**CHAPTER 1**

* 1. **INTRODUCTION:**

The exchange of information between two nodes is known as communication. Communication can be of two types. One is control communication that is exchange of control information required for the connection or path establishment. Another is data communication where the data is transformed from one system to other. The channel is defined as the communication path between the two systems.

The connection can be wired or wireless. In wired connections the path is established before the communication and it is stable. Here the communication is divided into two types:

**Synchronous communication:** Synchronous communication requires that the clocks in the transmitting and receiving devices are synchronized – running at the same rate – so the receiver can sample the signal at the same time intervals used by the transmitter. No start or stop bits are required. For this reason “synchronous communication permits more information to be passed over a circuit per unit time.”

**Asynchronous communication:** is transmission of data, generally without the use of an external clock signal, where data can be transmitted intermittently rather than in a steady stream. Any timing required to recover data from the communication symbols is encoded within the symbols. The most significant aspect of asynchronous communications is that data is not transmitted at regular intervals, thus making possible variable bit rate, and that the transmitter and receiver clock generators do not have to be exactly synchronized all the time.

The asynchronous communication generally uses the handshake protocols for communication. The handshake protocol consists of two common steps with other according to the respective protocol. Whenever the system gets the data and to be sent to another system then it sends the request to receiver. The receiver upon getting the request from the sender it wills acknowledge to sender by sending signals. The fig 1.1 shows the handshake protocol communication.

Handshaking is an automated process of negotiation that dynamically sets parameters of a communications channel established between two entities before normal communication over the channel begins. It follows the physical establishment of the channel and precedes normal information transfer. The handshaking process usually takes place in order to establish rules for communication when a computer sets about communicating with a foreign device. When a computer communicates with another device like a modem, printer, or network server, it needs to handshake with it to establish a connection. Handshaking can negotiate parameters that are acceptable to equipment and systems at both ends of the communication channel, including transfer rate, coding alphabet, parity, interrupt procedure, and other protocol or hardware features. Handshaking is a technique of communication between two entities.

Fig1.1: Handshake between two systems.

In asynchronous serial communication the [physical protocol layer](https://en.wikipedia.org/wiki/Physical_protocol_layer), the data blocks are code words of a certain [word length](https://en.wikipedia.org/wiki/Word_length), for example [octets](https://en.wikipedia.org/wiki/Octet_(computing)) ([bytes](https://en.wikipedia.org/wiki/Bytes)) or [ASCII characters](https://en.wikipedia.org/wiki/ASCII_characters), delimited by start bits and stop bits. A variable length space can be inserted between the code words. No bit synchronization signal is required. This is sometimes called [character oriented communication](https://en.wikipedia.org/w/index.php?title=Character_oriented_communication&action=edit&redlink=1). Examples are the [RS-232](https://en.wikipedia.org/wiki/RS-232)C serial standard, and [MNP2](https://en.wikipedia.org/w/index.php?title=MNP2&action=edit&redlink=1) and [V.2](https://en.wikipedia.org/wiki/List_of_ITU-T_V-Series_Recommendations) modems and older.

Asynchronous communication at the [data link layer](https://en.wikipedia.org/wiki/Data_link_layer) or higher protocol layers is known as [statistical multiplexing](https://en.wikipedia.org/wiki/Statistical_multiplexing), for example [asynchronous transfer mode](https://en.wikipedia.org/wiki/Asynchronous_transfer_mode) (ATM). In this case the asynchronously transferred blocks are called [data packets](https://en.wikipedia.org/wiki/Data_packet), for example ATM cells. The opposite is [circuit switched](https://en.wikipedia.org/wiki/Circuit_switched) communication, which provides constant bit rate, for example [ISDN](https://en.wikipedia.org/wiki/ISDN) and [SONET/SDH](https://en.wikipedia.org/wiki/SONET/SDH).

The packets may be encapsulated in a [data frame](https://en.wikipedia.org/wiki/Data_frame), with a [frame synchronization](https://en.wikipedia.org/wiki/Frame_synchronization) bit sequence indicating the start of the frame, and sometimes also a [bit synchronization](https://en.wikipedia.org/wiki/Bit_synchronization) bit sequence, typically 01010101, for identification of the bit transition times. Note that at the physical layer, this is considered as synchronous serial communication. Examples of packet mode data link protocols that can be/are transferred using synchronous serial communication are the [HDLC](https://en.wikipedia.org/wiki/HDLC), [Ethernet](https://en.wikipedia.org/wiki/Ethernet), [PPP](https://en.wikipedia.org/wiki/Point-to-point_protocol) and [USB](https://en.wikipedia.org/wiki/USB) protocols.

**Three way handshake protocol:** Within TCP/IP RFCs, the term "handshake" is most commonly used to reference the TCP three-way handshake. For example, the term "handshake" is not present in RFCs covering FTP or SMTP. One exception is Transport Layer Security, TLS, setup, FTP RFC 4217. In place of the term "handshake", FTP RFC 3659 substitutes the term "conversation" for the passing of commands.

A simple handshaking protocol might only involve the receiver sending a message meaning "I received your last message and I am ready for you to send me another one." A more complex handshaking protocol might allow the sender to ask the receiver if it is ready to receive or for the receiver to reply with a negative acknowledgement meaning "I did not receive your last message correctly, please resend it" (e.g., if the data was corrupted en route).

Handshaking facilitates connecting relatively heterogeneous systems or equipment over a communication channel without the need for human intervention to set parameters.

Establishing a normal TCP connection requires three separate steps as shown in fig 1.2:

Step 1: The first host (Alice) sends the second host (Bob) a "synchronize" (SYN) message with its own sequence number ‘x’,{\displaystyle x} which Bob receives.

Step 2: Bob replies with a synchronize-acknowledgment ([SYN-ACK](https://en.wikipedia.org/wiki/SYN-ACK)) message with its own sequence number ‘y’ {\displaystyle y}and acknowledgement number ‘x+1’{\displaystyle x+1}, which Alice receives.

Step 3: Alice replies with an acknowledgment message with acknowledgement number ‘y+1’{\displaystyle y+1}, which Bob receives and to which he doesn't need to reply. In this setup, the synchronize messages act as service requests from one server to the other, while the acknowledgement messages return to the requesting server to let it know the message was received.

One of the most important factors of three-way handshake is that, in order to exchange the starting sequence number the two sides plan to use, the client first sends a segment with its own initial sequence number{\displaystyle x}, then the server responds by sending a segment with its own sequence number {\displaystyle y}and the acknowledgement number {\displaystyle x+1}, and finally the client responds by sending a segment with acknowledgement number{\displaystyle y+1}.

The reason for the client and server not using the default sequence number such as 0 for establishing connection is to protect against two incarnations of the same connection reusing the same sequence number too soon, which means a segment from an earlier incarnation of a connection might interfere with a later incarnation of the connection.

Sender Receiver

Fig 1.2: 3-way handshake protocol

**Handshake protocols:** There are generally 5 types of handshake protocoland they are as follows:

1. Bundled data protocol: The term bundled-data refers to a situation where the data signals use normal Boolean levels to encode information, and where separate request and acknowledge wires are bundled with the data signals, The term ‘bundled-data’ hints at the timing relationship between the data signals and the handshake signals, whereas the term ‘single-rail’ hints at the use of one wire to carry one bit of data.
2. 4-phase dual rail protocol: The 4-phase dual-rail protocol encodes the request signal into the data signals using two wires per bit of information that has to be communicated. In essence it is a 4-phase protocol using two request wires per bit of information d; one wire d:t is used for signaling a logic 1 (or true), and another wire d:f is used for signaling logic 0 (or false). When observing a 1-bit channel one will see a sequence of 4-phase handshakes where the participating “request” signal in any handshake cycle can be either d:t or d:f . This protocol is very robust; two parties can communicate reliably regardless of delays in the wires connecting the two parties – the protocol is delay-insensitive.
3. 2-phase dual rail protocol: The 2-phase dual-rail protocol also uses 2 wires fd:t; d:f g per bit, but the information is encoded as transitions (events) as explained previously. On an N-bit channel a new codeword is received when exactly one wire in each of the N wire pairs has made a transition. There is no empty value; a valid message is acknowledged and followed by another message that is acknowledged.
4. 2-phase handshake protocol: Here when sender wants to send the data the req line is made high and receiver will raise the ack and the data transfer is done. For the next communication the req line is made low for data transfer. It is faster of all protocol used. Fig 1.3 depicts this protocol.
5. 4-phase handshake protocol: When the sender has the data to send it raises the req line and receiver will sense this and raises the ack and data transfer is done. After the completion of communication both req and ack are made 0. It is known as the return to zero protocol. Fig 1.4 represents this protocol.

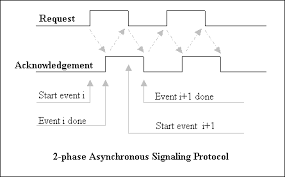


Fig 1.3: 2-phase handshake protocol

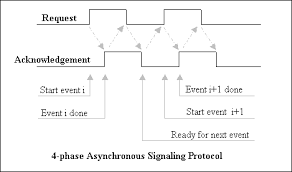


Fig 1.4: 4-phase handshake protocol

**1.4 Literature survey:**

1. Vincent Wing-Yun Sit, Chiu-Sing Choy, and Cheong-Fat Chan, proposes a theory to use a Four-Phase Handshaking Asynchronous Static RAM Design for Self-Timed Systems. The motivation of designing asynchronous memory arises from the recent development of asynchronous processors. As different from the conventional design, the proposed asynchronous static RAM can 1) communicate with other asynchronous systems based on a four-phase handshaking control protocol and 2) generate the read/write completion signals with increased average speed by the variable bit-line load concept. The techniques investigated include 1) dual-rail voltage sensing completion detection for read operation and 2) multiple delays completion generation for write operation. In this paper, the performances of these techniques are evaluated for 1-Mb memory with four regions of bit-line segmentation. The simulated and measured results are presented and compared.
2. The Family of 4-phase Latch Protocols is reviewed by Graham Birtwistle and Kenneth S. Stevens with the respective VLSI designs. Standard micro-pipelines use simple two-phase control circuits. The latches employed on AMULET1 are level sensitive, so two- to four-phase converters are required in each latch controller. To avoid this overhead an investigation has been carried out into four-phase micro-pipeline control circuits; this has thrown up several design issues relating to cost, performance and safety, and forms a useful illustration of asynchronous design techniques. A complete family of untimed asynchronous 4-phase pipeline protocols is derived and characterized. This family contains all untimed protocols where data becomes valid before the request signal rises. Starting with a specification of the most parallel such protocol, rules are provided for concurrency reduction to systematically generate the family of all 137 related protocols that can be pipelined. Graphical and textual nomenclatures are developed to represent protocol properties and behaviors. The protocols are categorized according to their behaviors when composed into linear and structured parallel pipelines. Six basic categories emerge, along with several properties such as a single state that determines whether a protocol is fully or half buffered. When equivalence classes are calculated for parallel pipeline behaviors they are dominated by 15 shapes (all of which are delay-insensitive) which are related by a simple lattice. Several published circuits are shown to map to 16 of our 137 family members. This work enhances the understanding of handshake protocols, their properties, and relationships between different implementations in terms of concurrency and behavioral properties.
3. Tradeoffs between RTO and RTZ in Asynchronous Design by Matheus T. Moreira, Julian J. H. Pontes, Ney L. V. Calazans compares both the protocols in the networ scenario. Classically, quasi-delay-insensitive asynchronous circuits based on weak-conditioned half-buffer employ the return- to-zero, 4-phase handshake protocol. This work scrutinizes the alternative return-to-one protocol and analyzes the effects of using it in practical circuits. A pipelined shift and add multiplier serves as case study. Return-to-one and return-to-zero versions of the circuit provide ground for extensive comparison. Experimental results point to reductions in static power and in forward propagation delay of up to 35% and 12%, respectively, when using return-to-one. Also, results indicate that mixing return-to-zero and return-to-one leads to dynamic power savings.
4. Analysis of the Experimental Results for Quantum Key Distribution Cryptography in IEEE 802.11 Networks by Xu Huang, Shirantha Wijesekera, Dharmendra Sharma proposes the dual rail protocols for the wireless networks. This paper presents a model of integration of the Three-stage quantum cryptography protocol and its variants into the IEEE 802.11 wireless communication standard. The three-stage protocol is introduced during the quantum handshake as a replacement to the BB’84 protocol. Compared to the quantum handshake using BB’84, integrating the three-stage protocol offers several benefits such as enhanced data rates, fewer steps needed to establish the final key, longer distances, and increased number of photons that can be used during the transmission process. In addition, this paper presents a multi-agent software approach to implement the quantum handshake using the three-stage protocol.
5. Joint Time and Spatial Reuse Handshake Protocol for Underwater Acoustic Communication Networks by Roee Diamant, Lutz Lampe. In most existing handshake-based collision avoidance (CA) protocols, nodes in the communication range of the transmitter or the receiver are kept silent during an ongoing communication session (CS). In underwater acoustic communication (UWAC), this restriction results in low throughput and long transmission delay. In this paper, we utilize the long propagation delay in the underwater acoustic channel and the (possible) sparsity of the network topology, and formalize conditions for which a node can transmit even when it is located within the communication range of a node participating in a CS. We consider these conditions as design constraints and present a distributed CA handshake-based protocol, which, by jointly applying spatial and time reuse techniques, greatly improves channel utilization. Our simulation results show that our protocol outperforms existing handshake-based protocols in terms of throughput and transmission delay. These gains come at the price of some reduction in fairness in resource allocation.
6. Return-to-One DIMS Logic on 4-phase m-of-n Asynchronous Circuits explained by Matheus T. Moreira, Ricardo A. Guazzelli, Ney L. V. Calazans. Asynchronous design techniques are gaining attention in the scientific community for their ability to cope with current technologies’ problems that the synchronous paradigm may fail to cope with. In fact, fully synchronous SoCs may soon become unfeasible to build. Among multiple asynchronous design styles, the quasi delay insensitive (QDI) stands out for its robustness to delay variations. When coupled to DI codes like m-of-n and to four-phase handshake protocols, the QDI style produces the dominant asynchronous template currently in use. This paper evaluates the use of the Return-to-One 4-phase handshake protocol on Delay- Insensitive Minterm Synthesis (DIMS) logic blocks. Results point that this protocol leads to significant reductions on power consumption when compared to classic Return-to-Zero protocols. No extra hardware is required by the evaluated protocol, as the only required modification is that OR gates are replaced by AND gates, adding no extra delay to the resulting circuit.

**1.3 Objective:**

The self study includes the overall survey of the 4-phase handshake protocol and with the comparison of 2-phase handshake protocol. It covers the characteristics and applications of the 4-phase protocol. The working of 4-phase handshake protocol is done using socket communication and with the send and recv mechanism.

**CHAPTER 2**

**SYSTEM SPECIFICATION**

Working O.S.: Fedora

Language used: C programming

Tools : Sockets

**CHAPTER 3**

**DESIGN**

The system design includes two phases:

1. **Signalling**: Asynchronous circuits represent a class of circuits which are not controlled by a global clock but by the data themselves. In fact, an asynchronous circuit is composed of individual modules which communicate to each other by means of point-to-point communication channels. Therefore, a given module becomes active when it senses the presence of incoming data. It then computes them and sends the result to the output channels. Communications through channels are governed by a protocol which requires a bi-directional signalling between senders and receivers (request and acknowledge). They are called Handshaking protocols which are the basis of the sequencing rules of asynchronous circuits. As presented above, the implementation of a four-phase handshaking protocol requires sensing the presence of data in phase 1 and phase 3. In order to do so, dedicated logic and special encoding are necessary for sensing data validity/invalidity and for generating the acknowledgement signal.
2. Data/ Request encoding: Considering that one bit has to be transferred through a channel using the four phase protocol, one bit has to encode three different values: invalid, valid at ‘1’, valid at ‘0’. Two wires (A0, A1) are then required to encode the three states. This technique is called dual-rail encoding (table1).
3. Acknowledge / Completion signal generation: The acknowledgement signal is generated by taking advantage of the data-encoding. A Nor gate is usually used to sense the dual-rail encoding output for generating the completion signal. Dual rail encoding is easily extended to N rails. It is called 1-of-N encoding. This encoding data scheme is useful to reduce the number of electrical transitions involved in a given computation which reduces the power consumption. For the sake of DPA resistance, 1-of-N encoding ensures that the same number of transitions is required to encode the values 0 to N-1.
4. **Balanced data path:** As an example, consider the xor function, generally a dual-rail xor gate implementation. Every computations of this dual-rail xor gate involve a fixed and constant number of transitions regardless of the data values. Hence, the opportunity to have data independent power consumption i.e. not correlated to the processed data is exactly the goal to achieve for DPA resistant chip. The Muller gate (C-element) generates an up-transition when up-transitions occur at all its inputs, and generates a down-transition when down-transitions occur at all its inputs. The Muller C-element’s truth table and symbol are however, the QDI implementation of a function is notalways balanced. In such cases, the gate structure ismodified to ensure that all data paths and control paths arebalanced and do involve a constant number of transitionsasynchronous logic permit to achieve a very high level ofDPA resistance. However, it also showed that place androute steps are unbalancing some paths, thus creatingsources of leakage. Let’s now precisely analyze this pointusing a formal approach.

**CHAPTER 4**

**IMPLEMENTATION**

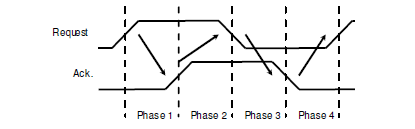


Fig 4.1: 4-phase handshake protocol

The four phases of the protocol can be depicted as:

Phase 1: Sender wants to transfer data / enable to receiver Raises req

Phase 2: Receiver detects rise in req, raises ack

Phase 3: Sender detects rise in ack, lowers req

Phase 4: Receiver detects low req, lowers ack

In phase 1, the sender will raise the req line whenever the data is ready to be sent. On sensing the raise in req, the respective receiver will raise the ack line. This is the second phase. Only when sender senses the raise in the ack line of receiver then the data will be sent across the network. The receiver will receive the data. After the receiver successfully gets all the data it will lower its ack line to indicate the successful transmission. This is the third phase. The fourth and the last phase is sender will lower its req line. And the data transfer will be over.

The 4-phase handshake protocol makes sure that both req and ack are low at the end of the transmission of data. That is the reason why the 4-phase handshake protocol is called as Return To Zero (RTZ) protocol.

This can be implemented using semaphores, inter process communication, pipes and sockets.

Source code for 4-phase handshake protocol is

At sender:

o1.send(D)

{

while(ack) wait(A);

data=D;

sync();

req=1;

R.signal();

while(!ack) wait(A);

req=0;

R.signal();

}

At receiver:

o2.recv()

{

while(!req) wait(R);

D=data;

ack=1;

A.signal();

while(req) wait(R);

ack=0;

A.signal();

return D;

}

**CHAPTER 5**

**RESULT AND ANALYSIS**

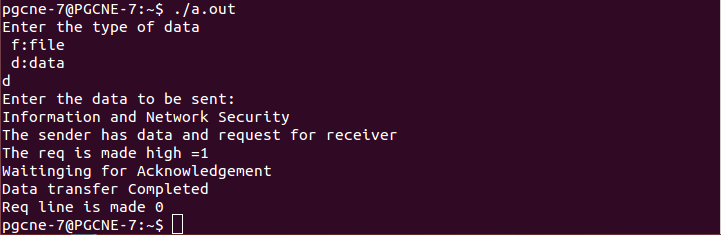


Fig 5.1: Output at Sender end

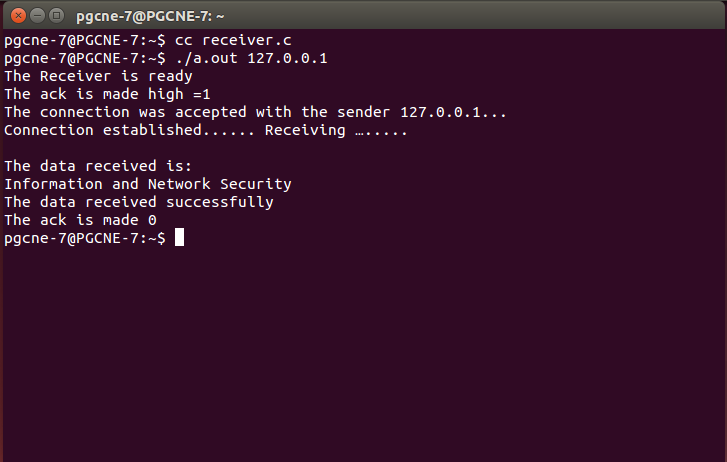


Fig 5.2: Output at Receiver end

The analyzation of the result can be summarized into the following:

* Connection establishes between sender and receiver.
* Sender sends the data only when ACK of receiver is set to 1.
* After the data transfer both REQ and ACK will be set to 0.
* Slow when compared to 2-phase protocol.

**CHAPTER 6**

**CONCLUSION**

The self-study report concentrates on the survey of 4-phase handshake protocol and the comparison with those of other handshake protocols. This report depicts the synchronization provided by the protocol in the asynchronous transmission. It specifies the advantages of 4-phase handshake protocol over the 2-phase handshake protocol. The implementation of the 4-phase handshake protocol is done using the socket communication in Fedora platform. The future study can be carried on the time consumption of the protocol and the power issues can be addressed.

**REFERENCES**

[1]ECE620\_special\_topics\_asynchronous\_circuits\_part2\_pdf

[2] https://en.wikipedia.org/wiki/Handshaking

[3]www.areata.org/.../2004\_Modeling\_and\_synthesis\_of\_multi-rail\_multi-protocol\_QD

[4]citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.124.818&rep=rep1.pdf

[5]  DPA on quasi delay insensitive asynchronous circuits: formalization and improvement by G. F. Bouesse; M. Renaudin; S. Dumont; F. Germain Design, Automation and Test in EuropeYear: 2005 Pages: 424 - 429 Vol. 1, DOI: 10.1109/DATE.2005.124

[6]www.mi.parisdescartes.fr/~mea/cours/DU/TLS\_SSL\_DUsec.pdf

[7]https://www.ics.forth.gr/carv/async/demo/literature/handshakeprotocols.pdf

[8] Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE’05)1530-1591/05 IEEE

[9] IEEE journal of solid-state circuits, vol. 34, no. 1, january 1999.

[10] Quantum Cryptography Based Key Distribution in IEEE 802.11 Networks –Wijesekera, Xu Huang, Dharmendra Sharma 2011 IEEE

[11] 2011 Ninth IEEE/IFIP International Conference on Embedded and Ubiquitous Computing