA	CpoP2	SDIV	SDIVcc	-	-	F

Table 6. op2 for Branch Instructions

cond [3:0]	op = "00" 2	6	7	0x3A
0	BN	FBN	FBN	TN
1	BE	FBNE	CB123	TE
2	BLE	FBLB	CB123	TLE
3	BLE	FBUL	CB13	TLE
4	BLEU	FBLG	CB1	TLEU
5	BCS	FBUG	CB23	TCS
6	BNEG	FBG	CB2	TNEG
7	BVS	FBU	CB3	TVS
8	BA	FBA	CBA	TA
9	BNEG	FBE	CB0	TNE
A	BG	FBUE	CB03	TG
B	BGE	FBGE	CB02	TGE
C	BGE	FBUGE	CB023	TO
D	BCC	FBLE	CB01	TCC
E	BPOS	FBULE	CB013	TPOS
F	BVC	FBO	CB012	TVC

Register Transfer Notation is designed for each instruction at each step. At T0, the microprocessor always get instruction from memory. At the same time address for the next instruction has already been determined, except in Branch instructions. At T1, operand1 is taken from Source Register rs1, while the operand2 can be taken from Source Register rs2 or immediate depends on the i value. At T2, Arithmetic Logic Unit (ALU) in Execute Stage performs ADD operation. At T3, the Execute Stage has finished with the operation. The Memory Access Stage perform no operation, since memory access is only performed in Memory-Access instructions. At T4, the result of the operation is stored in Destination Registers rd. RTN Design for ADD instruction is shown in table 7.

Table 7. RTN for Arithmetic Instructions.

Step	RTN for Arithmetic Instructions ADD	SUB	UMUL
T0	IR ← M[PC] : nPC ← PC + 4;		
T1	opr1 ← R[rs1]: opr2 := (IR<13>=0 → R[rs2]): opr2 := (¬IR<13>=0 → R[IR<12..0>]);		
T2	res := opr1 + opr2;	res := opr1 - opr2;	res1 := (opr1 * opr2)<63..32>: res2 := (opr1 * opr2)<31..0>;
T3	nop		
T4	R[rd] ← res: PC ← nPC;	R[rd] ← res: PC ← nPC;	R[rd] ← res2: R[x] ← res1: PC ← nPC;

RTN for SUB instruction is also shown in

Table 7. At T0 and T1, the microprocessor performs the same task as ADD instruction. At T2, Artihmetic Logic Unit (ALU) in Execute Stage performs SUB operation. At T3, the Execute Stage has finished with the operation and no operation is performed. At T4, the result of the operation is stored in Destination Registers rd.

RTN for UMUL instruction is also shown in table 7 At T0 and T1, the microprocessor performs the same task as ADD and SUB instruction. At T2, Artihmetic Logic Unit (ALU) in Execute Stage performs UMUL operation. The 64-bit result of this operation is stored in signal res1 and res2. At T3, the Execute Stage has finished with the operation and no operation is performed. At T4, the res1 signal is stored in Destination Register rd, while the res2 signal is stored in the register x.

RTN for logic instructions is shown in Table 8. At T0 and T1, the microprocessor performs the same task as arithmetic instructions. At T2, ALU in Execute Stage performs respective logic instructions, for example OR, AND, and XOR. At T3, the Execute Unit

performs no operation. At T4, the result of the operation is stored in Destination Registers rd.

Table 8. RTN for Logic Instructions

Step	RTN for Arithmetic Instructions		
	OR	AND	XOR
T0	IR ← M[PC] : nPC ← PC + 4;		
T1	opr1 ← R[rs1]: opr2 := (IR<13>=0 → R[rs2]): opr2 := (¬IR<13>=0 → R[IR<12..0>]);		
T2	res := opr1 v sopr2;	res := opr1 ∧ opr2;	res := opr1 ⊻ opr2;
T3	nop		
T4	R[rd] ← res: PC ← nPC;		

RTN for logic instructions is shown in Table 8. At T0 and T1, the microprocessor performs the same task as arithmetic instructions. At T2, ALU in Execute Stage performs respective logic instructions, for example OR, AND, and XOR. At T3, the Execute Unit performe no operation. At T4, the result of the operation is stored in Destination Registers rd.

RTN for Memory-Access Instructions is shown in table 9 At T0 and T1, the microprocessor performs the same task as the previous types of instruction, unless for the STORE instructions, the value to be stored in Memory must first be taken from the R[rd] and be stored in the temporary register MDR2. At T2, Execute unit performs the ADD operation and the result is stored as MAR. At T3, the value in the temporary register MDR2 is stored in Memory at adress MAR.

Table 9. RTN for Memory-Access Instructions.

Step	RTN for Arithmetic Instructions	
	LOAD	STORE
T0	IR ← M[PC] : nPC ← PC + 4;	
T1	opr1 ← R[rs1]: opr2 := (IR<13>=0 → R[rs2]): opr2 := (¬IR<13>=0 → R[IR<12..0>]) ;	MDR2 ← R[rd];
T2	MAR ← opr1 + opr2;	
T3	MDR ← M[MAR];	M[MAR] ← MDR:
T4	R[rd] ← MDR: PC ← nPC;	PC ← nPC;

RTN for Control-Transfer instructions is shown in table 10 . At T0, the Fetch Stage gets instruction from memory. At the same time address for the next instruction nPC has already been determined. At T1, CALL instructions assign signal disp30 from Least Significant 30 Bit (LSB) of Instruction Register. The BRANCH instructions, on the other hand, assign signals a, cond, disp22 from the bit 29 of IR, the bit 28 downto 25, from the Least Significant 22 Bits of IR respectively, and icc from the bit 23 downto 20 of Processor State Register (PSR). At T2 and T3, both units perform no operation. At T4, the CALL instructions perform Control Transfer, which updates PC with the displacement taken from the 30 LSB of IR. The BRANCH instructions, at T4 also perform PC update, if annul bit a is equal 0, PC is added by disp22, otherwise PC is set to nPC.

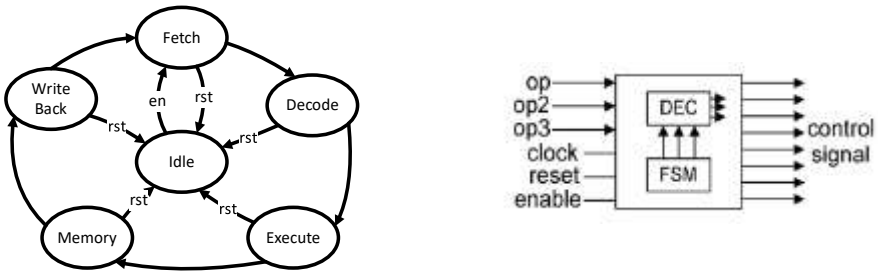
Table 10. RTN for Control-Transfer Instructions.

Step	RTN for Control Transfer Instructions	
	CALL	BRANCH
T0	IR ← M[PC] : nPC ← PC + 4;	
T1	disp30 := IR<29..0>;	a := IR<29>: cond := IR<28..25>: icc := PSR<23..20>: disp22 := IR<21..0>: specific Branch decode(icc);

T2	nop;	
T3	nop;	
T4	PC ← PC + IR<29..0>#2@0;	a=1 → PC ← nPC: a=0 → PC + disp22;

The datapath in general is the flow of data in a system and designed based on the RTN. The dotted-line box contains datapath. The dashed line contains the processor Core, which is Datapath and Control Unit. The Control Unit is designed using Finite State Machine as shown in Figure 1 (a). The Microprocessor is in the Idle state. At the present of enable signal en, the State moves to Fecth, Decode, Execute, Memory, and Write Back automatically unless the signal rst is present to force the next state to Idle.

Diagram of the Control Unit is shown in Figure 1 (b). The output of the FSM is connected to Decoder to produce the output signals, which description is shown in table 11 At T0, nPC, IR, and ROM are enabled to perform instruction fetch from ROM. At T1, en_dec is enabled to get operand 1 from Register rs1 and operand 2 from either Register rs2 or simm13 depending on the i bit. Mean while, the signal en_rd, en_rs1 are enabled, and the signal en_rs2 is enabled when i = '0'. At T2, en_exec_out is enabled to indicate the Execute Stage is working, signal alu_ctrl depends on the instructions. At T3, signals are active only in Memory-Access instructions. At T4, the result of the operation is stored in the Register rd, the next instruction is ready. Signal en_wb, en_wrt, and en_pc are enabled.



(a) Proposed Finite State Machine (b) Diagram of Control Unit

Figure 1. Control Unit Design.

Table 11. Control signal design and description.

Control Signal	Description	T0	T1	T2	T3	T4
en_pc	PC enable	0	0	0	0	1
en_npc	nPC enable	1	0	0	0	0
en_ir	IR enable	1	0	0	0	0
en_dec	op1 and op2_reg enable	0	1	0	0	0
en_mar	MAR enable	0	0	0	0	0
en_mdr2	MDR2 enable	0	0	0	0	0
en_psr	PSR enable	0	0	0	0	0
en_mdr	MDR enable	0	0	0	0	0
en_wb	Write back enable	0	0	0	0	1
wr_ram	Write/read RAM	0	0	0	0	0
aluctrl	ALU control	00000	00000	11001*	00000	00000
en_rom	ROM enable	1	0	0	0	0
en_exe_out	Execute Unit enable	0	0	1	0	0
en_rd	Read enable	0	1	0	0	0
en_rs1	rs1 enable	0	1	0	0	0
en_rs2	rs2 enable	0	1	0	0	0
en_wrt	Write enable	0	0	0	0	1

*instruction dependent.

The proposed designed is shown in Figure 2. The Core is functionally tested by connecting it to Read-Only Memory (ROM) and Random-Access Memory (RAM). ROM is

used to store the instruction, which is the test vector, while RAM is used to store the temporary result of the instructions.

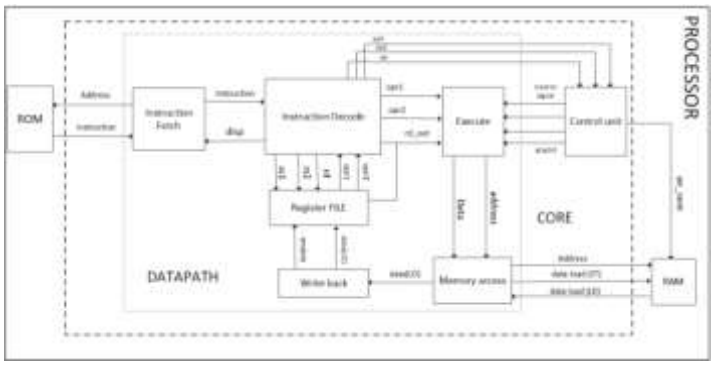
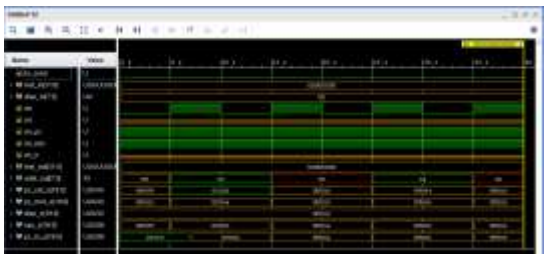


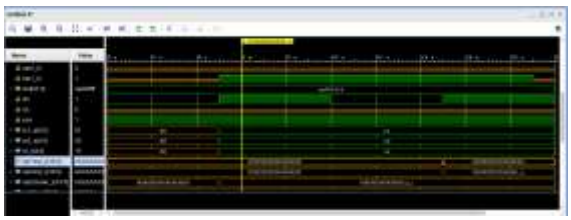
Figure 2. Proposed architecture.

3. RESULTS AND DISCUSSIONS

This section discusses the result and discussion of this work. The Microprocessor core is connected to ROM and RAM as previously shown in Figure 2. Instructions are used as test vectors. The temporary results are stored in registers and memory. Simulation result is shown in Figure 3. Signal values are shown in Figure 3 (a), while the decoded signals are shown in Figure 3 (b).



(a) Signal values from Fetch unit



(b) Instruction decode signal

Figure 3. Simulation result.

Reports are shown in this sections based on the FPGA implementation.

The power report and the thermal property of the design is shown in Table 12. The power consumption of the design is 0.266 W, 0.173 W of which is used for Dynamics, while 0.093 W is used to power up Device Static. The Dynamics part of the power consumption is used to power up Signals (0.047 W), Logic (0.039 W), and I/O (0.086 W). The junction temperature at 28.1 ºC and the thermal margin of 56.9 ºC, which mean the proposed design works in the normal condition.

Table 12. Power report of the proposed design.

On-Chip Power		Thermal properties	
Total On-Chip Power	0.266 W	Junction Temperature	28.1 ºC
Dynamic	0.173 W	Thermal Margin	56.9 ºC
Signals	0.047 W	Thermal Resistance	11.5 ºC/W
Logic	0.039 W		
I/O	0.086 W		
Device Static	0.093 W		

Timing report of the top 10 signal in the design is shown in Table 13. The most delayed paths have 0.288 ns total delay, with 0.178 logic delay and 0.110 net delay, which means 61.8% consist of logic delay.

Table 13. Timing report of the proposed design for top 10 path.

Path No.	Delay (ns)			Logic %
	Total	Logic	Net	
1	0.288	0.178	0.110	61.8
2	0.288	0.178	0.110	61.8
3	0.288	0.178	0.110	61.8
4	0.279	0.178	0.101	63.8
5	0.276	0.158	0.118	57.3
6	0.274	0.158	0.116	57.6
7	0.274	0.158	0.116	57.6
8	0.272	0.158	0.114	58.2
9	0.270	0.158	0.112	58.6
10	0.270	0.158	0.112	58.6

4. CONCLUSION

In this research, a SPArc Subset General-Purpose Microprocessor was designed and implemented using 5-stage direct connection between stages. These stages are Fetch, Decode, Execute, Memory, and Write Back. The proposed architecture works as functionally tested using VHDL simulation and can execute instructions as shown that the register contains the result of the operations. The proposed architecture comprises of the Datapath, and the Control Unit. The design was implemented in FPGA synthesize, resulting the maximum of 58 MHz and 61.8% logic as shown in timing report. The proposed architecture also works in the desired thermal margin of 56.9 ºC and the junction temperature of 28.1 ºC. The thermal resistance also shows good property of 11.5 ºC/W, which means the system will not easily increase as it operates. This design will be integrated with the floating point unit and used to realize cognitive processor in the near future (Bachri et al., 2019; Sereati et al., 2020).

REFERENCES

Bachri, K. O., Khayam, U., Soedjarno, B. A., Sumari, A. D. W., & Ahmad, A. S. (2019). Cognitive artificial-intelligence for doernenburg dissolved gas analysis interpretation. *Telkomnika (Telecommunication Computing Electronics and Control)*, *17*(1), 268–274. https://doi.org/10.12928/TELKOMNIKA.v17i1.11612

Fengler, A., Jung, P., & Caire, G. (2021). SPARCs for Unsourced Random Access. *IEEE Transactions on Information Theory*, *67*(10), 6894–6915. https://doi.org/10.1109/TIT.2021.3081189

Frühauf, J.-L., Hawich, M., & Blume, H. (2024). Enhancing RISC-V Processor Performance in Harsh Environments through Data Cache Optimization. *2024 Panhellenic Conference on Electronics & Telecommunications (PACET)*, 1–4. https://doi.org/10.1109/PACET60398.2024.10497077

Hemanthkumar, P. B., Anireddy, S. R., T, J. F., & R, V. (2022). Introduction to ARM processors & its types and Overview to Cortex M series with deep explanation of each of the processors in this Family. *2022 International Conference on Computer Communication and Informatics (ICCCI)*, 1–8. https://doi.org/10.1109/ICCCI54379.2022.9740768

Hennessy, J. L., & Patterson, D. A. (2019). *Computer Architecture: A Quantitative Approach* (6th ed.). Elsevier Inc.

Hepola, K., Multanen, J., & Jääskeläinen, P. (2024). Energy-Efficient Exposed Datapath Architecture With a RISC-V Instruction Set Mode. *IEEE Transactions on Computers*, *73*(2), 560–573. https://doi.org/10.1109/TC.2023.3337313

Hidri, L., Al-Samhan, A. M., & Mabkhot, M. M. (2019). Bounding Strategies for the Parallel Processors Scheduling Problem with No-Idle Time Constraint, Release Date, and Delivery Time. *IEEE Access*, *7*, 170392–170405. https://doi.org/10.1109/ACCESS.2019.2954905

Jyotsna, K. A., Siddharth, S. S., Kumar, P. S., & Madhavi, B. K. (2022). Design of 32-Bit ARM Processor Data Path Units utilizing DVS Current Mode Technique. *2022 IEEE International Symposium on Smart Electronic Systems (ISES)*, 263–266. https://doi.org/10.1109/iSES54909.2022.00060

Kulshreshtha, A., Moudgil, A., Chaurasia, A., & Bhushan, B. (2021). Analysis of 16-Bit and 32-Bit RISC Processors. *2021 7th International Conference on Advanced Computing and Communication Systems, ICACCS 2021*, 1318–1324. https://doi.org/10.1109/ICACCS51430.2021.9441873

Lindholm, T., Yellin, F., Bracha, G., & Buckley, A. (2014). *The Java® Virtual Machine Specification, Java SE 8 Edition-Addison-Wesley Professional* (8th ed.). Oracle.

Malatesta, F., Weigand, R., & Andersson, J. (2023). GR765: SPARC and RISC-V Multiprocessor System-on-Chip. *2023 European Data Handling & Data Processing Conference (EDHPC)*, 1–4. https://doi.org/10.23919/EDHPC59100.2023.10396360

Meyer-Baese, U. (2021). Embedded Microprocessor System Design using FPGAs. In *Embedded Microprocessor System Design using FPGAs*. Springer International Publishing. https://doi.org/10.1007/978-3-030-50533-2

Naffziger, S., Lepak, K., Paraschou, M., & Subramony, M. (2020). AMD Chiplet Architecture for High-Performance Server and Desktop Products. *2020 IEEE International Solid-State Circuits Conference (ISSCC 2020)*, 570. https://doi.org/10.1109/ISSCC19947.2020.9063103

Phangestu, A. E., Mujiono, I. T., Kom, M. I., & Zaini, S. A. (2022). Five-Stage Pipelined 32-Bit RISC-V Base Integer Instruction Set Architecture Soft Microprocessor Core in VHDL. *2022 International Seminar on Intelligent Technology and Its Applications (ISITIA)*, 304–309. https://doi.org/10.1109/ISITIA56226.2022.9855292

Pomianowski, A. (2021). RDNA™ 2 Gaming Architecture. *2021 IEEE Hot Chips 33 Symposium (HCS)*, 1–18. https://doi.org/10.1109/HCS52781.2021.9567555

Rotem, E., Yoaz, A., Rappoport, L., Robinson, S. J., Mandelblat, J. Y., Gihon, A., Weissmann, E., Chabukswar, R., Basin, V., Fenger, R., Gupta, M., & Yasin, A. (2022). Intel Alder Lake CPU Architectures. *IEEE Micro*, *42*(3), 13–19. https://doi.org/10.1109/MM.2022.3164338

Sakamoto, M., Katsuno, A., Inoue, A., Asakawa, T., Ueno, H., Morita, K., & Kimura, Y. (2002). *Microarchitecture and Performance Analysis of a SPARC-V9 Microprocessor for Enterprise Server Systems*.

Savadatti, M. B., R, V., L, S. S. M., V, S., & Reddy, T. J. (2024). Data Transfer Using AMBA Bus: An Empirical Approach. *2024 2nd International Conference on Artificial Intelligence and Machine Learning Applications Theme: Healthcare and Internet of Things (AIMLA)*, 1–6. https://doi.org/10.1109/AIMLA59606.2024.10531613

Schöne, R., Ilsche, T., Bielert, M., Velten, M., Schmidl, M., & Hackenberg, D. (2021). Energy Efficiency Aspects of the AMD Zen 2 Architecture. *2021 IEEE International Conference on Cluster Computing (CLUSTER)*, 562–571. https://doi.org/10.1109/Cluster48925.2021.00087

Sereati, C. O., Sumari, A. D. W., Adiono, T., & Ahmad, A. S. (2020). Towards cognitive artificial intelligence device: An intelligent processor based on human thinking emulation. *Telkomnika (Telecommunication Computing Electronics and Control)*, *18*(3), 1475–1482. https://doi.org/10.12928/TELKOMNIKA.v18i3.14835

Soundari, D. V., Ganesh, M. K. S., Raman, I., & Karthick, R. (2021). Enhancing network-on-chip performance by 32-bit RISC processor based on power and area efficiency. *Materials Today: Proceedings*, *45*, 2713–2720. https://doi.org/10.1016/j.matpr.2020.11.550

Suárez, D., Almeida, F., & Blanco, V. (2023). *Comprehensive Analysis of Energy Efficiency and Performance of ARM and RISC-V SoCs*. https://doi.org/10.21203/rs.3.rs-3405993/v1

Tang, G. M., Qu, P. Y., Ye, X. C., & Fan, D. R. (2018). Logic Design of a 16-bit Bit-Slice Arithmetic Logic Unit for 32-/64-bit RSFQ Microprocessors. *IEEE Transactions on Applied Superconductivity*, *28*(4). https://doi.org/10.1109/TASC.2018.2799994

Vera, X. (2020). Inside Tiger Lake: Intel's Next Generation Mobile Client CPU. *2020 IEEE Hot Chips 32 Symposium (HCS)*, 1–26. https://doi.org/10.1109/HCS49909.2020.9220443

Waterman, A., & Asanovi´c, K. A. (2017). *The RISC-V Instruction Set Manual Volume I: User-Level ISA Document Version 2.2 Editors*.

Xiang Lim, D., & G, S. K. (2019). Pipelined MIPS Simulation: A plug-in to MARS simulator for supporting pipeline simulation and branch prediction. In *2019 IEEE International Conference on Engineering, Technology and Education (TALE)*.

Yue, A., & Mehta, S. (2023). An Application-Oriented Approach to Designing Hybrid CPU Architectures. *2023 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 92–102. https://doi.org/10.1109/ISPASS57527.2023.00018

Zhang, W., Zhang, Y., & Zhao, K. (2021). Design and Verification of Three-stage Pipeline CPU Based on RISC-V Architecture. *2021 5th Asian Conference on Artificial Intelligence Technology (ACAIT)*, 697–703. https://doi.org/10.1109/ACAIT53529.2021.9731161