International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 10 Issue: 09 | Sep 2023 www.irjet.net p-ISSN: 2395-0072

DESIGN AND IMPLEMENTATION OF I2C AND UART BLOCK IMPLEMENTATION FOR RISC-V SOC

Sarala T[1], Kiran K N[2], Pragathi Panda[3], Raksha Nagendra[4], Suguna Chandrasekhar[5]

[1]*Sarala T, Assistant Professor, Dept. of ECE, BNM Institute of Technology, Karnataka, India.*
[2]*Kiran K N, Assistant Professor, Dept. of ECE, BNM Institute of Technology, Karnataka, India.*
[3]*Pragathi Panda, Student, Dept. of ECE, BNM Institute of Technology, Karnataka, India.*
[4]*Raksha Nagendra, Student, Dept. of ECE, BNM Institute of Technology, Karnataka, India.*
[5]*Suguna Chandrasekar, Student, Dept. of ECE, BNM Institute of Technology, Karnataka, India.*

Abstract - *Embedded processors are considered to be the processing engines of latest smart IoT devices. For many years now, the processors were primarily based on the Arm Instruction Set Architecture (ISA). In recent times, the RISC-V ISA standard which is both free and open has attracted the attention among researchers from the industry and academia and is well on its way in becoming mainstream. Several companies are already involved in the design of RISC-V processors, whilst many important operating systems and major tool chains have started to support RISC-V. Along with the objective to further explore the RISC-V processor, this paper aims at focusing on the architecture of the RISC-V processor and digital block peripherals which include the I2C and UART along with a detailed survey on RISC-V technologies is done. This paper summarizes the representative peripherals of RISC-V digital block architecture.*

Key Words: I2C(Inter-Integrated Circuit) , RISC V (Reduced Instruction Set Computer) , UART (Reduced Instruction Set Computer) , SOC (System on Chip), VLSI (Very Large Scale Integrated Circuit) Implementation.

1.INTRODUCTION

The RISC-V ISA stands out for its load store architecture, bit patterns that make it easier for CPU multiplexers to function, IEEE 754 floating-point, and architectural neutrality. To speed up sign extension, it fixes the placement of the most important elements. The instruction set is intended to be used in a variety of ways. It is flexible width and extensibility allow for the constant addition of new encoding bits.It supports several subsets and three word-widths, including 32, 64, and 128 bits [1]. For the three word-widths, the definitions of each subset differ a little bit. The subsets support small embedded device applications, computers with advanced vector processors including the large scale 19 inch rack mounted parallel computers. Because a shortage of memory address space is the most irrecoverable fault in instruction set design, the instruction set space for the 128-bit extended version of the ISA was set aside due to 60 years of industry experience.

Artificial intelligence, augmented reality, automotive, cloud servers, computer devices and controllers, general purpose processors, Internet of Things, Machine Learning, Network Edge, and Virtual Reality are just a few of the industries that employ RISC V.

The RISC V architecture (RV12) is shown in the Fig 1. The RV12 is highly configurable with a single-core RV32I and RV64I compliant RISC CPU which is used in embedded fields. TheRV12 is also from a 32 or 64-bit CPU family depending on the industrial standard RISC V instruction set. The RV12 executes a Harvard architecture for simultaneous access to instruction as well as data memory. It includes a 6-stage pipeline that helps in optimizing. overlapping between the execution as well as memory accesses to improve efficiency. This architecture mainly includes Branch Prediction, Data Cache, Debug Unit, Instruction Cache, & optional Multiplier or Divider Units.

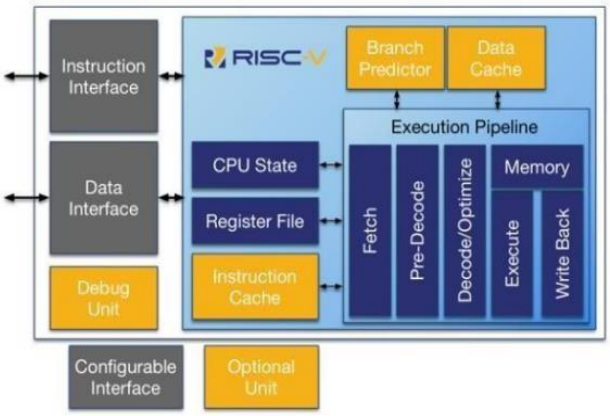

Fig-1: RISC V architecture [2]

The RISC-V execution pipeline include five stages like Instruction Fetch (IF), Instruction Decode (ID), EXecute (EX), MEMory access (MEM) & Register Write-Back (WB) as shown in Fig 1. In Instruction Fetch or IF stage, a single instruction is read from the program counter (PC) and instruction memory which is updated to the next instruction. Once RVC Support is allowed, then the Instruction Pre-Decode stage will decode a 16-bit-compressed instruction into a native 32-bit instruction.

1.1 Objectives

- To familiarize with the ASIC flow that includes RTL, verification, synthesis (SDC, genus), placement and routing (innovus), timing (tempus) analysis, and PV checks.
- To implement digital block of the RISC processor.
- To integrate the blocks to form a RISC SoC.

1.2 LITERATURE SURVEY

In this paper, low power optimization procedures are used to realize the RISC-V processor with an objective to reduce the power of the processor. Also, the clock tree optimization and clock gating techniques are proposed to reduce dynamic power along with Multi-Vth, Power Shut Off (PSO) and Multi Supply Voltage (MSV). The techniques also help to reduce the leakage power. The Encounter RTL Compiler is used to do the synthesis and to generate the netlist at gate level. The Cadence Encounter Digital Implementation System is used to design the VLSI backend design flow. The Common Power Format is used to capture the power intent, which in turn aids to implement the low power processor.[1]

Andrew S. Waterman et.al., in their work "Improving Energy Efficiency and Reducing Code Size with RISC-V Compressed" have proposed a system that aims to both energy efficiency and performance of the RISC-V Instruction Set Architecture. The authors propose a "RISC-V Compressed (RVC)" and a "variable-length instruction set extension." It is to be noted that the most frequent instructions are encoded in the RVC, thus resulting in programs using RVC to be 25% smaller than programs using the RISC-V. The instruction cache misses incurred by the RVC programs in also less.[2]

Pasquale Davide Schiavone et.al., in the work "Arnold: an eFPGA Augmented RISC-V SoC for Flexible and Low-Power IoT End-Nodes." The authors have proposed a novel SoC that exploits body biasing to reduce the leakage power of the embedded FPGA fabric. The SoC proposed by the authors is also equipped to overcome the challenges of IoT applications such as having the ability to interface sensors and accelerators with non-standard interfaces, processing tasks on the go and so on.[3]

2. Methodology

A synchronous protocol that combines the best elements of SPI and UARTs is called the Inter Integrated Circuit Bus (I2C, marked "1 square C"). The I2C is by design a two wire, serial device which take turns sending and receiving bus that only allows a single way of communication at a time. I2C was created with the goal of making it simple to connect microprocessor/microcontroller systems to the peripheral chips found in televisions. Two bus routes make up the I2C bus. Specifically, Serial Data SDA and Serial Clock SCL.

The SDA line oversees sending serial data between devices, whereas the SCL line oversees producing synchronization clock pulses. I2C devices can be connected to an I2C bus, which is a shared bus system. Both 'Master' and 'Slave' devices may be used with devices connected to the 12C bus. Controlling the communication is done by the 'Master' device, which starts and stops data transport. transferring data while producing the required synchronization clock pulses. The 'Master' and 'Slave' devices may function either as transmitters or receivers. 'Slave' devices wait for the Master's commands and reply when they get them. The Master device exclusively produces the synchronization clock signal, regardless of whether a master is working as a transmitter or receiver. Multimasters on same bus are supported by 12C.

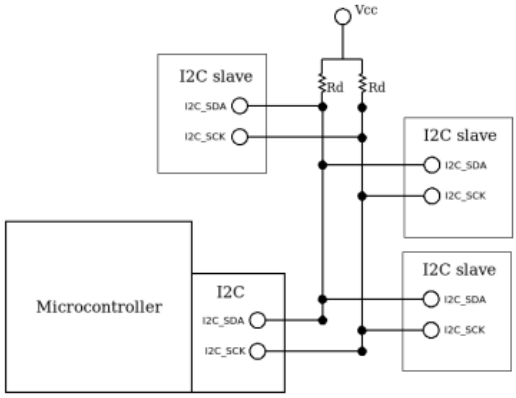

Fig -2: I2C architecture

The following is a list of the steps involved in connecting with a 12C slave device:

1. The master device pulls the Serial clock line (SCL) to "HIGH".

2. When the SCL line reaches logic "HIGH," the master device pulls the data line (SDA), signalling the "Start" condition for data transfer.

3. The master device sends the address of the "slave" device it wants to speak to over the SDA line. The SCL line generates clock pulses to synchronize the slave device's bit reception. The MSB is usually transferred first when transferring data. The data in the bus is still valid during the 'HIGH' phase of the clock signal.

4. In accordance with the specification, the master device offers the proper Read or Write bit (Bit value = 1 Read operation, Bit value = 0 Write operation).

5. The slave device waits for the acknowledgement bit from the master device. Its address is delivered on the bus together with the order for the Read/Write operation.

6. Every time a slave device connected to the bus receives an address request from a master device, the slave device compares the address requested with the address provided. When the slave device reacts, it crosses the SDA line with an acknowledgment bit (Bit value = 1).

7. After receiving the acknowledgement signal, if the desired operation is "Write to device," the Master device sends the 8-bit data to the Slave device through the SDA line. If the requested action is "Read from device," data is transmitted through the SDA line from the slave device to the master.

8. The master device waits for the acknowledgement bit from the device and sends an acknowledgement bit to the Slave device when a write operation is finished, and a byte transfer is finished.

9. When the clock line SCL reaches logic "HIGH" (indicating the "STOP" condition), the master device halts the transfer by pulling the SDA line "HIGH-I."

Table -1: Power report of I2C

Type	Instances	Area	Area %	Leakage Power (nW)	Leakage Power %	Internal Power (nW)	Internal Power %
sequential	150	1256.166	63.8	35.973	63.2	11772.666	80.8
inverter	32	22.230	1.1	0.688	1.2	24.016	0.2
clock_gating_integrated_cell	13	84.474	4.3	2.815	4.9	1726.374	11.8
logic	329	606.708	30.8	17.416	30.6	1054.293	7.2
physical_cells	0	0.000	0.0	0.000	0.0	0.000	0.0
total	524	1969.578	100.0	56.893	100.0	14577.349	100.0

The aggregate power consumed by all the instances present in the design is 524W. The leakage power is 56.896nW and internal power is 14577.349nW.

Table -2: Power report of UART

Type	Instances	Area	Area %	Leakage Power (nW)	Leakage Power %	Internal Power (nW)	Internal Power %
sequential	96	551.646	58.6	18.473	61.3	4957.952	83.6
inverter	22	15.048	1.6	0.435	1.4	5.927	0.1
clock_gating_integrated_cell	8	51.984	5.5	1.733	5.8	576.549	9.7
logic	161	322.164	34.2	9.481	31.5	387.508	6.5
physical_cells	0	0.000	0.0	0.000	0.0	0.000	0.0
total	287	940.842	100.0	30.122	100.0	5927.937	100.0

The total power consumption by all the instances present in the design is 287W. The leakage power is 30.122nW and internal power is 5927.937nW.

3. Results & Discussion

The steps leading up to the final design of the I2C and UART blocks include the successful implementation of the codes. Each gate was placed on to the digital block of the chip abiding the timing, area, and power constraints. After performing all the steps required in the configuration of the block of the chip according to the SUTRA methodology, the following screenshots, are the results of I2C and UART obtained.

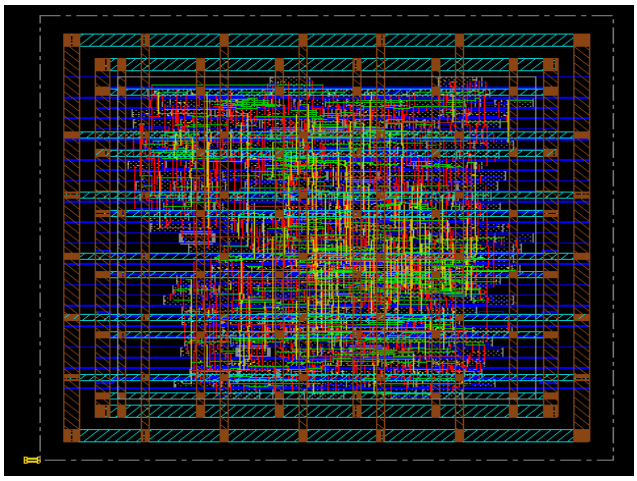

Fig -3: Routing for I2C

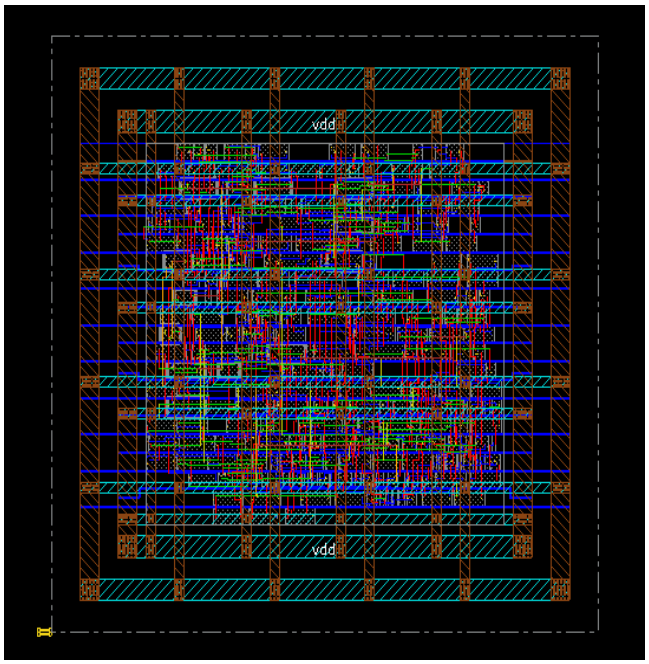

Fig -4: Routing for UART

The routing for I2C and UART screenshots depict the physical success in the designing of the block. The functionality verification is done by running a testbench against the code built. This is done using the Xilinx Vivado tool. Upon running

the code, the simulations obtained are shown in Fig 5 are obtained.

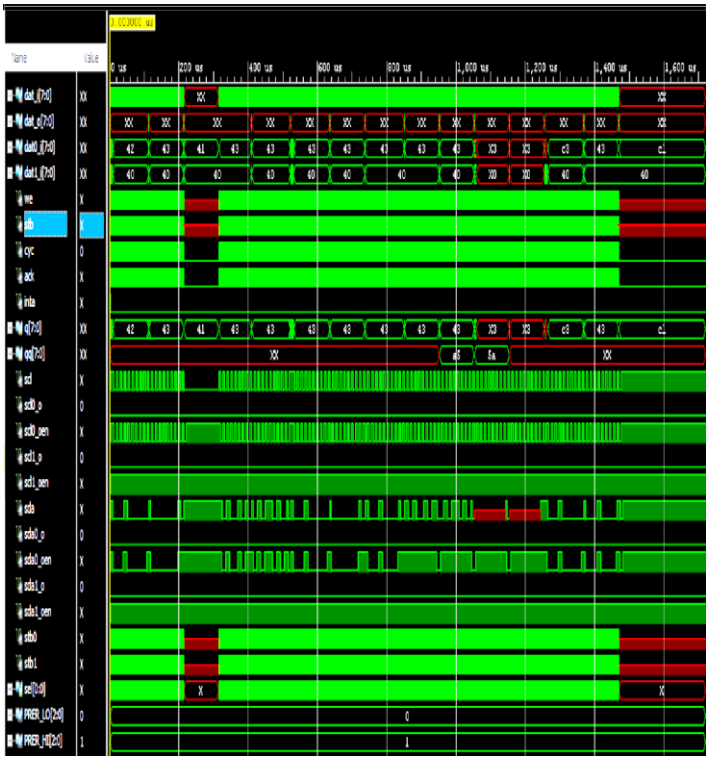

Fig -5: Simulation for I2C

In the above simulation for I2C screenshot, it is observed that the select line, sel chooses one of the slaves – stb0 or stb1. In this case, stb0 is chosen which is linked to the respective data in data0_i[7:0]. This data is input to the master and the successful transfer is observed on the line q[7:0].

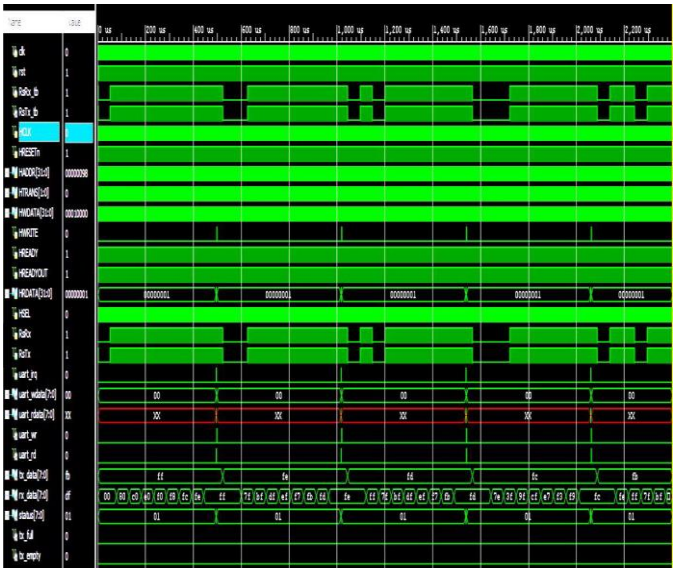

Fig 6 –: UART simulation

The simulation for UART depicts the verification of UART working.

It is observed that the data transfer occurs in cycles of 8 from tr_data[7:0] to rx_data[7:0]. After successful transfer of each data byte, an acknowledgement signal UART_rd as a pulse is generated. This data successfully verifies the physical and functional working of the I2C and UART blocks.

4. CONCLUSIONS

The RISC-V architecture is creating a tremendous impression on SoC designers and architects due to its capabilities. It is estimated that a significant portion of designs, revenues and unit shipments will be due to AI SoCs that use the RISC-V architecture in the next few years.

This proposed work mainly aims at integrating the peripherals with the RISC core. I2C and UART are peripheral communication protocols targeted for low power, low complexity, low cost and low performance applications. These protocols are part of every SOC, including the complex mobile SOC and server SOC platforms. Hence learning protocols is an important aspect for every design and verification engineer.

REFERENCES

[1] Shashi Kumar V, Gurusiddayya Hiremath "Low Power Implementation of RISC- V Processor" IOSR Journal of VLSI and Signal Processing (IOSR-JVSP) Volume 6, Issue 3, Ver. II (May. -Jun. 2016), PP 59-64 e-ISSN: 2319 – 4200, p-ISSN No.: 2319 – 4197 www.iosrjournals.

[2] Andrew S. Waterman "Improving Energy Efficiency and Reducing Code Size with RISC-V Compressed", Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of Master of Science, Plan II.

[3] Pasquale Davide Schiavone, Davide Rossi Member, IEEE, Alfio Di Mauro, Frank Gürkaynak, Timothy Saxe, Mao Wang, Ket Chong Yap, Luca Benini Fellow, IEEE "Arnold: an eFPGA-Augmented RISC-V SoC for Flexible and Low-Power IoT End-Nodes", Submitted on 25 Jun 2020

[4] Z Cao , Q Lv , Y Wang , M Wen , N Wu , C Zhang "A Compression Instruction Set Design based on RISC-V for Network Packet Forwarding", IOP Conf. Series: Journal of Physics: Conf. Series 1026 (2018) 012001 doi :10.1088/1742- 6596/1026/1/012001

[5] Etki Gür; Zekiye Eda Sataner; Yusuf H. Durkaya; Salih Bayar, "FPGA Implementation of 32-bit RISC-V Processor with Web-Based Assembler- Disassembler", 2018 International Symposium on Fundamentals of Electrical

Engineering (ISFEE) ,01-03 November 2018, DOI: 10.1109/ISFEE.2018.8742406.

[6] W. A. Arbaugh, D. J. Farber, and J. M. Smith, "A secure and reliable bootstrap architecture," in Security and Privacy, 1997. Proceedings.,1997 IEEE Symposium on. IEEE, 1997, pp. 65–71.

[7] I. Lebedev, K. Hogan, and S. Devadas, "Secure boot and remote attestation in the sanctum processor," in 2018 IEEE 31st Computer Security Foundations Symposium (CSF). IEEE, 2018, pp. 46–60.

[8] S. Sau, J. Haj-Yahya, M. M. Wong, K. Y. Lam, and A. Chattopadhyay, "Survey of secure processors," in Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS), 2017 International Conference on. IEEE, 2017, pp. 253–260.

[9] X. Ruan, "Boot with integrity, or dont boot," in Platform Embedded Security Technology Revealed. Springer, 2014, pp. 143–163.

[10] Michael Gautschi et al. "Near-threshold RISC-V core with DSP extensions for scalable IoT endpoint devices". In: IEEE Transactions on Very Large Scale Integration (VLSI) Systems 25.10 (2017), pp. 2700–2713.

[11] Matthew R Guthaus et al. "OpenRAM: An open source memory compiler". In: 2016 IEEE/ACM Interna tional Conference on Computer-Aided Design (ICCAD). IEEE. 2016, pp. 1–6.

[12] Ted Marena. "RISC-V: high performance embedded SweRV™ core microarchitecture, performance and CHIPS Alliance." In: Western Digital Corporation (2019)

[13] A. Waterman, Y. Lee, D. A. Patterson, and K. Asanovic, "The RISC V instruction set manual, volume I: user-level ISA, version 2.0," Tech. Rep. UCB/EECS-2014-54, EECS Department, University of California, Berkeley, May 2014.

[14] T. Sarala, Shivashankar, R. Gatti and S. S. Palle, "Isolation and Dispensing of DC-to-DC Power Supply to High Power Electrical Application Circuits," 2021 International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT), Bangalore, India, 2021,pp.970-974,doi:10.1109/RTEICT52294.2021.9574011.