ISTANBUL TECHNICAL UNIVERSITY

ELECTRICAL-ELECTRONICS FACULTY

MODEL BASED DESIGN AND IMPLEMENTATION OF SECURE IOT NETWORK USING SIMULINK

SENIOR DESIGN PROJECT

Heval Ronahi HALİTOĞLU
Oğuzhan TURAN

ELECTRONICS AND COMMUNICATION ENGINEERING

DEPARTMENT

JUNE 2021

ISTANBUL TECHNICAL UNIVERSITY

ELECTRICAL-ELECTRONICS FACULTY

MODEL BASED DESIGN AND IMPLEMENTATION OF SECURE IOT NETWORK USING SIMULINK

SENIOR DESIGN PROJECT

Heval Ronahi HALİTOĞLU
(040150114)
Oğuzhan TURAN
(040150094)

ELECTRONICS AND COMMUNICATION ENGINEERING

DEPARTMENT

Project Advisor: Prof. Dr. Sıddıka Berna ÖRS YALÇIN

JUNE 2021

We are submitting the Senior Design Project entitled as “MODEL BASED DESIGN AND IMPLEMENTATION OF SECURE IOT NETWORK USING SIMULINK”. The Senior Design Project Interim Report has been prepared as to fulfill the relevant regulations of the Electronics and Communication Engineering Department of Istanbul Technical University. We hereby confirm that we have realized all stages of the Senior Design Project Interim Report by ourselves, and we have abided by the ethical rules with respect to academic and professional integrity .

Heval Ronahi HALİTOĞLU ....
(040150114)

Oğuzhan TURAN ....
(040150094)

To our beloved family,

FOREWORD

Since the beginning of our undergraduate senior design project, we would like to offer endless thanks to our valuable professor, Sıddıka Berna ÖRS YALÇIN, for continuing her contact with us at every stage of the thesis and taking her time.Secondly, we would like to offer our gratitude to our mentor Res. Assist. Mehmet Onur DEMİRTÜRK.

We would like to thank our precious teachers and friends for their vision and quality awareness, besides engineering education at Istanbul Technical University, one of the oldest technical universities in the world, and our families who always supported us until we received this diploma.

June 2021 Heval Ronahi HALİTOĞLU

Oğuzhan TURAN

TABLE OF CONTENTS

ABBREVIATIONS

AES	: Advanced Encryption Standard
AWGN	: Additive White Gaussian Noise
AXI	: Advanced Extensible Interface
BPM	: Beats per Minute
DES	: Data Encryption Standard
ELF	: Executable and Linkable Format
FPGA	: Field Programmable Gate Array
HDL	: Hardware Description Language
IDE	: Integrated Development Environment
IEEE	: The Institute of Electrical and Electronics Engineers
IETF	: Internet Engineering Task Force
FPGA	: Field Programmable Gate Array
IoT	: Internet of Things
IP	: Intellectual Property
LUT	: Look-Up Table
LAN	: Local Area Network
MATLAB	: Matrix Labrotary
MDD	: Model Driven Development
OQPSK	: Offset Quadrature Phase-Shift Keying
PDF	: Probability Density Function
RC5	: Rivest Cipher 5
RISC	: Reduced Instruction Set Computer
RSA	: Rivest-Shamir-Adleman
RTL	: Register Transfer Level
UART	: Universal Asynchronous Receiver Transmitter
XOR	: Exclusive OR
XSA	: Xilinx Shell Archive
3DES	: Triple Data Encryption Standard

LIST OF TABLES

LIST OF FIGURES

MODEL BASED DESIGN AND IMPLEMENTATION OF SECURE IOT NETWORK USING SIMULINK

SUMMARY

Internet of Things (IoT) is a network of connected sensors, computers, and digital devices that can communicate with one another over the internet to share and transfer data. IoT has a wide variety of applications in the fields of manufacturing, transportation, energy, healthcare and smart building systems. According to the its application IoT involves extra devices being connected, extra networking to connect these devices, extra programming to direct the devices and networking and a massive volume of extra data pouring into the internet. Each of these layers come up with additional security issues. Consequently, IoT devices usually lack the security from outside impacts to prevent hacking. Cryptography algorithms are widely used in security issues. Even though use of traditional cryptosystems can improve security, another problem arises for those implemented cryptosystems which is increase the need of power for the processors thus makes the device less preferable. Cryptography algorithms are also extremely resource constrained devices in terms of computing capabilities, power and area usage.

Aim of this project is to create a model-based design and implementation of an IoT system that uses low power processors for speed and security. The hardware-software co-design process was utilized to make it easier to design this sensor node and decrease engineering expenses. According to this research, IoT and its architecture were investigated. An IoT nodes are generally consist of sensors, actuators and processors communicate with each other to serve its application. IoT node in this project is created for health application in health emergency service. Some data such as human temperature and heart rate is taken from human body for analyzing and reacting according to its data values.

For model-based design of IoT network, it has made some researches. After project's application is determined, Simulink is used for modeling an IoT node. IoT nodes can communicate with usage of standard communication protocols. Zigbee communication protocol is selected for its features such as its frequency, data rate and range for this project. Main objective for this part was to find some applied communication module for an example. Zigbee module had founded on Simulink File Exchange and it is selected which provides two main operations: modulation and demodulation. After communication protocol chosen, ideas for application were executed. First, vital health parameters will be taken from sensors. Then, the sensor data will be transmitted via Zigbee communication protocol and then received data will be used to actuate some alarms for warning health care providers.

After this step, it has researched cryptographic algorithm for securing modeled IoT node. Rivest Cipher 5 (RC5) cryptography algorithm was chosen according project's application; since, RC5 algorithm is fast and adaptable for different lengths of bits.

Also, it can be easily manipulated to make it faster or more secure. For length of word is selected as 32 bits for this application. Encryption and decryption algorithms of RC5 are written as Matrix Laboratory (MATLAB) code and implemented into Simulink blocks. The whole model is investigated and it is decided that the system works properly. After modeling step, Embedded Coder in Simulink is utilized for generating C code to transfer the system to software platform. Introduced code are transferred into Xilinx Vitis platform. In Vitis, C code is written observe the system in MicroBlaze which is Xilinx's virtual processor. According to project's IoT node, it has produced an Intellectual Property (IP) structure for embed to a module. Generated hardware module has exported into Xilinx Vivado Design Suite and system is observed in simulation. As a result, monitorable healthcare system is created as Simulink model and verified with MicroBlaze.

SIMULINK KULLANARAK GÜVENLİ IOT DÜĞÜMÜNÜN MODELLEME TABANLI TASARIM VE UYGUNLANMASI

ÖZET

Nesnelerin İnterneti (IoT), verileri paylaşmak ve aktarmak için internet üzerinden birbirleriyle iletişim kurabilen sensörler, bilgisayarlar ve dijital cihazlardan oluşan bir ağdır. IoT, üretim, ulaşım, enerji, sağlık ve akıllı bina sistemleri alanlarında çeşitli uygulamalara sahiptir. Bir IoT sistemi kullanılan cihazların birbiri ile bağlanmasını, bu cihazları birbirine bağlamak için gerekli ağ yapılarını, bu ağları yönlendirmek için programlamayı ve bunların doğurduğu büyük miktarda veriyi içerir.Bu sistemin oluşturduğu her bir katman ek güvenlik sorunlarıyla karşılaşır. Sonuç olarak, IoT cihazları genellikle dışarıdan gelebilecek herhangi bir saldırıya açık olup bu etkileri azaltabilecek güvenlikten yoksundur. Kriptografi algoritmaları bu tarz güvenlik konularında yaygın olarak kullanılmaktadır. Geleneksel şifreleme sistemlerinin kullanılması ise güvenliği artırabilse de, uygulanan bu şifreleme sistemleri, işlemciler için güç ihtiyacını artıran ve böylece cihazı daha az tercih edilir hale getiren başka bir sorun ortaya çıkartır. Ayrıca kriptografi algoritmaları kullanan cihazlar bilgi işlem yetenekleri, güç ve alan kullanımı açısından son derece kaynak kısıtlı cihazlardır.

Bu projenin amacı, hız ve güvenlik için düşük güçlü işlemciler kullanan bir IoT sisteminin model tabanlı bir tasarımını ve uygulamasını oluşturmaktır. Bir IoT sensör düğümünü tasarlamayı kolaylaştırmak ve mühendislik maliyetlerini azaltmak için donanım-yazılım ortak tasarımı bir arada kullanılmıştır. Bu araştırma için IoT ve mimarisi araştırıldı ve bir IoT sistemi genellikle, uygulamasına hizmet etmek için birbirleriyle iletişim kuran sensörler, aktüatörler ve işlemcilerden oluşur.Bu projedeki IoT sistemi, sağlık acil servisindeki sağlık uygulamaları için oluşturulmuştur. İnsan vücudundan vücut sıcaklığı ve kalp atış hızı gibi bazı veriler alınarak, bu değerleri analiz etmek ve değerlendirmek için kullanılır.

IoT ağının model tabanlı tasarımı için bazı araştırmalar yapılmıştır. Bir IoT düğümünün modellenmesi için Simulink kullanıldı. Oluşturulan IoT düğümleri, standart iletişim protokollerinin yardımı iletişim kurabilir. Bu proje için frekansı, veri hızı ve aralığı gibi özellikleri nedeniyle Zigbee iletişim protokolü seçilmiştir. Bu bölümün ana amacı olarak örnek amaçlı bazı modellenmiş iletişim modülleri bulundu. Araştırmalar sonucu bulunan Zigbee modülü Simulink File Exchane üzerinde kurulmuş olup kendi model yapısı içerisinde modülasyon ve demodülasyon işlemlerini gerçekleştiren iki ana modulden oluşmaktadır.letişim protokolü seçildikten sonra uygulama fikirleri hayata geçirilmeye başlandı. İlk olarak, sistem yapısı içerisindeki sensörlerden hayati sağlık parametreleri alınacaktır. Ardından ise sensör verileri Zigbee iletişim protokolü aracılığıyla iletilecek ve alınan veriler, sağlık çalışanlarını uyarmak amacıyla bazı alarmları etkinleştirmek için kullanılacaktır.

Bu adımdan sonra modellenen IoT düğümünün güvenliğini sağlamak için kriptografik algoritmalar araştırılmıştır. RC5 kripto algoritması hızlı olması ve farklı bit uzunlukları

için ayarlanabilmesi özelliklerinden dolayı bu projeye uygulamak amacıyla tercih edilmiştir.RC5, oluşturulan IoT düğümünü daha hızlı veya daha güvenli hale getirmek için kolayca veriyi manipüle edilebilir. Bu uygulamada word uzunluğu 32 bit olarak seçilmiştir. RC5'in şifreleme ve şifre çözme algoritmaları MATLAB kodu kullanarak yazıldı ve ardından Simulink bloklarına uygulandı. Tüm model incelenrek sistemin çalışması incelendi. Modelleme adımından sonra, sistemi yazılım platformuna aktarmak için Simulink'teki Embedded Coder aracı kullanılmıştır. Bu araç ile modüllerimize ait C kodları üretimi sağlanır. Bu üretilen kodlar daha sonrsında Xilinx Vitis platformuna aktarılır. Bu platform üzerinden bir donanım modulüne kendi yapısını gömmek amacıyla bir IP yapısı üretilir. Oluşturulan donanım modülü Xilinx Vivado Design Suite'e aktarılmıştır. Daha sonrasında bu yapının denenmesi amacıyla Vivado üzerindeki görsel bir mikroişlemci olan MicroBlaze kullanılması tercih edilmiştir. Sistemin doğru çalışıp çalışmadığını gözlemlemek için çeşitli durumlara bağlı olarak simülasyonlar gerçekleştirilmiştir. Sonuç olarak Simulink ile modellenen yapı ilk olarak software platformuna ardından da hardware platformuna aktarılarak MicroBlaze ile çalışması gözlemlenmiştir.

1. INTRODUCTION

Nowadays, Internet of Things (IoT) [6] has expanded to a global infrastructure with the number of connected devices being in multiples of the number of people worldwide. Large communication data and inter-connected gadgets also introduce many new cases in which breaches of privacy and information can occur. New generation Internet of Things (IoT) applications require low power, and small area secure systems that encrypt data.

The challenge that we wish to address, focusses on Model Driven Development (MDD) [7] of embedded system design software and hardware and implementing them on secure IoT applications. This project aims to create a modified version of the algorithm that is derived from the hardware with cooperation of hardware and software to create an improved version to ease implementation and accelerate manufacturability of the IoT device that the algorithm will be implemented for security to create a secure, low cost and rapidly produced IoT devices. It is used interoperation of MATLAB, Simulink, Vitis and Vivado for this project. Field Programmable Gate Array (FPGA) [8] is utilized to put the system on a trial and modify the algorithm on Xilinx Vivado program to create a hardware-software hybrid system. [9]

In the second part of the thesis, information about the architecture of the internet of things, application areas and security problems are given. In the third part, the plan of the system to be realized is made. In the fourth chapter, the software implementation of the system is explained, and in the fifth chapter, the hardware implementation. In the sixth and last chapter, the results and information about future studies are given.

2. INTERNET OF THINGS (IOT)

The term of Internet of Things was first proposed by a British technologist Kevin Ashton in 1999. IoT connects devices to the internet, allowing software to collect data and manage these devices to improve efficiency, create new services, or achieve its application goals [10].

IoT has been defined by different organizations and research centers over time. One of them is defined by Internet Engineering Task Force (IETF) as “the network of physical objects or "things" embedded with electronics, software, sensors, and connectivity to enable objects to exchange data with the manufacturer, operator and/or other connected devices” [11]. On the other hand, The Institute of Electrical and Electronics Engineers (IEEE) is described “Internet of Things” on its IEEE “Internet of Things” 2014 special report as “A network of items—each embedded with sensors—which are connected to the Internet” [12].

Nowadays, IoT become considerable technology that offers many solutions in daily basis. Data collected and analyzed via IoT systems can be utilized for optimization, efficiency and management based on its applications. To understand and maintain IoT systems, one must first understand their architectural layers.

2.1 Applications of IoT

2.1.1 Manufacturing

IoT is a real-time network of connecting sensors, computers, and digital devices that can communicate with one another over the internet to share and transfer data. Usage of IoT in manufacturing may help automation, efficiency, energy management, proactive maintenance, and connected supply chain management. It can collect data from the production environments, machinery, vehicles, and materials. Collecting these data without any human interruption can help for automation of workflow/processes and optimization of design and production systems [13]. On the other hand, increasing energy demand in manufacturing requires to be aware of energy usage. Tracking and associating data in real-time via IoT can supply more efficient usage of energy. It can help also continuously observing and error diagnosing production elements for decrease time and money that spend for maintenance. Demand, supply and feedback information can be taken for meaningful results for manufacturing management.

2.1.2 Transportation

As the world population continuously increasing, low-cost and efficient transportation systems in every field play an essential role in sustainable development. Data collected and interpreted from IoT system in transportation can used location management, driver safety, improved traffic management, reduced time and energy used for transportation and supply chain management [14]. IoT is possible to supply enhancement in cost and time management for more daily basis such as street lighting, optimized suggested routes, parking reservations, public transportation, driver and pedestrian safety, autonomous driving [15].

2.1.3 Smart Cities

IoT can help centralized, optimized and efficient use for smart cities based on continuous monitoring and associated data using systems, devices interoperability. IoT smart cities application can be divided some main aspects such as traffic management, environmental monitoring, home energy management, waste management, and etc. Proactive traffic management is enabled by measurements and predictions from a large number of sensor's data, which has the potential to significantly improve efficiency and minimize congestion [16]. It can supply information about Water Level Management, Soil Humidity, Gas concentrations, Structural Inclinations, Lighting Conditions, Infrared radiation [17]. Utilized / recycled waste can be determined and reported continuously for waste management. The monitoring system can easily be integrated via embedded sensors with a building management/automation system to operate in order to keep the determined parameter's value within predetermined security/health criteria for home energy management [18].

2.1.4 Healthcare

IoT applications have the potential to transform reactive medical systems into proactive wellness systems. Current medical research relies on resources that are deficient in critical real-world data. For medical examination, it usually uses residual data, controlled surroundings, and volunteers. Through analyzed, collected, and testing real-time data, IoT system provide efficiently used data [19]. Real-time applications can be exampled such as monitoring of temperature for vaccines remotely, sensors for air quality, sleep monitor, examination of drug effectiveness, capturing vital data, bio-metrics scanners for remote care, proactive scheduling for refilling medication, and etc [20].

2.1.5 Agriculture

IoT systems used in agriculture is a high-tech and efficient approach for sustainable agriculture and food production. It is a method of integrating connected devices and cutting-edge technologies into agriculture. The basis applications are mainly focused on climate conditions, precision farming, smart greenhouse, data analytic, and et [21]. These systems provide efficient and optimized production in each aspects of agriculture.

2.2 IoT Architecture

IoT architecture is generally described with five different layer in which data flows from sensors attached to “things” through a network and eventually on to a corporate data center or the cloud for processing, analysis and storage. All layers are can be seen in clearly in below Figure.

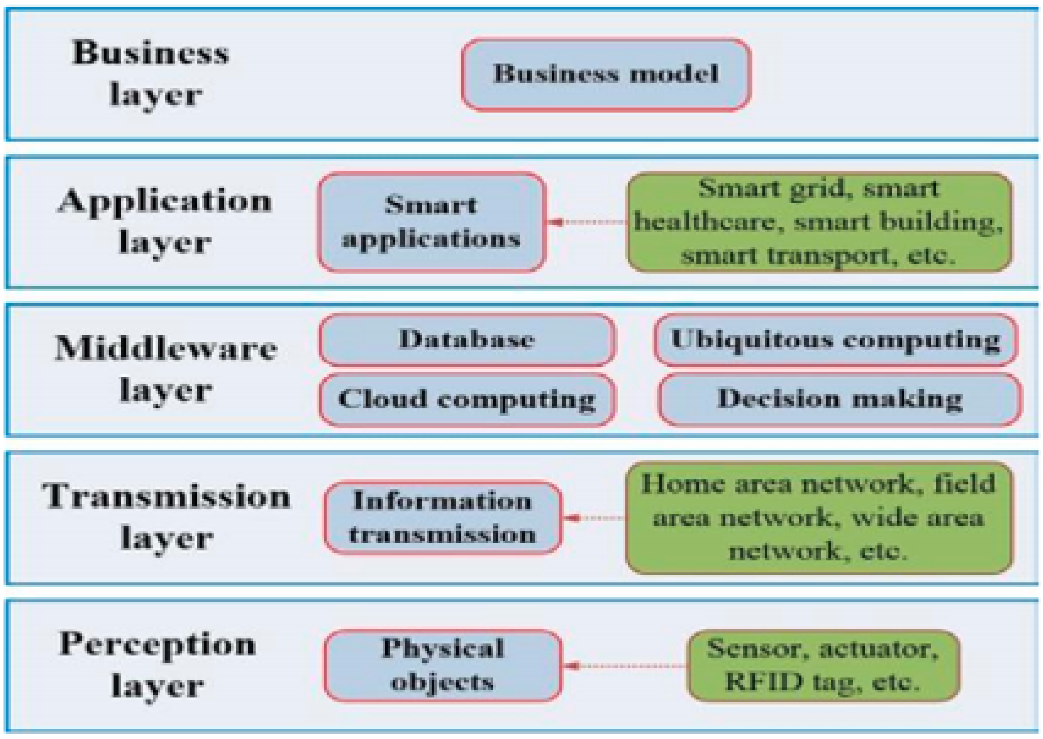


Figure 2.1: IoT Architecture [2]

- Perception Layer is a physical layer that basically consist of sensors. These devices take some physical information from environment; also, indicates other smart objects in environment.

- Transmission Layer is used for transferring the sensor data from perception layer to processing layer through a network which are wireless, 3G, Local Area Network (LAN), Bluetooth, Zigbee.

- Processing Layer is called also middle-ware later which is used for storage, analyzation, and processing huge amount of data which is taken from transport layer. Databases, cloud computing, and big data processing modules are used in this later.

- Application Layer provides application specific services are provided according to their purpose. It can be defined various applications according to used Internet of things, for example, smart houses, smart cities, and smart health.

- Business Layer is fulfilled for managing whole IoT systems, consist of its applications business and user privacy [22].

2.3 Security Issue of IoT

The purpose of IoT is about create appropriate connections between things. It has becoming widely usage for different area. Therefore, there is a large increase in the number of data used and collected. These collected data are generally personal and private. According to that, securing IoT devices and network is one of the essential issues in these systems. Security constraints can be divided into three main topics which are based on hardware, software and network. In these project, it focused on hardware part. Cryptology algorithms are utilized for securing hardware part of IoT systems. Different algorithms could be use such as Rivest-Shamir-Adleman (RSA), Data Encryption Standard (DES), Triple DES (3DES), Advanced Encryption Standard (AES), Twofish, Blowfish, RC2, and RC5. Each of these algorithms come with its drawbacks which are computational and energy constraints, memory constraints and tamper resistant packaging [23]. For this project, RC5 cryptology algorithm is used.

2.3.1 RC5 Cryptology Algorithm

2.3.1.1 Mathematical Background for RC5 Algorithm

In the last years, secure systems are more required with the development of the Internet and IoT. Therefore, cryptographic algorithms provide protection of the data. There are asymmetric and symmetric algorithms. For the symmetric ones such as DES, AES and RC5's security level depends on the key length [24]. RC5 is a simple cryptographic algorithm which designed by Ronald L. Rivest in 1994. While process, same key is used for encryption and decryption which makes the algorithm symmetric. It has many parameters such as word length, key length and number of rounds. Thanks to all parameters, RC5 can be exchanged easily faster to more secure or exact opposite. Moreover, there are only basic operations in algorithms which makes RC5 suitable for both hardware and software. It comprises of three parts: an encryption, a decryption and a key expansion [3]. Fig.2.2 shows the operations of RC5 algorithm. First operation is expanded to key K to S. Then, plain text which is the data to encrypt is connected to encryption algorithm with the key; so, cipher text is obtained. S is needed again since RC5 is symmetric algorithm. This time, cipher text is connected to decryption algorithm to obtain the plain text again.

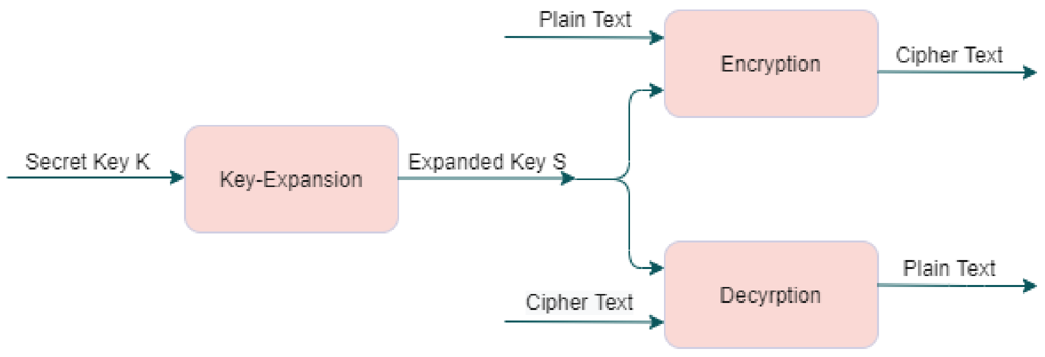


Figure 2.2: Representation of RC5 Algorithm

RC5 Keys and Parameters

w – word size in bits, usually 16,32 and 64.

u – w/8 which is word size in bytes.

r – number of rounds, 0 to 255.

b – number of bytes in secret key K, 0 to 255.

K – b-byte secret key.

A – low-order bit positions of input/output.

B – high-order bit positions of input/output.

S – key array which is generated from K.

L – temporary array for operations in key-expansion.

P – a magic constant which is used in key-expansion.

Q – a magic constant which is used in key-expansion.

e – 2.71828... which is base of the natural logarithm.

φ – 1.618… which is golden ratio.

c – b/u which is number of words in key.

t – 2(r+1) which is size of table S.

Key Expansion

The key expansion algorithm stands for obtaining S while filling K. It uses two constant values which is seen in Fig.2.3. Odd function in P and Q rounds up the result to the nearest odd integer.

$$P_w = \mathrm{Odd}((e-2)2^w)$$
$$Q_w = \mathrm{Odd}((\phi-1)2^w)$$

Figure 2.3: Magic Constants [3]

When the constants are ready to use, key K is needed to copied into L array which is K [0. . . b-1] into L [0. . . c-1] to convert key bytes to words. As a following operation, array S is initialized with the magic constants. Finally expanded key is mixed. A is low-order plain text and B is high-order plain text. As seen, modulus, addition, rotation operations are used which can be easily done with typical microprocessor. Fig.2.4 demonstrates three operations explained above.

```
for $i = b - 1$ downto 0 do
        $L[i/u] = (L[i/u] \lll 8) + K[i]$;
```

```
$S[0] = P_w$;
for $i = 1$ to $t - 1$ do
        $S[i] = S[i-1] + Q_w$;
```

```
$i = j = 0$;
$A = B = 0$;
do $3 * \max(t, c)$ times:
        $A = S[i] = (S[i] + A + B) \lll 3$;
        $B = L[j] = (L[j] + A + B) \lll (A + B)$;
        $i = (i + 1) \bmod(t)$;
        $j = (j + 1) \bmod(c)$;
```

Figure 2.4: Pseudo Code of Key Expansion [3]

Encryption

Input is given as two halves of 2w-bit register which equals A and B for this case. After S[0] and S[1] assigned to A and B, new operations are done according to number of rounds r and algorithm is repeated. Inside of the for structure, there are just exclusive or (XOR), rotational shift, and addition operations exist. Algorithm can be seen in Fig 2.5. Summary of encryption is seen in Fig.2.6 which represents one round. A is treated with B with the operation XOR, then shifted left. Then key is used for A, and the cipher text is ready for A. For B, same path is followed. At the end of the last loop cipher texts is obtained as a result of A and B.

```
$A = A + S[0];$
$B = B + S[1];$
for $i = 1$ to $r$ do
        $A = ((A \oplus B) \lll B) + S[2 * i];$
        $B = ((B \oplus A) \lll A) + S[2 * i + 1];$
```

Figure 2.5: Pseudo Code of Encryption [3]

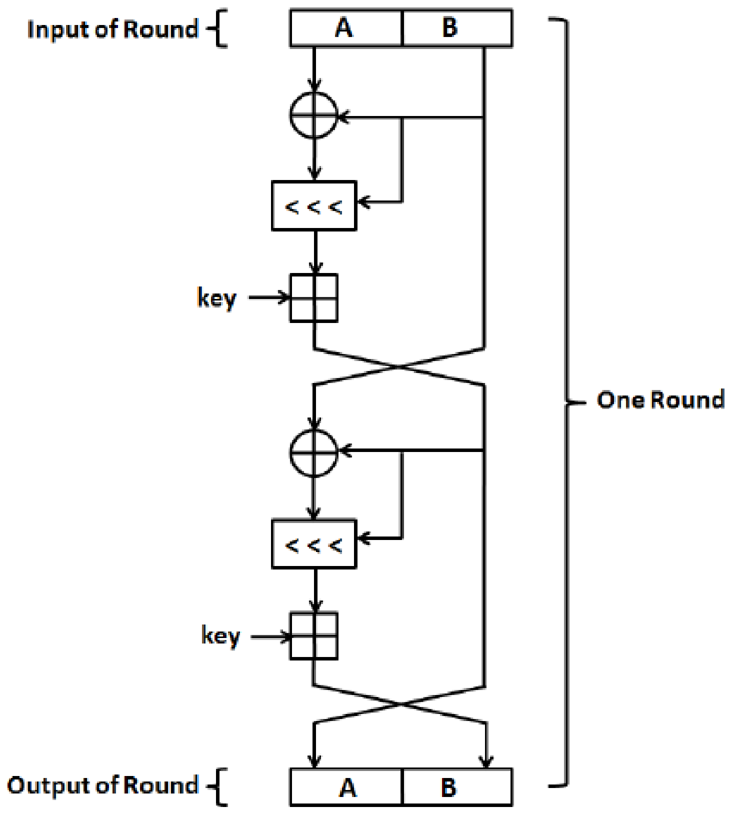


Figure 2.6: Block Diagram of Encryption for One Round [4]

Decryption

For the decryption algorithm, encryption steps are implemented back to front. Rotational operation is made backwards which means right. This time plain texts are achieved as results A and B.

```
for $i = r$ downto 1 do
        $B = ((B - S[2 * i + 1]) \ggg A) \oplus A;$
        $A = ((A - S[2 * i]) \ggg B) \oplus B;$
$B = B - S[1];$
$A = A - S[0];$
```

Figure 2.7: Pseudo Code of Decryption [3]

2.4 Zigbee Communication Protocol

For this project application, ZigBee is choosen to create IoT Transmission Layer. ZigBee is a set of high-level communication protocols based on the IEEE 802.15.4 standard. ZigBee is utilized for automation and remote control applications with a low data rate, low power consumption, and low cost [25]. It is used for applications such as home automation, medical device data collection, and other low-power, low-bandwidth needs. It is intended for small-scale projects that require a wireless connection [26].

Parameters	ZigBee Value
Tranmission Range(meters)	1-100
Battery Life (days)	100-1000
Network Size(of nodes)	>64000
Throughput (kb/s)	20-250

Table 2.1: Basic ZigBee Specification [1]

2.5 Programs Used to Implement Project

In this section, programs and tools that are utilized for implementing the project is investigated. All is used for modelling the system and simulating the system to check its results.

2.5.1 MATLAB and Simulink

MATLAB is a platform that provides analyzing and designing systems. Key of the program is MATLAB language which is matrix based. With MATLAB, apps can be built, data can be analyzed and visualized, algorithms can be developed for embedded applications and more [27]. Simulink is MATLAB and model-based design tool that provides simulating system before hardware [28]. It ensures multidomain modeling and simulation and reusability of this models. Also, models which are not made in Simulink can be added for combining. Other ability of the Simulink is testing and automatically generating C and Hardware Description Language (HDL) code that operates the same as the model. After, it can be implemented directly to FPGA [29]. For this project, Simulink tool which is an Embedded Coder is used. Embedded Coder generates C and C++ code from model with a code efficiency for asked processor [30].

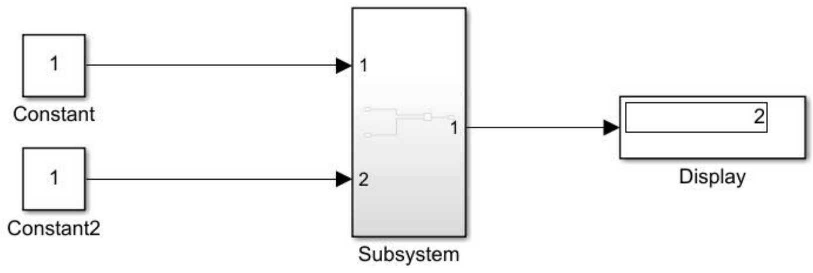


Figure 2.8: Example Simulink System

Fig.2.8 shows an example model which is built in Simulink. It can be seen that some constant blocks, created subsystem and display block exist. Code can be easily generated from subsystem with this setup.

2.5.2 Xilinx Vivado Design Suite

Vivado Design Suite is a tool that enhances performance by designing, integrating and implementing systems. With Vivado, implementations can be made faster and more optimized by using some tools such as route and place, and all stages can be seen [31]. Moreover, in Vivado, Register Transfer Level (RTL) schematics and behavioral simulations can be obtained. In this project, MicroBlaze and Universal Asynchronous

Receiver Transmitter (UART) used together in Vivado design. MicroBlaze is Reduced Instruction Set Computer (RISC) based soft processor for embedded applications which ensures peripheral and memory combined with minimum cost [32]. UART stands for carrying out data conversions that is received from peripheral device [33]. Fig.2.9 shows the system of MicroBlaze and UART system. After block design is set with export hardware selection Xilinx Shell Archive (XSA) file is generated for the embedded software part. XSA file has hardware specifications of processor.

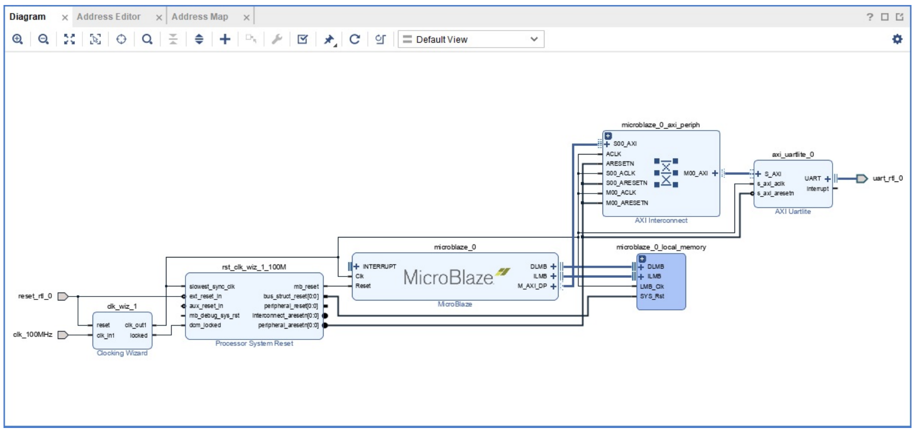


Figure 2.9: MicroBlaze System with UART

2.5.3 Xilinx Vitis Platform

Vitis is one of the latest tools of Xilinx that combines all software into itself. Embedded software applications are developed and used in embedded processors with usage of Vitis. It includes C/C++ editor, project management, focused special tools to configure FGPA and so on. XSA file is imported into Vitis for a platform project. In this project, application is used as software project and it may contain more than source file. It produces a binary output which is Executable and Linkable Format (ELF) file [5]. Fig.2.10 shows the project types in Vitis. As a result, processor system is set with MicroBlaze and UART connected to it. Afterwards, XSA is generated and used to create platform project.

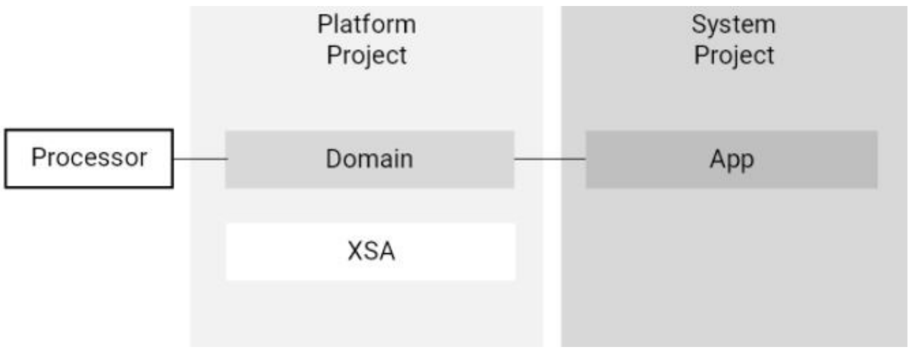


Figure 2.10: Project Types in Vitis [5]

3. MODEL BASED IMPLEMENTATION

3.1 Designing Simulink Model with ZigBee Protocol

IoT module is built for intensive care patients. Simulink model consists of 4 main parts: rooms for patients, RC5 algorithm, transmitter, and receiver with output. First, room area and Simulink blocks will be explained.

3.1.1 Room Area

Model is designed to check four patient's temperature and heart rate data one after another. Number of patients may be increased easily after that point. Fig.3.1 shows the exterior look of the patient's data subsystems. While one data is transferred, other patients' data is set as 0. After one is completely transferred, next one shows the instant data of the patient.

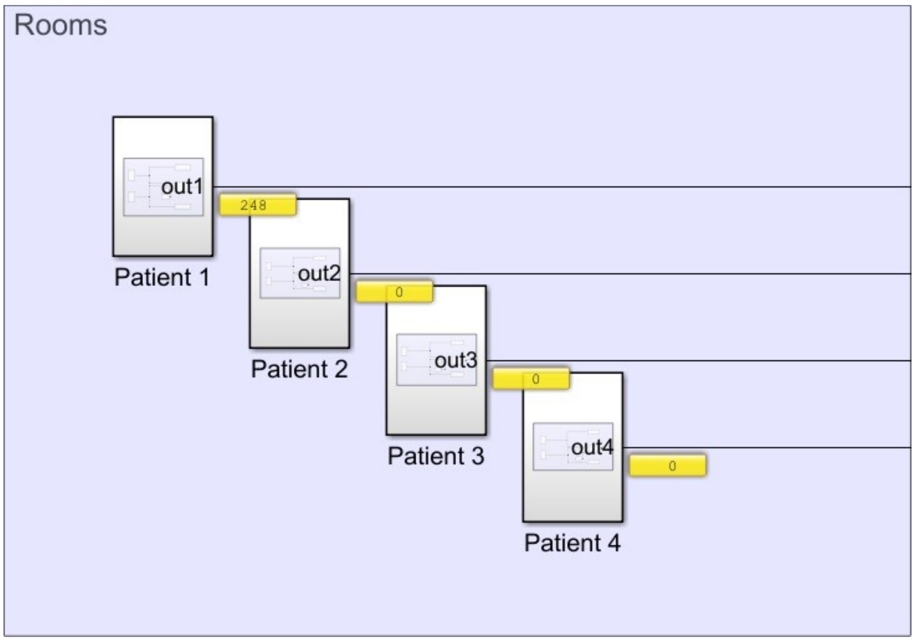


Figure 3.1: Room Area

When inside of the patient subsystem studied as in Fig.3.2, it can be seen that two more subsystems appear which are temperature and heart rate. In that subsystems, data is generated in particular sample times. MATLAB function to control values involves levels and data according to its level that appear at the output. As seen, values display logically for human body temperature and heart rate. There are also two clocks which are used as enables for every patient. This clock operation provides generate patient's data in particular times which while is generated others are set to 0.

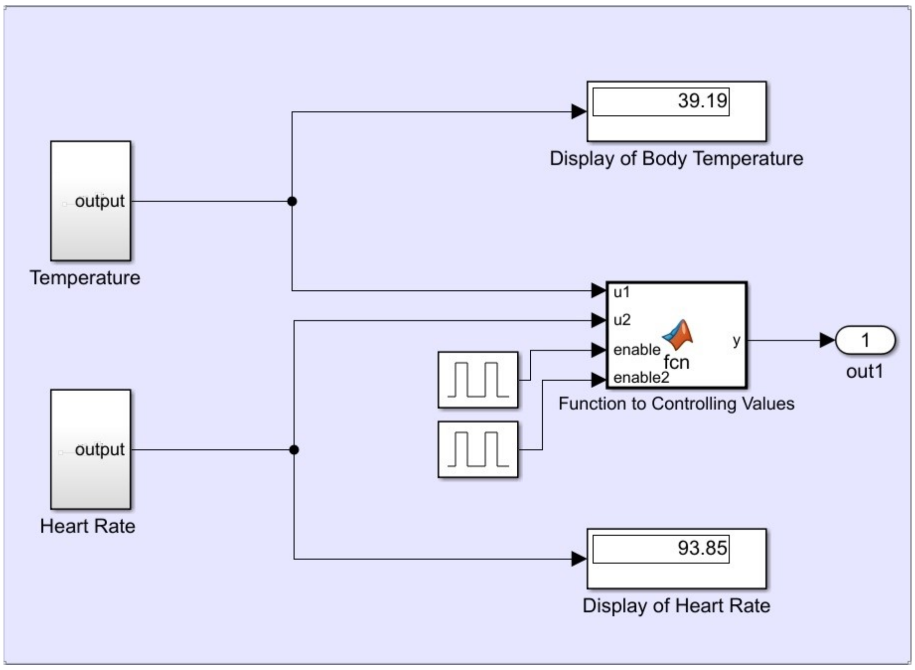


Figure 3.2: Inside of the Patient Subsystem

As seen in Fig.3.3, random number for temperature of human body is generated by PS Random Number and taken by sensor. To have controlled data, controlled temperature source block is required. S port stands for delivering the data into B port and connected to sensor's A port. Temperature sensor consists of three ports which are A, B and T. A port exists for positive heat in contrast of B. T is obtained by the equation T = TA - TB which is same with T == A.T – B.T equation [34]. To take the value only in A port, B port is connected to thermal reference which means TB = 0. Any sensor in Simscape library needs Solver Configuration block to obtain simulation. Sensors have T output value as Kelvin, then the system needs PS-Simulink Converter block. PS-converter with apply affine conversion box checked in Simulink operates Kelvin to Celsius conversion [35]. For the heart rate system, same path is followed.

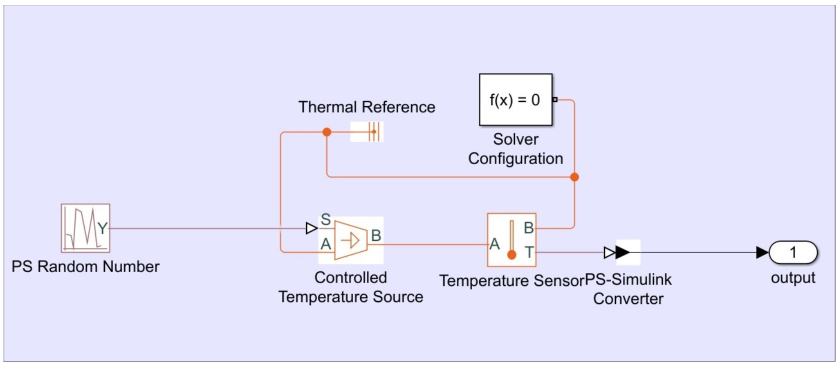
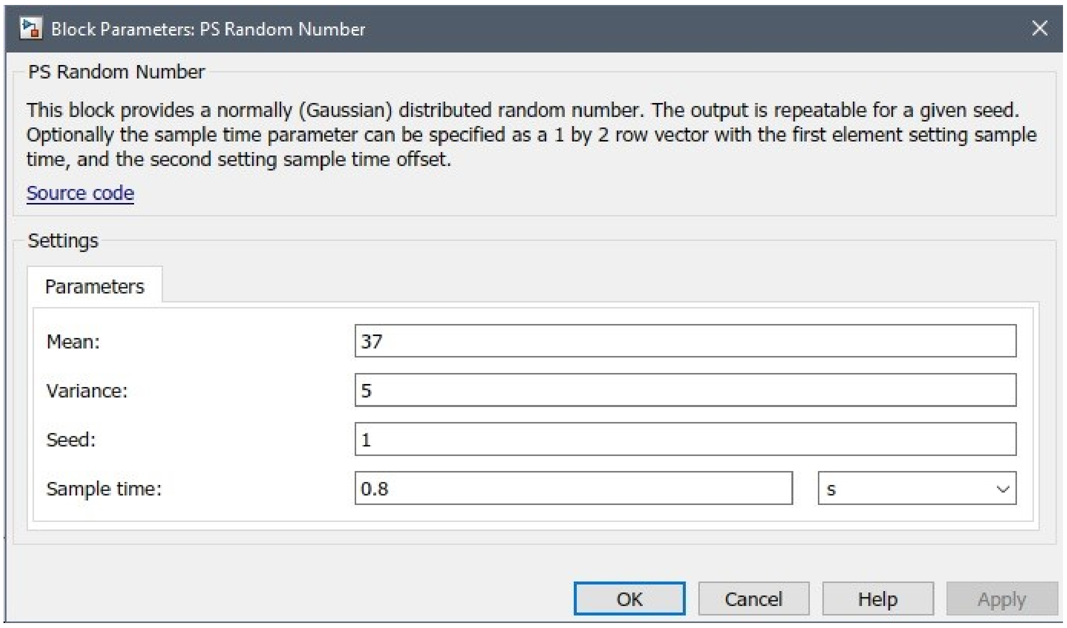


Figure 3.3: Inside of the Temperature Subsystem

To obtain source values which is generated by Gaussian function in PS Random Number, parameters are set as figure. With mean = 37 and variance = 5, probability density function is obtained which has values waving between 30 and 44. Sample time is set as 0.8 seconds, because every bit is transferred with sample time 0.025 and 32 bits in total, which will be explained later. The settings can be seen in Fig 3.4.


Block Parameters: PS Random Number

PS Random Number

This block provides a normally (Gaussian) distributed random number. The output is repeatable for a given seed. Optionally the sample time parameter can be specified as a 1 by 2 row vector with the first element setting sample time, and the second setting sample time offset.

Source code

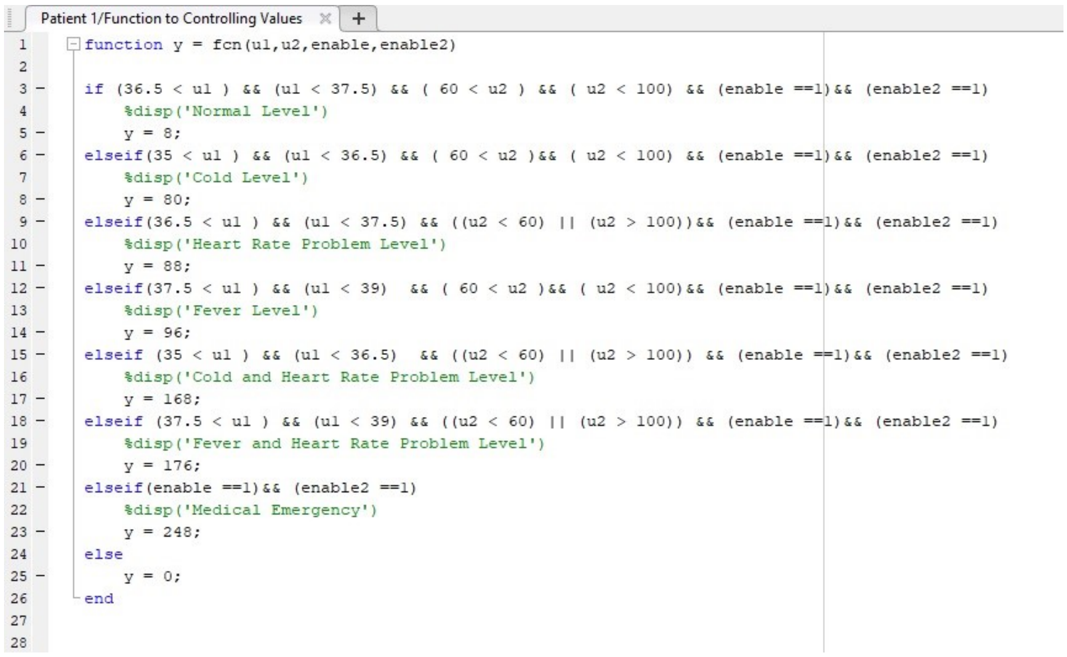
Settings

Parameters

Mean:	37
Variance:	5
Seed:	1
Sample time:	0.8 s

Figure 3.4: PS Random Number Block Settings

To produce acceptable values for the model-based design, average values and ranges for body temperature and heart rate are investigated. Since, body temperature changes depends on gender, activity in a day, age etc., average values are accepted. 37°C (98.6°F) is the mean of the temperature for humankind. Normal body temperature, fever, low and high body temperature are main groups for ranges [36]. Again, for the heart rate ranges, even though values are changing depends on so many variables such as age, activity, body size etc., 60-100 beats per minute (bpm) is accepted as normal [37]. For creating specific data according to taken by sensor, function block is manipulated by if-else structure. Therefore, rate of taken vital data can give specific data for producing output of function block. For instance, if patient has 33.9 °C temperature and 60.5 bpm heart rate, the values fit with medical emergency level; therefore, the last else if block is active. y = 248 is delivered as data to encrypt, as seen in Fig.3.5. Also, there are other levels which are combinations temperature and heart rate value. If problem is just about heart rate, 3rd level is active and data is generated.



```
function y = fcn(u1,u2,enable,enable2)

if (36.5 < u1 ) && (u1 < 37.5) && ( 60 < u2 ) && ( u2 < 100) && (enable ==1)&& (enable2 ==1)
    %disp('Normal Level')
    y = 8;
elseif(35 < u1 ) && (u1 < 36.5) && ( 60 < u2 )&& ( u2 < 100) && (enable ==1)&& (enable2 ==1)
    %disp('Cold Level')
    y = 80;
elseif(36.5 < u1 ) && (u1 < 37.5) && ((u2 < 60) || (u2 > 100))&& (enable ==1)&& (enable2 ==1)
    %disp('Heart Rate Problem Level')
    y = 88;
elseif(37.5 < u1 ) && (u1 < 39)  && ( 60 < u2 )&& ( u2 < 100)&& (enable ==1)&& (enable2 ==1)
    %disp('Fever Level')
    y = 96;
elseif (35 < u1 ) && (u1 < 36.5)  && ((u2 < 60) || (u2 > 100)) && (enable ==1)&& (enable2 ==1)
    %disp('Cold and Heart Rate Problem Level')
    y = 168;
elseif (37.5 < u1 ) && (u1 < 39) && ((u2 < 60) || (u2 > 100)) && (enable ==1)&& (enable2 ==1)
    %disp('Fever and Heart Rate Problem Level')
    y = 176;
elseif(enable ==1)&& (enable2 ==1)
    %disp('Medical Emergency')
    y = 248;
else
    y = 0;
end
```

Figure 3.5: MATLAB Function to Generate Patient's Data

Generated data which equals y in this function is not selected randomly. Since, UART can show 0 to 255 which is 8-bit, maximum value is set according this information. After that, first 3-bit is arranged for rooms that provides to enhance the model to 8 patients. Next 3-bit is set for levels since there are 7 levels as seen in function for controlling values. The last 2-bit is left for alarm depends on the level. For instance, 248 is set in Fig3.6. Room number, level and alarm equal to 1,7, and 3 respectively. At the output part of the model, function for this operation will be investigated in details.

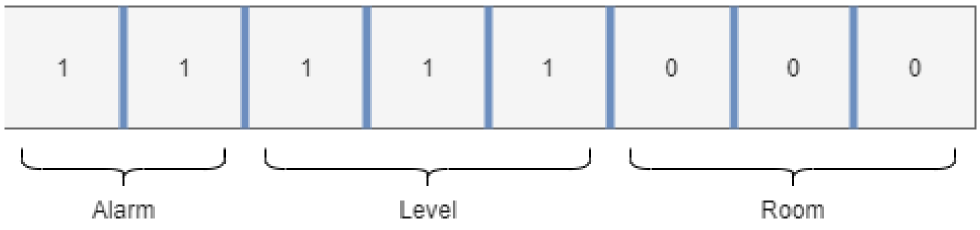


Figure 3.6: Logic Behind of the Produced Data Value

3.1.2 RC5 Algorithm Area

Second part of the model is the RC5 algorithm. For this part, MathWorks website is checked for possibility of the prepared code for this purpose and RC5 Encryption Algorithm [38] is found. All functions are organized and manipulated for the model. This area consists of 4 main subparts which are data-room selection function, encryption, decryption and subsystem for parameters and key expansion as seen in Fig 3.7. Primarily, data-room selection function will be explained. This function operates like multiplexer. Since, the model has 4 rooms and patient's data must be transferred in order, this structure is needed. 2 clock signals assist for enabling data for particular room. One clock is set to 1.6 seconds; however, other is 3.2. Since 32-bit is adapted by the receiver in 0.8 seconds. The function has 6 inputs which are u1 to u4 data inputs and 2 enables. Output y shows 248 as data to be input for encryption. Inside of this block can be seen in Fig.3.8.

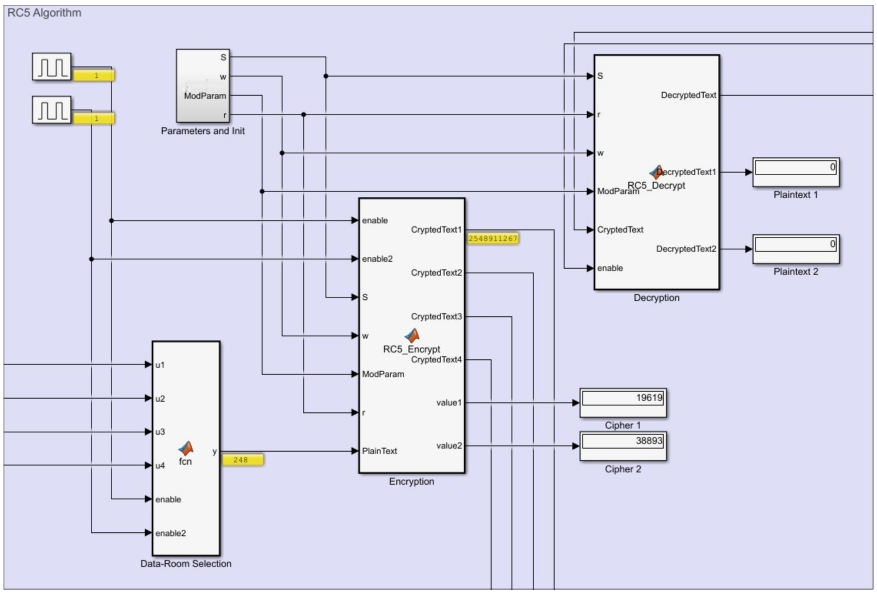
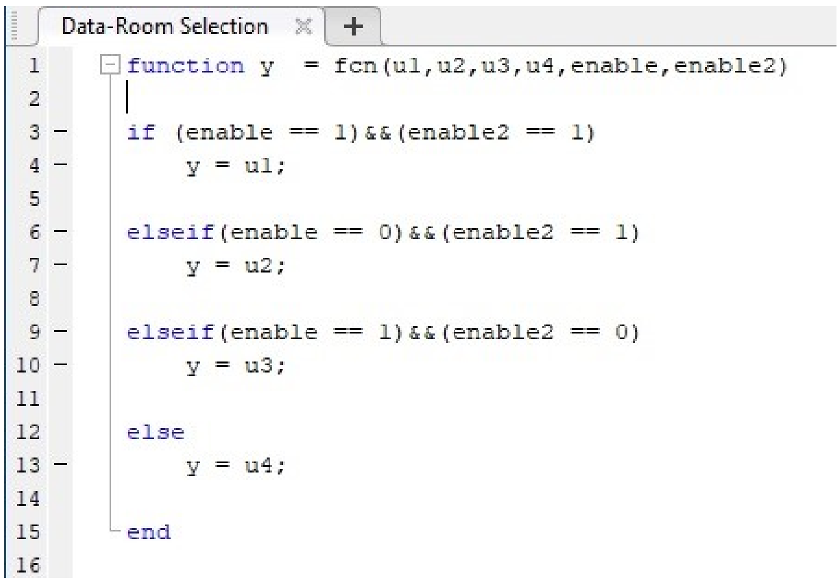


Figure 3.7: RC5 Algorithm Area



```
function y  = fcn(u1,u2,u3,u4,enable,enable2)

if (enable == 1)&&(enable2 == 1)
    y = u1;

elseif(enable == 0)&&(enable2 == 1)
    y = u2;

elseif(enable == 1)&&(enable2 == 0)
    y = u3;

else
    y = u4;

end
```

Figure 3.8: Data-Room Selection Function Block

Parameters can be seen clearly at Fig 3.9. RC5 operates two-word length; so, w is selected as 16 to obtain 32-bit cipher. Other important parameters are b and r. Number of rounds which is r is set 6 to make the algorithm fast. Also, b equals the byte number of secret key K and it is selected as 8, and K array initialized with this value. P and Q are magic constants, t is size of table, c is number of words in key are all constants according to b, r and w. All about parameters block can be seen in Fig 3.10. There is one difference in parameters which is studied in mathematical background of RC5 algorithm. As an addition, ModParam parameter exist. It has a value of 65536 to use in other functions to be able to make mod operation according to 32-bit.

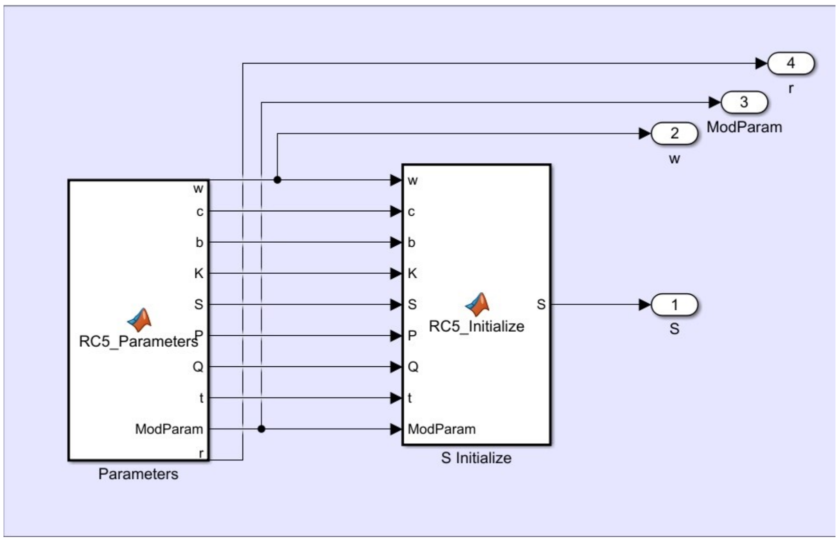
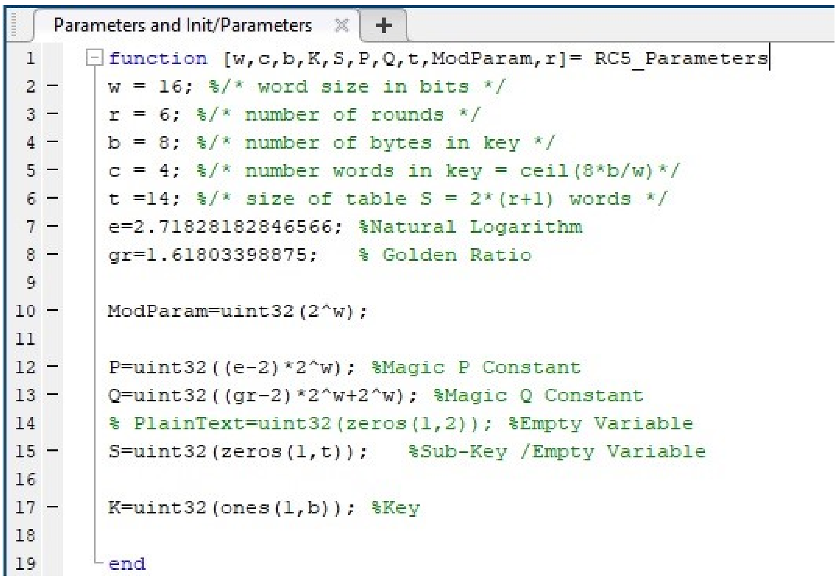


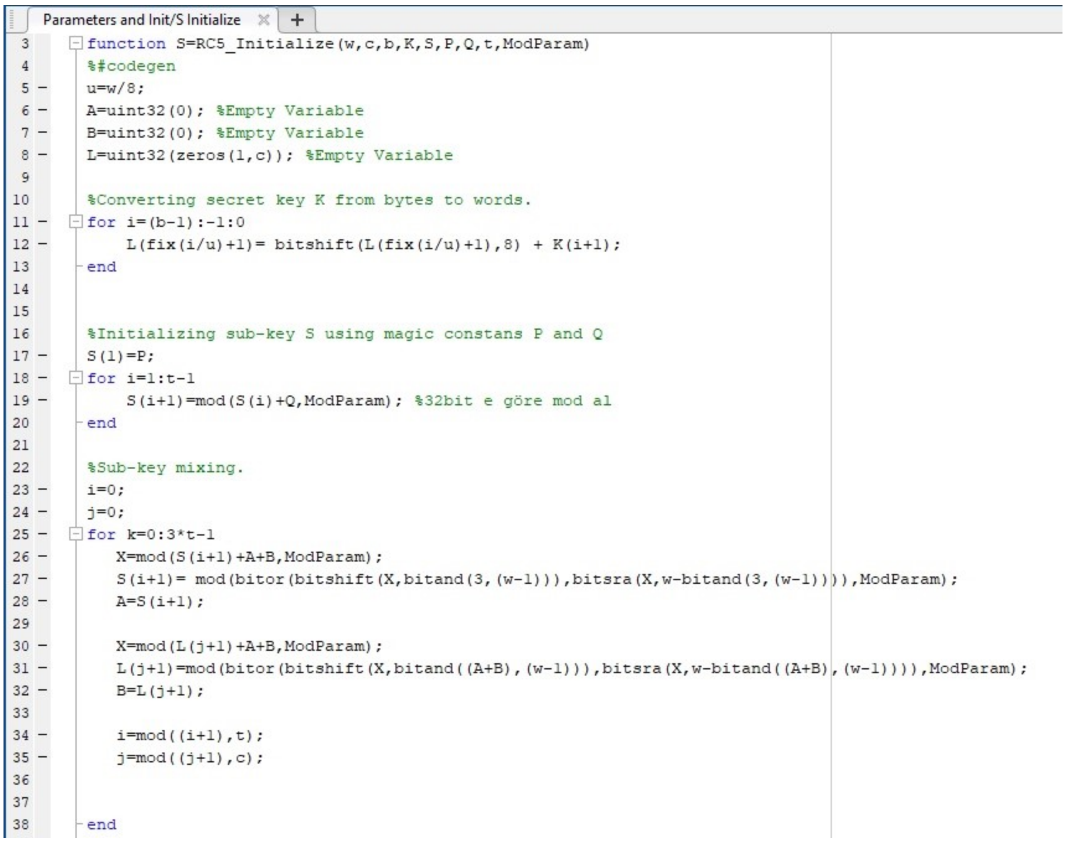
Figure 3.9: Parameters and Init Function Subsystem



```
function [w,c,b,K,S,P,Q,t,ModParam,r]= RC5_Parameters
w = 16; %/* word size in bits */
r = 6; %/* number of rounds */
b = 8; %/* number of bytes in key */
c = 4; %/* number words in key = ceil(8*b/w)*/
t =14; %/* size of table S = 2*(r+1) words */
e=2.71828182846566; %Natural Logarithm
gr=1.61803398875;    % Golden Ratio

ModParam=uint32(2^w);

P=uint32((e-2)*2^w); %Magic P Constant
Q=uint32((gr-2)*2^w+2^w); %Magic Q Constant
% PlainText=uint32(zeros(1,2)); %Empty Variable
S=uint32(zeros(1,t));    %Sub-Key /Empty Variable

K=uint32(ones(1,b)); %Key

end
```

Figure 3.10: Parameters Function

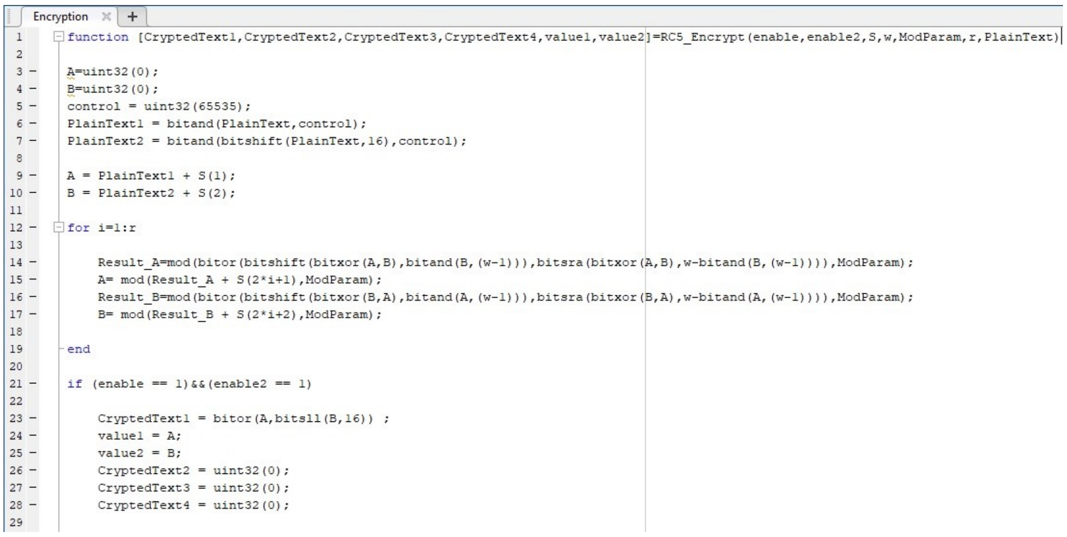
After that point, RC5 algorithm starts. First step is expanding secret key K which is S Initialize block for this project as seen in Fig.3.11. S output of the function will be used for both encryption and decryption. Also, key expansion has three main parts: copying K to temporary array L, initializing S with magic constants and mixing the key, as explained before. A and B is set as empty for start and assigned S[i] at the sub key mixing part. Rotational left shift algorithm can be seen in mixing part. This part is just put in order for prevent any run issue in Simulink. Most of the logic remains as same as original one.



```matlab
function S=RC5_Initialize(w,c,b,K,S,P,Q,t,ModParam)
%#codegen
u=w/8;
A=uint32(0); %Empty Variable
B=uint32(0); %Empty Variable
L=uint32(zeros(1,c)); %Empty Variable

%Converting secret key K from bytes to words.
for i=(b-1):-1:0
    L(fix(i/u)+1)= bitshift(L(fix(i/u)+1),8) + K(i+1);
end


%Initializing sub-key S using magic constans P and Q
S(1)=P;
for i=1:t-1
    S(i+1)=mod(S(i)+Q,ModParam); %32bit e göre mod al
end

%Sub-key mixing.
i=0;
j=0;
for k=0:3*t-1
   X=mod(S(i+1)+A+B,ModParam);
   S(i+1)= mod(bitor(bitshift(X,bitand(3,(w-1))),bitsra(X,w-bitand(3,(w-1)))),ModParam);
   A=S(i+1);

   X=mod(L(j+1)+A+B,ModParam);
   L(j+1)=mod(bitor(bitshift(X,bitand((A+B),(w-1))),bitsra(X,w-bitand((A+B),(w-1)))),ModParam);
   B=L(j+1);

   i=mod((i+1),t);
   j=mod((j+1),c);


end
```

Figure 3.11: S Initialize Function Block

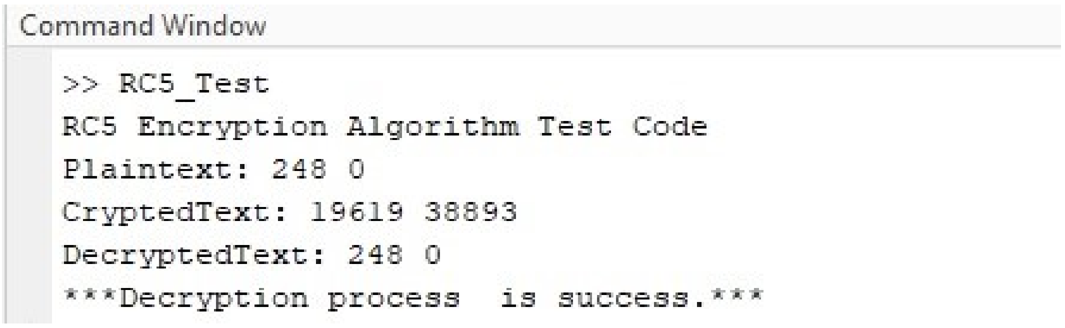
After initializing, encryption code is put in order. At this step, some additions are made to prepared code. First of all, RC5 algorithm has 2 plain texts to be encrypted, which are A and B. However, for this project, both of the texts are concatenated which can be seen in line 23 in Fig3.12. Therefore, newer plain text is 32-bit. Algorithm is implemented in for structure, which has rotational left and mod operation. Enables are set for room and value1 and value2 is put the code for checking encrypted plain texts. As seen in test results, first plain text is selected as 248 and the second one is 0. Then, 19619 is calculated for 248 and, 38893 is for 0. As a result, 19619 is low-order 16 bit and 38893 is high-order 16 bit which gives 2548911267 in decimal. Test results can be shown in Fig.3.13. While one room is selected, other rooms are 0, just to prevent mess. The same path is followed for the rest of the data.



Encryption

```
function [CryptedText1,CryptedText2,CryptedText3,CryptedText4,value1,value2]=RC5_Encrypt(enable,enable2,S,w,ModParam,r,PlainText)

A=uint32(0);
B=uint32(0);
control = uint32(65535);
PlainText1 = bitand(PlainText,control);
PlainText2 = bitand(bitshift(PlainText,16),control);

A = PlainText1 + S(1);
B = PlainText2 + S(2);

for i=1:r

    Result_A=mod(bitor(bitshift(bitxor(A,B),bitand(B,(w-1))),bitsra(bitxor(A,B),w-bitand(B,(w-1)))),ModParam);
    A= mod(Result_A + S(2*i+1),ModParam);
    Result_B=mod(bitor(bitshift(bitxor(B,A),bitand(A,(w-1))),bitsra(bitxor(B,A),w-bitand(A,(w-1)))),ModParam);
    B= mod(Result_B + S(2*i+2),ModParam);

end

if (enable == 1)&&(enable2 == 1)

    CryptedText1 = bitor(A,bitsll(B,16)) ;
    value1 = A;
    value2 = B;
    CryptedText2 = uint32(0);
    CryptedText3 = uint32(0);
    CryptedText4 = uint32(0);

```

Figure 3.12: Encryption Function Block

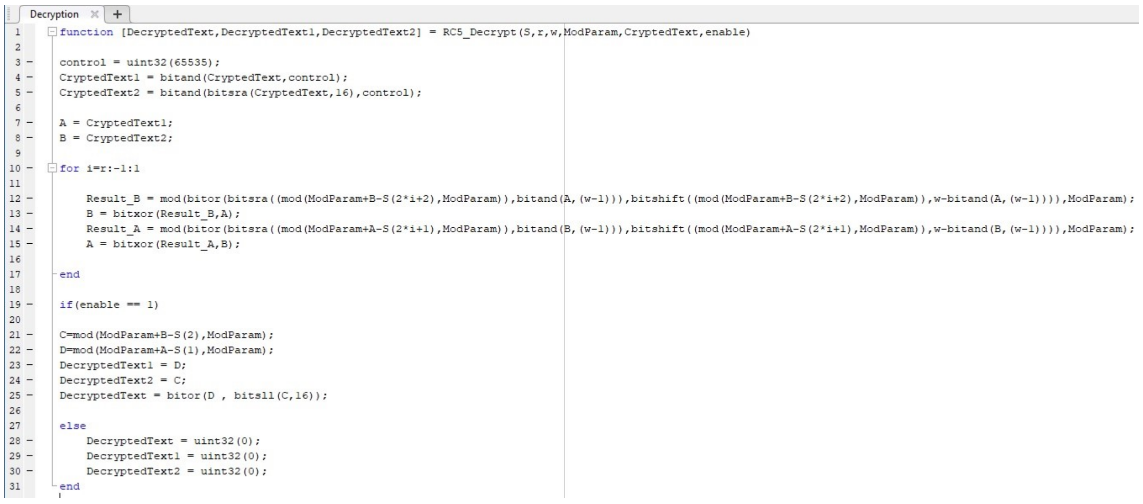


Command Window

```
>> RC5_Test
RC5 Encryption Algorithm Test Code
Plaintext: 248 0
CryptedText: 19619 38893
DecryptedText: 248 0
***Decryption process  is success.***
```

Figure 3.13: Data Encryption Results for 248

After 32-bit cipher text is obtained, this result is transferred bit by bit which will be explained in transmitter part. Assuming the data is successfully reached at the input of the decryption block, low-order 16-bit and high-order 16-bit is seperated primarily. Later, the opposite algorithm according to encryption is made, which equals the structure in Fig 3.14. This time, rotational right is used and loop is set as r = 6 to 1. If statement has C and D. Again, they stand for checking whether the result is what is expected or not. Moreover, bitor operation is made for the final result which will be 248 for the input 2548911267.



```
function [DecryptedText,DecryptedText1,DecryptedText2] = RC5_Decrypt(S,r,w,ModParam,CryptedText,enable)

control = uint32(65535);
CryptedText1 = bitand(CryptedText,control);
CryptedText2 = bitand(bitsra(CryptedText,16),control);

A = CryptedText1;
B = CryptedText2;

for i=r:-1:1

    Result_B = mod(bitor(bitsra((mod(ModParam+B-S(2*i+2),ModParam)),bitand(A,(w-1))),bitshift((mod(ModParam+B-S(2*i+2),ModParam)),w-bitand(A,(w-1)))),ModParam);
    B = bitxor(Result_B,A);
    Result_A = mod(bitor(bitsra((mod(ModParam+A-S(2*i+1),ModParam)),bitand(B,(w-1))),bitshift((mod(ModParam+A-S(2*i+1),ModParam)),w-bitand(B,(w-1)))),ModParam);
    A = bitxor(Result_A,B);

end

if(enable == 1)

C=mod(ModParam+B-S(2),ModParam);
D=mod(ModParam+A-S(1),ModParam);
DecryptedText1 = D;
DecryptedText2 = C;
DecryptedText = bitor(D , bitsll(C,16));

else
    DecryptedText = uint32(0);
    DecryptedText1 = uint32(0);
    DecryptedText2 = uint32(0);
end
```

Figure 3.14: Decryption Function Block

3.1.3 Transmitter Area

After encryption, data is needed to be transmitted to receiver. For this purpose, Zigbee module is found as said before. Transmitter area can be seen in Fig.3.15. It has three parts: bit serializer, transmitter and transmitter selection. Every room has its transmitter and main idea is transmitting room's data one after another and bit by bit. After encrypted data produced in MATLAB function, decimal number went out of the block. Nevertheless, data is needed to transfer bit by bit. In this purpose, serializer 1D block which provides by Simulink is used and data is transferred to transmitter. As seen in Fig.3.16, decimal number is converted to 32-bit number; therefore, parallel data is obtained. Integer to bit converter has output of true, false, false, true, true etc. for 32-bit and serializer delivers them bit by bit.

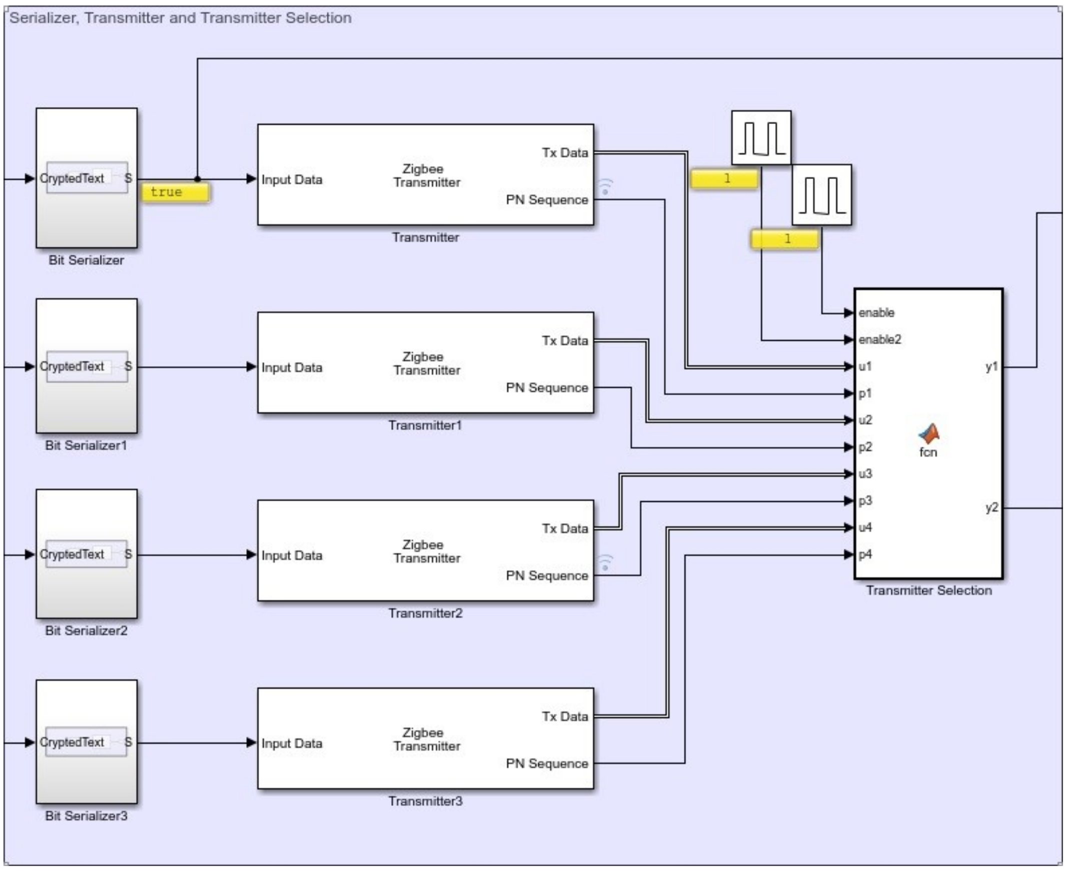


Figure 3.15: Transmitter Area

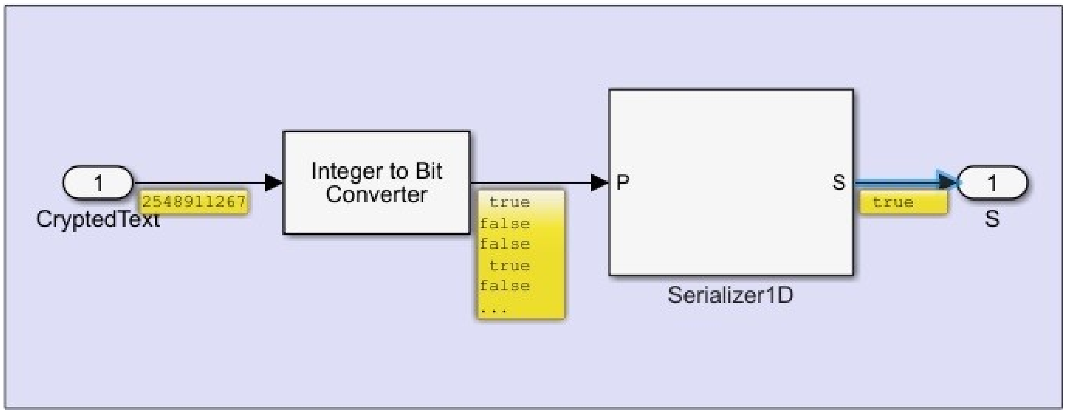


Figure 3.16: Bit Serializer Subsystem

Next part is transmitter part which is used as prepared. Fig.3.17 shows the interior look of the transmitter subsystem. There are two different outputs and data is modulated at this part with Offset Quadrature Phase-Shift Keying (OQPSK) block. This transmitter is driven with a 0.025 seconds sample time.

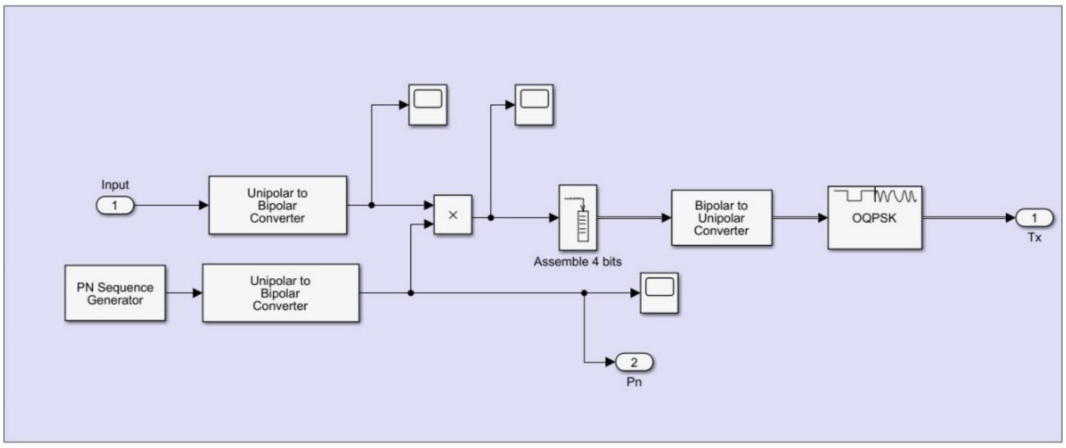
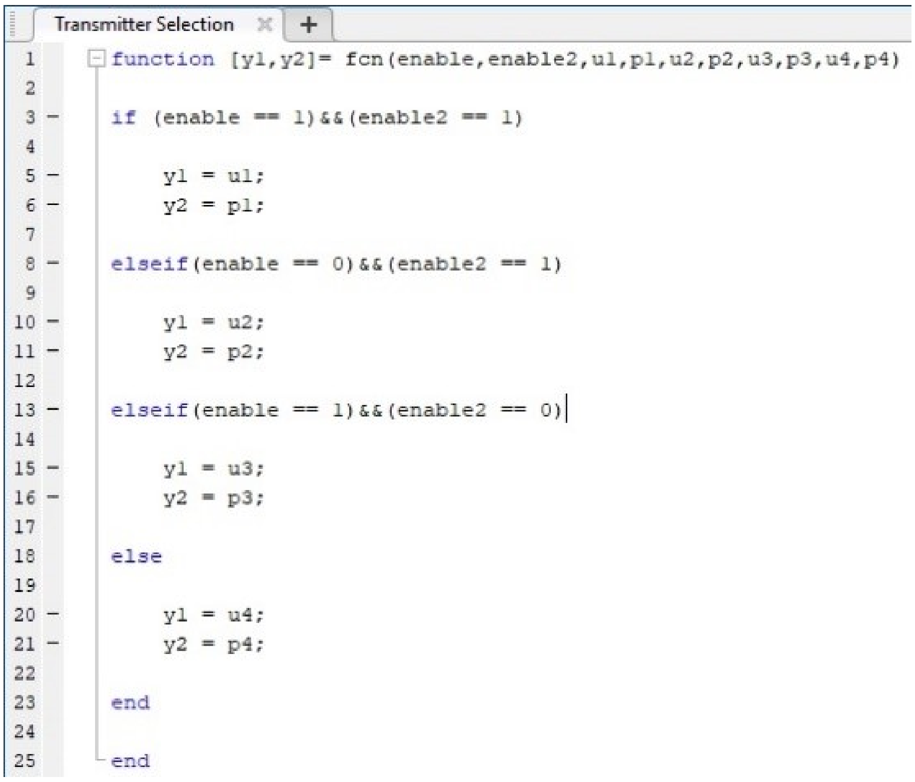

Input
1
Unipolar to Bipolar Converter
PN Sequence Generator
Unipolar to Bipolar Converter
×
Assemble 4 bits
Bipolar to Unipolar Converter
OQPSK
1
Tx
2
Pn

Figure 3.17: Transmitter Block

Before going receiver and output side, all transmitters must be controlled because model has only receiver. Therefore, two clocks are added to provide enabling at particular times. For both of them set as true, first room's data is transmitted. Again, this function behaves like multiplexer. The reason that multiplexer block is not used is giving errors when the model is tried to run. Since there are two different outputs that come out from transmitter, y1 and y2 defined to be transferred out of the transmitter selection function as can be seen in details in Fig.3.18.



```
function [y1,y2]= fcn(enable,enable2,u1,p1,u2,p2,u3,p3,u4,p4)

if (enable == 1)&&(enable2 == 1)

    y1 = u1;
    y2 = p1;

elseif(enable == 0)&&(enable2 == 1)

    y1 = u2;
    y2 = p2;

elseif(enable == 1)&&(enable2 == 0)

    y1 = u3;
    y2 = p3;

else

    y1 = u4;
    y2 = p4;

end

end
```

Figure 3.18: Transmitter Selection Function Block

3.1.4 Reciever and Output Area

First part of this section is receiver area. It has Additive White Gaussian Noise (AWGN), receiver and error calculation part as in Fig.3.19. Output y1 of the transmitter selection function goes to AWGN before reaching receiver; since, this defines noise in nature. After data is demodulated in receiver system, it remains as 1-bit form. For solving output data problem, it was understood that data should be turned into bytes. Therefore, demodulated data is sent to designed shift register that will be explained later.

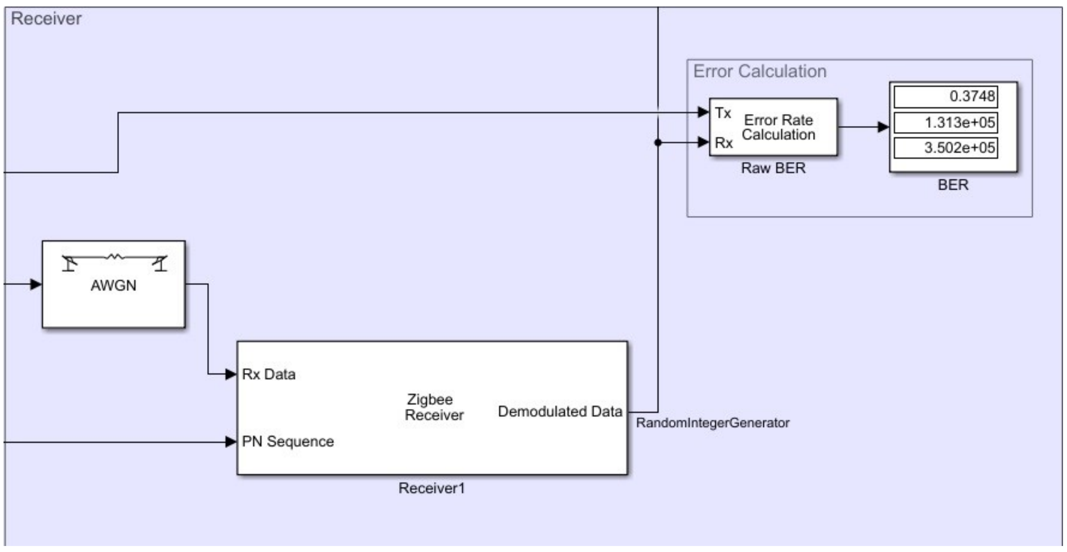


Figure 3.19: Receiver Area

Fig.3.20 shows the inside of the receiver. These blocks provide demodulation of the data produces as Boolean value every 0.025 seconds. Error calculation part does not work correctly since data is driven different than this area. When 0.025 seconds value per bit is tried to change, some errors occurs and result came wrong. So, error calculation part is passed for now.

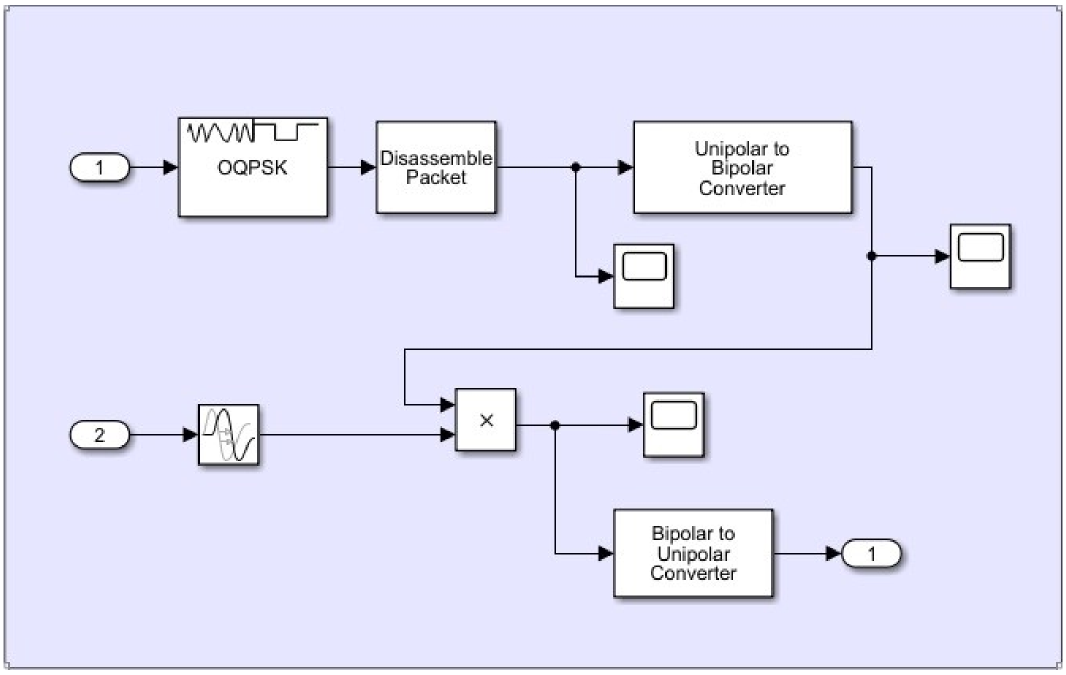


Figure 3.20: Receiver Area

Second part is output area. There are 3 main parts there: shift register, function to investigate text, and displays for room, level and alarm as in Fig.3.21.

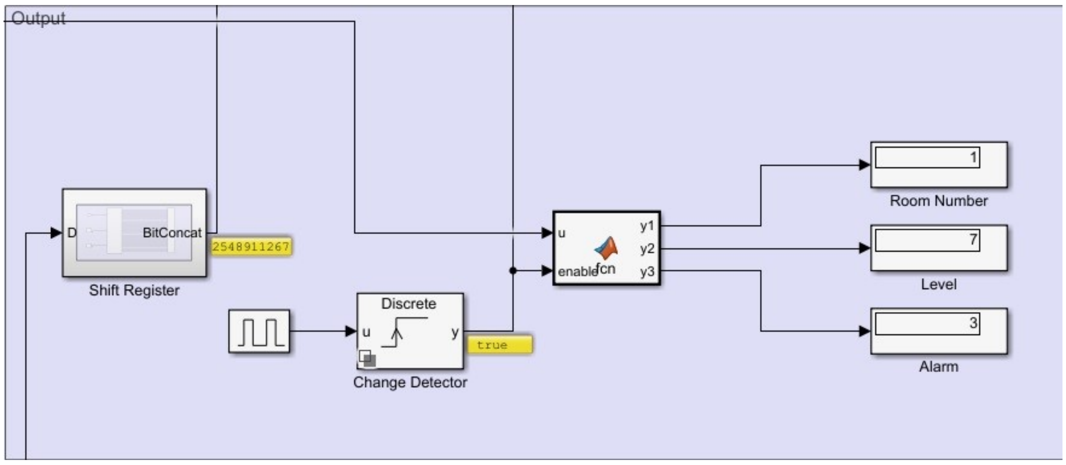


Figure 3.21: Output Area

First of all, shift register will be handled. To obtain whole number data again, the system requires shift register later receiver. Shift register which is available in Simulink library gave errors for this module. Then, subsystem was created for this reason and shift register is designed in Simulink. 32 D flip flops are connected for parallelization of the data. The reason why we need this subsystem is holding all the bits until all of them are received. Every Q value in FF concatenated with a block that is called “bit concat” to obtain final result. After this process, output data is obtained immediately before new data is produced at patient1 subsystem. Fig.3.22 and Fig3.23 shows the design of the shift register and the results at 0.2 seconds. Flip flops hold the values from least significant bit to most significant bit and Q to Q31 is connected according to place. Clock is set to 0.025 and constant value stands for clear (!CLR) connection.

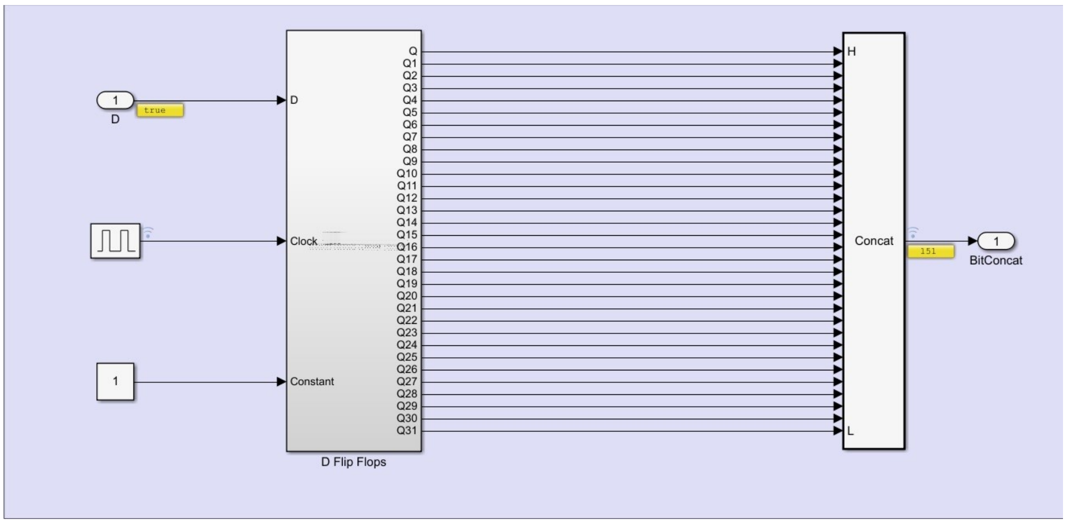


Figure 3.22: Shift Register Subsystem

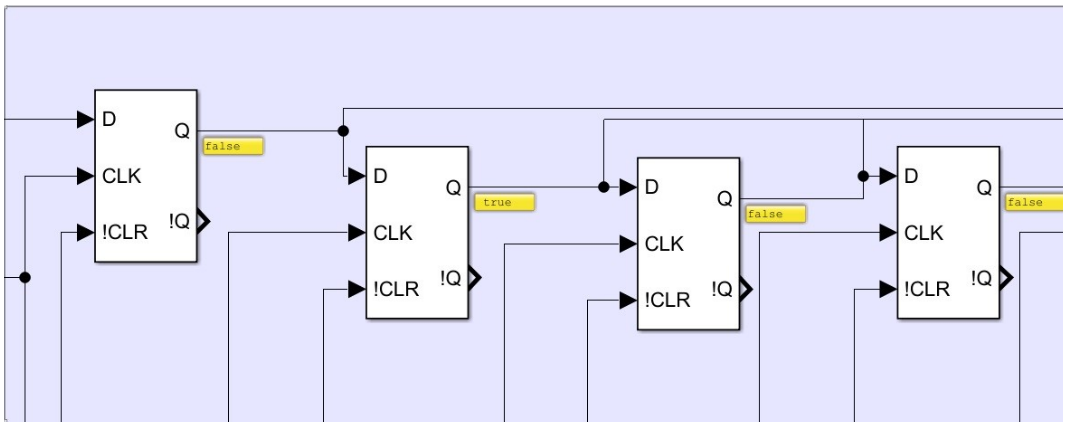
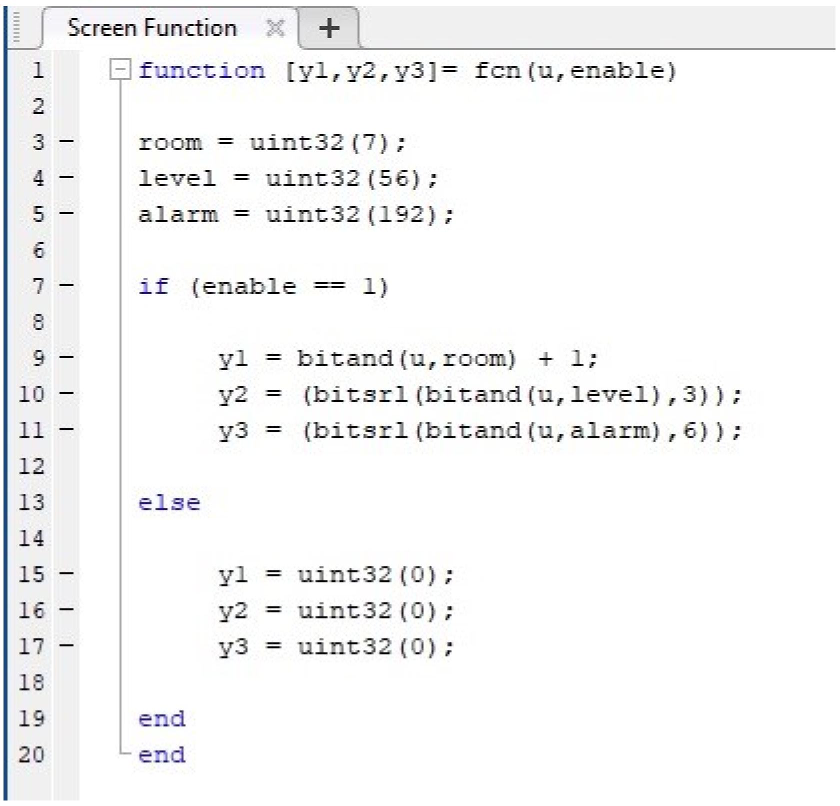


Figure 3.23: D Flip Flops Subsystems

When encrypted text is obtained in 0.8 seconds, output of the shift register is sent to decryption block to provide to system the real data. Then, the result of decryption is connected to function to investigate data. This function is the implementation the idea of Fig.3.6. To learn where data comes from, data u is put the operation bitand with 7 which equals 8-bit 00000111. For the next 3-bit level is defined as 56 that is 8-bit 00111000. For the alarm same logic is followed. Last of all, change detector is used for this part. The reason behind it is seeing the results at display when clock rises at 0.8 seconds. When the first bit of the second room is transferred to shift register, display is reset. Implementation of the logic into code can be seen in Fig.3.24. It separates data into 3 parts as room, level and alarm.



```
function [y1,y2,y3]= fcn(u,enable)

room = uint32(7);
level = uint32(56);
alarm = uint32(192);

if (enable == 1)

     y1 = bitand(u,room) + 1;
     y2 = (bitsrl(bitand(u,level),3));
     y3 = (bitsrl(bitand(u,alarm),6));

else

     y1 = uint32(0);
     y2 = uint32(0);
     y3 = uint32(0);

end
end
```

Figure 3.24: Function to Screen Data

4. HARDWARE IMPLEMENTATION

4.1 Implementing Model to FPGA

4.1.1 Simplfying Model

The aim is generating C code from the Simulink model, then transferring the model to FPGA. Fig.4.1 shows the model after removing transmitter,receiver and output area. Moreover, room number is reduced to 2 to obtain fast and simple system. Final model has just 2 data inputs, 3 enable inputs and a result output. Data types should be arranged as like as model, otherwise some errors occur depending on this types.

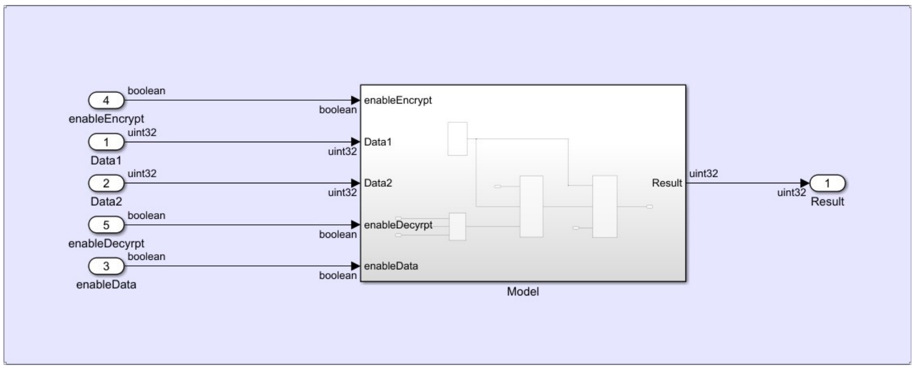


Figure 4.1: Simplified Model

When Model subsystem is investigated as in Fig.4.2, it is clearly seen that there are just 4 functions. Data1 and Data2 inputs of data-room selection function is set as taking data from rooms. They are defined as just inputs. Real values will be given later part of this section. Fig.4.3 shows the function of selection. Again, it has duty of multiplexer with the simplest functional way. SInit, encryption and decryption functions are simplified to 2 rooms and there is no more change inside of the functions.

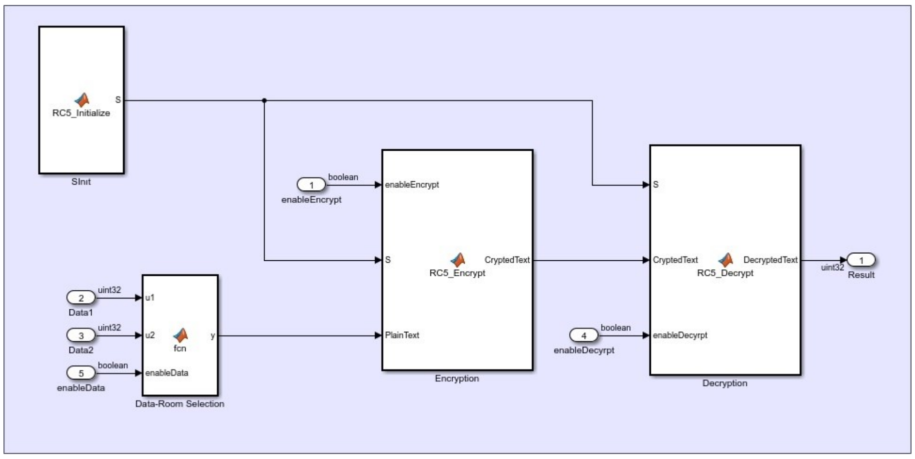
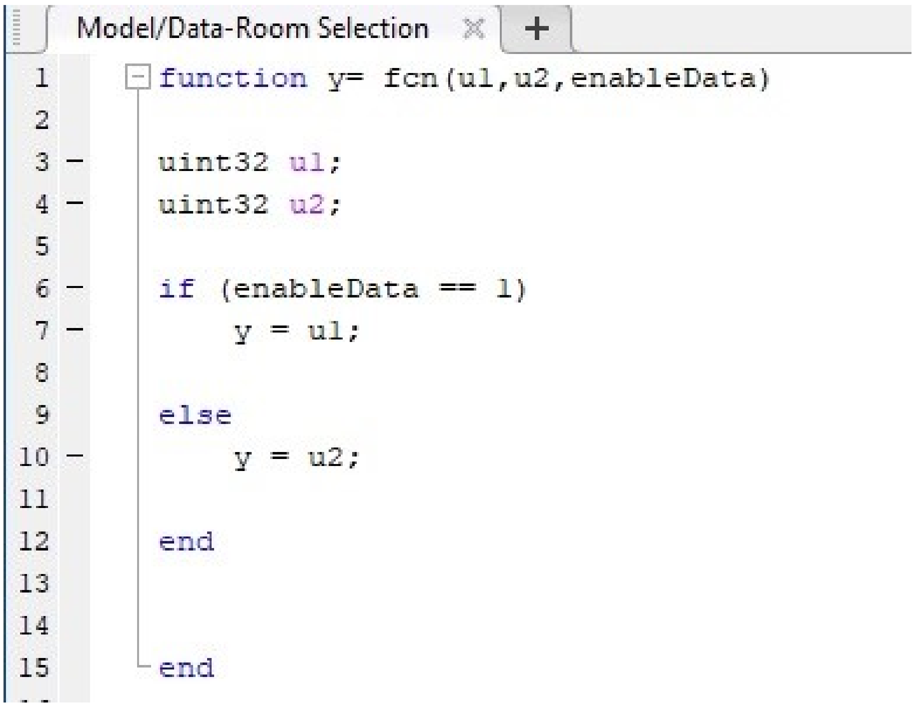


Figure 4.2: Model Subsystem



```
function y= fcn(u1,u2,enableData)

uint32 u1;
uint32 u2;

if (enableData == 1)
    y = u1;

else
    y = u2;

end


end
```

Figure 4.3: Simplified Data-Room Selection Function

After all process, the model is ready to be generated C code. As in Fig.4.4, subsystem is selected and right clicked on it. C/C++ code and Embedded Coder Quick Start is selected, respectively. From the opening screen, settings are done for the custom processor. Then the model is ready to transfer.

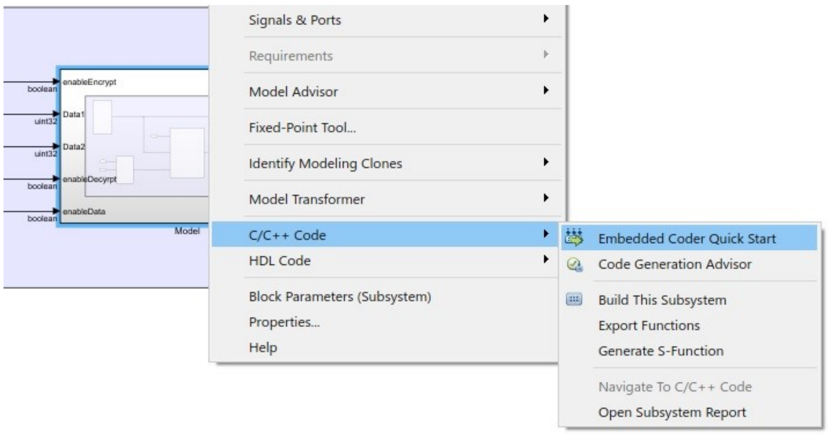
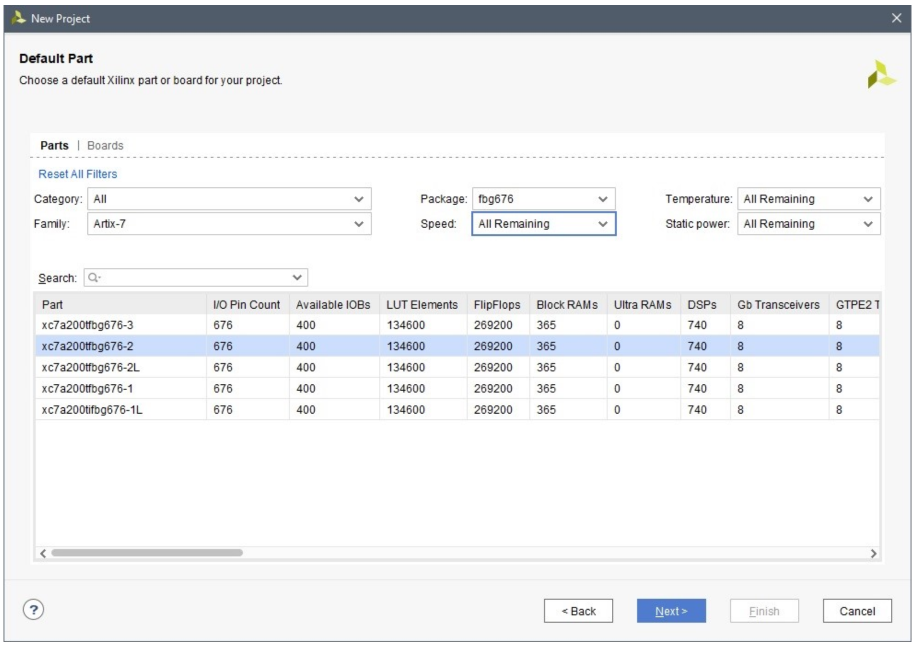


Figure 4.4: Generating Code from Subsystem

4.1.2 Generating MicroBlaze and UART Environment

C code is generated. This section includes checking the generated C code in FPGA. First of all, project is created in Vivado and board is selected as seen in Fig.4.5. The board is selected to encounter enough input and output pins and Lookup Table (LUT) elements; moreover, this is the board that we are familiar with which worked before for another projects.



Default Part

Choose a default Xilinx part or board for your project.

Category: All

Family: Artix-7

Package: fbg676

Speed: All Remaining

Temperature: All Remaining

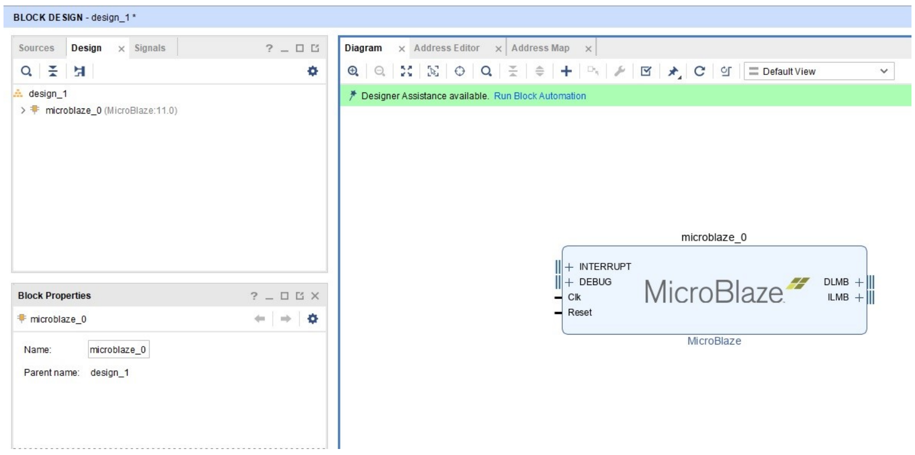
Static power: All Remaining

Search:

Part	I/O Pin Count	Available IOBs	LUT Elements	FlipFlops	Block RAMs	Ultra RAMs	DSPs	Gb Transceivers	GTPE2 T
xc7a200tfbg676-3	676	400	134600	269200	365	0	740	8	8
xc7a200tfbg676-2	676	400	134600	269200	365	0	740	8	8
xc7a200tfbg676-2L	676	400	134600	269200	365	0	740	8	8
xc7a200tfbg676-1	676	400	134600	269200	365	0	740	8	8
xc7a200tifbg676-1L	676	400	134600	269200	365	0	740	8	8

Figure 4.5: Board Selection Screen in Vivado Platform

After that, Create Block Design is selected from the IP Integrator part in flow navigator to add MicroBlaze and UART. First MicroBlaze is added as in Fig.4.6; however, some settings are needed to be changed.



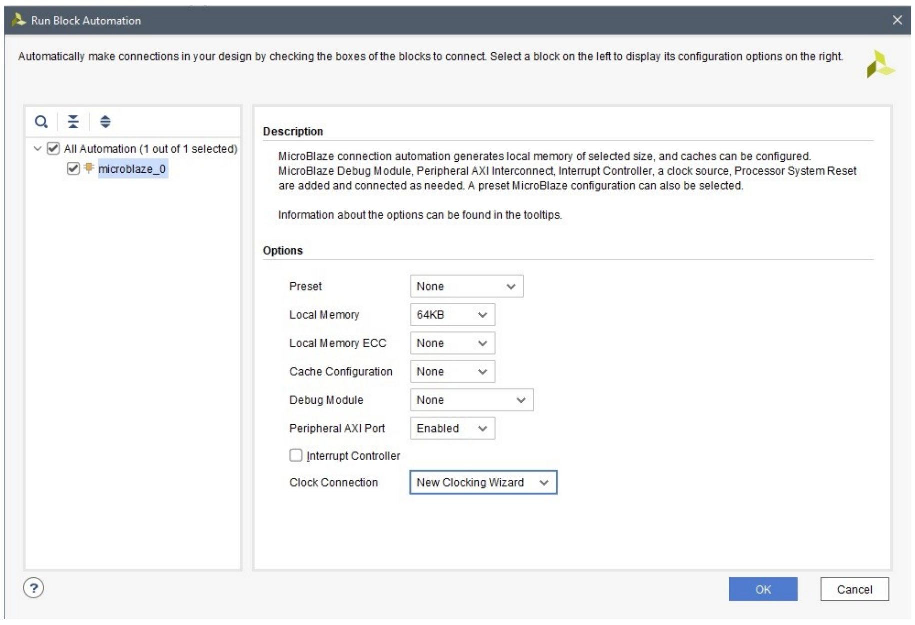
Designer Assistance available. Run Block Automation

Name: microblaze_0

Parent name: design_1

Figure 4.6: MicroBlaze in Vivado

After clicking on Run Block Automation, pop up window comes as in Fig.4.7. Local memory is set to 64KB to avoid insufficient system for the model. Also, debug module is set to none. Then, Vivado generates blocks automatically and make connections. In this case, Clocking Wizard, Peripheral Advanced eXtensible Interface (AXI) Port and local memory is generated.



Automatically make connections in your design by checking the boxes of the blocks to connect. Select a block on the left to display its configuration options on the right.

Description

MicroBlaze connection automation generates local memory of selected size, and caches can be configured. MicroBlaze Debug Module, Peripheral AXI Interconnect, Interrupt Controller, a clock source, Processor System Reset are added and connected as needed. A preset MicroBlaze configuration can also be selected.

Information about the options can be found in the tooltips.

Options

Preset	None
Local Memory	64KB
Local Memory ECC	None
Cache Configuration	None
Debug Module	None
Peripheral AXI Port	Enabled
☐ Interrupt Controller	
Clock Connection	New Clocking Wizard

Figure 4.7: Settings for MicroBlaze

Just before adding UART to the system, clock is set to single ended capable because the model is connected to one clock. Then, UART is added and clicked on Run Connection

Automation. Vivado makes all proper connections between inputs and outputs as seen in Fig.4.8. From the sources part, block design is found and right clicked on it to create HDL wrapper. This operation exists because block design cannot be synthesized. The wrapper is synthesized. The C code has already generated and needed to be checked. For this purpose, as a last process of Vivado is exporting hardware to integrate software and hardware for this stage.

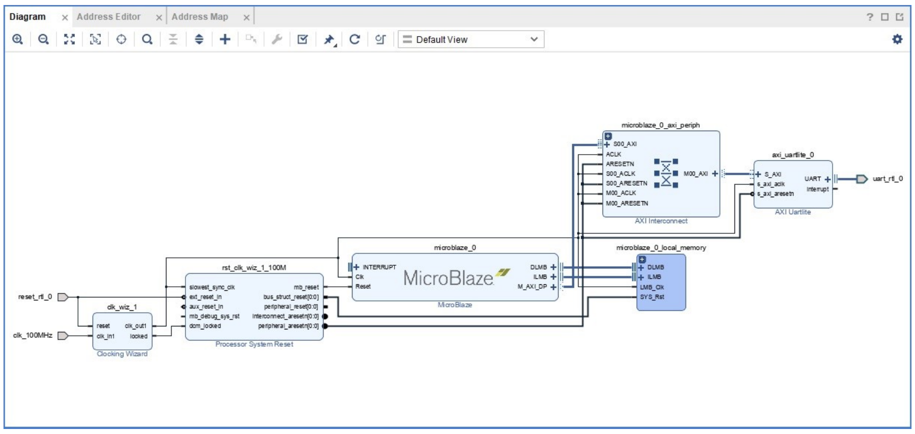
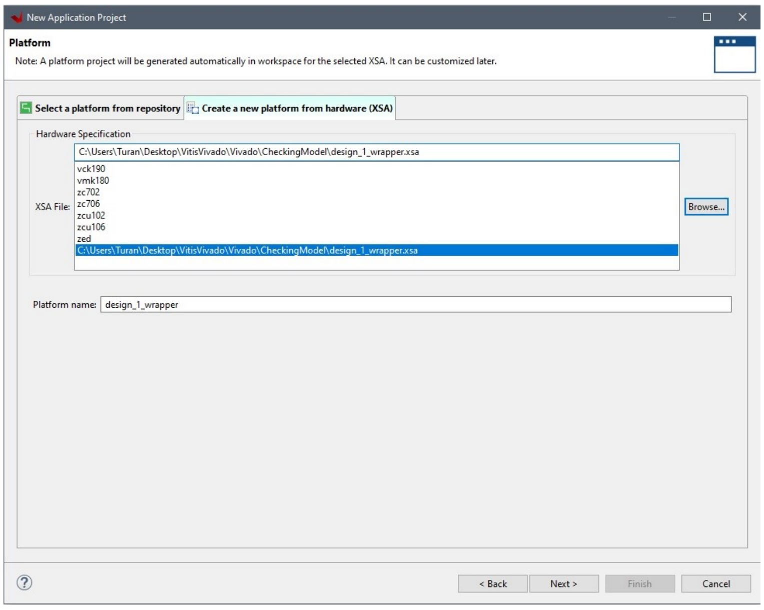


Figure 4.8: Completed Block Design in Vivado

4.1.3 Creating ELF File

In Vivado, from the tools tab Launch Vitis Integrated Development Environment (IDE) is selected and a workspace is created. When export hardware process is done, there exist a file with .xsa extension. Later, .xsa file is selected that is like in Fig.4.9 for the hardware and empty application project is obtained.



New Application Project

Platform

Note: A platform project will be generated automatically in workspace for the selected XSA. It can be customized later.

Select a platform from repository | Create a new platform from hardware (XSA)

Hardware Specification

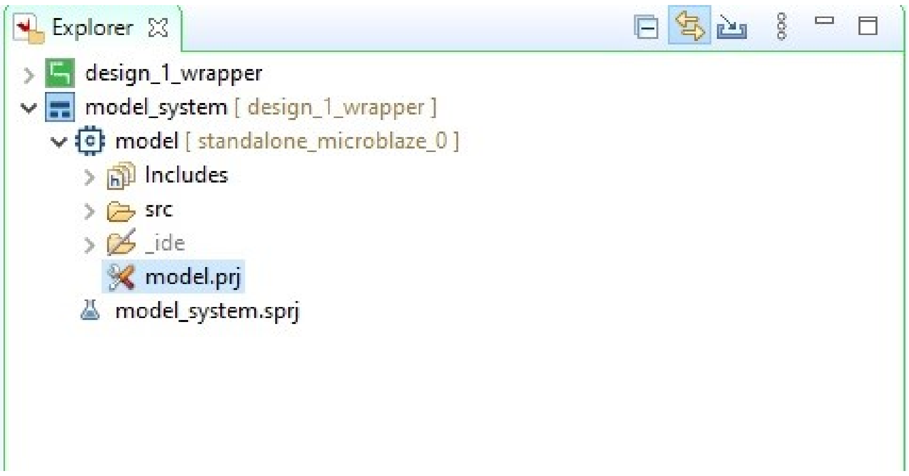
XSA File: C:\Users\Turan\Desktop\VitisVivado\Vivado\CheckingModel\design_1_wrapper.xsa

```
vck190
vmk180
zc702
zc706
zcu102
zcu106
zed
C:\Users\Turan\Desktop\VitisVivado\Vivado\CheckingModel\design_1_wrapper.xsa
```

Platform name: design_1_wrapper

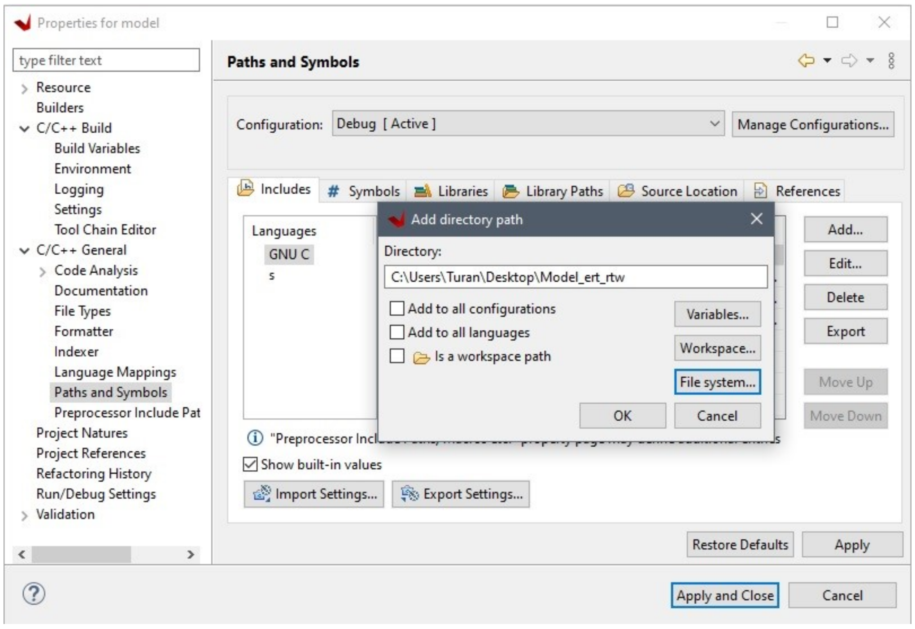
Figure 4.9: Adding Exported Hardware to Vitis Platform

So, Fig.4.10 shows the hierarchy of the Vitis application project. There are 2 main operations as a start. First, generated code folder should be added to Includes for the system. For this, right clicked on model-system. C/C++ general/Paths and Symbols path is selected as in Fig.4.11 and folder is added to workspace. Second, another C file is necessary to src tab because elf file will be generated. The c file called checkModel is created.



```
Explorer
> design_1_wrapper
v model_system [ design_1_wrapper ]
  v model [ standalone_microblaze_0 ]
    > Includes
    > src
    > _ide
      model.prj
    model_system.sprj
```

Figure 4.10: View of the Vitis Application Project



Configuration: Debug [Active]

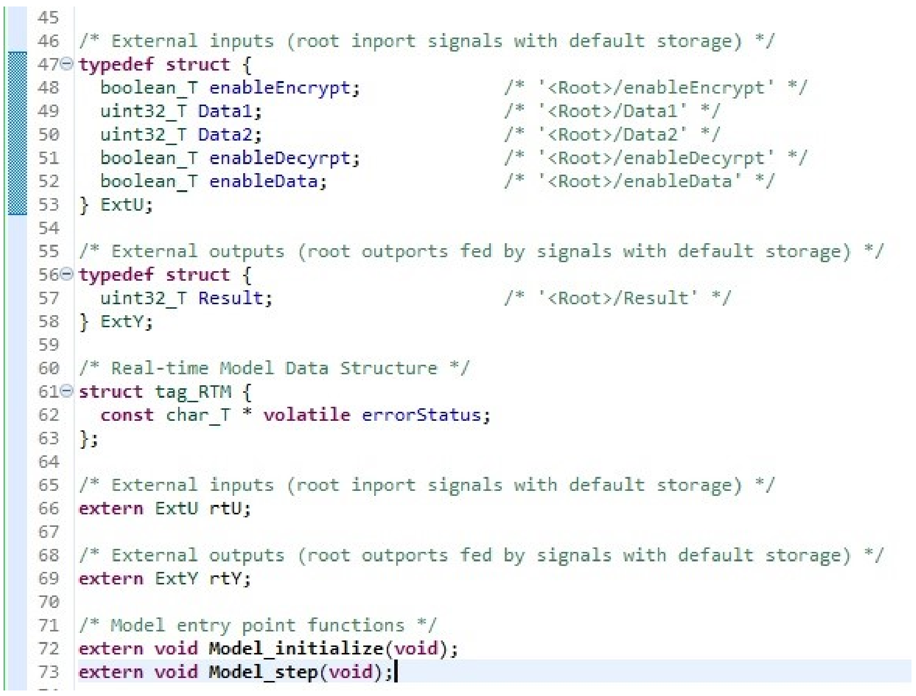
Directory: C:\Users\Turan\Desktop\Model_ert_rtw

☐ Add to all configurations
☐ Add to all languages
☐ Is a workspace path

☑ Show built-in values

Figure 4.11: Adding Generated Folder to Vitis Platform

Generated code is analyzed and seen that there are external inputs and outputs which are rtU and rtY, respectively. Inputs and outputs are as same as Simulink model names. Moreover, generated functions that are Model_initialize and Model-step is seen in Fig.4.12. Model_step is necessary to be called in the test function because it is mainly the C code of all operations in Simulink model. In other words, encryption, decryption and data-room selections are buried in Model_step.



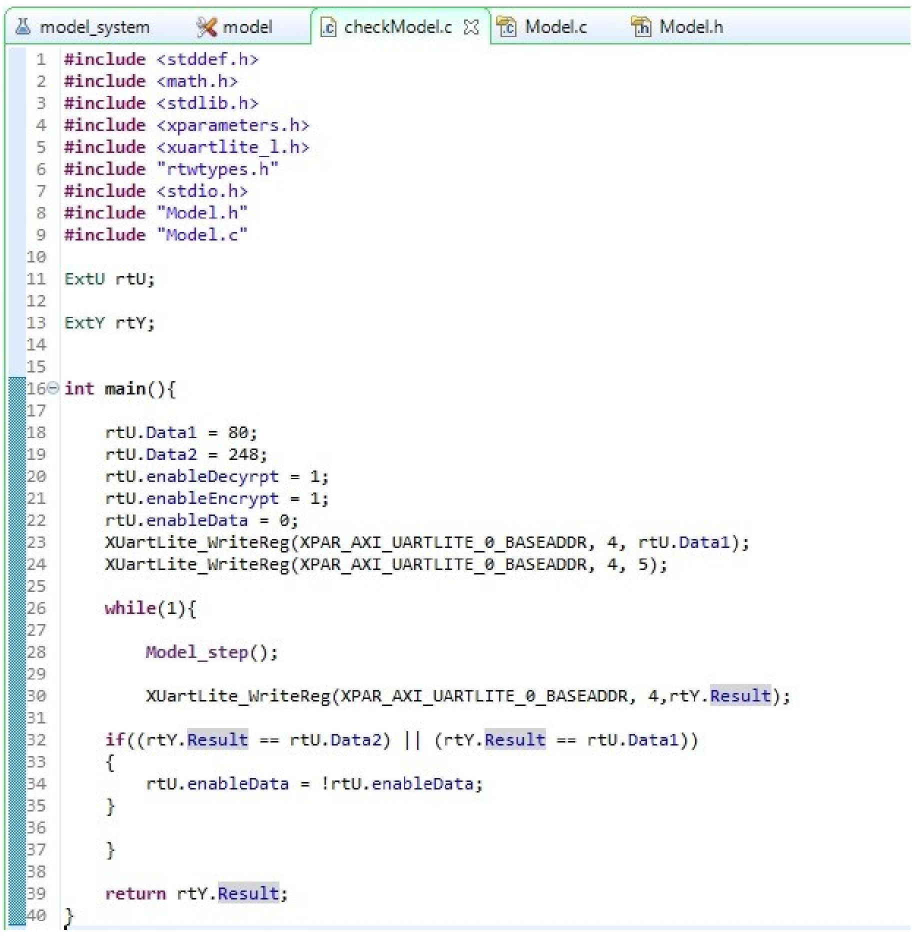
```c

/* External inputs (root inport signals with default storage) */
typedef struct {
  boolean_T enableEncrypt;             /* '<Root>/enableEncrypt' */
  uint32_T Data1;                      /* '<Root>/Data1' */
  uint32_T Data2;                      /* '<Root>/Data2' */
  boolean_T enableDecyrpt;             /* '<Root>/enableDecyrpt' */
  boolean_T enableData;                /* '<Root>/enableData' */
} ExtU;

/* External outputs (root outports fed by signals with default storage) */
typedef struct {
  uint32_T Result;                     /* '<Root>/Result' */
} ExtY;

/* Real-time Model Data Structure */
struct tag_RTM {
  const char_T * volatile errorStatus;
};

/* External inputs (root inport signals with default storage) */
extern ExtU rtU;

/* External outputs (root outports fed by signals with default storage) */
extern ExtY rtY;

/* Model entry point functions */
extern void Model_initialize(void);
extern void Model_step(void);
```

Figure 4.12: Generated Header File Model.h

The rest of the work is writing the proper code for observing the result in UART and checking the system according to particular values assigned. Fig.4.13 shows the code. All libraries that we need are added. For the UART, <xuartlite_l.h> is added as seen in line 5. External input rtU and external output rtY are also called just before the main function. Data1 is set to 80 and Data2 is set to 248, which assumed as room1 and room2 data. Then, encryption and decryption blocks are enabled. To select Data1 to transfer, enableData input is set to 0. To provide observing the results of Data1, line of code is written as below:

- XUartLite_WriteReg(XPAR_AXI_UARTLITE_0_BASEADDR, 4, rtU.Data1);

Therefore, it is expected to see Data1 first. However, observation is not about the checking the system as a start. The aim of this operation is to see whether any data can be seen or not. Afterwards, it is desired to observe another value, so 24th is written and 5 is expected to be monitored. Actually, validating part starts with while loop. Inside of the while, the function Model_step is called which has all operations that is generated from Simulink model. To see result, this time rtY.Result is selected while data which is particular first room is selected. Next step is to change the data when first one is seen as a result. To provide it, if statement is built. Since, enableData is Boolean type, when !rtu.enableData command is implemented to the system, data which is particular to 2nd room is started to send. Last operation for Vitis is creating elf file from this test code. For this, the project is built and elf file created into the workspace.

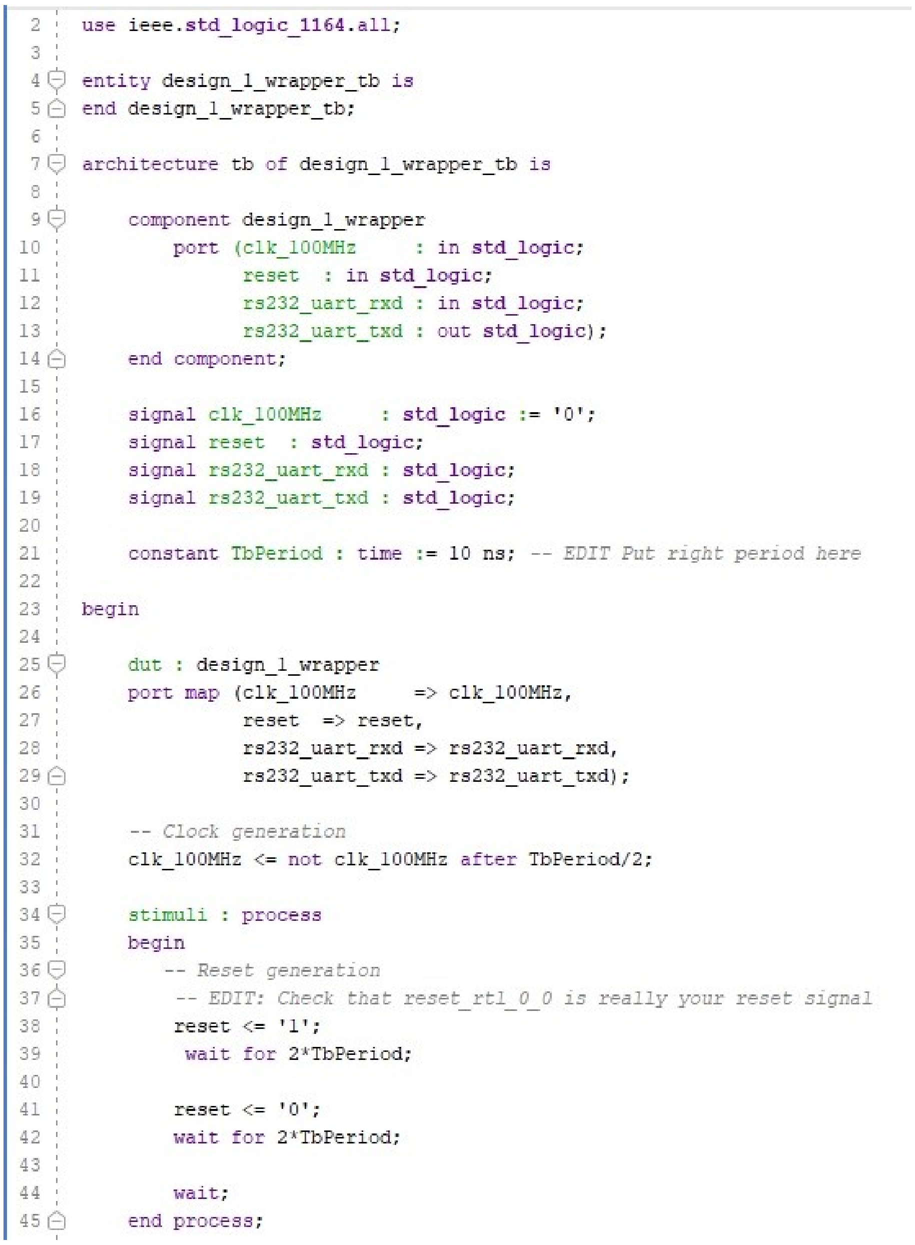


```c
#include <stddef.h>
#include <math.h>
#include <stdlib.h>
#include <xparameters.h>
#include <xuartlite_l.h>
#include "rtwtypes.h"
#include <stdio.h>
#include "Model.h"
#include "Model.c"

ExtU rtU;

ExtY rtY;


int main(){

    rtU.Data1 = 80;
    rtU.Data2 = 248;
    rtU.enableDecyrpt = 1;
    rtU.enableEncrypt = 1;
    rtU.enableData = 0;
    XUartLite_WriteReg(XPAR_AXI_UARTLITE_0_BASEADDR, 4, rtU.Data1);
    XUartLite_WriteReg(XPAR_AXI_UARTLITE_0_BASEADDR, 4, 5);

    while(1){

        Model_step();

        XUartLite_WriteReg(XPAR_AXI_UARTLITE_0_BASEADDR, 4,rtY.Result);

    if((rtY.Result == rtU.Data2) || (rtY.Result == rtU.Data1))
    {
        rtU.enableData = !rtU.enableData;
    }

    }

    return rtY.Result;
}
```

Figure 4.13: C Code of checkModel

4.1.4 Observating Results

When the operations in Vitis platform are done, it is time to write a testbench for the system and associate elf file to the system. Fig.4.14 shows the test bench code. It is written for the UART to control signals which is received and transferred. Reset and clock operations are set.

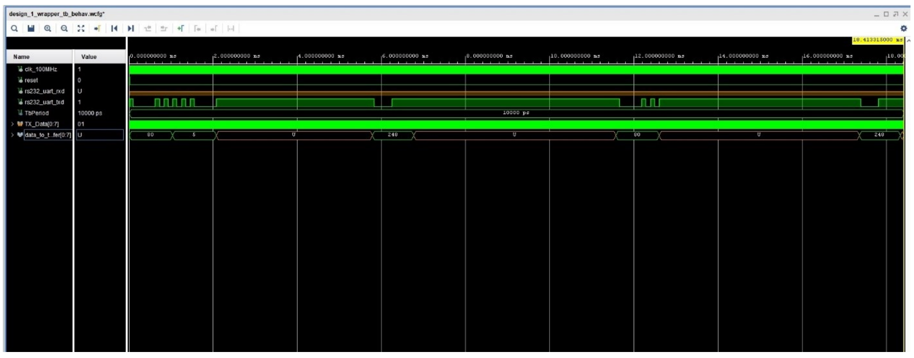


```vhdl
use ieee.std_logic_1164.all;

entity design_1_wrapper_tb is
end design_1_wrapper_tb;

architecture tb of design_1_wrapper_tb is

    component design_1_wrapper
        port (clk_100MHz     : in std_logic;
              reset  : in std_logic;
              rs232_uart_rxd : in std_logic;
              rs232_uart_txd : out std_logic);
    end component;

    signal clk_100MHz     : std_logic := '0';
    signal reset  : std_logic;
    signal rs232_uart_rxd : std_logic;
    signal rs232_uart_txd : std_logic;

    constant TbPeriod : time := 10 ns; -- EDIT Put right period here

begin

    dut : design_1_wrapper
    port map (clk_100MHz     => clk_100MHz,
              reset  => reset,
              rs232_uart_rxd => rs232_uart_rxd,
              rs232_uart_txd => rs232_uart_txd);

    -- Clock generation
    clk_100MHz <= not clk_100MHz after TbPeriod/2;

    stimuli : process
    begin
       -- Reset generation
        -- EDIT: Check that reset_rtl_0_0 is really your reset signal
        reset <= '1';
         wait for 2*TbPeriod;

        reset <= '0';
        wait for 2*TbPeriod;

        wait;
    end process;
```

Figure 4.14: Testbench Code

Last of all, associating elf file for both design and simulation is needed. For this operation, in tools tab Associate ELF files is selected and generated elf file from checkModel.c is added. To see results, Run Simulation is selected as behavioral one. After it is simulated sufficiently, some observations are made in Fig.4.15. To see transferred data, data_to_transfer[0:7] is investigated in simulation. 8-bit can be observed as data. As seen, values are 80, 5, U, 248, U, 80, U, 248, respectively. So, before the while loop in C code Data1 and 5 are assigned to be observed, first. First two results are matched with the code. After that, Model_step function that includes both encryption and decryption operations are called and it is started. Results remains

‘U’ which means ‘don’t cares’ until all operations are made and the output has new value. While, it is seen in simulation for a particular time, enableData is set to its not. Model_step is repeated with a starting value of 80 and after a while 80 is seen as a result. So, while one is transferred, it is time to send the following.



Name	Value
clk_100MHz	1
reset	0
rs232_uart_rxd	U
rs232_uart_txd	1
TbPeriod	10000 ps
TX_Data[0:7]	01
data_to_t..fer[0:7]	U

Figure 4.15: Behavioral Simulation Results

As a result, to see whether designed model can be implemented on FPGA or not, model is simplified to 2 rooms. As seen in results, it works correctly and after that point the room number can be increased to 8 rooms for now.

5. CONCLUSIONS AND RECOMMENDATIONS

5.1 Practical Application of this Study

The aim of this project is mainly focused on secure and efficient usage for healthcare application of a IoT system. It can be used especially for the obtaining and using of vital health data from the human body and depending on these data, it will be possible to quickly reach the patient and make necessary interventions in emergency situations. Practically, this project can improve healthcare applications in daily or emergency basis. On the other hand, project is based on model based design. Therefore, it decreases time and money spent on the IoT Network designing and implementing to present it to market.

5.2 Realistic Constraints

Even though, model based design is reducing the time spent for designing, the problem occurs mostly to do changes modeled platform into software platform. While model becomes more complex, created code for designs is becoming more incomprehensible and complicated. According to application, it would increase time for implementation for some cases.

5.2.1 Social, Environmental and Economic Impact

Nowadays, Internet of Things has expanded to a global infrastructure with the number of connected devices being in multiples of the number of people worldwide. Large communication data and inter-connected gadgets also introduce many new ways in which breaches of privacy and information can occur. In order to overcome the problem of the information applied in IoT can be easily read and used for malicious purposes, application of cryptosystems on processors used for IoT is a necessary process. On the other hand, usage of Model Based Design will reduce the time and money spent on the IoT Network designing to present it to market. This cooperation of this two main application would create an impact for secure and low cost IoT system for general usage.

5.2.2 Cost Analysis

The main cost factor of this project is A FPGA evaluation board that faculty management meets. In this project, personal computers are used. In addition, it was utilized Xilinc Vivado development environment, Xilinx Vitis Software development environment, Matlab programming and computing platform and Matlab's Simulink tool are utilized for modeling, implementing and debugging. Since, these platforms can be used free for academic purpose in our faculty, there are no other cost factor.

5.2.3 Standards

We aim to implement a system that is secure IoT healthcare applications. Thus, our project responsible to implement safe systems. According to expected outcomes of this project, it may include any written standard for this type research. Also, IEEE and NIST standards were followed for the completed works. This project asserts to be honest and realistic in stating claims or estimates based on the available data. (IEEE-Code of Ethics) This project claims to improve the understanding of technology, its appropriate application and potential consequences. (IEEE-Code of Ethics) This project claims to accept responsibility in making engineering decisions consistent with the safety, health and welfare of the public, and to disclose promptly factors that might endanger the public/environment. (IEEE-Code of Ethics)

5.2.4 Health and Safety Concerns

The aim of this project was not to create or implement any risky or harmful products that could damage users. It is used in IoT based products that are used in smart healthcare systems which have many nodes required secure connection..

5.3 Future Work and Recommendations

For future works, implemented crypto algorithms can be chosen differently depending on project application. Traditional cryptosystems are a compelling process to implement on IoT processors since power required for implementation is substantially higher than the intended levels. Overcoming this problem can be done via modifying processors that have acceptable power consumption levels with cryptographic operations. This aim would create an impact in economy for the information in IoT is secure within the region of intended power consumption levels in devices that are used.

REFERENCES

[1] B. Mihajlov and M. Bogdanoski, "Overview and analysis of the performances of zigbee- based wireless sensor networks," *International Journal of Computer Applications*, vol. 29, p. 30, 2011.

[2] L. Xuyang, K. Lam, K. Zhu, C. Zheng, X. Li, Y. Du, L. Chunhua, and P. Pong, "Overview of spintronic sensors, internet of things, and smart living," 08 2016.

[3] R. Rivest, "The rc5 encryption algorithm," in *FSE*, 1994.

[4] H. Gill, "Selection of parameter 'r' in rc5 algorithm on the basis of prime number," 03 2014, p. 3.

[5] Getting started with the vitis software platform. [Online]. Available: https://www.xilinx.com/html_docs/xilinx2020_2/vitis_doc/vitis_embedded_getstarted.html

[6] R. Fatima, R. Manal, and M. Tomader, "Cryptography in e-health using 5g based iot: A comparison study." New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3372938.3372955

[7] model-driven development (mdd). [Online]. Available: https://searchsoftwarequality.techtarget.com/definition/model-driven-development

[8] Field programmable gate array (fpga). [Online]. Available: https://www.xilinx.com/products/silicon-devices/fpga/what-is-an-fpga.html

[9] P. Hoai Luan, T. H. Tran, T. Phan, D. Le Vu Trung, D. Lam, and Y. Nakashima, "Double sha-256 hardware architecture with compact message expander for bitcoin mining," *IEEE Access*, vol. 8, pp. 1–1, 01 2020.

[10] H. B. S. Jankowski, J. Covello and J. Ritchie. (2014, 09) The internet of things: Making sense of the next mega-trend. [Online]. Available: https://www.goldmansachs.com/insights/pages/internet-of-things/iot-report.pdf

[11] ITU-T. (2012, 06) Internet of things global standards initiative. [Online]. Available: https://www.itu.int/en/ITU-T/gsi/iot/Pages/default.aspx

[12] A. B. R. Minerva and D. Rotond. Towards a definition of the internet of things (iot). [Online]. Available: https://iot.ieee.org/images/files/pdf/IEEE_IoT_Towards_Definition_Internet_of_Things_Issue1_14MAY15.pdf

[13] C. Yang, W. Shen, and X. Wang, “Applications of internet of things in manufacturing,” *2016 IEEE 20th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pp. 670–675, 2016.

[14] A. Ryzhokhin. How the iot is improving transportation and logistics). [Online]. Available: https://ardas-it.com/how-the-iot-is-improving-transportation-and-logistics

[15] F. Zantalis, G. Koulouras, S. Karabetsos, and D. Kandris, “A review of machine learning and iot in smart transportation,” *Future Internet*, vol. 11, no. 4, 2019. [Online]. Available: https://www.mdpi.com/1999-5903/11/4/94

[16] ——, “future internet a review of machine learning and iot in smart transportation,” *Future Internet*, vol. 11, 04 2019.

[17] L. Pawar, R. Bajaj, J. Singh, and V. Yadav, “Smart city iot: Smart architectural solution for networking, congestion and heterogeneity,” 05 2019, pp. 124–129.

[18] E. Tragos, V. Angelakis, A. Fragkiadakis, D. Gundlegård, C.-S. Nechifor, G. Oikonomou, H. Pöhls, and A. Gavras, “Enabling reliable and secure iot-based smart city applications,” 03 2014, pp. 111–116.

[19] Real world iot applications in different domains. [Online]. Available: https://www.edureka.co/blog/iot-applications/#healthcare

[20] (2020, 04) Iot in healthcare – connected devices, telemedicine and remote monitoring. [Online]. Available: https://www.embitel.com/blog/embedded-blog/iot-in-healthcare-connected-devices-telemedicine-and-remote-monitoring

[21] 5 applications of iot in agriculture - making agriculture smarter. [Online]. Available: https://www.biz4intellia.com/blog/5-applications-of-iot-in-agriculture/

[22] P. Sethi and S. Sarangi, “Internet of things: Architectures, protocols, and applications,” *Journal of Electrical and Computer Engineering*, vol. 2017, pp. 1–25, 01 2017.

[23] M. M. Hossain, M. Fotouhi, and R. Hasan, “Towards an analysis of security issues, challenges, and open problems in the internet of things,” in *2015 IEEE World Congress on Services*, 2015, pp. 21–28.

[24] T. Nie, Y. Li, and C. Song, “Performance evaluation for cast and rc5 encryption algorithms,” in *2010 International Conference on Computing, Control and Industrial Engineering*, vol. 1, 2010, pp. 106–109.

[25] M. Kollam and B. S R, “Zigbee wireless sensor network for better interactive industrial automation,” 12 2011, pp. 304–308.

[26] Zigbee. [Online]. Available: https://tr.wikipedia.org/wiki/ZigBee

[27] Math. graphics. programming. [Online]. Available: https://www.mathworks.com/products/matlab.html/

[28] Simulation and model-based design. [Online]. Available: https://www.mathworks.com/products/simulink.html/

[29] Simulink for system modeling and simulation. [Online]. Available: https://www.mathworks.com/solutions/system-design-simulation.html/

[30] Embedded coder. [Online]. Available: https://www.mathworks.com/products/embedded-coder.html/

[31] Vivado design suite user guide. [Online]. Available: https://www.xilinx.com/support/documentation/sw_manuals/xilinx2018_2/ug910-vivado-getting-started.pdf

[32] Microblaze. [Online]. Available: https://www.xilinx.com/products/intellectual-property/microblazecore.html#overview

[33] Universal asynchronous receiver/transmitter (uart) for keystone devices ug. [Online]. Available: https://www.ti.com/lit/ug/sprugp1/sprugp1.pdf

[34] Temperature sensor. [Online]. Available: https://www.mathworks.com/help/physmod/simscape/ref/temperaturesensor.html/

[35] Ps-simulink converter. [Online]. Available: https://www.mathworks.com/help/physmod/simscape/ref/pssimulinkconverter.html/

[36] Body temperature. [Online]. Available: https://www.healthlinkbc.ca/medical-tests/hw198785/

[37] What should my heart rate be? [Online]. Available: https://www.medicalnewstoday.com/articles/235710/

[38] Rc5 encryption algorithm. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/73672-rc5-encryption-algorithm/

APPENDICES

APPENDIX A

```c
/*
 * Academic License - for use in teaching, academic research, and meeting
 * course requirements at degree granting institutions only.  Not for
 * government, commercial, or other organizational use.
 *
 * File: Model.c
 *
 * Code generated for Simulink model 'Model'.
 *
 * Model version                  : 1.143
 * Simulink Coder version         : 9.4 (R2020b) 29-Jul-2020
 * C/C++ source code generated on : Mon Jun 14 02:46:56 2021
 *
 * Target selection: ert.tlc
 * Embedded hardware selection: Custom Processor->Custom Processor
 * Code generation objectives:
 *    1. Execution efficiency
 *    2. RAM efficiency
 * Validation result: Not run
 */

#include "Model.h"

/* External inputs (root inport signals with default storage) */
ExtU rtU;

/* External outputs (root outports fed by signals with default storage) */
ExtY rtY;

/* Real-time model */
static RT_MODEL rtM_;
RT_MODEL *const rtM = &rtM_;

/* Model step function */
void Model_step(void)
{
  int32_T i;
  int32_T j;
  int32_T k;
  uint32_T rtb_S[14];
  uint32_T L[4];
  uint32_T A;
  uint32_T B_0;
  uint32_T X_0;
  uint32_T q0;
  uint32_T qY;
  uint32_T qY_0;

  /* MATLAB Function: '<S1>/SIn&#x131;t' */
  for (i = 0; i < 14; i++) {
    rtb_S[i] = 0U;
  }
```

```

A = 0U;
B_0 = 0U;
L[0] = 0U;
L[1] = 0U;
L[2] = 0U;
L[3] = 0U;
for (i = 0; i < 8; i++) {
  j = (int32_T)trunc((-(real_T)i + 7.0) / 2.0);
  q0 = L[j] << 8;
  L[j] = q0 + /*MW:OvSatOk*/ 1U;
  if (q0 + 1U < q0) {
    L[(int32_T)trunc((-(real_T)i + 7.0) / 2.0)] = MAX_uint32_T;
  }
}

rtb_S[0] = 47073U;
for (i = 0; i < 13; i++) {
  q0 = rtb_S[i];
  qY = q0 + /*MW:OvSatOk*/ 40503U;
  if (q0 + 40503U < q0) {
    qY = MAX_uint32_T;
  }

  rtb_S[i + 1] = qY - ((qY >> 16) << 16);
}

i = 0;
j = 0;
for (k = 0; k < 42; k++) {
  q0 = rtb_S[i];
  qY = q0 + /*MW:OvSatOk*/ A;
  if (qY < q0) {
    qY = MAX_uint32_T;
  }

  qY_0 = qY + /*MW:OvSatOk*/ B_0;
  if (qY_0 < qY) {
    qY_0 = MAX_uint32_T;
  }

  X_0 = qY_0 - ((qY_0 >> 16) << 16);
  X_0 = X_0 << 3 | X_0 >> 13;
  rtb_S[i] = X_0 - ((X_0 >> 16) << 16);
  A = rtb_S[i];
  q0 = L[j];
  qY = q0 + /*MW:OvSatOk*/ rtb_S[i];
  if (qY < q0) {
    qY = MAX_uint32_T;
  }

  qY_0 = qY + /*MW:OvSatOk*/ B_0;
  if (qY_0 < qY) {
    qY_0 = MAX_uint32_T;
  }

  X_0 = qY_0 - ((qY_0 >> 16) << 16);
  q0 = rtb_S[i];
  qY = q0 + /*MW:OvSatOk*/ B_0;
```

```
    if (qY < q0) {
      qY = MAX_uint32_T;
    }

    q0 = rtb_S[i];
    qY_0 = q0 + /*MW:OvSatOk*/ B_0;
    if (qY_0 < q0) {
      qY_0 = MAX_uint32_T;
    }

    X_0 = X_0 >> (16U - (qY_0 & 15U)) | X_0 << (qY & 15U);
    L[j] = X_0 - ((X_0 >> 16) << 16);
    B_0 = L[j];
    i = (int32_T)fmod((real_T)i + 1.0, 14.0);
    j = (int32_T)fmod((real_T)j + 1.0, 4.0);
  }

  /* End of MATLAB Function: '<S1>/SIn&#x131;t' */

  /* MATLAB Function: '<S1>/Data-Room Selection' incorporates:
   *  Inport: '<Root>/Data1'
   *  Inport: '<Root>/Data2'
   *  Inport: '<Root>/enableData'
   */
  if (rtU.enableData) {
    B_0 = rtU.Data1;
  } else {
    B_0 = rtU.Data2;
  }

  /* End of MATLAB Function: '<S1>/Data-Room Selection' */

  /* MATLAB Function: '<S1>/Encryption' incorporates:
   *  Inport: '<Root>/enableEncrypt'
   */
  i = (int32_T)(B_0 & 65535U);
  A = i + /*MW:OvSatOk*/ rtb_S[0];
  if (A < (uint32_T)i) {
    A = MAX_uint32_T;
  }

  i = (int32_T)(B_0 << 16 & 65535U);
  B_0 = i + /*MW:OvSatOk*/ rtb_S[1];
  if (B_0 < (uint32_T)i) {
    B_0 = MAX_uint32_T;
  }

  for (i = 0; i < 6; i++) {
    X_0 = (A ^ B_0) >> (16U - (B_0 & 15U)) | (A ^ B_0) << (B_0 & 15U);
    q0 = X_0 - ((X_0 >> 16) << 16);
    j = (int32_T)((i + 1U) << 1);
    qY = rtb_S[j] + /*MW:OvSatOk*/ q0;
    if (qY < q0) {
      qY = MAX_uint32_T;
    }

    A = qY - ((qY >> 16) << 16);
    X_0 = (B_0 ^ A) >> (16U - (A & 15U)) | (B_0 ^ A) << (A & 15U);
    q0 = X_0 - ((X_0 >> 16) << 16);
    qY = rtb_S[j + 1] + /*MW:OvSatOk*/ q0;
    if (qY < q0) {
      qY = MAX_uint32_T;
    }

    B_0 = qY - ((qY >> 16) << 16);
  }

  if (rtU.enableEncrypt) {
    B_0 = B_0 << 16 | A;
  } else {
    B_0 = 0U;
  }

  /* End of MATLAB Function: '<S1>/Encryption' */

  /* MATLAB Function: '<S1>/Decryption' incorporates:
   *  Inport: '<Root>/enableDecyrpt'
   */
  A = B_0 & 65535U;
  B_0 >>= 16;
  for (i = 0; i < 6; i++) {
    qY = B_0 + /*MW:OvSatOk*/ 65536U;
    if (B_0 + 65536U < B_0) {
      qY = MAX_uint32_T;
    }

    qY_0 = qY - /*MW:OvSatOk*/ rtb_S[((6 - i) << 1) + 1];
    if (qY_0 > qY) {
      qY_0 = 0U;
    }

    qY = B_0 + /*MW:OvSatOk*/ 65536U;
    if (B_0 + 65536U < B_0) {
      qY = MAX_uint32_T;
    }

    j = (6 - i) << 1;
    q0 = qY - /*MW:OvSatOk*/ rtb_S[j + 1];
    if (q0 > qY) {
      q0 = 0U;
    }

    X_0 = (q0 - ((q0 >> 16) << 16)) >> (A & 15U) | (qY_0 - ((qY_0 >> 16) << 16))
      << (16U - (A & 15U));
    B_0 = (X_0 - ((X_0 >> 16) << 16)) ^ A;
    qY = A + /*MW:OvSatOk*/ 65536U;
    if (A + 65536U < A) {
      qY = MAX_uint32_T;
    }

    q0 = rtb_S[j];
    qY_0 = qY - /*MW:OvSatOk*/ q0;
    if (qY_0 > qY) {
      qY_0 = 0U;
    }

    qY = A + /*MW:OvSatOk*/ 65536U;
    if (A + 65536U < A) {
      qY = MAX_uint32_T;
    }

    q0 = qY - /*MW:OvSatOk*/ q0;
    if (q0 > qY) {
      q0 = 0U;
    }

    X_0 = (q0 - ((q0 >> 16) << 16)) >> (B_0 & 15U) | (qY_0 - ((qY_0 >> 16) << 16)
      << (16U - (B_0 & 15U));
    A = (X_0 - ((X_0 >> 16) << 16)) ^ B_0;
  }

  if (rtU.enableDecyrpt) {
    qY = B_0 + /*MW:OvSatOk*/ 65536U;
    if (B_0 + 65536U < B_0) {
      qY = MAX_uint32_T;
    }

    qY_0 = qY - /*MW:OvSatOk*/ rtb_S[1];
    if (qY_0 > qY) {
      qY_0 = 0U;
    }

    qY = A + /*MW:OvSatOk*/ 65536U;
    if (A + 65536U < A) {
      qY = MAX_uint32_T;
    }

    q0 = qY - /*MW:OvSatOk*/ rtb_S[0];
    if (q0 > qY) {
      q0 = 0U;
    }

    /* Outport: '<Root>/Result' */
    rtY.Result = (qY_0 - ((qY_0 >> 16) << 16)) << 16 | (q0 - ((q0 >> 16) << 16));
  } else {
    /* Outport: '<Root>/Result' */
    rtY.Result = 0U;
  }

  /* End of MATLAB Function: '<S1>/Decryption' */
}

/* Model initialize function */
void Model_initialize(void)
{
  /* (no initialization code required) */
}

/*
 * File trailer for generated code.
 *
 * [EOF]
 */
```

Listing A.1: Encryption.c

APPENDIX B

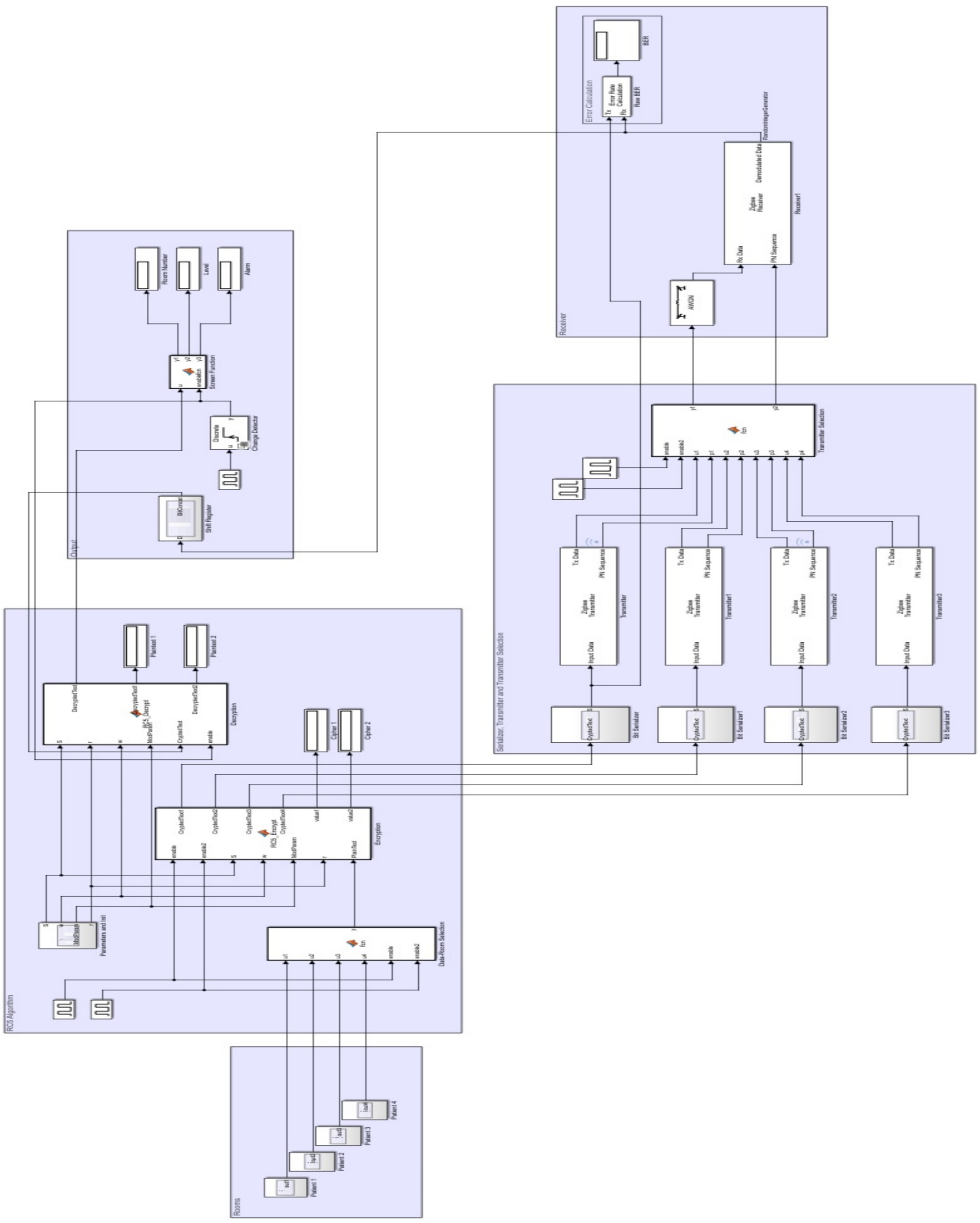


Figure A.1: Simulink Model

APPENDIX C

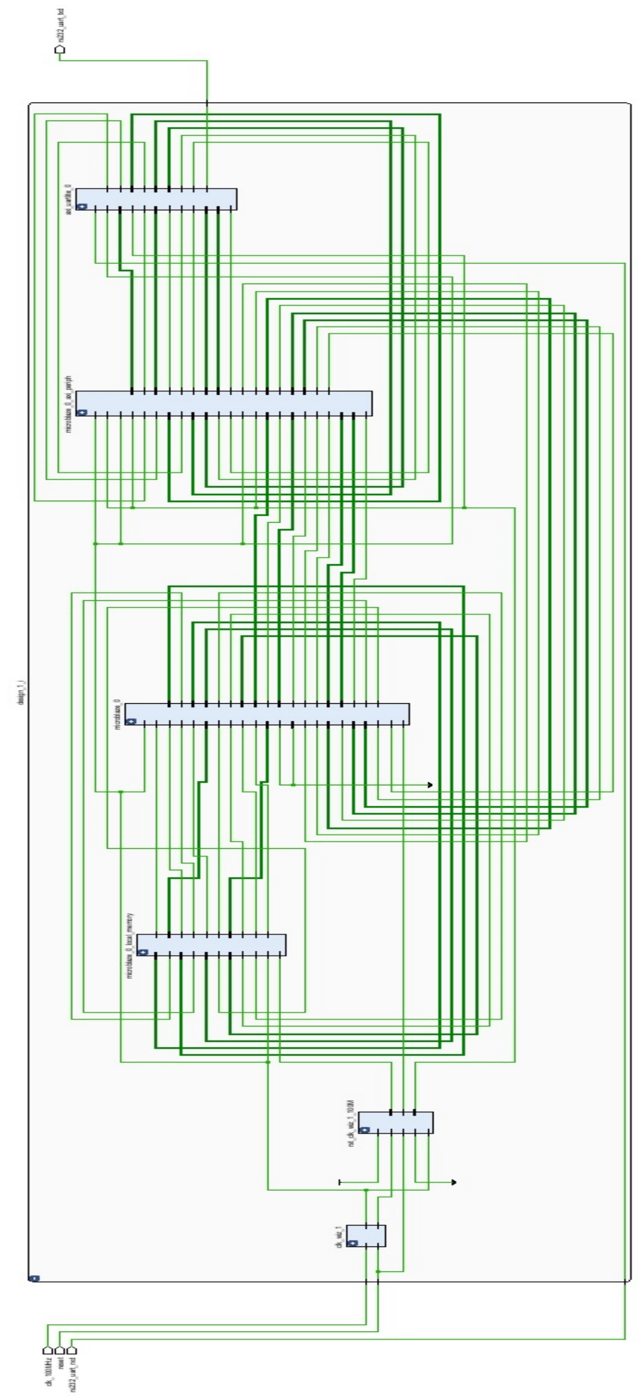


Figure B.1: RTL Schematic

CURRICULUM VITAE

Name Surname: Heval Ronahi Halitoglu

Place and Date of Birth: Van,1997

E-Mail: ronahihltgl@gmail.com

EDUCATION:

- **B.Sc.:** 2015-2021, Istanbul Technical University, Electric Electronic Faculty, Electronic and Communication Department
- **B.Sc.:** 08.2019-02.2020 , "Angel Kanchev" University of Ruse , Electric Electronic Faculty, Electric and Electronics Engineering Department(Exchange)
- **High School:** 2013-2014, Antalya Aldemir Attilla KOnuk Anatolian High School

PROFESSIONAL EXPERIENCE:

- 11.2020-Present, Part Time Employee, TÜBİTAK BİLGEM
- 06.2020-07.2020, Summer Intern, YongaTek, Verification Department
- 07.2019-08.2019, Summer Intern, TÜBİTAK, Integrated Circuit Design and Training Laboratory(TÜTEL)

CURRICULUM VITAE

Name Surname: Oğuzhan TURAN

Place and Date of Birth: İstanbul, 1997

E-Mail: turano15@itu.edu.tr

EDUCATION:

- **B.Sc.:** 2015-2021, Istanbul Technical University, Electric Electronic Faculty, Electronic and Communication Department
- **B.Sc.:** 2018-2019 Fall, Slovak University of Technology Erasmus Programme
- **High School:** 2011-2015, Adnan Menderes Anatolian High School

PROFESSIONAL EXPERIENCE:

- 06.2021-Present, Summer Intern,YONGATEK, Digital Design
- 10.2019-12.2019, Part Time Software Engineer Intern, INTERTECH
- 06.2019-07.2019, Summer Intern, INTERTECH