Research Article

A Smart Memory Controller for System on Chip-Based Devices

Mohammed Altaf Ahmed and Jaber Aloufi

Department of Computer Engineering, College of Computer Engineering & Sciences, Prince Sattam Bin Abdulaziz University, Alkharyj-11942, Saudi Arabia

Correspondence should be addressed to Mohammed Altaf Ahmed; m.altaf@psau.edu.sa

Received 6 February 2022; Accepted 22 April 2022; Published 5 May 2022

Academic Editor: Runwei Mo

Copyright © 2022 Mohammed Altaf Ahmed and Jaber Aloufi. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

This research paper deals with a system-on-chip (SoC) architecture design where multiple processors are inbuilt with other blocks of memory and control logic developed by nanomaterials. The multiprocessing-based SoC architecture is commonly used in the latest electronic devices such as smartphones, tablets, and smart wristwatches with large memory sizes. The data handing in these highly memory-dense devices is a critical task, and it needs special attention for the smooth operation of the device. This research proposed a smart controller to exchange data between various processors and input-output devices to tackle this challenge. A proposed controller block controls the data flow between memory and different SoC components and processors. A memory access controller (MAC) is presented in this research study to manage and accelerate data transmission speed and reduce the processors’ activity for SoC-based devices. The proposed MAC will integrate into the SoC with multiprocessing units, including gaming processors, at minimum hardware overhead and low power consumption. It improves the memory accessing efficiency and reduces the processors’ activity of a system. As a result, the system’s performance and power consumption improve at an acceptable level compared with the other conventional methods. This research is aimed at enhancing the performance of any SoC-based device where multiprocessing engines are inbuilt and flexible enough to serve various SoCs.

1. Introduction

In response to the rising demands of IoT devices, the 4G mobile phone devices cannot provide complete imminent services, which is required for the next-generation IoT-based system and other SoC-based systems [1, 2]. There is ample research conducted in this context of next-generation 5G mobile services [3, 4].

The traffic growth exponentially increases mobile phone networks from the generation of the cellular system 1G to 5G [5]. As per the survey of the CISCO Corporation, the data traffic on the mobile web becomes doubled from 2010 to 2011. They also stated that mobile traffic would increase by 1000x and above by 2021 compared to 2010 [6]. Due to this increased level of mobile traffic, the energy consumption of the mobile handset user increases with immense power. Therefore, it directly affects the expenditure. It is necessary to rethink the design to control these attributes and manage the network to reduce spending in a next-generation mobile phone system called the 5G generation. There is a future prediction of a fully connected society where everybody and everything is interconnected, on a common platform of the Internet of Things (IoT)—where many devices serve a single person. Therefore, the forthcoming 5G cellular infrastructure for massive data support will enable smart cities. There will be a need for an intelligent SoC design to support these 5G features in next-generation mobile phone devices and other SoC-based devices. The SoC of these devices should compromise with a high-end processing unit with an intelligent memory controller inbuilt feature.

In this context, a low-power, high-performance SoC development approach is accessible [7] that deals with the power management in the SoC for IoT-based devices. This SoC approach was developed on 65 nm technology and occupied an area of $1.0 \times 1.7 \text{ mm}^2$. Similarly, a method dealt with an adaptive power management (APM) design module that extensively supports learning techniques [8] and reduces power consumption in the IoT-SoC.
We are aware that the SoC is a single-chip integrated system and each block of it remains the intellectual property (IPs). The IPs are part of SoC, are available as processor core (one or more) peripheral devices, and are sometimes called macros. They are incorporated as per the requisition or demand of the SoC design. The SoC of these intelligent devices consists of some hardware and some software IPs. The hardware IPs are the processor, random access memory (RAM), and universal serial bus (USB) devices. The software IPs are the device drivers, the processor running algorithms, etc. IPs communicate together via specific channels called the on-chip bus, and it links all the IPs that belong to it [9]. This SoC bus works on a precise communication protocol containing various bus strategies. For instance, the IP begins communication to any other IP only when the bus is idle (not accessible by any further IPs). In this case, the IP with control over the bus is called the master, and other IPs responding to it are called slaves [10].

In case the master is accessing the SoC bus to convey their messages to other IPs at a particular instance, known as the active master. This SoC bus will remain idle during the absence of the active master. In recent times, multiprocessor systems have been drastically used to improve power performance compared to single processor systems in desktop segments. It caused the extensive application of multiple processor SoC cores, known as multiprocessor SoCs (MPSoCs) [11, 12]. These MPSoC cores are commonly available in the latest smartphone and other gaming devices. Therefore, some processors need to be offloaded to handle essential tasks and deal with such a high configurable multiprocessor-based SoC.

This paper is motivated by the future requirements of the SoC architecture of multiprocessor-based smart devices to fulfill consumers’ demands. We cannot ignore the snapdragon SoC as it explicitly concerns multiprocessor-based devices [13]. The majority of the market for android-based devices uses Snapdragon SoC in their devices. The single snapdragon SoC contains many processing units based on the ARM instruction set. In this research study, we proposed a MAC unit to perform tasks in parallel with the processor to enhance the speed and performance of the processing engines. Memories generally occupy the majority area of SoC, and it motivates this research to choose the MAC unit as a research topic to improve the processor performance. SoC architectures of the majority of digital systems mainly consist of nanomaterial-based memories. For example, the SoC of mobile phone devices, smart wristwatches, and mobile tablets comprises large memory sizes. The International Technological Roadmap for Semiconductor (ITRS) report is available for reference [14, 33] to convey that the memories occupy most of the SoC area. The MAC unit is small in hardware and occupies a tiny space overhead when integrating with SoC design when manufacturing the nanotechnological chip. Compared with the processor area (here, we consider a Snapdragon SoC), it holds a very thin area overhead of 0.0003% with low power consumption and a fast data transfer rate. Thus, the area overhead is negligible compared to a SoC area.

![Figure 1: SoC architecture with multiprocessing units.](image1)

![Figure 2: The Snapdragon multiprocessing is SoC.](image2)

![Figure 3: MAC data exchange principle.](image3)
The authors’ contributions to the article is noted as follows:

(i) Proposing a novel MAC unit architecture to perform memory accessing tasks while integrated with SoC at a fast transfer rate

(ii) Evaluate area overhead and power consumption and analyze the processor’s performance with and without integrating the MAC unit into the SoC of the smart device

(iii) Reduces the processor’s activity, and hence, it saves system power and temperature by integrating MAC unit into SoC

The paper is organized as follows: Section 2 outlines the SoC architecture of the mobile phone for previously available devices in terms of the services provided. Section 3 proposes the MAC architecture for mobile SoC. Simulation and ASIC Synthesis results are presented in Section 4. In this section, we have compared the area overhead of SoC by taking the example of Qualcomm’s processor Snapdragon SoC. Finally, we conclude the paper in Section 5.

### 2. SoC Architecture with Multiprocessing Units

A generic SoC architecture with multiprocessor units with various IP blocks is depicted in Figure 1. This SoC architecture uses the ARM Cortex-A15 multicore processor with advanced microcontroller bus architecture (AMBA) [15]. There are two different processing cores included in this SoC, namely, ARM Cortex A15, also known as a general purpose multiprocessor (GPMCP), and digital signal processing (DSP) processor [12]. These GPMCP and DSP processors are coupled with the memory controller core, enabling them to collaborate with other external flash memory, and system memory with the help of a high-performance AMBA advanced high-performance bus (AHB). In addition to these IPs, there is additional peripheral IP in the system, which supports UTMS (universal mobile telecommunications system) and GSM (global system for mobile communication). The external IPs are connected with the system through the high-performance bus known as AMBA advanced peripheral bus (APB). Since every mobile phone SoC consists of these two high-performance buses, AMBA-AHB and AMBA-APB [16], the system will work [17]. These two buses are responsible for data exchange between peripherals and processors. A bridge is used as an interface between the AMBA-AHB bus and the AMBA-APB bus to share information. A system architecture that uses AXI bus architecture is also available in recent research [18] for the FPGA-based dynamic partial reconfigurable security system for low area and power.

In SoC technology, trivial functions are conceived through the reusability of IPs. The reusability of IPs results in the rapidity of constructing a computer system based on a single-chip application. It is the root cause of the ever-growing fame of SoCs [19]. The constructional block diagram of a SoC is built up by reusing different modules. These reusable prevailing modules comprise an ARM processor core, peripherals modules like GSM and UMTS, memory controllers, etc. The precompiled libraries for the system-
Table 1: Signal description of memory controller for mobile SoC.

<table>
<thead>
<tr>
<th>Pin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rd</td>
<td>This signal enables the read operation when MAC wants to read data from FIFO.</td>
</tr>
<tr>
<td>rd_data</td>
<td>The data bus, which holds data, is read from FIFO to write into the memory through the state machine.</td>
</tr>
<tr>
<td>full</td>
<td>This input indicates to the MAC state machine about FIFO-full status.</td>
</tr>
<tr>
<td>empty</td>
<td>This input indicates that FIFO is empty to the MAC state machine, with no further data available to read.</td>
</tr>
<tr>
<td>ptr</td>
<td>There are two FIFO pointers to indicate reading and writing location and decide between empty and full status. wr</td>
</tr>
<tr>
<td>addr</td>
<td>Addr signal indicates the exact location of memory to write/read data from or into it.</td>
</tr>
<tr>
<td>Data</td>
<td>It is a data bus that carries data to write into and read from memory.</td>
</tr>
<tr>
<td>rd</td>
<td>Read enable to indicate read operations from memory.</td>
</tr>
<tr>
<td>fifo_wr</td>
<td>Enable signal from MAC state machine to write into the FIFO.</td>
</tr>
<tr>
<td>fifo-rd</td>
<td>Request from the peripheral to read data from FIFO.</td>
</tr>
<tr>
<td>Data-rd</td>
<td>Data bus carrying the data read from FIFO for any external device.</td>
</tr>
</tbody>
</table>

level programmer are also available in the form of software drivers. The multiprocessor SoC system is flexible to extend by adding new IPs.

The IP configuration in SoCs always describes the formal models. The multiprocessor SoC incorporates many IPs like different peripherals, processors, and memory cores. These IPs perform various functions at various speeds and work at different data exchange behaviors. Due to their unusual behaviors, their interconnections have to be comprehensive enough to handle the individual IP smoothly. Also, they must provide a standard approach to connect with all such IPs.

Usually, in the typical SoC architecture, all the IPs are connected with a common bus called a SoC bus. This SoC bus is linked with all the IPs of that particular SoC. Therefore, it has to set up a single platform for every IP connected to it, and the design is consequently called the platform-based design. This platform builds by integrating generic hardware (buses, processors, memory, etc.) and software (microcontroller code), and it can be remodeled to obtain differentiated products. Presently the typical SoC buses include Altera Avalon [20], IBM Core Connect, IDT IP Bus [21], Open Core Protocol [22], and Wishbone. This research work analyzes the AMBA [15] in detail, as it is the industry standard for creating SoCs. For example, AMBA is being used by many corporations like NVidia, Qualcomm, and Actel [23, 24] in their products.

The standard AMBA is rendered by ARM, which elicits as a generalized set of useful buses and is used in system-based integrated circuits (ICs). These system-based ICs are known as SoCs, traditional microcontrollers, and application-specific integrated circuits (ASICs). In 1996, the first time the ARM-based processors used this standard (AMBA) in their SoCs. Internally, AMBA consists of two types of buses: high-performance system buses to connect core IPs and a flexible peripheral bus to join many input outputs devices and the components. The AMBA-AHB connects various IPs, for example, a processor, a DSP, and advanced memory controllers as shown in the SoC architecture.

Similarly, the AMBA-APB (advanced peripheral bus) connects various peripheral IPs dedicated to that system. Additionally, AMBA provides the flexibility of upgrading the SoC. The SoC upgradation is possible by replacing some IPs with advanced versions (like the Bluetooth IP module) and adding IPs to enhance the functionality of the existing design (for example, adding an FM receiver, etc.). There is a bridge between the AHB and APB buses which is also shown in the multiprocessor SoC. This bridge is a standard bus-to-bus interface that provides communication between the IPs connected to different buses in a standardized way.

In this research, we are focused on specialized processing and integration. We know that the building of multiprocessor SoC requires many particular processing engines, which are custom designed to provide the user with the lowest power and higher performance [25]. A notable example is the Qualcomm Snapdragon (Snapdragon) [26, 27]. The SoC of Snapdragon processing is shown in Figure 2. The SoC consists of typical processing engines that integrate into a single piece of silicon [28]. There is a trend towards integrating more and more onto the SoC. The integration capabilities depend on the silicon traders’ IP technology and design abilities.

The benefits of integration result includes being cost-effective, low area, lower power, and low temperature. The SoC architect will make the appropriate design decisions and trade-offs among the many dependencies throughout the system. This insight gives the right system view and the ability to propagate features, accurately simulate use cases, and quickly respond to market demands. This research takes a step to enhance the system performance and speed of the devices using multiprocessing units to accept the consumers’ challenges and needs. The proposed MAC unit is an independent entity; it can easily integrate with the SoC with a reasonable transfer rate to get these features at low power consumption and a lite area overhead.

3. Implementation of Proposed MAC for SoC

The processor does intelligent operations like arithmetic and managerial processes. The real-time mobile systems incorporated today are built up with multiprocessors. These processors performed various functions by interacting with any
other processor in the same SoC. By doing this on one side, adding many processors to share the load on the other side. The processors waste their time by transferring data between themselves. Therefore, this research introduces the MAC to avoid this labor work of data transmission performed by any specialized processing engine in SoC-based devices. The MAC is a separate entity, and it integrates into SoC to perform such tasks. The research presents a real-time embedded system with fast data accessing support. It is known for its processor multitask support to achieve better yield and less power consumption while supporting future expansions. The system is called intelligent when it integrates the memory access controller for accessing the data between IP devices and memory, including cache. Hence, it can increase processing speed and reduce processing activity to save power. The active processor, among other processors in a specific SoC, excessively uses the memory while in use. Therefore, the MAC unit plays a significant role while the processor offloads memory accessing tasks in any specific SoC-based device.
Today, high-end smart processors are used in sophisticated real-time systems such as automotive, avionics, and some medical equipped devices where this MAC unit can be used for integration. The use of MAC in the systems provides CPU usage and high system throughput. Before the MAC application was narrow to computers, servers, etc., but now it has become an integrated system entity in a new generation system. The MAC unit can transfer data between the entities as follows:

(i) Between two memories
(ii) Between memory and IPs
(iii) Between two IPs
(iv) Between two processing systems

The MAC handles all data transfer tasks, freeing the processor to manage other essential functions for any selected embedded systems [28].

How data exchange between peripherals (together with memory blocks) and processors by using these buses in SoC as indicated in Figure 3. There are two masters over the bus, the CPU and MAC. Once the data is accessed from memory by either of these two masters, the appropriate memory address information is updated. It works in two modes. The first mode CPU is the master, and the second mode MAC will be the master over the bus. The CPU sends the request to the bus for access in the first mode. A bus will grant access to the CPU in response to this request. The CPU will operate on data accessing by updating data into an appropriate memory location after getting a permit on a bus. On another side, when MAC is master over a bus, it can access any peripheral and memory connected to the bus in the second mode.

3.1. Case when MAC in SoC. There is an example of a case when MAC is in the SoC.

Example 1. For example, when MAC transfers the data between UART and memory. It reads the UART registers and appropriate memory locations through the bus and released the bus grant once the operation finished. If the processor alone performs this operation, it requires more instruction cycles. This way, it needed more time to transfer the data between a memory and an external peripheral connected. Instead, the MAC engine handles this transfer operation by offloading the processor for sharing of data. An arbitration mechanism is used to grant access between the processor and MAC. The data transfer procedure is represented in Figure 4 by the MAC unit between peripherals and memory, data copied from memory and transferred to the USB by MAC itself. The data length of 15 bytes is considered.

Benefits of being MAC in multiprocessor-based devices:

(i) The power consumption reduces to a great extent. If the CPU runs for longer, it consumes more power, which the presence of the MAC will overcome by offloading the CPU
(ii) MAC is an individual entity, and it performs data transfer operations individually in parallel to CPU operation. Thereby, it may be used in the simulation of a multiprocessing environment and helps in improving the processor’s bandwidth.

(iii) It makes the CPU idle for more time and frees up the processor to perform a more advanced task in future product improvement.

3.2. Proposed MAC Architecture. The proposed memory access controller for multiprocessor-based SoC is shown in Figure 5. The MAC engine with first-in-first-out (FIFO) handles the extra load of the multiprocessors units to free the individual processor to perform another essential task in parallel. This offloading of the processing units in smartphone devices takes place by MAC’s proposed design method. The architecture consists of MAC in the writing mode and MAC in the reading mode, and a FIFO block on both sides is used for synchronization. In the first part, the MAC performs reading operations for any IP (peripheral) from FIFO. Similarly, the MAC performed writing operations into the FIFO for any external IP (peripheral) in the second part.

Once the writing or reading operation is complete, the data will be available for any external device or IP to read or write from FIFO. The external device then accesses the data from FIFO.

Two memory controllers are implemented in Verilog hardware description language for accessing the data. The buffer memories (FIFO) on both sides are used to store the data temporarily to avoid data loss and mismatches. Hence, it synchronizes the speed of the peripheral and controller. The MAC signals and the description are summarized in Table 1. The external IP will write the data into the FIFO in writing mode. This data is then read by MAC and stored in the memory. The status of available data in FIFO will be updated by the pointer signal ‘ptr’. Therefore, the MAC engine will read the appropriate data from FIFO and write it into the memory. Thus, the memory contains the data by writing any external IP into it.

Table 3: ASIC area report for proposed MAC.

<table>
<thead>
<tr>
<th>Design</th>
<th>Cell</th>
<th>Cell area (nm²)</th>
<th>Net area (nm²)</th>
<th>Total area (nm²)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed MAC</td>
<td>2201</td>
<td>31272</td>
<td>16220</td>
<td>67497</td>
</tr>
</tbody>
</table>

Table 4: Area comparison between the proposed MAC and the other existing approaches.

<table>
<thead>
<tr>
<th>Design</th>
<th>Cell</th>
<th>Cell area (nm²)</th>
<th>Net area (nm²)</th>
<th>Total area (nm²)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed MAC</td>
<td>2201</td>
<td>31272</td>
<td>16220</td>
<td>67497</td>
</tr>
<tr>
<td>DMA [30]</td>
<td>3901</td>
<td>56179</td>
<td>21237</td>
<td>99417</td>
</tr>
<tr>
<td>DMA [31]</td>
<td>3637</td>
<td>48312</td>
<td>25177</td>
<td>93289</td>
</tr>
<tr>
<td>DMA [32]</td>
<td>4637</td>
<td>62749</td>
<td>30256</td>
<td>101389</td>
</tr>
<tr>
<td>DMA [33]</td>
<td>4213</td>
<td>56274</td>
<td>28096</td>
<td>10017</td>
</tr>
</tbody>
</table>

Writing of memory is carried out as follows:

(i) The experiment results are taken by considering 8 K of memory

(ii) Memory size (Mem-array) = 8 K

(iii) Each location of memory contains 64-bit data = 8 bytes

(iv) Total location required for 8 K data (max-add) = 210

(v) Address location will vary from zero ‘0’ to the $2^{10}$

The data is written into the memory for all 1000 address locations, and each address location contains 8 k of data. If 64-bit is the data line, it requires $2^{10}$ address locations. Max-add is indicating as $2^{10}$. The writing operation can be performed and expressed in equation (1).

\[ \text{MEMORY}\text{[Add]} \leq \text{Data}\_\text{wr}. \]
The data is already available in the memory on the other side of the MAC reading operation. The MAC engine will read it from memory and write it into the FIFO for any connected external IP. Once the data is available in FIFO, the external IP will come and read it. The MAC reads the status of FIFO by pointer ‘ptr’ signal. The reading operation can be performed and expressed in equation (2).

\[
\text{Data}_{\text{rd}} = \text{Memory} [\text{Address}]. \tag{2}
\]

The IP request for accessing the memory decides that MAC has to work in writing or reading mode. If the request is for writing to the memory, the MAC first checks for the data request is for how many bytes (length), and then it grants writing into the FIFO to the requested IP. The MAC continuously checks the full-empty status of FIFO and depends on the availability of data in FIFO. Whereas if the request is to read the data from memory, the MAC first checks the IP request, calculates the bytes, and allows the IP to read the data from FIFO. The data in the FIFO is available by the MAC after reading it from memory. The flow of reading and writing data for any requested IP is shown in Figure 6. Multiple requests and reading writing operations can perform in parallel by the proposed MAC unit in any SoC-based system.

### 4. Results

Verilog HDL is used to design the MAC unit. The simulation and synthesis are performed on the EDA tool. The top-level Verilog module includes two-part transmitting (writing) and receiving (reading) parts, and each part uses a MAC unit with a FIFO block. The simulation is performed on the Xilinx simulator. A case where an external device is writing data into the memory is shown in Figure 7. In this case, the external IP wants to write into the memory by sending a write request. It writes data into the FIFO on the approval of its request by a grant signal. Once the data is available in FIFO, the MAC will read it and write it into the memory. The MAC unit continuously reads the FIFO pointer to act during this process. On the other side, when external IP wants to read data from memory, the MAC unit will read data from memory and write it into the FIFO for any requested external IP. The read-write request handled by the MAC unit itself to offload the processor is shown in the graph in Figure 8 and Table 2.

FIFO pointer in each operation increments and appropriate information about data availability is communicated with the MAC. Depending on the pointer location, MAC will perform operations. We consider 64 as the FIFO depth for both the reading and writing sides. In this research, we implement a controller that can allow multiple IPs to access the memory without any involvement of the processors. As a result, processor performance and speed will increase drastically, and it can only concentrate on its operations other than memory access.

The developed MAC unit can integrate into mobile phone SoC to perform enhanced features data accessing memory to offload the processor. Hence, it increases the processing speed and power optimization of any processor. It can merely be understood by considering an example of wave generation. In a wave generation process for SoC, the MAC will perform a data exchange operation instead of a

![Figure 10: Area comparison graph.](image)

![Table 5: Data transfer rate for the proposed MAC and other existing methods.](table)

<table>
<thead>
<tr>
<th>Design</th>
<th>Max frequency (MHz)</th>
<th>Mega cycles per sec</th>
<th>Transfer rate (Mbytes/sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed DMA</td>
<td>350</td>
<td>350</td>
<td>1400</td>
</tr>
<tr>
<td>DMA [30]</td>
<td>320</td>
<td>320</td>
<td>1200</td>
</tr>
<tr>
<td>DMA [31]</td>
<td>320</td>
<td>320</td>
<td>1200</td>
</tr>
<tr>
<td>MAC [32]</td>
<td>250</td>
<td>250</td>
<td>1000</td>
</tr>
<tr>
<td>DMA [33]</td>
<td>250</td>
<td>250</td>
<td>1000</td>
</tr>
</tbody>
</table>
As a result, processor utilization will reduce drastically, directly affecting processor speed and power consumption. The purpose of a wave generator is to generate waves continuously [29]. The continuous stream will create without interruption by continually accessing the data from the look-up-table (LUT) to the digital-to-analog converter (DAC) data register. If there is a MAC unit available in the SoC architecture, the updating of data from LUT to the DAC register will take place by the MAC unit, and hence processor activity reduces significantly.

Large processor bandwidth and power are required when the wave’s frequency and the number of samples increase. In case of the absence of the MAC unit in mobile device SoC, it forces the processor to perform this hectic task of data updating into the DAC register. Therefore, in this way, the processor consumes many instruction cycles to copy the data from LUT to the DAC register, affecting its performance significantly.

Instead, the MAC unit can easily do this task of data accessing from memory to the DAC register at the required time interval. Additionally, the MAC is not interrupt-driven like a processor. In this case, the processor does not need to do anything as long as the wave generates. As a result, a large amount of power will save because, generally, the processor consumes more energy than the MAC unit. The processor acts in power-down mode if it is not in use for data transfer operation, and hence, there will be a reasonable saving of static and dynamic power consumed. Thus, the MAC helps optimize the system’s power consumption where a significant portion of work is to transfer data. The comparison graph between processor utilization with and without MAC is plotted and shown in Figure 9, and it is not scaled. We can see from the chart how speed and power performance affect any mobile phone SoC when the MAC unit is integrated.

The design of the MAC unit is performed on the ASIC platform. The ASIC synthesis results are received on the synopsis tool design compiler with 32 nm² technology. The MAC unit is implemented on ASIC to calculate the area overhead and to integrate with Qualcomm’s Snapdragon SoC. The design of MAC uses only 2201 cells with a cell area of 31272 nm², a net area of 16220 nm², and a total area of 67497 nm². The synthesis result for area calculation is tabulated in Table 3. The design frequency, the power consumption, and the data transfer rate of the MAC block are computed. The existing approaches of the MAC are also implemented, and the obtained results are compared with the proposed MAC unit. The area comparison is tabulated in Table 4 and indicated in the graph in Figure 10.

The design frequency and the data transfer rate of the proposed MAC unit and the other available conventional methods of memory access controller are tabulated in Table 5. The comparison graph is shown in Figure 11.

The power consumption comparison for the proposed and other approaches are listed in Table 6 and the graph in Figure 12.

The commercially available SoC having the transistor count, including only the processing units, is 32 billion MOSFETs by 2019. For example, Qualcomm’s Snapdragon processor gate count is more than 3 billion transistors [34, 35]. If we consider one cell having four gates or 16 transistors (4-transistors per logic gate), three billion transistors have 187 million cells.

The proposed design in this research used only 2201 cells with 32 nm² technology. Therefore, integrating this controller with the Snapdragon SoC results in negligible area overhead with enhanced power optimization and speed features. The processor activity reduces to a great extent, and hence, it optimizes the system’s power consumption. The result of the

![Figure 11: Data transfer rate comparison.](image-url)
A memory access controller for the multiprocessor-based SoC devices is proposed in the presented research. The architecture of a memory controller is proposed to fulfill the consumer’s demands for IoT devices. The SoC consists of a multiprocessing unit, and MAC integration decreases the processing activity and power consumption by offloading processors from the memory-accessing task. As a result, it increases the speed of the processors to perform more essential tasks. The processor performance comparison is noticed with and without integrating a memory controller into the SoC while developing this nanomaterial-based product. It has been seen that the memory controller’s presence will drastically affect processing speed and the power consumption of the processing units. The MAC unit is minimal, and it takes a low area overhead at low power consumption and fast data rate transfer. The hardware overhead is only a limitation. The MAC unit is a separate entity, and therefore, it may use for any future enhancement of SoC design with a lite area overhead.

5. Conclusion

The authors declare no conflicts of interest to report regarding the present study.

Acknowledgments

The authors extend their appreciation to the Deputyship for Research & Innovation, Ministry of Education in Saudi Arabia for funding this research work through the project number (IF-PSAU-2021/01/18656).

References


Conflicts of Interest

The authors extend their appreciation to the Deputyship for Research & Innovation, Ministry of Education in Saudi Arabia for funding this research work through the project number (IF-PSAU-2021/01/18656).

Data Availability

The simulation screen results and excel file graphs data used to support the findings of this study are included within the article.


