REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE DJILLALI LIABES
DE SIDI BEL ABBES
FACULTÉ DES SCIENCES DE L'INGÉNIEUR
DÉPARTEMENT D'ÉLECTRONIQUE

Mémoire présenté par : **Mr Soufi Abdelkader**

Pour Obtenir le Diplôme de Magister en Electronique

Option : ***« Signal et Télécommunications »***

Thème

Implémentation d'un égaliseur de canal d'un système OFDM sans file

Mémoire soutenu le :

Devant le jury composé de :

Président : Mr DJEBBARI Ali : Professeur Univ SBA

Membre : Mr NAOUM Rafah : Professeur Univ SBA

Membre : Mr ELAHMAR Sid Ahmed : Maître de conférences Univ SBA

Membre : Mr DJEBBAR Ahmed Bouzidi : Chargé de cours Univ SBA

Encadreur : Mr BOUZIANI Merahi : Maître de conférences Univ SBA

Année 2008

CONTENTS

LIST OF FIGURE

Abbreviation

4G	Four Generation
ABEL	Advanced Boolean Expression Language
AP	Access points
ASIC	Application Specific Integrated Circuit
ASSP	Application Specific Standard Product
ATE	Automatic Test Equipment
BSS	Basic Service Set
CDMA	Code Division Multiple Access
CDMA	Code Division Multiple Access
CLB	Configurable Logic Block
CPLD	Complex Programmable Logic Device
CSMA/CD	Carrier Sense Multiple Access network with Collision Detection
DCF	distributed coordination function
DES	Data Encryption Standard
DFT	Discrete Fourier Transform
DRAM	Dynamic Random Access Memory
DS	Distribution system
DSL	Digital Subscriber Line
DSP	Digital Signal Processor
DSSS	Direct Sequence Spread Spectrum
DTV	Digital Television
DVB	Digital Video Broadcasting
ECS	Schematic Editor
EDA	Electronic Design Automation
EDIF	Electronic Digital Interchange Format
EPROM	Erasable Programmable Read Only Memory
ESS	Extended Service Set
FAT	File Allocation Table
FEC	Forward Error Correction
FHSS	Frequency Hopping Spread Spectrum
FIFO	First In First Out
FIR	Finite Impulse Response (Filter) Fmax Frequency Maximum
FPGA	Field Programmable Gate Array
FSM	Finite State Machine
GPS	Geo-stationary Positioning System
GSM	Global System for Mobile communications
GUI	Graphical User Interface
HDTV	High Definition Television

HDTV	High Definition Television
I/O	Inputs and Outputs
IBSS	Independent BSS
IP	Intellectual Property
IRL	Internet Reconfigurable Logic
ISI	Inter-Symbol Interference
ISP	In-System Programming
JTAG	Joint Test Advisory Group
LFSR	linear feedback shift register
LLC	logical link control.
LSB	Least Significant Bit
LUT	Look Up Table
MAC	Media Access Control .
MP3	MPEG Layer III Audio Coding
MPDUs	MAC protocol data units
MPEG	Motion Picture Experts Group
MSB	Most Significant Bit
MSDU	MAC service data unit
NAV	Network Allocation Vector .
NRE	Non-Recurring Engineering (Cost)
OFDM :	Orthogonal Frequency Division Multiplexing
OSI	Open system Interconnect
PAL	Programmable Array Logic device
PCB	Printed Circuit Board
PCF	Point Coordination Function
PCI	Peripheral Component Interconnect
PCMCIA	Personal Computer Memory Card International Association
PCS	Personnel Communications System
PLA	Programmable Logic Array
PLCP	Physical Layer Convergence Procedure
PLD	Programmable Logic Device
PMD	Physical Medium Dependent
PROM	Programmable Read Only Memory
QAM	Square Amplitude Modulation
QML	Qualified Manufacturers Listing
QOS	Quality Of Service
QPRO	QML Performance Reliability of supply Off- the-shelf ASIC
RAM	Random Access Memory
RF	radio frequencies
ROM	Read Only Memory

SFN	Single Frequency Network
SPLD	Simple Programmable Logic Device
SRAM	Static Random Access Memory SRL16 Shift Register LUT
STA	Wireless medium and Station.
TPD	Time of Propagation Delay through the device
UMTS	Universal Mobile Telecommunications System
UMTS	Universal Mobile Telecommunications System
VHDL	VHISC High Level Description Language
VHSIC	Very High Speed Integrated Circuit
VSS	Visual Software Solutions
WCDMA	Wide-band Code Division Multiple Access
WLAN	Wireless Local Access Network
WLAN	Wireless LAN
XST	Xilinx Synthesis Technology

Acronyme

AP	Access points
BSS	Basic Service Set
CDMA	Code Division Multiple Access
CSMA/CD	Carrier Sense Multiple Access network with Collision Detection
DCF	distributed coordination function
DFT	Discrete Fourier Transform
DS	Distribution system
DSSS	Direct Sequence Spread Spectrum
DVB	Digital Video Broadcasting
ESS	Extended Service Set
FEC	Forward Error Correction
FHSS	Frequency Hopping Spread Spectrum
GSM	Global System for Mobile communications
HDTV	High Definition Television
IBSS	Independent BSS
ISI	Inter-Symbol Interference
LFSR	linear feedback shift register
LLC	logical link control.
MAC	Media Access Control .
MPDUs	MAC protocol data units
MSDU	MAC service data unit
NAV	Network Allocation Vector .
OFDM :	Orthogonal Frequency Division Multiplexing
OSI	Open system Interconnect
PCF	Point Coordination Function
PLCP	Physical Layer Convergence Procedure
PMD	Physical Medium Dependent
QAM	Square Amplitude Modulation
QOS	Quality Of Service
RF	radio frequencies
SFN	Single Frequency Network
STA	Wireless medium and Station.
UMTS	Universal Mobile Telecommunications System
WCDMA	Wide-band Code Division Multiple Access
WLAN	Wireless LAN

GLOSSARY OF TERMS

ABEL- Advanced Boolean Expression Language, low-level language for design entry, from Data I/O.

AIM – Advanced Interconnect Matrix in the CoolRunner II CPLD that provides the flexible interconnection between the PLA function blocks.

Antifuse - A small circuit element that can be irreversibly changed from being non-conducting to being conducting with ~100 Ohm. Anti-fusebased FPGAs are thus non-volatile and can be programmed only once (see OTP).

AQL- Acceptable Quality Level. The relative number of devices, expressed in parts-per-million (ppm), that might not meet specification or be defective. Typical values are around 10 ppm.

ASIC- Applications-Specific Integrated Circuit, also called a gate array Asynchronous Logic that is not synchronised by a clock. Asynchronous designs can be faster than synchronous ones, but are more sensitive to parametric changes, and are thus less robust.

ASSP- Application-Specific Standard Product. Type of high-integration chip or chipset ASIC that is designed for a common yet specific application.

ATM- Asynchronous Transfer Mode. A very-high-speed (megahertz to gigahertz) connection-oriented bit-serial protocol for transmitting data and real-time voice and video in fixed-length packets (48-byte payload, 5-byte header).

Back annotation- Automatically attaching timing values to the entered design format after the design has been placed and routed in an FPGA.

Behavioral language- Top-down description from an even higher level than VHDL.

Block RAM- A block of 2k to 4k bits of RAM inside an FPGA. Dual-port and synchronous operation are desirable.

CAD Computer- Aided Design, using computers to design products.

CAE Computer- Aided Engineering, analyses designs created on a computer.

CLB- Configurable Logic Block. Xilinx-specific name for a block of logic surrounded by routing resources. A CLB contains 2 or 4 look-up-tables (function generators) plus 2 or 4 flip-flops.

CMOS- Complementary Metal-Oxide-Silicon. Dominant technology for logic and memory. Has replaced the older bipolar TTL technology in most applications except very fast ones. CMOS offers lower power consumption and smaller chip size compared to bipolar and now meets or even beats TTL speed.

Compiler- software that converts a higher-language description into a lower-level representation. For FPGAs : the complete partition, place & route process.

Configuration- The internally stored file that controls the FPGA so that it performs the desired logic function. Also: The act of loading an FPGA with that file.

Constraints- Performance requirements imposed on the design, usually in the form of max allowable delay, or required operating frequency.

CoolCLOCK – Combination of the clock divider and clock doubler functions in CoolRunner II to further reduce power consumption associated with high speed clocked in internal device networks.

CPLD- Complex Programmable Logic Device, synonymous with EPLD. PAL-derived programmable logic devices that implement logic as sum-ofproducts driving macrocells. CPLDs are known to have short pin-to-pin delays, and can accept wide inputs, but have relatively high power consumption and fewer flip-flops, compared to FPGAs.

CUPL- Compiler Universal for Programmable Logic, CPLD development tool available from Logical Devices.

DataGATE – A function within CoolRunner II to block free running input signals, effectively blocking controlled switching signals so they do not drive internal chip capacitances to further reduce power consumption. Can be selected on all inputs.

Input Hysteresis - Input hysteresis provides designers with a tool to minimize external components. Whether using the inputs to create a simple clock source, or reducing the need for external buffers to sharpen up a slow or noisy input signal. Function found in CoolRunner II CPLDs (may also be referred to as Schmitt Trigger inputs in the text).

DCM- Digital Clock Manager, Provides zero-delay clock buffering, precise phase control and precise frequency generation on Xilinx Virtex II FPGAs

DCI – Digitally Controlled Impedance in the Virtex-II solution dynamically eliminates drive strength variation due to process, temperature, and voltage fluctuation. DCI uses two external high-precision resistors to incorporate

equivalent input and output impedance internally for hundreds of I/O pins.

Debugging- The process of finding and eliminating functional errors in software and hardware.

Density- Amount of logic in a device, often used to mean capacity. Usually measured in gates, but for FPGAs better expressed in Logic Cells, each consisting of a 4-input look-up table and a flip-flop.

DLL- Delay Locked Loop, A digital circuit used to perform clock management functions on and off-chip.

DRAM- Dynamic Random Access Memory. A low-cost\read-write memory where data is stored on capacitors and must be refreshed periodically. DRAMs are usually addressed by a sequence of twoaddresses, row address and column address, which makes them slower and more difficult to use than SRAMs.

DSP- Digital Signal Processing. The manipulation of analog data that has been sampled and converted into a digital representation. Examples are: filtering, convolution, Fast-Fourier-Transform, etc.

EAB- Embedded Array Block. Altera name for Block RAM in FLEX10K.

EDIF- Electronic Data Interchange Format. Industry-standard for specifying a logic design in text (ASCII) form.

EPLD- Erasable Programmable Logic Devices, synonymous with CPLDs. PAL-derived programmable logic devices that implement logic as sum-of-products driving macrocells. EPLDs are known to have short pin-to-pin delays, and can accept wide inputs, but have relatively high power consumption and fewer flip-flops than FPGAs.

Embedded RAM- Read-write memory stored inside a logic device. Avoids the delay and additional connections of an external RAM.

ESD- Electro-Static Discharge. High-voltage discharge can rupture the input transistor gate oxide. ESD-protection diodes divert the current to the supply leads.

5-volt tolerant- Characteristic of the input or I/O pin of a 3.3 V device that allows this pin to be driven to 5 V without any excessive input current or device breakdown. Very desirable feature.

FIFO- First-In-First-Out memory, where data is stored in the incoming sequence, and is read out in the same sequence. Input and output can be asynchronous to each other. A FIFO needs no external addresses, although all modern FIFOs are implemented internally with RAMs driven by circular read and write counters.

FIT- Failure In Time. Describes the number of device failures statistically expected for a certain number of device-hours. Expressed as failures per one billion device hours. Device temperature must be specified. MTBF can be calculated from FIT.

Flash- Non-volatile programmable technology, an alternative to Electrically-Erasable Programmable Read-Only Memory (EEPROM) technology. The memory content can be erased by an electrical signal. This allows in-system programmability and eliminates the need for ultraviolet light and quartz windows in the package.

Flip-flop- Single-bit storage cell that samples its Data input at the active (rising or falling) clock edge, and then presents the new state on its Q output after that clock edge, holding it there until after the next active clock edge.

Floor planning- Method of manually assigning specific parts of the design to specific chip locations. Can achieve faster compilation, better utilisation, and higher performance.

Footprint- The printed-circuit pattern that accepts a device and connects its pins appropriately. Footprint-compatible devices can be interchanged without modifying the pc-board.

FPGA- Field Programmable Gate Array. An integrated circuit that contains configurable (programmable) logic blocks and configurable (programmable) interconnect between these blocks.

Function Generator- Also called look-up-table (LUT), with N-inputs and one output. Can implement any logic function of its N-inputs. N is between 2 and 6, most popular are 4-input function generators.

GAL- Generic Array Logic. Lattice name for a variation on PALs Gate Smallest logic element with several inputs and one output. AND gate output is High when all inputs are High. OR gate output is High when at least one input is High. A 2-input NAND gate is used as the measurement unit for gate array complexity.

Gate Array- ASIC where transistors are pre-defined, and only the interconnect pattern is customised for the individual application.

GTL- Gunning Transceiver Logic, is a high speed, low power back-plane standard.

GUI- Graphic User Interface. The way of representing the computer output on the screen as graphics, pictures, icons and windows. Pioneered by Xerox and the Macintosh, now universally adopted, e.g by Windows95.

HDL- Hardware Description Language.

Hierarchical design- Design description in multiple layers, from the highest (overview) to the lowest (circuit details). Alternative: Flat design, where everything is described at the same level of detail. Incremental design Making small design changes while maintaining most of the layout and routing.

Interconnect- Metal lines and programmable switches that connect signals between logic blocks and between logic blocks and the I/O.

IOB or I/O- Input/Output Block. Logic block with features specialised for interfacing with the pc-board.

ISO9000- An internationally recognised quality standard. Xilinx is certified to ISO9001 and ISO9002.

IP- Intellectual Property. In the legal sense: Patents, copyrights and trade secrets. In integrated circuits: pre-defined large functions, called cores, that help the user complete a large design faster.

ISP- In-System Programmable device. A programmable logic device that can be programmed after it has been connected to (soldered into) the system pc-board. Although all SRAM-based FPGAs are naturally ISP, this term is only used with certain CPLDs, to distinguish them from the older CPLDs that must be programmed in programming equipment.

JTAG- Joint Test Action Group. Older name for IEEE 1149.1 boundary scan, a method to test pc-boards and also ICs.

LogiBLOX™ - Formerly called X-Blox. Library of logic modules, often with user-definable parameters, like data width. (Very similar to LPM).

Logic Cell- Metric for FPGA density. One logic cell is one 4-input lookup table plus one flip-flop.

LPM- Library of Parameterised Modules, library of logic modules, often with user-definable parameters, like data width. Very similar to LogiBlox.

LUT- Look-Up-Table, also called function generator with N inputs and one output. Can implement any logic function of its N inputs. N is between 2 and 6, most popular are 4-input LUTs.

Macrocell- The logic cell in a sum-of-products CPLD or PAL/GAL.

Mapping- Process of assigning portions of the logic design to the physical chip resources (CLBs). With FPGAs, mapping is a more demanding and more important process than with gate arrays.

MTBF- Mean Time Between Failure. The statistically relevant up-time between equipment failure. See also FIT.

Netlist- Textual description of logic and interconnects. See XNF and EDIF.

NRE- Non-Recurring Engineering charges. Start-up cost for the creation of an ASIC, gate array, or HardWire™. Pays for lay-out, masks, and test development. FPGAs and CPLDs do not require NRE.

Optimisation- Design change to improve performance. See also: Synthesis.

OTP- One-Time Programmable. Irreversible method of programming logic or memory. Fuses and anti-fuses are inherently OTP. EPROMs and EPROM-based CPLDs are OTP if their plastic package blocks the ultraviolet light needed to erase the stored data or configuration.

PAL- Programmable Array Logic. Oldest practical form of programmable logic, implemented a sum-of-products plus optional output flip-flops.

Partitioning- In FPGAs, the process of dividing the logic into subfunctions that can later be placed into individual CLBs. Partitioning precedes placement.

PCI- Peripheral Component Interface. Synchronous bus standard characterised by short range, light loading, low cost, and high performance. 33-MHz PCI can support data byte transfers of up to 132 megabytes per second on 36 parallel data lines (including parity) and a common clock. There is also a new 66-MHz standard.

PCMCIA- Personal Computer Memory Card Interface Association, also: People Can't Memorise Computer Industry Acronyms. Physical and electrical standard for small plug-in boards for portable computers.

Pin-locking- Rigidly defining and maintaining the functionality and timing requirements of device pins while the internal logic is still being designed or modified. Pin-locking has become important, since circuit-boardfabrication times are longer than PLD design implementation times.

PIP- Programmable Interconnect Point. In Xilinx FPGAs, a point where two signal lines can be connected, as determined by the device configuration.

Placement- In FPGAs, the process of assigning specific parts of the design to specific locations (CLBs) on the chip. Usually done automatically.

PLA – Programmable Logic Array. The first and most flexible programmable logic configuration with two programmable planes providing any combination of 'AND' and 'OR' gates and sharing of AND terms across multiple OR's. This architecture is implemented in the CoolRunner and CoolRunner II devices.

PLD- Programmable Logic Device. Most generic name for all programmable logic: PALs, CPLDs, and FPGAs.

QML- Qualified Manufacturing Line. For example, ISO9000.

Routing- The interconnection, or the process of creating the desired interconnection, of logic cells to make them perform the desired function. Routing follows after partitioning and placement.

Schematic- Graphic representation of a logic design in the form of interconnected gates, flip-flops and larger blocks. Older and more visually intuitive alternative to the increasingly more popular equationbased or high-level language textual description of a logic design.

Select-RAM- Xilinx-specific name for a small RAM (usually 16 bits), implemented in a LUT.

Simulation- Computer modelling of logic and (sometimes) timing behaviour of logic driven by simulation inputs (stimuli, or vectors).

SPROM- Serial Programmable Read-Only Memory. Non-volatile memory device that can store the FPGA configuration bitstream. The SPROM has a built-in address counter, receives a clock and outputs a serial bitstream.

SRAM- Static Random Access Memory. Read-write memory with data stored in latches. Faster than DRAM and with simpler timing requirements, but smaller in size and about 4-times more expensive than DRAM of the same capacity.

SRL16 - Shift Register LUT, an alternative mode of operation for every function generator (look up table) which are part of every CLB in Virtex and Spartan FPGAs. This mode increases the number of flip-flops by 16. Adding flip-flops enables fast pipelining - ideal in DSP applications.

Static timing- Detailed description of on-chip logic and interconnect delays.

Sub-micron- The smallest feature size is usually expressed in micron (µ= millionth of a meter, or thousandth of a millimetre) The state of the art is moving from 0.35µ to 0.25µ, and may soon reach 0.18µ. The wavelength of visible light is 0.4 to 0.8µ. 1 mil = 25.4µ.

Synchronous- Circuitry that changes state only in response to a common clock, as opposed to asynchronous circuitry that responds to a multitude of derived signals. Synchronous circuits are easier to design, debug, and modify, and tolerate parameter changes and speed upgrades better than asynchronous circuits

Synthesis- Optimisation process of adapting a logic design to the logic resources available on the chip, like look-up-tables, Longline, dedicated carry. Synthesis precedes Mapping.

SystemI/O- technology incorporated in Virtex II FPGAs that uses the SelectI/O-Ultra™ blocks to provide the fastest and most flexible electrical interfaces available. Each user I/O pin is individually programmable for any of the 19 single-ended I/O standards or six differential I/O standards, including LVDS, SSTL, HSTL II, and GTL+. SelectI/O-Ultra technology delivers 840 Mbps LVDS performance using dedicated Double Data Rate (DDR) registers.

TBUFs- Buffers with a 3-state option, where the output can be made inactive. Used for multiplexing different data sources onto a common bus. The pull-down-only option can use the bus as a wired AND function.

Timing- Relating to delays, performance, or speed.

Timing driven- A design or layout method that takes performance requirements into consideration.

UART- Universal Asynchronous Receiver/Transmitter. An 8-bit-parallelto- serial and serial-to-8-bit-parallel converter, combined with parity and start-detect circuitry and sometimes even FIFO buffers. Used widely in asynchronous serial-communications interfaces, (e.g. modems).

USB- Universal Serial Bus. A new, low-cost, low-speed, self-clocking bitserial bus (1.5 MHz and 12 MHz) using 4 wires (Vcc, ground, differential data) to daisy-chain up to 128 devices.

VME- Older bus standard, popular with MC68000-based industrial computers.

XNF File- Xilinx-proprietary description format for a logic design (Alternative: EDIF).

CHAPTER 1
INTRODUCTION

Wireless communications is an emerging field, which has seen enormous growth in the last several years. The huge uptake rate of mobile phone technology, Wireless Local Area Networks (WLAN) and the exponential growth of the Internet have resulted in an increased demand for new methods of obtaining high capacity wireless networks.

Most WLAN systems currently use the IEEE802.11b standard, which provides a maximum data rate of 11 Mbps. Newer WLAN standards such as IEEE802.11a and HiperLAN2, are based on OFDM technology and provide a much higher data rate of 54 Mbps. However systems of the near future will require WLANs with data rates of greater than 100 Mbps, and so there is a need to further improve the spectral efficiency and data capacity of OFDM systems in WLAN applications.

For cellular mobile applications, we will see in the near future a complete convergence of mobile phone technology, computing, Internet access, and potentially many multimedia applications such as video and high quality audio. In fact, some may argue that this convergence has already largely occurred, with the advent of being able to send and receive data using a notebook computer and a mobile phone. Although this is possible with current 2G (2nd Generation) Mobile phones, the data rates provided are very low (9.6 kbps – 14.4 kbps), limiting the usefulness of such a service.

The goal of third and fourth generation mobile networks is to provide users with a high data rate, and to provide a wider range of services, such as voice communications, videophones, and high speed Internet access. The higher data rate of future mobile networks will be achieved by increasing the amount of spectrum allocated to the service and by improvements in the spectral efficiency. OFDM is a potential candidate for the physical layer of fourth generation mobile systems.

1.1 THIRD GENERATION WIRELESS SYSTEMS

Third generation mobile systems such as the Universal Mobile Telecommunications System (UMTS), and CDMA2000 will be introduced over the next 1-5 years (2002 onwards). These systems are striving to provide higher data rates than current 2G systems such as the Global System for Mobile communications (GSM), and IS-95. Second generation systems are mainly targeted at providing voice services, while 3rd generation systems will shift to more data oriented services such as Internet access.

Third generation systems use Wide-band Code Division Multiple Access (WCDMA) as the carrier modulation scheme. This modulation scheme has a high multipath tolerance, flexible data rate, and allows a greater cellular spectral efficiency than 2G systems. Third generation systems will provide a significantly higher data rate (64 kbps – 2 Mbps) than second-generation systems (9.6 – 14.4 kbps). The higher data rate of 3G systems will be able to support a wide range of applications including Internet access, voice communications and mobile videophones. In addition to this, a large number of new applications will emerge to use the permanent network connectivity, such as wireless appliances, notebooks with built in mobile phones, remote logging, wireless web cameras, car navigation systems, and so forth.

In fact most of these applications will not be limited by the data rate provided by 3G systems, but by the cost of the service.

The demand for use of the radio spectrum is very high, with terrestrial mobile phone systems being just one of many applications vying for suitable bandwidth. These applications require the system to operate reliably in non-line-of-sight environments with a propagation distance of 0.5 - 30 km, and at velocities up to 100 km/hr or higher. This operating environment limits the maximum RF frequency to 5 GHz, as operating above this frequency results in excessive channel path loss, and excessive Doppler spread at high velocity. This limits the spectrum available for mobile applications, making the value of the radio spectrum extremely high. In Europe auctions of 3G licenses of the radio spectrum began in 1999. In the United Kingdom, 90 MHz of bandwidth was auctioned off for £22.5 billion [9]. In Germany the result was similar, with 100 MHz of bandwidth raising $46 billion (US) [7]. This represents a value of around $450 Million (US) per MHz. The length of these license agreements is 20 years [8] and so to obtain a reasonable rate of return of 8% on investment, $105 Million (US) per MHz must be raised per year. It is therefore vitally important that the spectral efficiency of the communication system is maximized, as this is one of the main limitations to providing a low cost high data rate service. [1]

1.2 4TH GENERATION SYSTEMS AND BEYOND

Research has just recently begun on the development of 4th generation (4G) mobile communication systems. The commercial rollout of these systems is likely to begin around 2008 - 2012, and will replace 3rd generation technology. Few of the aims of 4G networks have yet been published, however it is likely that they will be to extend the capabilities of 3G networks, allowing a greater range of applications, and improved universal access. Ultimately 4G networks should encompass broadband wireless services, such as High Definition Television (HDTV) (4 - 20 Mbps) and computer network applications (1 - 100 Mbps). This will allow 4G networks to replace many of the functions of WLAN systems. However, to cover this application, cost of service must be reduced significantly from 3G networks. The spectral efficiency of 3G networks is too low to support high data rate services at low cost. As a consequence one of the main focuses of 4G systems will be to significantly improve the spectral efficiency.

In addition to high data rates, future systems must support a higher Quality Of Service (QOS) than current cellular systems, which are designed to achieve 90 - 95% coverage, i.e. network connection can be obtained over 90 - 95% of the area of the cell. This will become inadequate as more systems become dependent on wireless networking. As a result 4G systems are likely to require a QOS closer to 98 - 99.5%. In order to achieve this level of QOS it will require the communication system to be more flexible and adaptive. In many applications it is more important to maintain network connectivity than the actual data rate achieved. If the transmission path is very poor, e.g. in a building basement, then the data rate has to drop to maintain the link. Thus the data rate might vary from as low as 1 kbps in extreme conditions, to as high as 20 Mbps for a good transmission path. Alternatively, for applications requiring a fixed data rate, the QOS can be improved by allocating additional resources to users with a poor transmission path.

A significant improvement in spectral efficiency will be required in order for 4G systems to provide true broadband access. This will only be achieved by significant advances in multiple aspects of cellular network systems, such as network structure, network

management, smart antennas, RF modulation, user allocation, and general resource allocation.

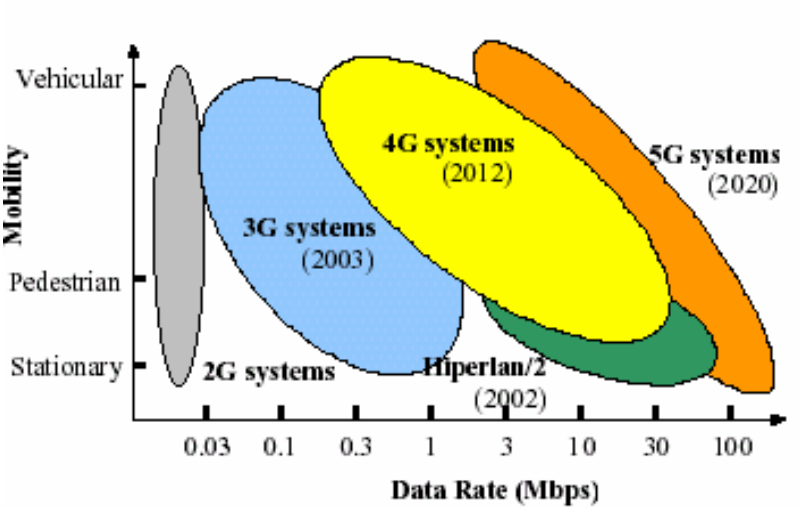


FIGURE 1-1, CURRENT AND FUTURE MOBILE SYSTEMS. THE GENERAL TREND WILL BE TO PROVIDE HIGHER DATA RATES AND GREATER MOBILITY. DERIVED FROM [5].

1.3 ORTHOGONAL FREQUENCY DIVISION MULTIPLEXING

The first solution of the bandwidth efficiency problem of multi-tone transmission (not the complexity problem) was probably the "Kineplex" system. The Kineplex system was developed by Collins Radio Co. For data transmission over an H.F. radio channel subject to severe multi-path fading. In that system, each of 20 tones is modulated by differential 4-PSK without filtering. The spectra are therefore of the *sin(kf)/f* shape and strongly overlap. However, similar to modern OFDM, the tones are spaced at frequency intervals almost equal to the signaling rate and are capable of separation at the receiver.

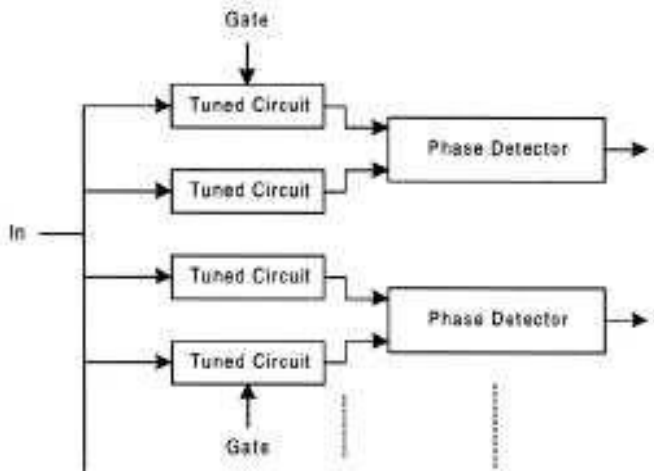


FIGURE 1.2. THE COLLINS KINEPLEX RECEIVER. [1]

Orthogonal Frequency Division Multiplexing (OFDM) is an alternative wireless modulation technology to CDMA. OFDM has the potential to surpass the capacity of CDMA systems and provide the wireless access method for 4G systems. OFDM is a modulation scheme that allows digital data to be efficiently and reliably transmitted over a radio channel, even in multipath environments. OFDM transmits data by using a large number of narrow bandwidth carriers. These carriers are regularly spaced in frequency, forming a block of spectrum. The frequency spacing and time synchronization of the carriers is chosen in such a way that the carriers are orthogonal, meaning that they do not cause interference to each other. This is despite the carriers overlapping each other in the frequency domain. The name 'OFDM' is derived from the fact that the digital data is sent using many carriers, each of a different frequency (Frequency Division Multiplexing) and these carriers are orthogonal to each other, hence Orthogonal Frequency Division Multiplexing.

1.4 MULTI-CARRIER SYSTEM FUNDAMENTALS [2]

Let [D0, D1,...., DN-1,] denote data symbols. Digital signal processing techniques, rather than frequency synthesizer, can be deployed to generate orthogonal sub-carriers. The DFT as a linear transformation maps the complex data symbols [D0, D1,...., DN-1,] to OFDM symbols [d0, d1,...., dN-1,] such that:

$$d_k = \sum_{n=0}^{N=1} D_n \, e^{2\pi n \frac{k}{N}} \tag{1.1}$$

The linear mapping can be represented in matrix form as:

$$\bar{d} = \bar{W} \cdot \bar{D} \qquad \text{where} \qquad \bar{W} = \begin{bmatrix} 1 & .... & 1 & 1 \\ 1 & W & .... & W^{N-1} \\ 1 & W^2 & & W^{2(N-1)} \\ & .... & & ... \\ 1 & W^{N-1} & & W^{N(N-1)} \end{bmatrix} \quad \text{and} \quad W = e^{j2\pi/N}$$

$\bar{W}$ is a symmetric and orthogonal matrix. After FFT, a cyclic pre/postfix of lengths k_1 and k_2 and will be added to each block (OFDM symbol) followed by a pulse shaping block. Proper pulse shaping has an important effect in improving the performance of OFDM systems in the presence of some channel impairments. The output of this block is fed to a D/A at the rate of f_s, and low-pass filtered. A basic representation of the equivalent complex baseband transmitted signal is:

$$x(t) = \sum_{n=0}^{N-1} \left\{ D_n e^{j2\pi \frac{n}{N} f_s t} \right\} \tag{1.2}$$

for $$-\frac{k_1}{f_s} < t < \frac{N + k_2}{f_s}$$

A more accurate representation of OFDM signal including windowing effect is:

$$x(t)=\sum_{l=-\infty}^{\infty}\sum_{k=-k_t}^{N+k_2}\sum_{n=0}^{N-1}\left\{D_{n!}e^{j2\pi\frac{n}{N}k}\right\}w\left(t-\frac{k}{f_s}-lT\right) \tag{1.3}$$

D_{nl} Represents the n_{th} data symbol transmitted during the ***l-th*** OFDM block, $T=(N+k_1+k_2)/f_s$ is the OFDM block duration, and $w(t)$is the window or pulse shaping function. The extension of the OFDM block is equivalent to adding a cyclic pre/postfix in the discrete domain. The received signal for a time-varying random channel is:

$$r(t)=\int_0^{\infty}x(t-\tau)h(t,\tau)d\tau+n(t) \tag{1.4}$$

The received signal is sampled at $t=k/f_s$ *for* $k=\{-k_1,.....N+k_2-1\}$ With no inter-block interference and assuming that the windowing function Satisfies $w(n-l)=\delta_{nl}$ the output of the FFT block at the receiver is

$$\tilde{D}_m=\frac{1}{N}\sum_{k=0}^{N-1}r_k e^{j2\pi m\frac{k}{2N}} \tag{1.5}$$

where

$$r_k=\sum_{n=0}^{N-1}H_n D_n e^{j2\pi\frac{n}{2N}k}+n(k) \tag{1.6}$$

A complex number H_n is the frequency response of the time-invariant Channel $h(t-\tau)$ at frequency n/T . So,

$$\tilde{D}_m=\begin{cases}H_n D_n+N(n), n=m\\ N(n),\ n\neq m\end{cases}$$

n(t) is white Gaussian noise with a diagonal covariance matrix of $E(n(k)n(l))=\delta l$ Therefore, the noise components for different sub-carriers are not correlated,

$$E(\bar{n}(k)n^*(l))=W\delta IW^T=\delta I$$

Where $\bar{n}(k)$ is the vector of noise samples at $n(k),.....,n(k-N)$

A detailed mathematical analysis of OFDM in multi-path Rayleigh fading is presented in the Appendix.

In 1971 Weinstein introduced the idea of using a Discrete Fourier Transform (DFT) for implementation of the generation and reception of OFDM signals, the key components of an OFDM system are the Inverse DFT in the transmitter and the DFT in the receiver, eliminating the requirement for banks of analog subcarrier oscillators. This presented an opportunity for an easy implementation of OFDM. These components must implement respectively.

$$d_n=\frac{1}{\sqrt{N}}\sum_{k=0}^{N-1}D_k W_N^{kn},\ n=0,....,N-1 \tag{1.7}$$

and

$$D_n = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} d_k W_N^{-kn}, \; n = 0,\ldots..,\; N-1 \tag{1.8}$$

These operations perform reversible linear mappings between N complex data symbols $\{D\}$ and N complex line symbols $\{d\}$.

A general N-to-N point linear transformation requires N^2 multiplications and additions. This would be true of the DFT and IDFT if each output symbol were calculated separately. However, by calculating the outputs simultaneously and taking advantage of the cyclic properties of the multipliers $e^{\pm j2\pi nk/N}$ Fast Fourier Transform (FFT) techniques reduce the number of computations to the order of $N \log N$. The FFT is most efficient when N is a power of two. Several variations of the FFT exist, with different ordering of the inputs and outputs, and different use of temporary memory. One variation, decimation in time, is shown below.

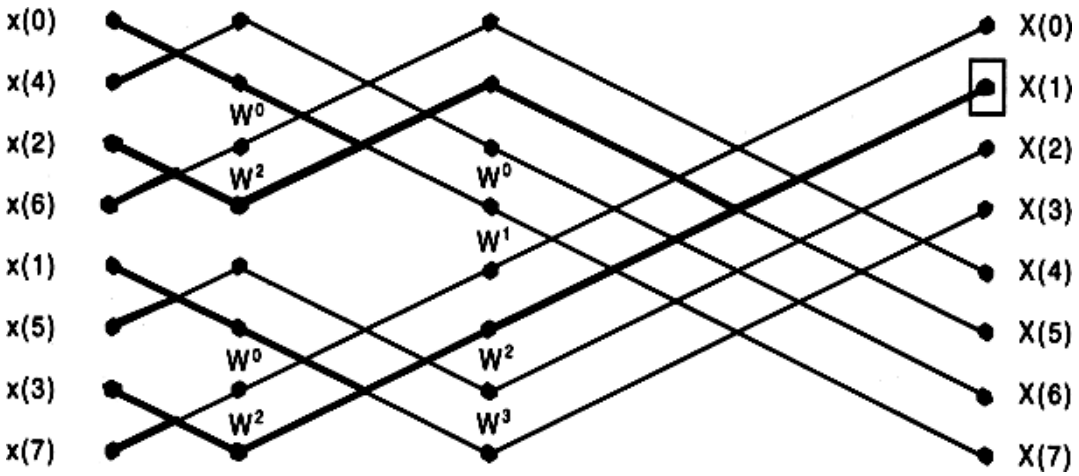


FIGURE 1.3. PARTIAL FFT (DIT) [1]

One of the benefits of an OFDM system with an FFT structure is the fact that it lends itself to a repetitive structure very well. This structure is preferred compared to the required filtering complexity in other wideband systems. Two common structures are shown in Figure 1.4

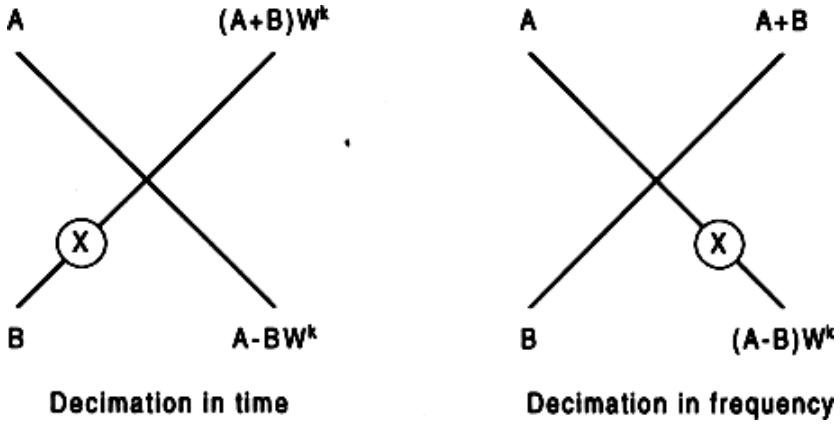


FIGURE 1.4. TWO DIFFERENT TECHNIQUES FOR FFT BUTTERFLY [1]

Two main differences between decimation in time (DIT) and decimation in frequency (DIF) are noted [1]. First, for DIT, the input is bit-reversed and output is in natural order, while in DIF the reverse is true. Secondly, for DIT complex multiplication is performed before the add-subtract operation, while in DIF the order is reversed. While complexity of the two structures is similar in typical DFT, this is not the case for partial FFT [2]. The reason is that in the DIT version of partial FFT, a sign change (multiplication by 1 and -1) occurs at the first stages, but in the DIF version it occurs in later stages.

Figure 1.5 shows the architecture of an OFDM system capable of using a further stage of modulation employing both in-phase and square modulators. This configuration is common in wireless communication systems for modulating baseband signals to the required IF or RF frequency band. It should be noted that the basic configuration illustrated does not account for channel dispersion, which is almost always present. The channel dispersion problem is solved by using the cyclic prefix which will be described later.

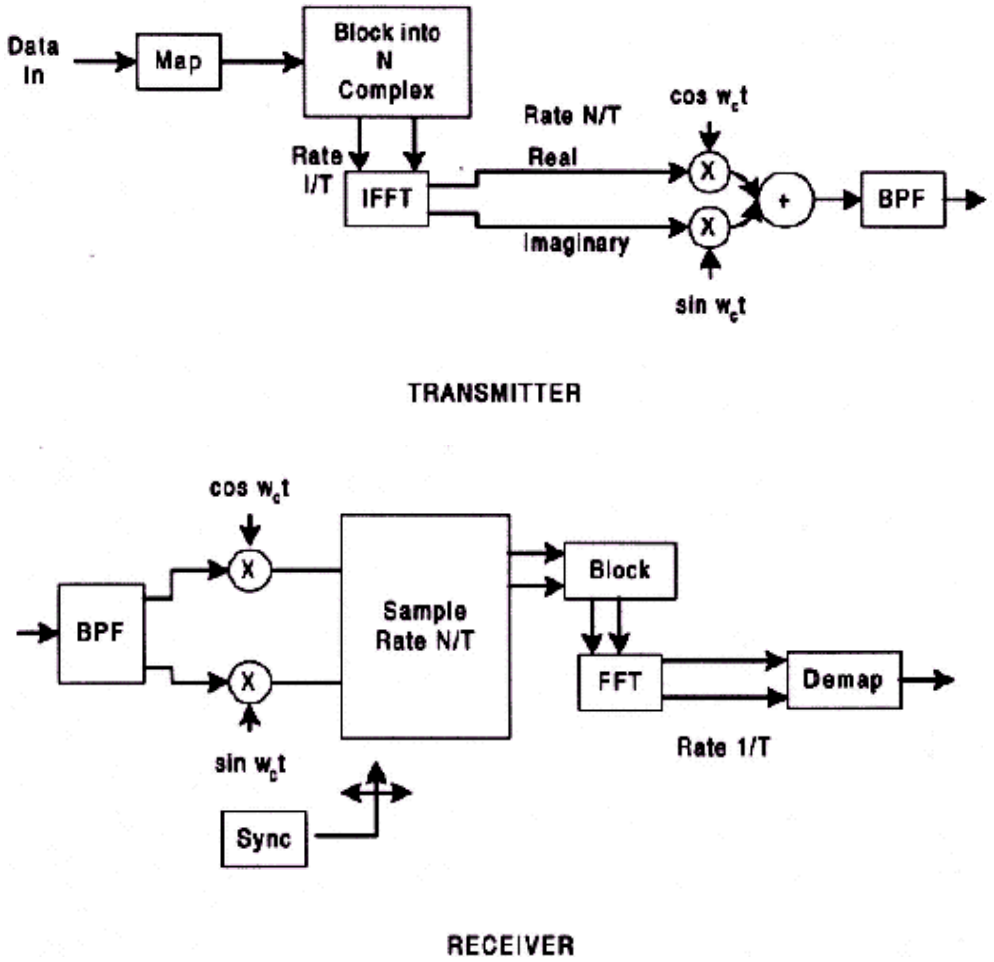


FIGURE 1.5. SYSTEM WITH COMPLEX TRANSMISSION [1].

Small sets of input bits are first assembled and mapped into complex numbers which determine the constellation points of each sub-carrier. In most wireless systems, smaller constellation is formed for each sub-carrier. In wire-line systems, where the signal-to-noise ratio is higher and variable across the frequency range, the number of bits assigned to each sub-carrier may be variable. If the number of sub-carriers is not a power of two, then it is common to add symbols of value zero to the input block so that the advantage of using such a block length in the FFT is achieved.

1.5 ORTHOGONALITY

Signals are orthogonal if they are mutually independent of each other. Orthogonality is a property that allows multiple information signals to be transmitted perfectly over a common channel and detected, without interference. Loss of orthogonality results in blurring between these information signals and degradation in communications.

If any two different functions within the set are multiplied, and integrated over a symbol period, the result is zero, for orthogonal functions. Another way of thinking of this is that if we look at a matched receiver for one of the orthogonal functions, a subcarrier in the case of OFDM, then the receiver will only see the result for that function. The results from all other functions in the set integrate to zero, and thus have no effect.

$$\int_0^{\tau} s_t(t)\, s_j(t)\, dt = \begin{cases} C & i = j \\ 0 & i \neq j \end{cases} \qquad 1.8$$

Equation (1-15) shows a set of orthogonal sinusoids, which represent the subcarriers for an unmodulated real OFDM signal.

$$s_k(t) = \begin{cases} \sin(2\pi k f_0 t) & 0 < t < T \; k = 1,2,....M \\ 0 & otherwise \end{cases} \qquad (1.9)$$

These subcarriers are orthogonal to each other because when we multiply the waveforms of any two subcarriers and integrate over the symbol period the result is zero. Multiplying the two sine waves together is the same as mixing these subcarriers. This results in sum and difference frequency components, which will always be integer subcarrier frequencies, as the frequency of the two mixing subcarriers has integer number of cycles. Since the system is linear we can integrate the result by taking the integral of each frequency component separately then combining the results by adding the two sub-integrals. The two frequency components after the mixing have an integer number of cycles over the period and so the sub-integral of each component will be zero, as the integral of a sinusoid over an entire period is zero. Both the sub-integrals are zeros and so the resulting addition of the two will also be zero, thus we have established that the frequency components are orthogonal to each other.

1.6 GUARD PERIOD

For a given system bandwidth the symbol rate for an OFDM signal is much lower than a single carrier transmission scheme. For example for a single carrier BPSK modulation, the symbol rate corresponds to the bit rate of the transmission. However for OFDM the system bandwidth is broken up into N_c subcarriers, resulting in a symbol rate that is N_c times lower than the single carrier transmission. This low symbol rate makes OFDM naturally resistant to effects of Inter-Symbol Interference (ISI) caused by multipath propagation.

Multipath propagation is caused by the radio transmission signal reflecting of objects in the propagation environment, such as walls, buildings, mountains, etc. These multiple signals arrive at the receiver at different times due to the transmission distances being different. This spreads the symbol boundaries causing energy leakage between them.

The effect of ISI on an OFDM signal can be further improved by the addition of a guard period to the start of each symbol. This guard period is a cyclic copy that extends the length of the symbol waveform. Each subcarrier, in the data section of the symbol, (i.e. the OFDM symbol with no guard period added, which is equal to the length of the IFFT size used to generate the signal) has an integer number of cycles. Because of this, placing copies of the symbol end-to-end results in a continuous signal, with no discontinuities at the joins. Thus by copying the end of a symbol and appending this to the start results in a longer symbol time. Figure 1-6 shows the insertion of a guard period.

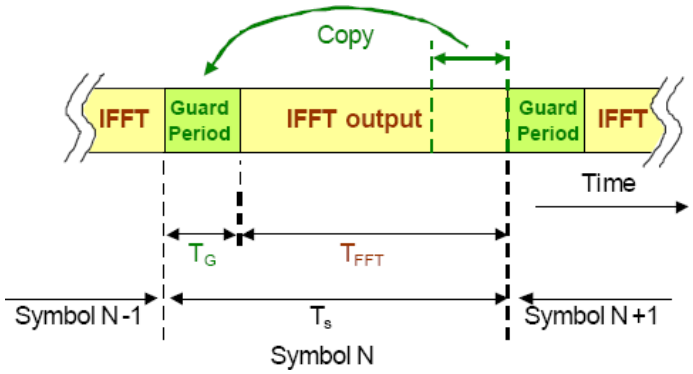


FIGURE 1.6, ADDITION OF A GUARD PERIOD TO AN OFDM SIGNAL[5]

The total length of the symbol is $T_S = T_G + T_{FFT}$, where T_s is the total length of the symbol in samples, T_G is the length of the guard period in samples, and T_{FFT} is the size of the IFFT used to generate the OFDM signal. In addition to protecting the OFDM from ISI, the guard period also provides protection against time-offset errors in the receiver.

1.6.1 Protection Against Time Offset

To decode the OFDM signal the receiver has to take the FFT of each received symbol, to work out the phase and amplitude of the subcarriers. For an OFDM system that has the same sample rate for both the transmitter and receiver, it must use the same FFT size at both the receiver and transmitted signal in order to maintain subcarrier orthogonality. Each received symbol has $T_G + T_{FFT}$ samples due to the added guard period. The receiver only needs T_{FFT} samples of the received symbol to decode the signal. The remaining T_G samples are redundant and are not needed. For an ideal channel with no delay spread the receiver can pick any time offset, up to the length of the guard period, and still get the correct number of samples, without crossing a symbol boundary.

Because of the cyclic nature of the guard period changing the time offset simply results in a phase rotation of all the subcarriers in the signal. The amount of this phase rotation is proportional to the subcarrier frequency, with a subcarrier at the Nyquist frequency changing by 180° for each sample time offset. Provided the time offset is held constant from symbol to symbol, the phase rotation due to a time offset can be removed out as part of the channel equalization. In multipath environments ISI reduces the effective length of the guard period leading to a corresponding reduction in the allowable time offset error.

1.6.2 Protection Against ISI

In an OFDM signal the amplitude and phase of the subcarrier must remain constant over the period of the symbol in order for the subcarriers to maintain orthogonality.

If they are not constant it means that the spectral shape of the subcarriers will not have the correct *sinc* shape, and thus the nulls will not be at the correct frequencies, resulting in Inter-Carrier Interference. At the symbol boundary the amplitude and phase change suddenly to the new value required for the next data symbol. In multipath environments ISI causes spreading of the energy between the symbols, resulting in transient changes in the amplitude and phase of the subcarrier at the start of the symbol. The length of these transient effects corresponds to the delay spread of the radio channel. The transient signal is a result of each multipath component arriving at slightly different times, changing the received subcarrier vector. Figure 1.7 shows this effect. Adding a guard period allows time for the transient part of the signal to decay, so that the FFT is taken from a steady state portion of the symbol.

This eliminates the effect of ISI provided that the guard period is longer than the delay spread of the radio channel. The remaining effects caused by the multipath, such as amplitude scaling and phase rotation are corrected for by channel equalization.

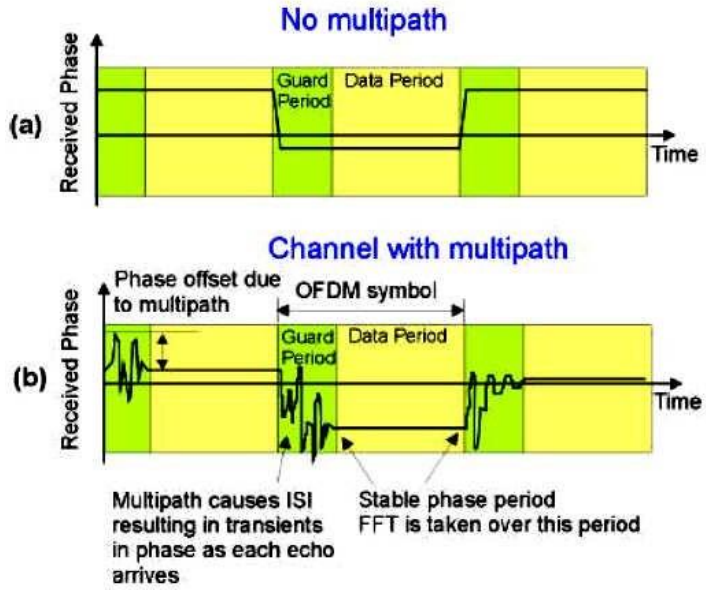


FIGURE 1-7, FUNCTION OF THE GUARD PERIOD FOR PROTECTING AGAINST ISI [5].

The guard period protects against transient effects due to multi-path, removing the effects of ISI, provided it is longer than the channel delay spread. This example shows the instantaneous phase of a single carrier for 3 symbols.

The addition of guard period removes most of the effects of ISI; however in practice, multi-path components tend to decay slowly with time, resulting in some ISI even when a relatively long guard period is used.

1.7 SYNCHRONIZATION

Multi-carrier modems, like any other digital communication modems, require a reliable synchronization scheme. Parallel transmission of N symbols results in a longer symbol duration, consequently there is less sensitivity to the timing offset. In other words, unlike single carrier systems in which a timing jitter can create inter-symbol interference, it does not violate orthogonality of transmitted waveforms in a multi-carrier system. However, frequency offset is detrimental to OFDM systems and has an important role in system design. Phase noise is another critical impairment in wireless OFDM.

1.7.1 Synchronization and System Architecture

A synchronization sequence in an OFDM system is shown in Figure 1.8.

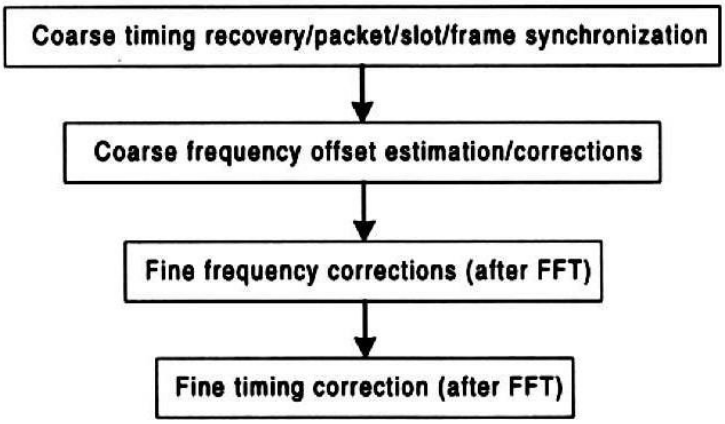


FIGURE 1.8: SYNCHRONIZATION SEQUENCE IN OFDM [1].

Initially, a coarse frame (or packet) synchronization is required to provide timing information regarding the OFDM symbol. Then, a frequency correction prior to FFT is implemented to reduce the effect of inter-channel interference. Consequently, further stages of fine timing and frequency offset are implemented. These may be combined in a joint estimation block. In general, frame timing and frequency correction are more complicated and require higher hardware/software resources. Timing and frequency correction techniques for OFDM can be classified into two categories: data aided and non-linear techniques. Data aided techniques use a known bit pattern or pilot signal to estimate the timing or frequency offset. Non-linear techniques use the cyclostationarity characteristics of the signal to extract the desired harmonic component by using a non-linear operation. Use of data-aided techniques is applicable to many OFDM digital communication systems such as HDTV and Wireless LAN. Pilot signal fields are provided in existing standards.

1.7.1.1 TIMING AND FRAME SYNCHRONIZATION

Initially a frame (packet or slot) synchronization circuit is required to detect the frame starting point. This is usually achieved by correlating the incoming signal with a known preamble. The same circuit is sometimes used to adjust for initial gain control. Therefore, the threshold of the detection circuit should be adjusted accordingly. A general block diagram is shown in Figure 1.9. Since timing offset does not violate orthogonality of the symbols it can be compensated after the receiver FFT.

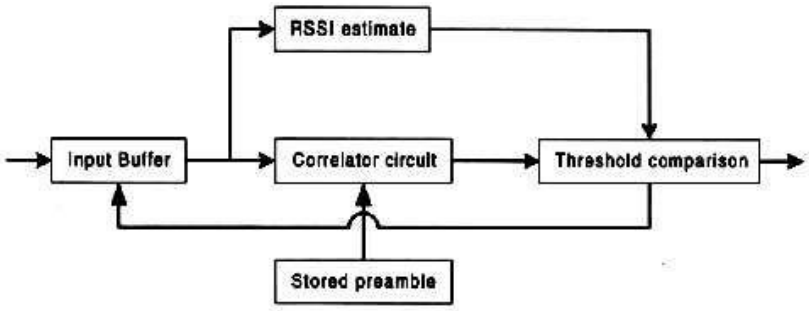


FIGURE 1.9: FRAME SYNCHRONIZATION [1].

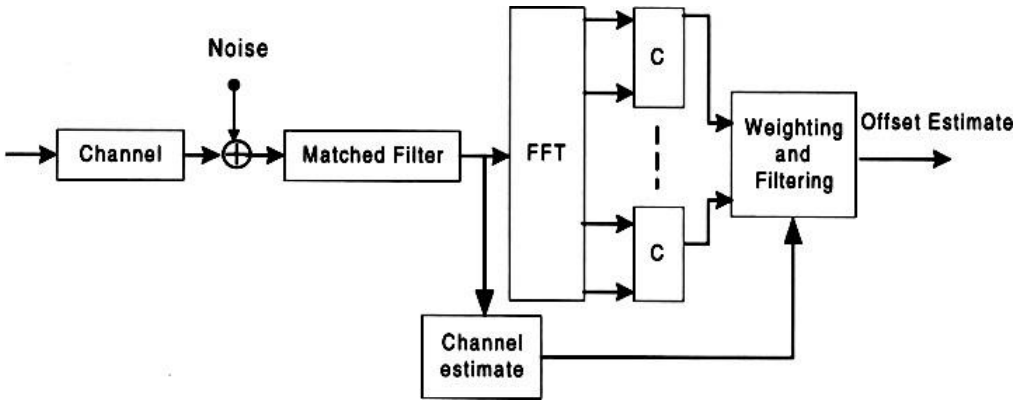


FIGURE 1.10: TIMING OFFSET ESTIMATE.[1]

Frequency offset estimation in packet data communications requires fast acquisition. Hence, phase-lock techniques are not applicable for initial coarse correction. For this purpose, a pilot or training sequence should be transmitted. This can be achieved by sending a pilot in one or two bins and estimate the frequency offset by measuring the energy spread in other bins, or by using a training sequence or pilot in every bin and measuring the phase difference by repeating the same pattern. Usually, the second approach is more reliable, as additive noises in different bins are independent and the estimate convergence will be faster.

1.7.1.2 FREQUENCY OFFSET ESTIMATION

Frequency offset should be corrected before the receiver FFT. The FFT can be used as a frequency offset detector. Before correction for frequency offset, C/I will not be at the acceptable level for reliable data transmission. Therefore, It is preferable that entire bins should be used for frequency estimation purposes initially. We assume a known data pattern is used for synchronization. After FFT, at the receiver, we have:

$$D_m = \sum_{k=0}^{N-1} \sum_{n=0}^{N-1} D_n e^{j2\pi \frac{k}{N}(n-m+\Delta f)} + N_m. \tag{1.10}$$

If the same block is repeated, the result would be

$$D_m^{\,j2\pi\Delta f} + N'_m \,. \tag{1.11}$$

Therefore, we can obtain a proper estimate of Δf by averaging the product of every bin from one symbol and the same bin from the previous symbol, and then averaging the result.

$$\frac{1}{N}\sum_{m=0}^{N-1}\left(\frac{1}{l}\sum_{i=0}^{l-1} y_{mi}\,\overline{y'}_{m(i+1)}\right). \tag{1.12}$$

This process is shown in Figure 1.11

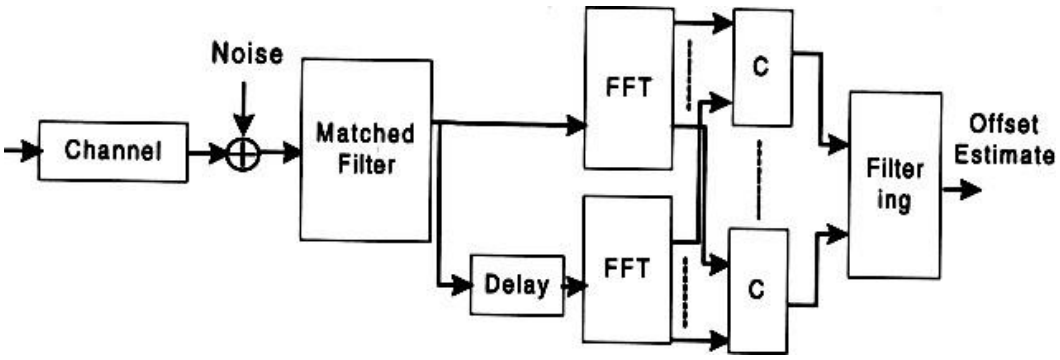


FIGURE 1.11: FREQUENCY OFFSET ESTIMATION.[1]

The number of repetitions depends on the signal-to-noise ratio and the required accuracy of frequency estimation. Double averaging accelerates the convergence of the estimate significantly. This scheme works in low signalto-noise ratios as well. However, this technique is limited to Frequency offset of less than $\Delta f = 1/2T$ the initial frequency offset could be higher than that limit, and requires an initial coarse acquisition. A Typical coarse frequency detection in single and multi-carrier systems is the Maximum Likelihood estimation technique.

$$f_{est} = \max_f \sum D_{n+l+\tau} D^*_{n+l} \tag{1.13}$$

A digital implementation of the technique is shown in the Figure 1.12 where $C(f_i)$ denotes a matched filter tuned to transmit filter impulse response shifted by frequency offset of f_i.

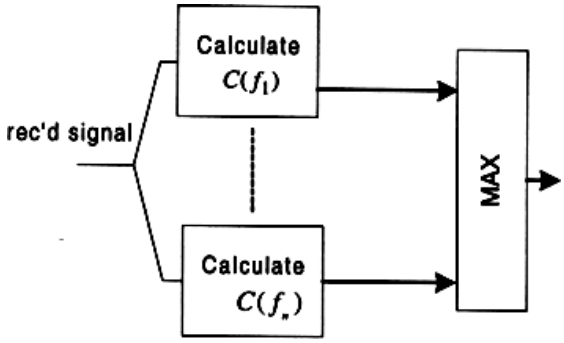


FIGURE 1.12 A DIGITAL IMPLEMENTATION OF COARSE FREQUENCY OFFSET ESTIMATION USING MAXIMUM LIKELIHOOD CRITERIA [1].

2. OFDM GENERATION AND RECEPTION

OFDM signals are typically generated digitally due to the difficulty in creating large banks of phase lock oscillators and receivers in the analog domain. Figure 1-13 shows the block diagram of a typical OFDM transceiver. The transmitter section converts digital data to be transmitted, into a mapping of subcarrier amplitude and phase. It then transforms this spectral representation of the data into the time domain using an Inverse Discrete Fourier Transform (IDFT). The Inverse Fast Fourier Transform (IFFT) performs the same operations as an IDFT, except that it is much more computationally efficiency, and so is used in all practical systems. In order to transmit the OFDM signal the calculated time domain signal is then mixed up to the required frequency.

The receiver performs the reverse operation of the transmitter, mixing the RF signal to base band for processing, then using a Fast Fourier Transform (FFT) to analyze the signal in the frequency domain. The amplitude and phase of the subcarriers is then picked out and converted back to digital data. The IFFT and the FFT are complementary function and the most appropriate term depends on whether the signal is being received or generated. In cases where the signal is independent of this distinction then the term FFT and IFFT is used interchangeably.

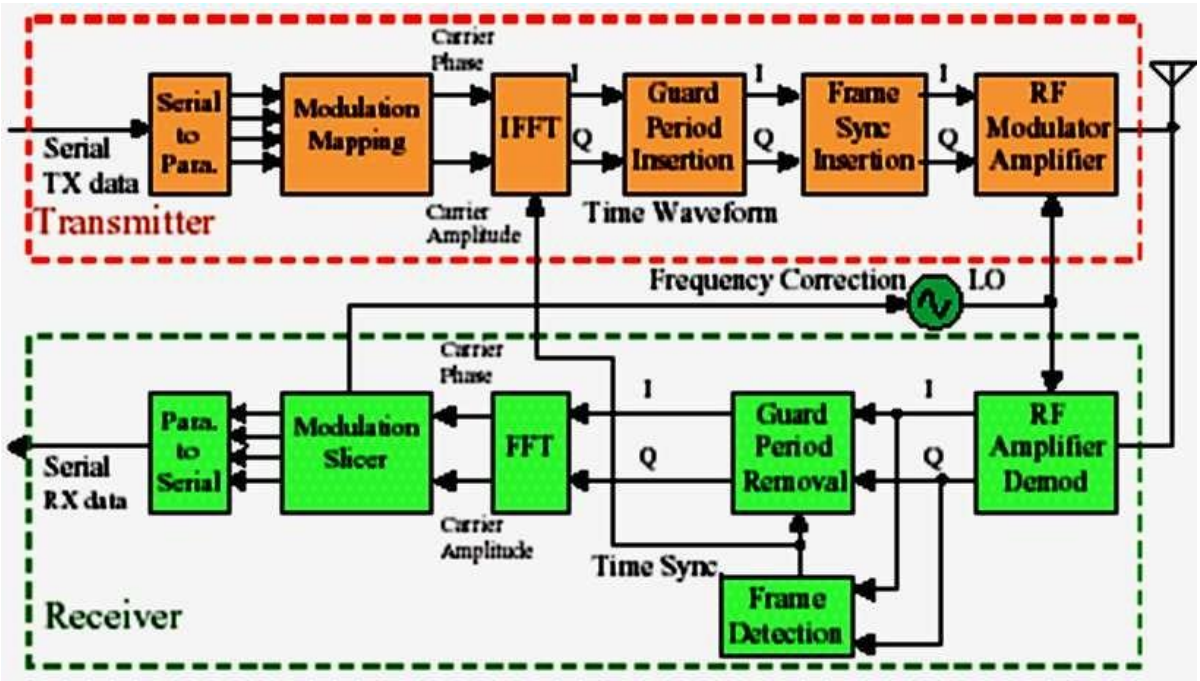


FIGURE 1.13 : BLOCK DIAGRAM SHOWING A BASIC OFDM TRANSCEIVER [5]

2.1 SERIAL TO PARALLEL CONVERSION

Data to be transmitted is typically in the form of a serial data stream. In OFDM, each symbol typically transmits 40 - 4000 bits, and so a serial to parallel conversion stage is needed to convert the input serial bit stream to the data to be transmitted in each OFDM

symbol. The data allocated to each symbol depends on the modulation scheme used and the number of subcarriers. For example, for a subcarrier modulation of 16-QAM each subcarrier carries 4 bits of data, and so for a transmission using 100 subcarriers the number of bits per symbol would be 400.

When an OFDM transmission occurs in a multipath radio environment, frequency selective fading can result in groups of subcarriers being heavily attenuated, which in turn can result in bit errors. These nulls in the frequency response of the channel can cause the information sent in neighboring carriers to be destroyed, resulting in a clustering of the bit errors in each symbol. Most Forward Error Correction (FEC) schemes tend to work more effectively if the errors are spread evenly, rather than in large clusters, and so to improve the performance most systems employ data scrambling as part of the serial to parallel conversion stage. This is implemented by randomizing the subcarrier allocation of each sequential data bit. At the receiver the reverse scrambling is used to decode the signal. This restores the original sequencing of the data bits, but spreads clusters of bit errors so that they are approximately uniformly distributed in time. This randomization of the location of the bit errors improves the performance of the FEC and the system as a whole.

2.2 SUBCARRIER MODULATION

Once each subcarrier has been allocated bits for transmission, they are mapped using a modulation scheme to a subcarrier amplitude and phase, which is represented by a complex In-phase and Quadrature-phase (IQ) vector. Figure 1.14 shows an example of subcarrier modulation mapping. This example shows 16-QAM, which maps 4 bits for each symbol. Each combination of the 4 bits of data corresponds to a unique IQ vector, shown as a dot on the figure. A large number of modulation schemes are available allowing the number of bits transmitted per carrier per symbol to be varied.

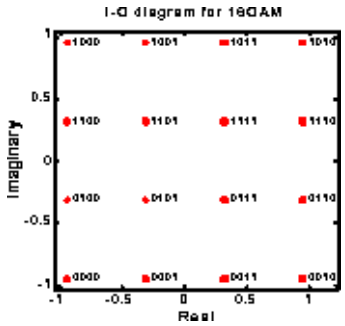


FIGURE 1-14, EXAMPLE IQ MODULATION CONSTELLATION. 16-QAM, WITH GRAY CODING OF THE DATA TO EACH LOCATION. (SCRIPT S0045) [5]

Subcarrier modulation can be implemented using a lookup table, making it very efficient to implement. In the receiver, mapping the received IQ vector back to the data word performs subcarrier demodulation. During transmission, noise and distortion becomes added to the signal due to thermal noise, signal power reduction and imperfect channel equalization. Figure 1.15 shows an example of a received 16-QAM signal with a SNR of 18 dB. Each of the IQ points is blurred in location due to the channel noise. For each received IQ vector the receiver has to estimate the most likely original transmission vector. This is achieved by finding the transmission vector that is closest to the received vector. Errors occur

when the noise exceeds half the spacing between the transmission IQ points, making it cross over a decision boundary.

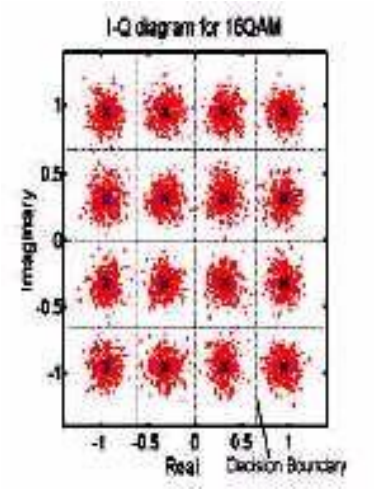


FIGURE 1-15, IQ PLOT FOR 16-QAM DATA WITH ADDED NOISE. (SCRIPT S0083) [5]

2.3 FREQUENCY TO TIME DOMAIN CONVERSION

After the subcarrier modulation stage each of the data subcarriers is set to an amplitude and phase based on the data being sent and the modulation scheme; all unused subcarriers are set to zero. This sets up the OFDM signal in the frequency domain. An IFFT is then used to convert this signal to the time domain, allowing it to be transmitted. Figure 1.16 shows the IFFT section of the OFDM transmitter. In the frequency domain, before applying the IFFT, each of the discrete samples of the IFFT corresponds to an individual subcarrier. Most of the subcarriers are modulated with data. The outer subcarriers are unmodulated and set to zero amplitude. These zero subcarriers provide a frequency guard band before the Nyquist frequency and effectively act as an interpolation of the signal and allows for a realistic roll off in the analog anti-aliasing reconstruction filters.

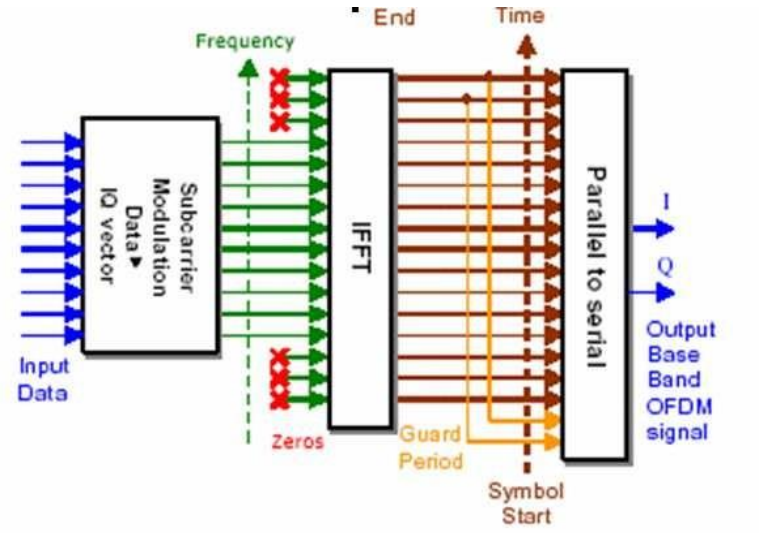


FIGURE 1.16: OFDM GENERATION [5]

2.4 RF MODULATION

The output of the OFDM modulator generates a base band signal, which must be mixed up to the required transmission frequency. This can be implemented using analog techniques as shown in Figure 1-17 or using a Digital Up Converter as shown in Figure 1;18. Both techniques perform the same operation, however the performance of the digital modulation will tend to be more accurate due to improved matching between the processing of the I and Q channels, and the phase accuracy of the digital IQ modulator.

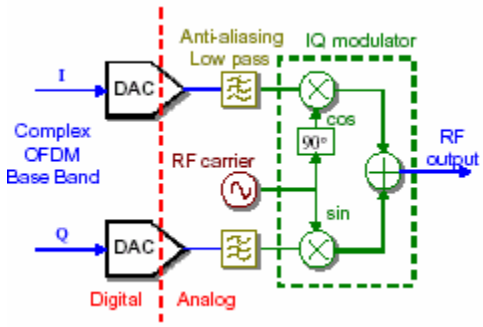


FIGURE 1.17, RF MODULATION OF COMPLEX BASE BAND OFDM SIGNAL, USING ANALOG TECHNIQUES.[5]

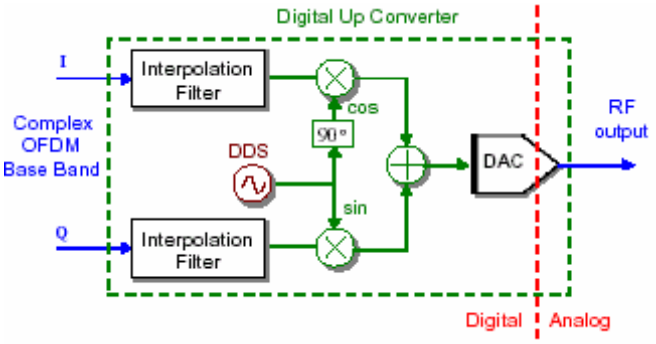


FIGURE 1-18, RF MODULATION OF COMPLEX BASE BAND OFDM SIGNAL, USING DIGITAL TECHNIQUES. (DDS = DIRECT DIGITAL SYNTHESIS) [5]

ADSL SYSTEMS

Two classes of ADSL have been standardized recently, with many options in each. Full rate ADSL can carry up to approximately 8 Mb/s downstream and 800 kb/s upstream. A simpler class, commonly called "ADSL Lite," carries up to approximately 1.5 Mb/s downstream and 500 kb/s upstream. In both cases, data rates can be adjusted to any value in steps *ADSL* 171 of 32 kb/s. An analog voice channel is provided on the same pair. The target error probability is per bit, with some required margin .

The two classes are somewhat compatible with each other. In both cases, sub-carriers are spaced 4312.5 Hz apart in both directions. After every 68 frames of data, a synchronization frame is inserted. Because of this and the use of cyclic prefixes, the net useful number of data frames is 4000 per second in all cases. One of the sub-carriers of the frame is devoted to synchronization. Adaptive bit allocation over the sub-carriers is performed in all cases. This process is critical to ensure system performance.

In the full rate downstream direction, a block of 255 complex data symbols, including several of value zero, are assembled. These will correspond to sub-channels 1 to 255. The lower ones cannot be used because of the analog voice channels, nor can the 255th. Therefore, the highest frequency allowed sub-carrier is centered at 1.095 MHz. Sub-carriers which cannot support at least a 4-point constellation at the desired error probability will also be unused. Conjugate appending is performed on the block, followed by a 512-point DFT. This results in frame of 512 real values. A cyclic prefix of 32 samples is added, and the resultant 2.208 M samples per second transmitted over the line.

Upstream, 31 sub-channels are processed, although (again) the lower few and the 31st cannot be used. The same processing is performed with a cyclic prefix of 4 samples. The upstream and downstream sub-channels may overlap. This provides a larger data rate, but requires the use of echo cancellation.

The bit streams may be treated as several multiplexed data channels. Each such channel may be optionally Reed-Solomon coded, with a choice of code and interleaving depth. Other optional codes include a CRC error check, and a 16-state 4-dimensional trellis code. The trellis code, when present, operates over the non-zero sub-carriers of a block, and is forced to terminate at the end of each block. 172 *ADSL* "ADSL Lite" is intended as a simpler lower cost system, with greater range of coverage because of the lower rate. One important difference is the elimination of filters at the customer's premises to separate the voice and data channels. The upstream channel is created identically to that of the full rate system, except that the first 6 sub-carriers must be zero.

The downstream transmitted sampled rate is reduced by a factor of two, to 1.104 M samples per second. The IDFT is performed over an initial block of 127 complex numbers, of which the first 32 must be zero. The highest sub-carrier is now at 543 kHz. In this case, the upstream and downstream sub-carriers do not overlap. The signal is treated as a single bit stream. Reed-Solomon and CRC coding are again optional, but there is no trellis coding. As of the time of publication of this text, many trials of various ADSL systems are in progress around the world. The ultimate market for high rate digital residential services is not clear, nor are the relative advantages anddis advantages of providing these services over competitive cable television facilities.

PHYSICAL LAYER TECHNIQUES FOR WIRELESS LAN

Wireless LAN 802.11 supports a variety of physical media and communication techniques ranging from Frequency Hopping Spread Spectrum (FHSS), Direct Sequence Spread Spectrum (DSSS), OFDM to infrared light systems. Infrared is a low cost solution for low-range applications while radio is a better solution for larger areas with little line of sight access, it could be a proper complement to the radio part. Allocated spectra for wireless LAN applications are typically in 2.4 GHz and 5 GHz range where the bandwidth is scarce and much in demand. WLAN system at 2.4 GHz must meet ISM band requirements. The IEEE 802.11 standard supports both FHSS and DSSS techniques for this band. Spread spectrum techniques are robust against interference and do not require adaptive equalization. However, for higher data rates the synchronization requirements of spread spectrum techniques are more restrictive and add to system complexity. For data rates of above 10 Mb/s OFDM system shows better performance.

OFDM for Wireless LAN

Multi-carrier modulation is a strong candidate for packet switched wireless applications and offers several advantages over single carrier approaches. For higher data rate applications ranging from 10 Mb/s up to 50Mb/s, an OFDM system is viable solution for the following reasons:

• *Robustness against delay spread*: Data transmission in wireless environment experiences delay spreads of up to 800 ns which covers several symbols at baud rates of 10 Mb/s and higher. In a single carrier system an equalizer handles detrimental effects of delay spread. When delay spread is beyond 4 symbols, use of maximum likelihood sequence estimator structure is not practical due to its exponentially increasing complexity. Linear equalizers are not suitable for this application either since in a frequency selective channel it amounts to significant noise enhancement. Hence, other equalizer structures such as decision feedback equalizers are used. Number of taps of the equalizer should be enough to cancel the effect of inter-symbol interference and perform as a matched filter too. In addition, equalizer coefficients should be trained for every packet, as the channel characteristics are different for each packet. A large header is usually needed to guarantee the convergence of adaptive training techniques. A multi-carrier system is robust against delay spread and does not need a training sequence. Channel estimation is required however. A typical channel impulse response for indoor wireless LAN is shown in Figure 1.19.

• *Fall-back mode:* Depending on the delay spread for different applications a different number of carriers is required to null the effect of delay spread. The structure of FFT lends itself to simple fallback mode by using the butterfly structure of FFT.

• *Computational efficiency:* Use of FFT structure at the receiver reduces the complexity to $N\ N$. As the number of carriers grows the higher efficiency can be achieved. A typical OFDM Wireless LAN is shown in the Figure 1.20.

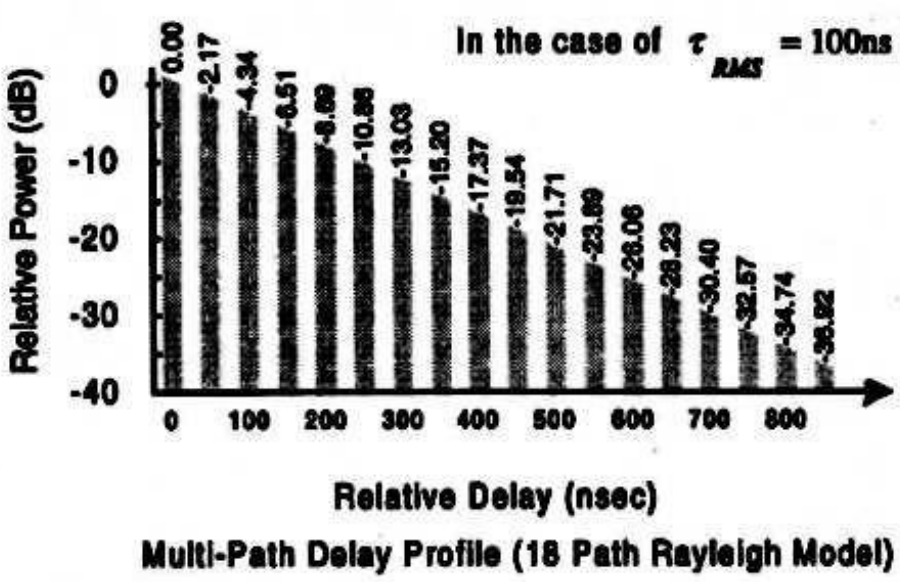


FIGURE 1.19. CHANNEL IMPULSE RESPONSE FOR TYPICAL WIRELESS LAN MEDIUM. [1]

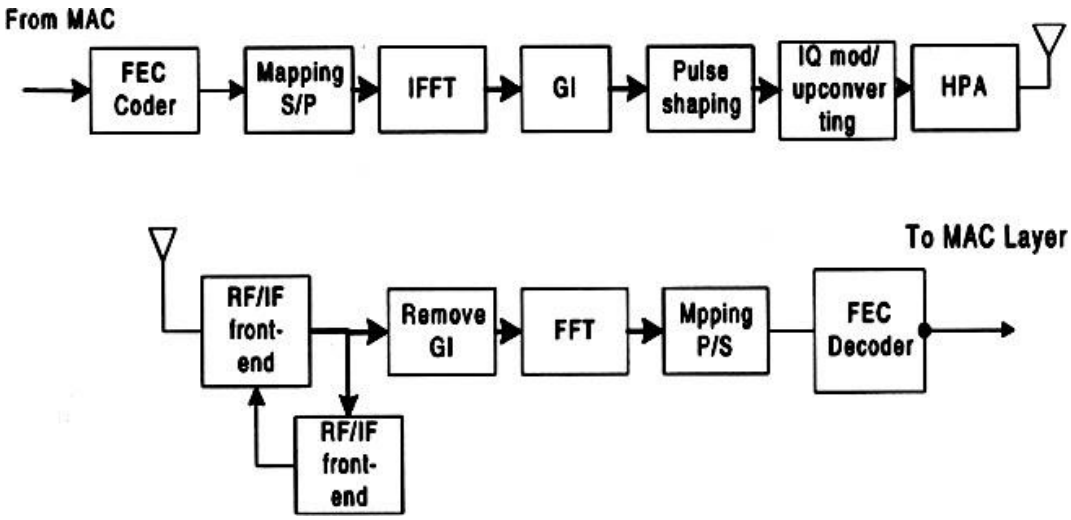


FIGURE 1.20. WIRELESS LAN SYSTEM STRUCTURE [1]

Fast synchronization: OFDM receivers are less sensitive to timing jitter compared to spread spectrum techniques. A typical OFDM wireless LAN transceiver is shown in Figure 1.20.

TRANSMITTER STRUCTURE

WLAN system at 5 GHz supports variable data rates by using different modulation schemes and therefore requires variable coding gain for each data rate. Currently, using BPSK, QPSK, 16 QAM and 64 QAM modulation supports data rates of 6 Mb/s up to 54 Mb/s. Different coding rates for convolutional coding are proposed depending on required protection Obviously, more crowded constellations need higher coding protection. A combination of error detection, such as CRC, and error correction, such as convolutional coding, in conjunction with interleaving are used to provide better coding gain in the presence of frequency selective fading. Convolutional coding is usually punctured to provide higher bandwidth efficiency. The number of carriers and modulation of data bits depends on

the required data rates. Typical number of carriers is 48-52 and a guard band is added to control the spectrum shape. Additional guard interval in the time domain is used for pulse shaping and controlling the ISI. A typical frame format of OFDM WLAN is shown in Figure 1.21.

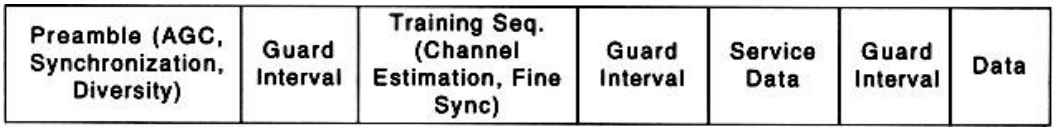


FIGURE 1.21. TYPICAL FRAME FORMAT FOR OFDM WLAN. [1]

Pulse shaping has an important role in robustness of the system against phase noise and frequency offset as explained in Chapter 5. In addition, pulse shaping provides a smoother transmit signal spectrum which is critical in an OFDM system due to higher peak-to-average ratio. Transmitter output has to meet strict criteria of out-of-band leakage. A spectrum mask, such as the one shown in Figure 1.22, is usually defined to control the effect of bandwidth regrowth caused by amplifier clipping.

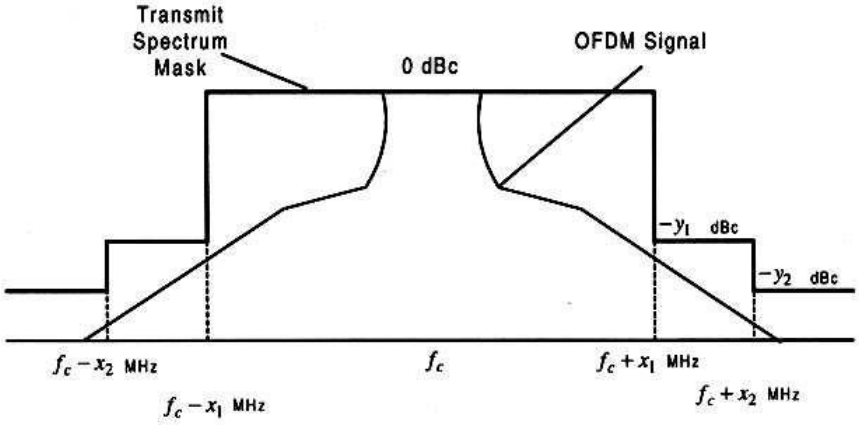


FIGURE 1.22. TRANSMITTER SPECTRUM MASK.[1]

RECEIVER STRUCTURE

Initially, the receiver must detect the frame, adjust AGC to proper level, utilize any diversity capability, and adjust for coarse frequency offset. Fast synchronization is critical in this application. High phase noise and frequency offset of low cost receiver oscillators (with about ±10 ppm frequency stability) interfere with the synchronization process and require robust algorithm design. A typical AGC block consists of a correlator followed by confirmation block. Since received signal level could vary significantly due to shadowing, a low threshold should be used followed by confirmation block, which utilizes the repetitive pattern of preamble sequence to reduce false alarm probability.

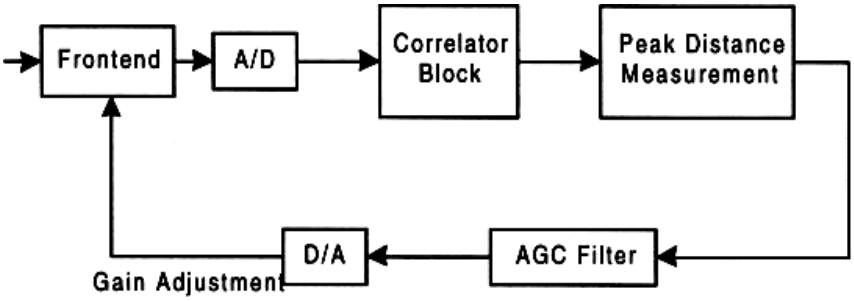


FIGURE 1.22. FRAME SYNCHRONIZATION AND AGC [1]

DIGITAL AUDIO BROADCASTING

DAB was the first commercial use of OFDM technology [19], [20]. Development of DAB started in 1987 and services began in U.K and Sweden in 1995. DAB is a replacement for FM audio broadcasting, by providing high quality digital audio and information services. OFDM was used for DAB due to its multipath tolerance. Broadcast systems operate with potentially very long transmission distances (20 - 100 km). As a result, multipath is a major problem as it causes extensive ghosting of the transmission. This ghosting causes Inter-Symbol Interference (ISI), blurring the time domain signal.

For single carrier transmissions the effects of ISI are normally mitigated using adaptive equalization. This process uses adaptive filtering to approximate the impulse response of the radio channel. An inverse channel response filter is then used to recombine the blurred copies of the symbol bits. This process is however complex and slow due to the locking time of the adaptive equalizer. Additionally it becomes increasing difficult to equalize signals that suffer ISI of more than a couple of symbol periods. OFDM overcomes the effects of multipath by breaking the signal into many narrow bandwidth carriers. This results in a low symbol rate reducing the amount of ISI. In addition to this, a guard period is added to the start of each symbol, removing the effects of ISI for multipath signals delayed less than the guard period (see section 2.3 for more detail). The high tolerance to multipath makes OFDM more suited to high data transmissions in terrestrial environments than single carrier transmissions.

Parameter	Transmission Mode			
	I	II	III	IV
Bandwidth	1.536 MHz	1.536 MHz	1.536 MHz	1.536 MHz
Modulation	DQPSK	DQPSK	DQPSK	DQPSK
Frequency Range (Mobile reception)	≤ 375 MHz	≤ 1.5 GHz	≤ 3 GHz	≤ 1.5 GHz
Number of subcarriers	1536	384	192	768
Symbol Duration	1000 µs	250 µs	125 µs	500 µs
Guard Duration	246 µs	62 µs	31 µs	123 µs
Total Symbol Duration	1246 µs	312 µs	156 µs	623 µs
Maximum Transmitter Separation for SFN	96 km	24 km	12 km	48 km

Table 1-1 shows the system parameters for DAB. DAB has four transmission modes. [2]

The transmission frequency, receiver velocity and required multipath tolerance all determine the most suitable transmission mode to use. Doppler spread is caused by rapid changes in the channel response due to movement of the receiver through a multipath environment. It results in random frequency modulation of the OFDM subcarriers, leading to signal degradation. The amount of Doppler spread is proportional to the transmission frequency and the velocity of movement. The closer the subcarriers are spaced together, the more susceptible the OFDM signal is to Doppler spread, and so the different transmission modes in DAB allow trade off between the amount of multipath protection (length of the guard period) and the Doppler spread tolerance.

The high multipath tolerance of OFDM allows the use of a Single Frequency Network (SFN), which uses transmission repeaters to provide improved coverage, and spectral efficiency. For traditional FM broadcasting, neighboring cities must use different RF frequencies even for the same radio station, to prevent multipath causes by rebroadcasting at the same frequency. However, with DAB it is possible for the same signal to be broadcast from every area requiring coverage, eliminating the need for different frequencies to be used in neighboring areas.

The data throughput of DAB varies from 0.6 - 1.8 Mbps depending on the amount of Forward Error Correction (FEC) applied. This data payload allows multiple channels to be broadcast as part of the one transmission ensemble. The number of audio channels is variable depending on the quality of the audio and the amount of FEC used to protect the signal. For telephone quality audio (24 kbps) up to 64 audio channels can be provided, while for CD quality audio (256 kb/s), with maximum protection, three channels are available. [1]

DIGITAL VIDEO BROADCASTING

The development of the Digital Video Broadcasting (DVB) standards was started in 199. DVB is a transmission scheme based on the MPEG-2 standard, as a method for point to multipoint delivery of high quality compressed digital audio and video. It is an enhanced replacement of the analogue television broadcast standard, as DVB provides a flexible transmission medium for delivery of video, audio and data services. The DVB standards specify the delivery mechanism for a wide range of applications, including satellite TV (DVB-S), cable systems (DVB-C) and terrestrial transmissions (DVB-T). The physical layer of each of these standards is optimized for the transmission channel being used. Satellite broadcasts use a single carrier transmission, with QPSK modulation, which is optimized for this application as a single carrier allows for large Doppler shifts, and QPSK allows for maximum energy efficiency. This transmission method is however unsuitable for terrestrial transmissions as multipath severely degrades the performance of high-speed single carrier transmissions. For this reason, OFDM was used for the terrestrial transmission standard for DVB. The physical layer of the DVB-T transmission is similar to DAB, in that the OFDM transmission uses a large number of subcarriers to mitigate the effects of multipath. DVB-T allows for two transmission modes depending on the number of subcarriers used. Table 1-2 shows the basic transmission parameters for these two modes. The major difference between DAB and DVB-T is the larger bandwidth used and the use of higher modulation schemes to achieve a higher data throughput. The DVB-T allows for three subcarrier modulation schemes: QPSK, 16-QAM (Square Amplitude Modulation) and 64- QAM; and a range of guard period lengths and coding rates. This allows the robustness of the transmission link to be traded at the expense of link capacity. Table 1-3 shows the data throughput and required SNR for some of the transmission combinations.

DVB-T is a uni-directional link due to its broadcast nature. Thus any choice in data rate verses robustness affects all receivers. If the system goal is to achieve high reliability, the data rate must be lowered to meet the conditions of the worst receiver. This effect limits the usefulness of the flexible nature of the standard. However if these same principles of a flexible transmission rate are used in bi-directional communications, the data rate can be maximized based on the current radio conditions. Additionally for multiuser applications, it can be optimized for individual remote transceivers.

parameter	2k Mod	8k Mode
Number subcarriers	1705	6817
Useful Symbol Duration (T_u)	896 µs	224 µs
Carrier Spacing (1/ T_u)	1116 Hz	4464 Hz
Bandwidth	7.61 MHz	7.61 MHz

Table 1-2 DVB transmission parameters [6].

Subcarrier Modulation	Code Rate	SNR for BER = 2×10^{-4} after Viterbi (dB)		Bit rate (Mbps) Guard Period (Fraction of Useful symbol duration)	
		Gaussian Channel	Rayleigh Channel	1/4	1/32
QPSK	½	3.1	5.4	4.98	6.03
QPSK	7/8	7.7	16.3	8.71	10.56
16-QAM	½	8.8	11.2	9.95	12.06
16-QAM	7/8	13.9	22.8	17.42	21.11
64-QAM	½	14.4	16.0	14.93	18.10
64-QAM	7/8	20.1	27.9	26.13	31.67

Note: Code rate can be any of the following values: 1/2, 2/3, 3/4, 5/6, 7/8. The Guard Period duration can be any following values: 1/4, 1/8, 1/16, 1/32.

HIPERLAN2 AND IEEE802.11A

Development of the European Hiperlan standard was started in 1995, with the final standard of HiperLAN2 being defined in June 1999. HiperLAN2 pushes the performance of WLAN systems, allowing a data rate of up to 54 Mbps [97]. HiperLAN2 uses 48 data and 4 pilot subcarriers in a 16 MHz channel, with 2 MHz on either side of the signal to allow out of band roll off. User allocation is achieved by using TDM, and subcarriers are allocated using a range of modulation schemes, from BPSK up to 64-QAM, depending on the link quality. Forward Error Correction is used to compensate for frequency selective fading. IEEE802.11a has the same physical layer as HiperLAN2 with the main difference between the standard corresponding to the higher-level network protocols used. Since the physical layer of HiperLAN2 is very similar to the IEEE802.11a standard these examples are applicable to both standards.

Standard	802.11b	802.11a	HiperLAN2
Spectrum	2.4 GHz	5.2 GHz	5.2 GHz
Modulation Technique	DSSS	OFDM	OFDM
~ Max physical rate	11 Mbps	54 Mbps	54 Mbps
~ Max data rate, layer 3	5 Mbps	32 Mbps	32 Mbps
Medium access control	CSMA/CA		TDMA/TDD
Connectivity	Conn. less	Conn. less	Conn. orientated

Table 1-4, Summary of characteristics of IEEE802.11b, IEEE802.11a and HiperLAN2. [7]

Parameter	Value
Channel Spacing	20 MHz
IFFT used for 20 MSPS	64
Data Subcarriers	48
Pilot Subcarriers	4
Carrier Spacing (F_c)	312.5 kHz (=20 MHz/64)
Nominal Bandwidth	16.25 MHz (=312.5 kHz × 52)
Useful Symbol Period	3.2 μsec (=1/ F_c)
Guard Period	0.8 μsec
Modulation Schemes	BPSK, QPSK, 16-QAM, 64-QAM
Coding Rate	1/2, 2/3, 3/4

Table 1-5, Physical Layer for HiperLAN2 and IEEE802.11a. Derived from [7]

INTRODUCTION

The following chapter gives an overview of how and where programmable logic devices are used. It gives a brief history of the programmable logic devices and goes on to describe the different ways of designing with PLDs.

2.1 The History of Programmable Logic

By the late 70's, standard logic devices were the rage and printed circuit boards were loaded with them. Then someone asked the question: "What if we gave the designer the ability to implement different interconnections in a bigger device?" This would allow the designer to integrate many standard logic devices into one part. In order to give the ultimate in design flexibility Ron Cline from Signetics (which was later purchased by Philips and then eventually Xilinx) came up with the idea of two programmable planes. The two programmable planes provided any combination of 'AND' and 'OR' gates and sharing of AND terms across multiple OR's.

This architecture was very flexible, but at the time due to wafer geometry's of 10um the input to output delay or propagation delay (Tpd) was high which made the devices relatively slow.

- Tow programmable planes
- Any combination of ANDs/ Ors
- Sharing of AND terms across multiple Ors
- Highest logic density available to user
- High Fuse count, Slower than PALs
- Programmable Logic Array PLA

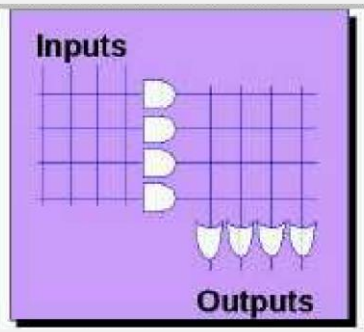


FIGURE 1.1 SPLD ARCHITECTURES PLA [8]

MMI (later purchased by **AMD**) was enlisted as a second source for the PLA array but after fabrication issues was modified to become the Programmable Array Logic (PAL) architecture by fixing one of the programmable planes. This new architecture differs from that of the PLA by having one of the programmable planes fixed - the OR array.

This PAL architecture had the added benefit of faster Tpd and less complex software but without the flexibility of the PLA structure. Other architectures followed, such as the PLD (Programmable Logic Device). This category of devices is often called Simple PLD (SPLD).

- One programmable plane –AND/ Fixed OR
- Finite combination of ANDs / Ors Medium Logic density available to user
- Lower Fuse Count, Faster than PLAs
- Programmable Array Logic -PAL

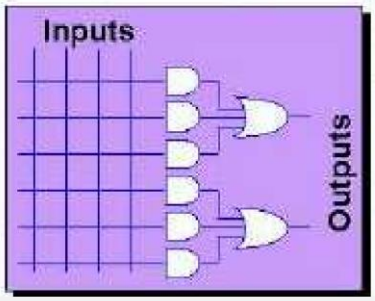


FIGURE 1.2 SPLD ARCHITECTURES PAL[8]

The architecture has a mesh of horizontal and vertical interconnect tracks. At each junction, there is a fuse. With the aid of software tools, the user can select which junctions will not be connected by "blowing" all unwanted fuses. (This is done by a device programmer or more commonly nowadays using In-System Programming or ISP).

Input pins are connected to the vertical interconnect and the horizontal tracks are connected to AND-OR gates, also called "product terms". These in turn connect to dedicated flip-flops whose outputs are connected to output pins.

PLDs provided as much as 50 times more gates in a single package than discrete logic devices! A huge improvement, not to mention fewer devices needed in inventory and higher reliability over standard logic.

Programmable Logic Device (PLD) technology has moved on from the early days with such companies as Xilinx producing ultra low power CMOS devices based on Flash technology. Flash PLDs provide the ability to program the devices time and time again electrically programming and ERASING the device! Gone are the days of erasing taking in excess of twenty minutes under an UV eraser.

2.2 Complex Programmable Logic Devices (CPLDs)

Complex Programmable Logic Devices (CPLD) are another way to extend the density of the simple PLDs. The concept is to have a few PLD blocks or macrocells on a single device with general purpose interconnect in between. Simple logic paths can be implemented within a single block. More sophisticated logic will require multiple blocks and use the general purpose interconnect in between to make these connections.

- Central, global Interconnect
- Simple, Deterministic timing
- Easily routed
- PLD tools add only interconnect
- Wide, Fast complex gating

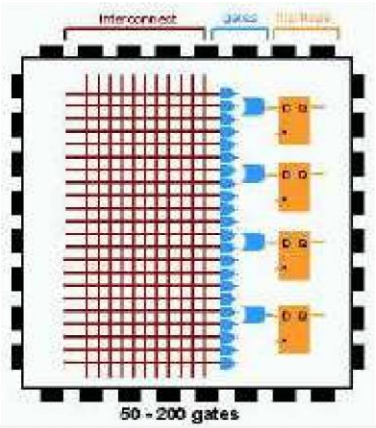


FIGURE 1.3 CPLD ARCHITECTURE[8]

CPLDs are great at handling wide and complex gating at blistering speeds e.g. 5ns which is equivalent to 200MHz. The timing model for CPLDs is easy to calculate so before you even start your design you can calculate your in to output speeds.

2.2.1 Why Use a CPLD?

CPLDs enable ease of design, lower development costs, more product revenue for your money, and the opportunity to speed your products to market...

2.2.1.1-Ease of Design:

CPLDs offer the simplest way to implement design. Once a design has been described, by schematic and/or HDL entry, a designer simply uses CPLD development tools to optimise, fit, and

simulate the design. The development tools create a file, which is then used to customise (program) a standard off-the-shelf CPLD with the desired functionality. This provides an instant hardware prototype and allows the debugging process to begin. If modifications are needed, design changes are just entered into the CPLD development tool, and the design can be re-implemented and tested immediately.

2.2.1.2-Lower Development Costs:

CPLDs offer very low development costs. Ease of design, as described above, allows for shorter development cycles. Because CPLDs are re-programmable, designers can easily and very inexpensively change their designs. This allows them to optimise their designs and continues to add new features to continue to enhance their products. CPLD development tools are relatively inexpensive and in the case of Xilinx, are free. Traditionally, designers have had to face large cost penalties such as re-work, scrap, and development time. With CPLDs, designers have flexible solutions thus avoiding many traditional design pitfalls.

2.2.1.3-More Product Revenue:

CPLDs offer very short development cycles, which means your products get to market quicker and begin generating revenue sooner. Because CPLDs are re-programmable, products can be easily modified using ISP over the Internet. This in turn allows you to easily introduce additional features and quickly generate new revenue from them. (This results in an expanded time for revenue). Thousands of designers are already using CPLDs to get to market quicker and then stay in the market longer by continuing to enhance their products even after they have been introduced into the field. CPLDs decrease Time To Market (TTM) and extend Time In Market (TIM).

2.2.1.4-Reduced Board Area:

CPLDs offer a high level of integration (large number of system gates per area) and are available in very small form factor packages. This provides the perfect solution for designers of products which must fit into small enclosures or who have a limited amount of circuit board space to implement the logic design. The CoolRunner CPLDs are available in the latest chip scale packages, e.g. CP56 which has a pin pitch of 0.5mm and is a mere 6mm by 6mm in size so are ideal for small, low power end products.

2.2.1.5-Cost of Ownership:

Cost of Ownership can be defined as the amount it costs to maintain, fix, or warranty a product. For instance, if a design change requiring hardware rework must be made to a few prototypes, the cost might be relatively small. However, as the number of units that must be changed increases, the cost can become enormous. Because CPLDs are re-programmable, requiring no hardware rework, it costs much less to make changes to designs implemented using them. Therefore cost of ownership is dramatically reduced. And don't forget the ease or difficulty of design changes can also affect opportunity costs. Engineers who are spending a lot of time fixing old designs could be working on introducing new products and features - ahead of the competition.

There are also costs associated with inventory and reliability. PLDs can reduce inventory costs by replacing standard discrete logic devices. Standard logic has a predefined function and in a typical design lots of different types have to be purchased and stocked. If the design is changed then there may be excess stock of superfluous devices. This issue can be

alleviated by using PLDs i.e. you only need to stock one device and if your design changes you simply reprogram. By utilising one device instead of many your board reliability will increase by only picking and placing one device instead of many. Reliability can also be increased by using the ultra low power CoolRunner CPLDs i.e. lower heat dissipation and lower power operation leads to decreased Failures In Time (FIT).

2.3 Field Programmable Gate Arrays (FPGAs)

In 1985, a company called Xilinx introduced a completely new idea. The concept was to combine the user control and time to market of PLDs with the densities and cost benefits of gate arrays. A lot of customers liked it - and the FPGA was born. Today Xilinx is still the number one FPGA vendor in the world!

An FPGA is a regular structure of logic cells or modules and interconnect which is under the designer's complete control. This means the user can design, program and make changes to his circuit whenever he wants. And with FPGAs now exceeding the 10 million gate limit (Xilinx Virtex II is the current record holder), the designer can dream big!

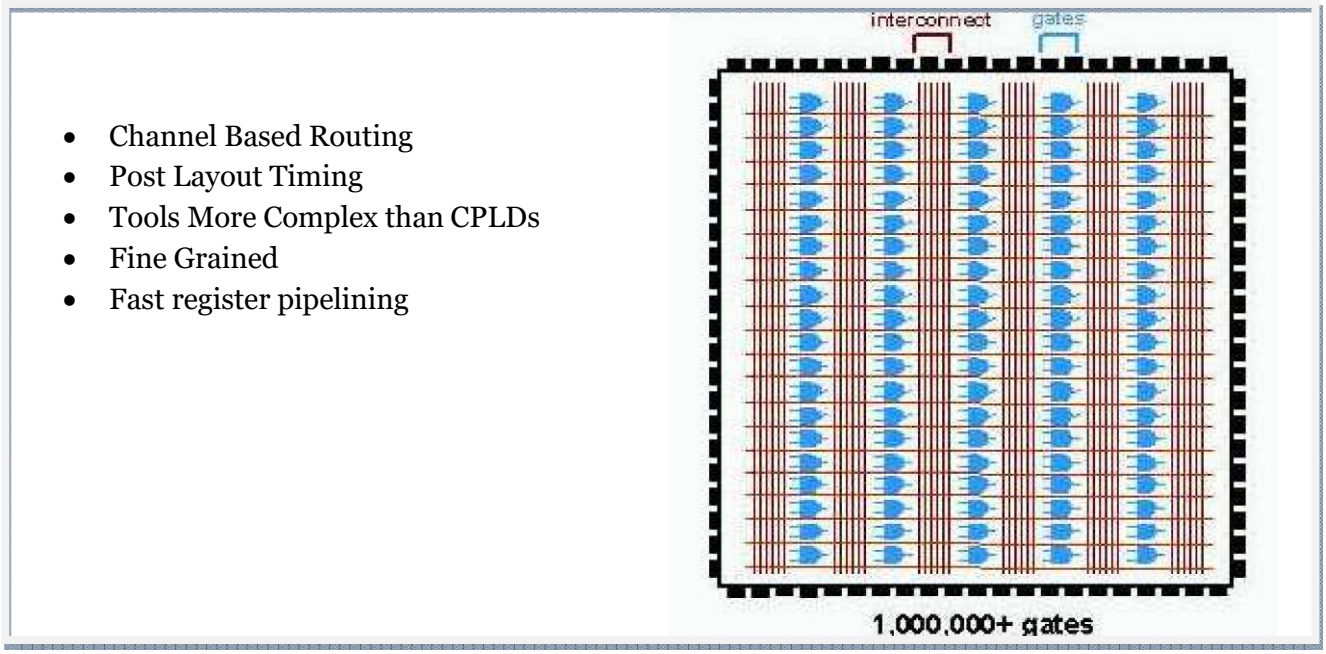


FIGURE 1.4 FPGA ARCHITECTURE[8]

With the introduction of the Spartan range of FPGAs we can now compete with Gate Arrays on all aspects - price, gate and I/O count, performance and cost! The new Spartan IIE will provide up to 300k gates at a price point that enables Application Specific Standard Product (ASSP) replacement. For example a Reed Solomon IP Core implemented in a Spartan II XC2S100 FPGA has an effective cost of $9.95 whereas the equivalent ASSP would cost around $20.

There are 2 basic types of FPGAs: SRAM-based reprogrammable and One-time programmable (OTP). These two types of FPGAs differ in the implementation of the logic cell and the mechanism used to make connections in the device.

The dominant type of FPGA is SRAM-based and can be reprogrammed by the user as often as the user chooses. In fact, an SRAM FPGA is reprogrammed every time it is powered-up because the FPGA is really a fancy memory chip! (That's why you need a serial PROM or system memory with every SRAM FPGA).

2-4 Design Integration

The integration of 74 series standard logic into a low cost CPLD is a very attractive proposition. Not only do you save Printed Circuit Board (PCB) area and board layers therefore reducing your total system cost but you only have to purchase and stock one generic part instead of up to as many as twenty pre-defined logic devices. In production the pick and place machine only has to place one part - therefore speeding up production. Less parts means higher quality and better Failure In Time (FIT) factor.

By using Xilinx CoolRunner devices (our family of ultra low power parts) in a design customers can benefit from low power consumption and reduced thermal emissions. This in turn leads to the reduction of the use of heat sinks (another cost saving) and a higher reliability end product.

2.4.1-Basic logic definitions

Standard logic

Fixed function devices which can be connected together to implement a system

Pld (CPLDs & FPGAs)

Devices which can be re-programmed to implement any function within the device's resources

Gate array (GA)

Blocks of gates that are customized at the fab by adding layers of metal interconnects

Standard Cell (ASIC)

ICs designed with cell libraries. All mask layers customized at the fab

2.5 The Basic Design Process

The availability of design software such as WebPACK ISE has made it much easier to design with programmable logic. Designs can be described easily and quickly using either a description language such as ABEL (Advanced Boolean Expression Language), VHDL (VHSIC Hardware Description Language), Verilog or via a schematic capture package. schematic capture is the traditional method that designers have used to specify gate arrays and programmable logic devices. It is a graphical tool that allows the designer to specify the exact gates he requires and how he wants them connected. There are 4 basic steps to using schematic capture.

Step one : After selecting a specific schematic capture tool and device library, the designer begins building his circuit by loading the desired gates from the selected library. He can use any combination of gates that he needs. A specific vendor and device family library must be chosen at this time (e.g. Xilinx XCR3256XL) but he doesn't have to know what device within that family he will ultimately use with respect to package and speed.

Step two: Connect the gates together using nets or wires. The designer has complete control of connecting the gates in whatever configuration is required for his application.

Step three: The input and output buffers are added and labelled. These will define the I/O package pins for the device.

Step four: The final step is to generate a netlist.

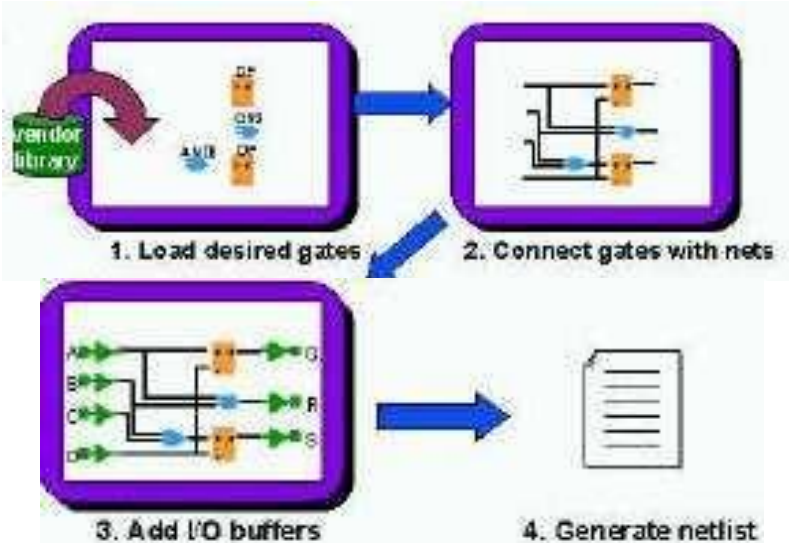


FIGURE 1.5 PLD DESIGN FLOW (SCHEMATIC CAPTURE) [8]

The netlist is a text equivalent of the circuit which is generated by design tools such as a schematic capture program. The netlist is a compact way for other programs to understand what gates are in the circuit, how they are connected and the names of the I/O pins.

In the example below, the netlist reflects the actual syntax for the circuit in the schematic. There is one line for each of the components and one line for each of the nets. Note that the computer assigns names to components (G1 to G4) and the nets (N1 to N8). When we implement this design, it will have input package pins A, B, C, D and output pins Q, R, S.

EDIF (Electronic Digital Interchange Format) is the industry-wide standard for netlists although there are many other including vendor- specific ones such as the Xilinx Netlist Format (XNF). If you have the design netlist, you have all you need to determine what the circuit does.

- Graphical description
- Created by humans
- For reading by humans

- Text file
- Output of schematic capture
- PC friendly
- EDIF is industry standard
- XNF is Xillinx Netlist Format

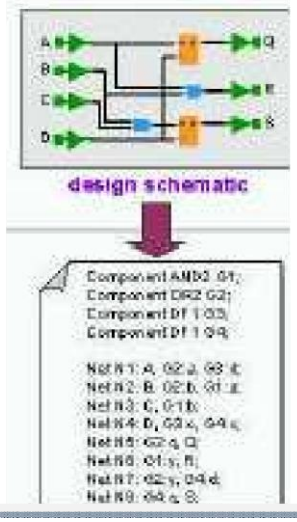


FIGURE 1.6 DESIGN SPECIFICAT ION – NETLIST

The examples on the previous pages are obviously very simplistic. A more realistic design of 10,000 equivalent gates is shown here.

The typical schematic page contains about 200 gates included the logic contained with soft macros. Therefore, it would require 50 schematic pages to create a 10,000 gate design! Each page needs to go through all the steps mentioned previously: adding components, interconnecting the

gates, adding I/Os and generating a netlist! This is rather time-consuming, especially if you want to design a 20k, 50k or larger design. Another inherent problem with using schematic capture is the difficulty in migrating between vendors and technologies. If you initially create your 10,000 gate design with FPGA vendor X and then want to migrate to a gate array, you would have to modify every one of those 50 pages using the gate array vendor's component library! There has to be a better way…

And of course, there is. It's called High Level Design (HLD), Behavioural or Hardware Description Language (HDL). For our purposes, these three terms are essentially the same thing.

The idea is to use a high-level language to describe the circuit in a text file rather than a graphical low-level gate description. The term Behavioural is used because in this powerful language, the designer describes the function or behaviour of the circuit in words rather than figuring out the appropriate gates needed to create the application.

There are two major flavours of HDL: VHDL and Verilog. Although it's not really important for you to know, VHDL is an acronym for "VHSIC High-level Design Language". And yes, VHSIC is another acronym "Very High Speed Integrated Circuit".

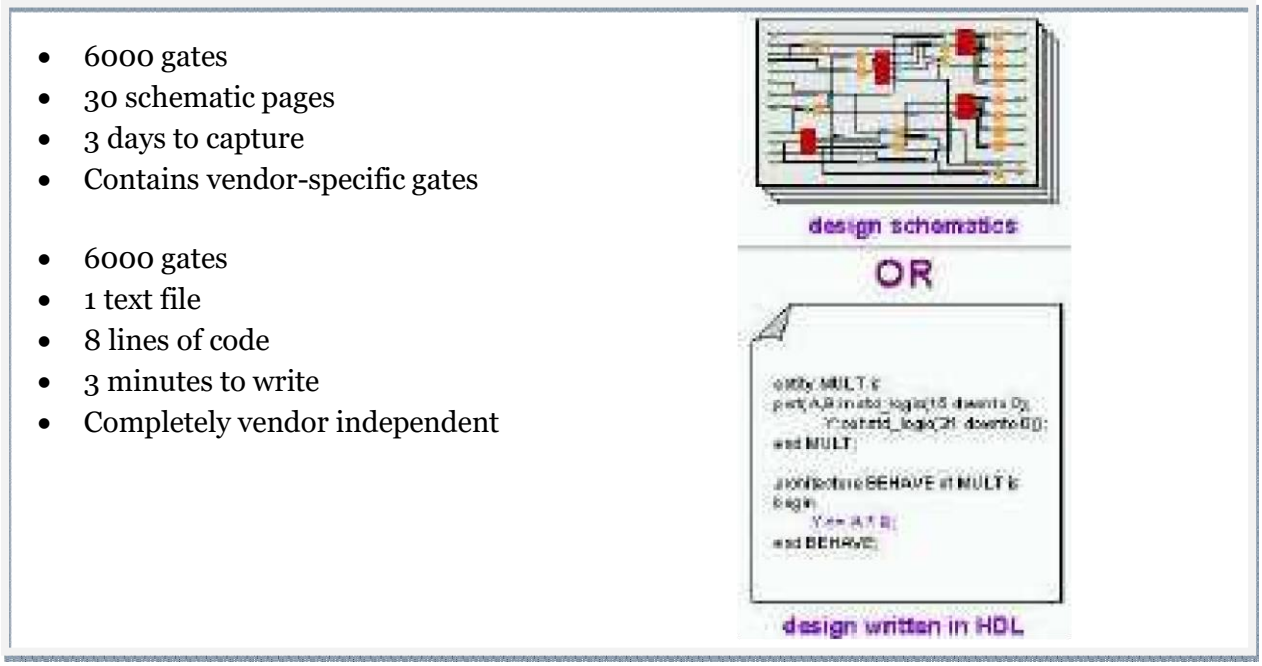


FIGURE 1.7 DESIGN SPECIFICATION – MULTIPLIER [8]

As an example we will design a 16 by 16 multiplier specified with a schematic and with an HDL file. A multiplier is a regular but complex arrangement of adders and registers which requires quite a few gates. Our example has two 16 bit inputs (A and B) and a 32 bit product Output (Y=A*B) - that's a total of 64 I/Os. This circuit requires Approximately 6,000 equivalent gates.

In the schematic implementation, all the required gates would have to be loaded, positioned on the page, interconnected, and I/O buffers added. About 3 days worth of work. The HDL implementation, which is also 6,000 gates, requires 8 lines of text and can be done in 3 minutes. This file contains all the information necessary to define our 16x16 multiplier!

So, as a designer, which method would you choose? In addition to the tremendous time savings, the HDL method is completely vendor- independent. That means that this same code could be used to implement a Xilinx FPGA as an LSI Logic gate array! This opens up tremendous design possibilities for engineers. For example, what if you wanted to create a 32X32 multiplier

Obviously, you would want to modify the work already done for the smaller multiplier. For

the schematic approach, this would entail making 3 copies of the 30 pages, then figuring out where to edit the 90 pages so that they addressed the larger bus widths. This would probably require 4 hours of graphical editing. For the HDL specification, it would be a matter of changing the bus references: change 15 to 31 in line 2 and 31 to 63 in line 3 (4 seconds)!

2.6 HDL File Change Example

Before (16x 16 multiplier):

```
entity MULT is
    port(A,B:in std_logic(15 downto 0);
    Y:out std_logic(31 downto 0));
    end MULT;
architecture BEHAVE of MULT is begin
    Y <= A * B;
    end BEHAVE;
```

After (32 x 32 multiplier):

```
entity MULT is
    port(A,B:in std_logic(31 downto 0);
    Y:out std_logic(63 downto 0));
    end MULT;
architecture BEHAVE of MULT is begin
    Y <= A * B;
    end BEHAVE;
```

So HDL is ideal for design re-use, you can share you 'library' of parts with other designers at your company therefore saving and avoid duplication of effort. I think you can see now why HDL is the way to design logic circuits!

So, now that we have specified the design in a behavioural description, how do we convert this into gates, which is what all logic devices are made of?

The answer is Synthesis. It is the synthesis tool that does the intensive work of figuring out what gates to use based on the high level description file provided by the designer. (Using schematic capture, the designer has to do this all this manually). Since the resulting netlist is vendor and device family specific, the appropriate vendor library must be used. Most synthesis tools support a large range of gate array, FPGA and CPLD device vendors.

In addition, the user can specify optimization criteria that the synthesis tool will take into account when selecting the gate-level selection or Mapping. Some of these options include: optimize the complete design for the least number of gates, optimize a certain section of the design for fastest speed, use the best gate configuration to minimize power, use the FPGA-friendly register rich configuration for state machines. The designer can easily experiment with different vendors, device families and optimization constraints thus exploring many different solutions instead of just one with the schematic approach.

To recap, the advantages of high level design & synthesis are many. It is much simpler and faster to specify your design using HLD. And much easier to make changes to the design by the designer or another engineer because of the self-documenting nature of the language. The

designer is relieved from the tedium of selecting and interconnecting at the gate level. He merely selects the library and optimization criteria (e.g. speed, area) and the synthesis tool will determine the results. The designer can thereby try different design alternatives and select the best one for the application. In fact, there is no real practical alternative for designs exceeding 10,000 gates.

2.7 Intellectual Property (IP) Cores

Intellectual Property (IP) Cores are defined as very complex pre-tested system-level functions that are used in logic designs to dramatically shorten development time. The IP Core benefits are:

- Faster Time-to-Market
- Simplifies the development process
- Minimal Design Risk
- Reduces software compile time
- Reduced verification time
- Predictable performance/functionality

IP Cores are similar to vendor-provided soft macros in that they simplify the design specification step by removing the designer from gate-level details of commonly used functions. IP Cores differ from soft macros in that they are generally much larger system-level functions such as PCI bus interface, DSP filter, PCMCIA interface, etc. They are extensively tested (and hence rarely free of charge) to offload the designer from having to verify the IP Core functions himself

2.8 Design Verification

To verify a programmable logic design we will probably use a simulator, which is a software program to verify the functionality and/or timing of a circuit.

The industry-standard formats used ensure that designs can be re-used and there is no concerns if a vendors changes their libraries - no rework is necessary, just a synthesis recompile. Even if the customer decides to move to a different vendor and/or technology, it is just a compile away after selecting the new library. It's even design tool independent so the designer can try synthesis tools from different vendors and pick the best results!

It is more common to have cores available in HDL format since that makes them easier to modify and use with different device vendors. After completing the design specification, you need to know if the circuit actually works as it's supposed to. That is the purpose of Design Verification. A simulator is used to well ... simulate the circuit. You need to provide the design information (via the netlist after schematic capture or synthesis) and the specific input pattern or Test Vectors that you want checked. The simulator will take this information and determine the outputs of the circuit.

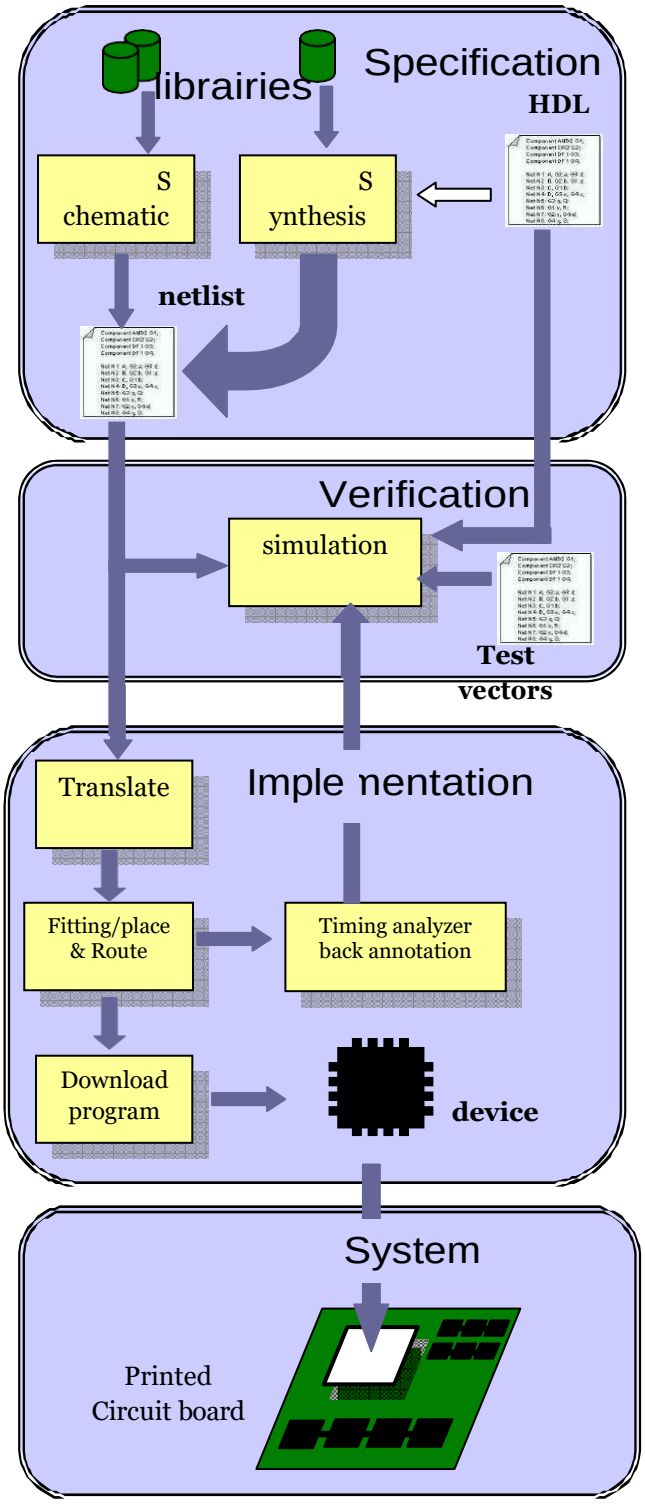


FIGURE 1.8 THE PLD DESIGN FLOW [8]

2.8.1-Functional Simulation

At this point in the design flow, we are doing a Functional Simulation which means we are only checking to see if the circuit gives us the right combinations of ones and zeros. We will do Timing Simulation a little later in the design flow.

If there are any problems, the designer goes back to the schematic or HDL file, makes the changes, re-generates the netlist and then reruns the simulation. Designers typically spent 50% of the development time going through this loop until the design works as required.

Using HDL offers an additional advantage when verifying the design. You can simulate directly from the HDL source file. This by passes the time-consuming synthesis process that would be required for every design change iteration. Once the circuit works correctly, we would need to run the synthesis tool to generate the netlist for the next step in the design flow - Device Implementation.

2.8.2 Device Implementation

We now have a design netlist that completely describes our design using the gates for a specific vendor/ device family and it has been fully verified. It is now time to put this in a chip, referred to as Device Implementation.

Translate consists of a number of various programs that are used to import the design netlist and prepare it for layout. The programs will vary among vendors. Some of the more common programs during translate include: optimisation, translation to the physical device elements, device-specific design rule checking (e.g. does the design exceed the number of clock buffers available in this device). It is during the stage of the design flow that you will be asked to select the target device, package, speed grade and any other device-specific options.

The translate step usually ends with a comprehensive report of the results of all the programs executed. In addition to warnings and errors, there is usually a listing of device and use I/O , which helps the designer to determine if he has selected the best device.

2.8.3 Fitting

For CPLDs, the design step is called Fitting to “Fit” the design to the target device. In the diagram above, a section of the design is fit to the CPLD. CPLDs are a fixed architecture so the software needs to pick the gates and interconnect paths that match the circuit. This is usually a fast process.

The biggest potential problem here is if the designer has previously assigned the exact locations of the I/O pins, commonly referred to as Pin Locking. (Most often this is from previous design iteration and has now been committed to the printed circuit board layout). Architectures (like the Xilinx XC9500 & CoolRunner CPLDs) that support I/O pin locking have a very big advantage. They permit the designer to keep the original I/O pin placements regardless of the number of design changes, use or required performance.

Pin locking is very important when using In-System Programming - ISP. This means that if you layout your PCB to accept a specific pin out then if you need to change the design you can re-programme confident that you pin out will stay the same.

2.8.4 Place and Route

For FPGAs, the Place and Route programs are run after Compile. “Place” is the process of selecting specific modules or logic blocks in the FPGAs where design gates will reside. “Route” as the name implies, is the physical routing of the interconnect between the logic blocks.

Most vendors provide automatic place and route tools so the user does not have to worry about the intricate details of the device architecture. Some vendors have tools that allow expert users to manually place and/or route the most critical parts of their designs and achieve better performance than with the automatic tools. Floorplanner is a form of such manual tools.

These two programs require the longest time to complete successfully since it is a very complex task to determine the location of large designs, ensure they all get connected correctly, and meet the desired performance. These programs however, can only work well if the target architecture has sufficient routing for the design. No amount of fancy coding can compensate for an ill-conceived architecture, especially if there is not enough routing tracks. If the designer faces this problem, the most common solution to is to use a larger device. And he will likely remember the experience the next time he is selecting a vendor.

A related program is called Timing-Driven Place & Route (TDPR). This allows users to specify timing criteria that will be used during device layout. A Static Timing Analyser is usually part of the vendor's implementation software. It provides timing information about paths in the design.

This information is very accurate and can be viewed in many different ways (e.g. display all paths in the design and rank them from longest to shortest delay).

In addition, the user at this point can use the detailed layout information after reformatting, and go back to his simulator of choice with detailed timing information. This process is called Back- Annotation and has the advantage of providing the accurate timing as well as the zeros and ones operation of his design.

In both cases, the timing reflects delays of the logic blocks as well as the interconnect. The final implementation step is the Download or Program.

2.8.5 Downloading or Programming

Download generally refers to volatile devices such as SRAM FPGAs. As the name implies, you download the device configuration information into the device memory. The Bitstream that is transferred contains all the information to define the logic and interconnect of the design and is different for every design. Since SRAM devices lose their configuration when the power is turned off, the bitstream must be stored somewhere for a production solution. A common such place is a serial PROM. There is an associated piece of hardware that connects from the computer to a board containing the target device.

Program is used to program all non-volatile programmable logic devices including serial PROMs. Programming performs the same function as download except that the configuration information is retained after the power is removed from the device. For anti fuse devices, programming can only be done one per device. (Hence the term One-Time Programmable, OTP).

2.8.6 System De bug

At this point in the design flow, the device is now working but we're not done yet. We need to do a System Debug - verify that our device works in the actual board. This is truly the moment of truth because any major problems here means the engineer has made a assumption on the device specification that is incorrect or has not considered some aspect of the signal required to/from the programmable logic device. If so, he will then collect data on the problem and go back to the drawing (or behavioural) board!

VHDL LANGUAGE

2.9 VHDL design Units

The VHDL language is specifically tailored to designing circuits at both behavioral and gate levels. Although it may be possible to use VHDL to program general purpose software, that is not the intention of the language. The very structure of the language suggests hardware design. There are three components to the basic VHDL program. They are: Entity, Architecture and Configuration.

Briefly, the entity describes the interface to the circuit, the architecture describes the behavior of the "black box", and the configuration binds the architecture and entity together as well as all internally declared components.

These are two other components found in more complicated VHDL programs. They are the package and package body. Briefly, the PACKAGE defines constants and interfaces to the routines found within the PACKAGE BODY. The following sections will detail these statements and also introduce the actual language constructs.

2.9.1 Entity

The entity construct declares the interface to the system. This interface can be viewed as a "black box". A very simple example would be to look a NAND gate. The simple NAND gate is pictured in Figure 1, and VHDL code in Listing 1.

(Note: In VHDL a comment is indicated by two dashes '--'. Also, by convention, reserved words are all in capitols.)

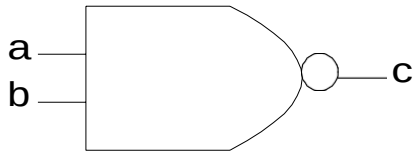


FIGURE 1. SIMPLE NAND GATE [9]

```
LIBRARY ieee;          -- use the ieee library
USE ieee.std_logic_1164.ALL;
ENTITY nand IS
   GENERIC(out_delay  : time := 5 ns); --Delay info
   PORT(a: IN std_logic;     -- Inputs
        b: IN std_logic;
        c: OUT std_logic);   -- Outputs
END nand;
```

Listing 1. Flip-Flop ENTITY Statement [9]

Here we define generics which are constants passed into components, usually counts or delays and the port statement allows us to define system I/O. The following are signal types which can be declared in the port statement:

IN	Input to the system
OUT	Output from the system
INOUT	A bi-directional SIGNAL
BUFFER	A register attached to an output. (note: it is normally impossible to read an output, but a buffer allows this)

Entities are also important in that it is possible to include passive processes within

the entity. The impact of this is that it is possible to include setup and hold checking. This is only one possible use, but the one of greatest importance.

2.9.2 Architecture

The architecture is the actual structure which makes up a given design. It is in here where we describe in some manner the actual behavior or function which defines the operation of the system. There are two methods used for writing VHDL, behavioral and structural models.

2.9.2.1 Behavioral Models

A behavioral model is one which defines the behavior of a system, how a system acts. We can flesh out the NAND gate above in a behavioral model as follows:

```
ARCHITECTURE behavior OF nand IS
   -- declare internal signals or aliases
BEGIN
   c <= NOT(a AND b);
END behavior;
```

Listing 2. Nand Architecture [9]

The above code could be taken as behavioral or functional as NAND is aVHDL primitive. A larger purely behavioral code will be introduced later.

2.9.2.2 Structural Models

A Structural model can be many levels deep, starting with primitive gates and building to describe a complete system. An example of structural code would be the following RS-Flip Flop constructed from the simple NAND gate above.

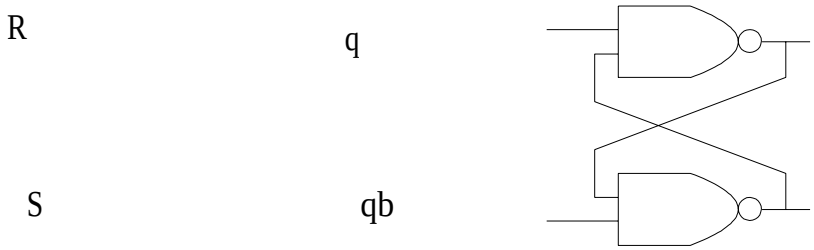


FIGURE 3. RS-FLIP FLOP [9]

```
ENTITY rsff IS

Port (R : IN std_logic;
S : IN std_logic;
q : OUT std_logic;
qb : OUT std_logic);

END rsff;

ARCHITECTURE behav OF rsff IS
COMPONENT nand            -- define our nand gate
GENERIC
(delay: time);            -- this is a copy of the
PORT(a: IN std_logic; -- nand entity statement
b: IN std_logic;
```

```
c: OUT std_logic);
END COMPONENT;
BEGIN
u1: nand              -- instantiate u1 as a nand component
GENERIC MAP(5 ns)               -- here you would change delay
values
PORT MAP(s, qb, q); -- map I/O to components

u2: nand              -- instantiate u2 as a nand component
GENERIC MAP(5 ns)
PORT MAP(q, r, qb);
END behav;
```

LISTING 4. RS FLIP FLOP EXAMPLE [9]

The above code illustrates many powerful features of VHDL. The Flip Flop (FF) is defined as two interconnected NAND gates, and the NAND gates are defined in separate VHDL code. This allows us to use different levels in coding. For example, we could then instantiate the RS-FF component to form a shift register, and from there a bank of registers, etc.

2.9.3 Configuration

The configuration section of VHDL allows us to bind our entities and architecture's together to form a single design unit. It is this which allows

us to have different architecture's bound to an entity statement. This allows us to start our coding with a pure behavioral function and then to progress through synthesis or hand coding to a structural level. This level would be more akin to a gate level description of the circuit. Either of these could then be bound, through the configuration, to the entity which defines the I/O.

Also, we could have different entities, one with setup and hold checking and one without. Thus once we knew that the interface meets setup and hold times, we could forgo the time checks for an increase in simulation speed. The following is the configuration for the above RS-FF.

```
CONFIGURATION con OF rsff IS FOR
  behav
        FOR u1, u2 : nand
             USE ENTITY work.nand(behavior);
        END FOR
     END FOR
  END con;
```

LISTING 5. RS FLIP FLOP CONFIGURATION [9]

This configuration binds the NAND component to the RS FF architecture. If there were multiple definitions for NAND, the compiler would be unable to know which one to use without this. The USE clause defines which ARCHITECTURE will be bound to the configuration. The syntax is:

```
USE ENTITY library.design_entity;
USE ARCHITECTURE library.design_architecture;
```

The USE statement allows us to mix and match design units. We can substitute behavioral and gate level models by changing a line.

2.9.4 Packages and Package Bodies

Packages allow convenient ways of defining functions and constants which will be used in multiple VHDL programs. Packages act like an ENTITY, they declare the interfaces to the functions and subprograms found in the package body. Packages can't contain the actual subprogram or function, they must be included in the package body. Packages are also used to hide a designs complexity. For example, at Sanders, we have defined a parity function parity() which will return the odd parity of any std_logic_vector() passed into it. This is a standard package which is available to all of our designers and can be used by including it in a given piece of code.

A package body just contains the functionality of a given package, much like the architecture associated with a given entity. The difference is that only one package body can be associated with a given package. Below is the package which contains the parity function:

```
        USE    ieee.std_logic_1164.ALL;
PACKAGE sanders IS
        FUNCTION       parity(s:IN
             std_logic_vector)RETURN
        std_logic;
      END sanders;
      PACKAGE BODY sanders IS
        FUNCTION       parity(s:IN std_logic_vector) RETURN
        std_logic IS
          VARIABLE inter : std_logic := '0';
        BEGIN
          FOR i IN s'RANGE LOOP
               inter := inter XOR s(i);
          END LOOP;
          RETURN (NOT(inter));
        END parity;
      END sanders
```

LISTING 6. PARITY PACKAGE [9]

2.9.4.1 Overloading

VHDL allows the use of overloaded functions. Overloaded functions are two or more functions with the same name which have different signal lists, which may return different values. This allows the creation of one function type, i.e. parity, but use different parameters. For example, we could declare two parity functions, one for std_logic types and one for bittypes:

```
FUNCTION        parity(s:                    IN
      std_logic_vector)  RETURN
   std_logic;
FUNCTION parity(s : IN bit_vector) RETURN bit;
```

LISTING 7. OVERLOADING [9]

The above two functions could then be called at any point within a program. The compiler would pick the function which matched the parameters passed in. (i.e. if parity(bit_vector) was called, the compiler would know to use the second function and not the first.) The only stipulation is that the functions must have unique parameters. Otherwise the compiler will not know which function to use and will generate an error.

VHDLEXAMPLES

2.10 MUX EXAMPLE

Below is a complete 8-1 MUX constructed hierarchicaly from two 4-1 MUX's. This example will give a good idea as to how VHDL models are constructed, and that they actually can simulate hardware effectively. This is purely behavioral and of the form accepted by most synthesis tools.

2.10.1 Behavioral 4-1 MUX

We will construct behavioral code to represent the 4-1 MUX pictured in figure 3. This code will be purely behavioral in nature, utilizing the VHDL case statement.

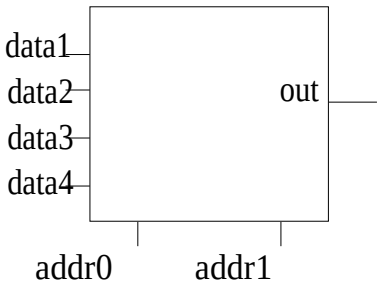


Figure 3. 4-1 MUX [9]

2.10.1.1 MUX Entity

The Interface to the 4-1 MUX is defined in the entity statement. The following code comprises the 4-1 MUX entity.

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY 4_1_mux IS
   GENERIC(out_delay : time := 5 ns);
PORT( data1: IN  std_logic;
      data2:    IN    std_logic;
      data3:    IN    std_logic;
      data4:     IN   std_logic;
      addr:   IN   std_logic_vector(1   DOWNTO   0);
      output: OUT std_logic);
END 4_1_mux;
```

LISTING 8: MUX ENTITY

2.10.1.2. Architecture

The actual behavior of the MUX is defined in the Architecture. The Fllowing code comprises the 4-1 MUX architecture.

```
     ARCHITECTURE  behav  OF  4_1_mux  IS
BEGIN
       mux_it:
       PROCESS(addr, data1, data2, data3, data4)
```

```
        BEGIN
          CASE addr IS
               WHEN "00" =>
                    output <= data1 AFTER out_delay;
               WHEN "01" =>
                    output <= data2 AFTER out_delay;
               WHEN "10" =>
                    output <= data3 AFTER out_delay;
               WHEN "11" =>
                    output <= data4 AFTER out_delay;
               WHEN OTHERS =>
                     ASSERT FALSE
                         REPORT "addr out of range!"
                              SEVERITY ERROR;
   END   PROCESS;   END
behav;
```

LISTING 9: MUX ARCHITECTURE [9]

2.10.1.3 Configuration

The VHDL code which binds the ENTITY and ARCHITECTURE together is in the CONFIGURATION. The following code comprises the configuration of the 4-1 MUX.

```
CONFIGURATION  config4  OF  4_1_mux  IS  FOR
   behav
   END    FOR;    END
config4;
```

Listing 10: MUX Configuration

Since no components internal to the architecture have been declared, the configuration is empty. This is often referred to as a default configuration and may be omitted.

2.10.2 Behavioral 8-1 MUX

A behavioral 8-1 MUX can be created by using the two behavioral 4-1 MUX's. This is a good example of structured VHDL.

2.10.2.1 Entity

The following code comprises the entity of the 8-1 MUX.

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY 8_1_mux IS
   GENERIC(out_delay : time := 5 ns);
PORT( data1: IN  std_logic;
      data2:    IN    std_logic;
      data3:    IN    std_logic;
      data4:    IN    std_logic;
      data5:    IN    std_logic;
      data6:    IN    std_logic;
      data7:    IN    std_logic;
      data8:     IN   std_logic;
      addr:   IN   std_logic_vector(3   DOWNTO   0);
      output: OUT std_logic);
END 8_1_mux;
```

LISTING 11 MUX ENTITY [9]

For this code, we define a delay for all signal assignments of 5ns. If we had written a flat behavioral model, we would use this delay for all gates. In this case, however, we are using a structural model and we will "map" values to the individual components.

2.10.2.2 Architecture

The following code comprises the architecture of the 8-1 MUX.

```
ARCHITECTURE behav8 OF 8_1_mux IS ALIAS
  addr3: std_logic IS addr(3);
      ALIAS       address:          std_logic_vector(1
        DOWNTO    0)                IS
      addr(1 DOWNTO 0);
      SIGNAL out1: std_logic;
      SIGNAL out2: std_logic;
        GENERIC(out_delay : time := 5 ns);-- this defines
        PORT( data1: IN std_logic;-- a default
                  data2: IN std_logic;-- delay of 5ns
                  data3: IN std_logic;-- if no MAP is
                  data4: IN std_logic;-- used
                  addr: IN std_logic_vector(1 DOWNTO
                                  0);
                  output:   OUT std_logic);
      END COMPONENT;
      u1: 4_1_mux
        GENERIC MAP(3.5 ns) -- a little slower
        PORT MAP(  data1 => data1,
                   data2 => data2,
                   data3 => data3,
                   data4 => data4,
                   addr => address,
                   output => out1);
      u2: 4_1_mux
        GENERIC MAP(5.5 ns) -- a little faster
        PORT MAP( data1 => data5,
                   data2 => data6,
                   data3 => data7,
                   data4 => data8,
                   addr => address,
                 output => out2);
      BEGIN          -- out2
        CASE addr3 IS
             WHEN '0' =>
                output <= out1;
             WHEN '1' =>
                output <= out2;
             WHEN OTHERS =>
                   ASSERT FALSE
                       REPORT "address error"
                            SEVERITY ERROR;
        END CASE;

  END  PROCESS;  END
behav8;
```

LISTING 12 MUX ARCHITECTURE [9]

A GENERIC MAP is used to specify different delays for components than the delays specified in the components. This allows an easy way of changing parameters passed into a component. The example above uses generics and generic maps to provide timing information. A GENERIC MAP will always override any previous GENERIC clauses. A generic can be thought of as a default value, and a generic map overides the default.

2.10.2.3 Configuration

The following code comprises the configuration for the 8-1 MUX.

```
CONFIGURATION config OF 4_1_mux IS FOR
  behav8
        FOR u1: 4_1_mux
             USE CONFIGURATION config4;
        END FOR;
        FOR u2: 4_1_mux
             USE CONFIGURATION config4;
        END FOR;
     END FOR;
  END config;
```

LISTING 13 MUX CONFIGURATION [9]

2.11 VHDL Test benches

After the actual VHDL code is written, the important task of testing the code comes. Simulators usually offer a built in test capability, but often it is more desirable to actually construct a VHDL test process. Once the final behavioral or functional code is complete, we can instantiate the top- level entity in a test bench and write a process which toggles the inputs and monitors the outputs. The test bench can be as simple as a state machine, or as complex as a process which reads inputs from a file and compares the outputs to a pre-generated file.

Testing can be done on a variety of levels. Simulators usually have their own built in simulation packages which consist of "forcing functions" to simulate your code. This has its limitations. For one, using VHDL it is possible to write complicated test benches comprised of multiple files or stimulus written in other languages and saved as a text file to be implemented using TEXTIO functions. These functions allow you to read and write text files.

2.11.1 MUX Test bench

In order to effectively test the MUX, we must write a test bench which will toggle the inputs. We can do this quite effectively with a VHDL test bench.

```
ENTITY testbench IS -- we must declare all signals to look
                    at
  PORT(   data1:   IN   std_logic;
        data2:    IN    std_logic;
        data3:    IN    std_logic;
        data4:    IN    std_logic;
        data5:    IN    std_logic;
        data6:    IN    std_logic;
        data7:    IN    std_logic;
        data8: IN  std_logic;
        addr:   IN   std_logic_vector(3   DOWNTO   0);
        output: OUT std_logic);
   END testbench;

ARCHITECTURE   behavior   OF   testbench   IS
  COMPONENT 8_1_mux
        GENERIC(out_delay : time := 5 ns);
        PORT( data1: IN std_logic;
              data2:    IN    std_logic;
              data3:    IN    std_logic;
              data4:    IN    std_logic;
              data5:    IN    std_logic;
              data6:    IN    std_logic;
              data7:    IN    std_logic;
              data8:     IN   std_logic;
              addr:  IN  std_logic_vector(3  DOWNTO  0);
               output: OUT std_logic);

  END COMPONENT; BEGIN
     u1: 8_1_mux
       PORT MAP( data1 => data1,
                 data2 => data2,
                 data3 => data3,
                 data4 => data4,
                 data5 => data5,
                 data6 => data6,
                 data7 => data7,
               data8 => data8,
                 addr => addr,
                 output => output);
     tester:
  PROCESS
    state : integer := 0; BEGIN
        WAIT  UNTIL  rising_edge(clock);
    CASE state IS
        WHEN 0 => -- initialize data1
             <= '0';
                  data2   <=   '0';
             data3  <=  '0';  data4
             <= '0'; data5 <= '0';
             data6  <=  '0';  data7
             <= '0'; data8 <= '0';
             addr <= "000";  state
```

```
     := 1;
WHEN 1 => -- test addressing data1
     <= '1';
     state := 2; WHEN 2
=>
          data1  <=  '0';
     data2 <= '1';  state
     := 3;
WHEN 3 => -- test the second mux data2
     <= '0';
          data5   <=   '1';
     addr <= "100"; state
     := 4;
     WHEN 4 =>
        data5 <= '0';
        state := 5;
     WHEN 5 => -- done
```

3.1 Introduction

One of the advantages of a multi-carrier system is its robustness against inter-symbol interference. The longer duration of OFDM symbols provides higher immunity against delay spread and ISI. As long as channel dispersion is not longer than the OFDM symbol guard interval, system performance does not degrade due to ISI and use of time domain equalization is not usually mandated. However, in case of higher data rates and channels with extensive time dispersion an equalizer is unavoidable. Albeit, the structure of the equalizer is different from that of single carrier systems. The purpose of equalization is not complete removal but restriction of inter-symbol interference to a tolerable extent. Frequency domain equalization, in the absence of inter-channel interference (ICI) is used to compensate for channel complex gain at each sub-carrier frequency. Using time-frequency duality, complex gain compensation after FFT is equivalent to a convolution of a FIR filter in time domain (residual equalization).

In this chapter, we review main differences of equalization in single and multi-carrier and follow up with analytical and practical aspects of time domain equalization. Then, frequency domain equalization and echo cancellation are discussed in sections 3 and 4. Using duality principle, we present a unified approach to time and frequency domain equalization.

3.2 Time Domain Equalization

A brief review of decision feedback equalizer structure and performance should help the reader to readily understand the analysis and design of equalizers for multi-carrier.

3.2.1 Decision Feedback Equalizers

In a decision feedback equalizer, the forward filter whitens the noise and produces a response with post-cursor ISI only, and the feedback filter then cancels that post-cursor ISI. The optimum forward filter for a zero-forcing DFE can be considered as cascade of a matched filter followed by a linear equalizer and a causal whitening filter whose transfer function can be found by spectral factorization of the channel power spectrum. The forward filter, therefore, first minimizes ISI, then noise power at the input of slicer, reintroducing causal ISI. Notice that the separation and ordering has only analytic significance. In practice, all forward filters are usually lumped together in one adaptive filter.

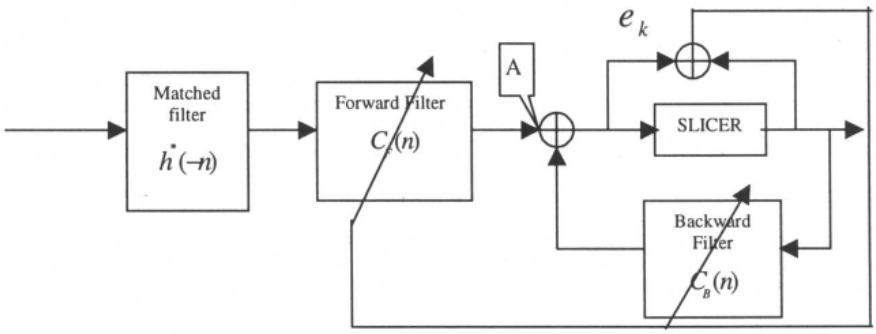


FIGURE 3.1. GENERAL DECISION FEEDBACK EQUALIZER. [1]

In a MSE decision feedback equalizer, the optimum forward filter minimizes the mean-square residual ISI and noise power. In both MSE-DFE and ZF-DFE we intend to minimize the error term at the input of the slicer, which in the absence of error propagation includes residual ISI and noise.

Let h(n) represent the channel impulse response, hence $h^*(-n)$ is the impulse response of the matched filter. Forward filter $c_F(n)$ performs ISI suppression and noise whitening. Feedback filter $c_B(n)$ eliminates the interference of previous symbols, hence $1+c_B(n)$ has to be a monic and causal filter. Therefore, its output should have post-cursor inter-symbol interference only, namely

$$h(n)*h^*(-n)*c_F(n)=1+c_B(n) \tag{3.1}$$

On the other hand the variance of the noise, at the output of the matched filter is

$$\sigma^2\int_{-\pi/2}^{\pi/2}S_h\left(e^{j\omega t}\right)d\omega \tag{3.2}$$

where S_h is power spectrum of the channel impulse response. In order to minimize noise power at the input of the slicer, should whiten the noise. Rewriting Equation 6.1 in the frequency domain, the forward filter transfer function is

$$C_F(z)=\frac{1+C_B(z)}{H(z)H^{*-1}\left(z^{-1}\right)} \tag{3.3}$$

The forward filter may apply further linear transformation to the input signal to meet some optimality criteria such as minimizing square error or peak distortion. To whiten a stochastic process with spectrum of $\sigma^2 S_h(z)$ it should satisfy the following equation:

$$\sigma^2 S_h(z)\frac{1+C_B(z)}{H(z)H^{*-1}(z^{-1})}\frac{1+C_B^*(z^{*-1})}{H^*(z^{*-1})H(z)}=\sigma_{DFE}^2. \tag{3.4}$$

Since

$$S_h(z) = H(z)H^{*-1}(z^{-1}) \tag{3.5}$$

from Equation 3 .4 we have:

$$S_h(z) = (1 + C_B(z))(1 + C_B^*(z^{*-1})) \tag{3.6}$$

The same procedure [4] can be followed for the MSE equalizer to obtain

$$S'_h(z) = (1 + C_B(z))(1 + C_B^*(z^{*-1})). \tag{3.7}$$

In the following, we discuss some important issues about the DFE with finite tap FIR feedback and forward filters as a preface for future discussions.

The input to the feedback filter is noise free, therefore the feedback filter eliminates the post-cursor ISI both in ZF-DFE and MSE-DFE. The error term used for training is memoryless and defined as:

$$e_k = x_k - \hat{x}_k. \tag{3.8}$$

Where x_k is transmitted symbol, which is the same as the output of the slicer when there is no error and $\hat{x}_k$ is the output of the equalizer. Minimizing $|e_k|^2$ requires that input signal be orthogonal to the error term. Since the input to the feedback filter is noise free, we design the forward filter to minimize the error power which leads to the following equations:

$$E(\varepsilon_k y_{k-i}^*) = E(x_k - \sum_j c_{F_i} y_{k-j} - \sum_{j'} c_{B_{j'}} x_{k-j'}) y_{k-i}^* = 0 \quad for \quad -\infty < i < \infty, \tag{3.9}$$

where:

$$y_k = \sum_{n=0} p_n x_{k-n} + \eta_k. \tag{3.10}$$

and

$$p_i = \sum_{n=0}^{l-i} h_n h_{n+i}^* \qquad i = 0, \cdots, l$$
$$p_{-i} = p_i^*. \tag{3.11}$$

where l is channel spread in symbols.

Therefore,

$$\sum c_{F_j} E\{\sum_n p_n x_{k-n-j} \sum_{n'} p_{n'}^* x_{k-n'-i}^*\} + \sum_j c_{F_j} E\{\eta_{k-j}\eta_k^*\} = \sum_{n=0}^{N} p_n E(x_k x_{k-n-i})$$

$$\Rightarrow \sum_j c_{F_j} \sum_n p_n p_{n+j-i}^* + N_0 \sum_j c_{F_j} p_j = p_{-i}^*. \tag{3.12}$$

is a colored Gaussian noise with the following auto-correlation function

$$E(\eta_i \eta_j^*) = \begin{cases} N_0 p_{i-j}, & |i-j| \le l \\ 0, & otherwise \end{cases}, \tag{3.13}$$

The solution to equation 3.9 is

$$C_F = \Gamma^{-1}\theta, \tag{3.14}$$

where

$$\Gamma_{ij} = E\{y_{k+i} y_{k+j}^*\}, \qquad 0 \le i, j < N. \tag{3.15}$$

N is the number of forward taps and

$$\theta_i = E\{x_k y_{k+i}\}, \qquad 0 \le i < N. \tag{3.16}$$

Coefficients of the backward filter can be found by

$$C_B = P^T C_F, \tag{3.17}$$

where

$$C_F = \begin{pmatrix} c_{F_0} \\ c_{F_{-1}} \\ \\ c_{F_{-N}} \end{pmatrix} \quad C_B = \begin{pmatrix} c_{B_1} \\ c_{B_2} \\ \\ c_{B_M} \end{pmatrix} \quad P = \begin{pmatrix} p_0 & p_1 & \cdots & p_M \\ p_1 & p_2 & & p_{M+1} \\ & & & \\ p_N & & & p_{N+M} \end{pmatrix}. \tag{3.18}$$

The forward filter c_F is anti-causal which means that the main tap is assumed to be the closest one to the slicer. The forward taps should remove the entire post-cursor. Practically, the matched filter is usually absorbed in the forward equalizer(We can interpret this as replacing the matched filter with a whitening matched filter) then h replaces p is Equations 6.12 and 6.16. Notice that the feedback filter is designed to minimize the post-cursor ISI at the input of the slicer. In practice, c_B and c_F are adjusted adaptively.

3.3 Equalization in DMT

Multi-carrier systems are robust against inter-symbol interference as long as orthogonality of adjacent symbols is preserved in the frequency domain. In other words, the symbol duration of the OFDM signal is extended to beyond T. However, if channel impulse response spread is more than the prefix extension, inter-symbol interference can degrade the performance.

There are several fundamental differences between time domain equalization requirements of OFDM channels and classic decision feedback equalizers:

a) An equalizer of an OFDM system does not need to cancel the ISI entirely but to limit its length. Unlike a single carrier system in which equalization minimizes ISI, we only need to

reduce it to a time span less than the length of guard interval.

b) The channel impulse response is modeled as an ARMA system.

Usually, the channel impulse response is too long. Therefore, a model of the form is appropriate.

$$\frac{A(z)}{1+B(z)} \tag{3.18}$$

c) The equalizer does not work as a DFE in data mode. Unless channel characteristics change rapidly, equalizer taps are adjusted during the training sequence and the same tap values are preserved during data mode.

d) The feedback filter is not in the loop and need not to be monic or causal. In single carrier systems, the slicer output is fed to the feedback filter to remove the post-cursor effect. However, truncating the impulse response does not require a monic feedback filter. A typical structure is shown in Figure3.2.

e) The error is not memoryless. The error term used in typical DFE is derived from the scalar input and output of the slicer. In a DMT equalizer the error is derived from the output of an impulse shortening filter and reference filter.

f) The location of the impulse response window has a big impact on th e performance of equalizer.

The overall structure of the equalizer in training mode is shown in the following diagram:

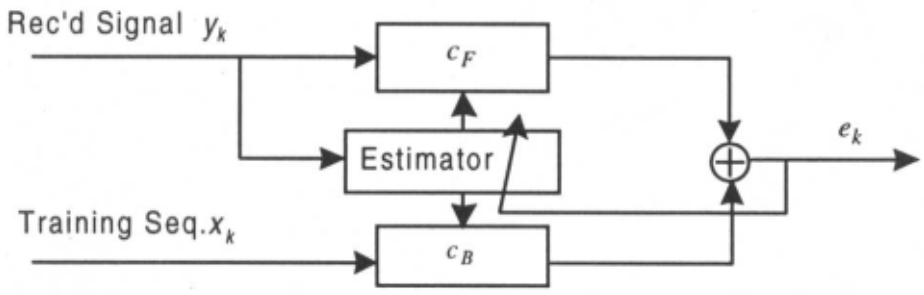


FIGURE 3.2. EQUALIZER CONFIGURATION IN TRAINING MODE. [1]

In steady state mode, the equalizer will include only The error term is

$$e_k = c_F * Y_k - c_B * X_k. \tag{3.19}$$

where X_k and Y_k are vectors of training and received samples respectively.

In general, we can use adaptive techniques to find near optimum equalizer coefficients. A large number of taps prevents us from using RLS type of algorithms. On the other hand, a wide spread of eigenvalues of the input signal covariance matrix can

slow down convergence of the equalizer. A different technique for equalizer tap adjustment is based on linear prediction theory as explained below.

During acquisition, the frame is repeated and can be used for equalizer training. In this approach, we use second order statistics of the channel to produce a one shot estimate of equalizer coefficients. In order to use the predictor format of linear regression techniques, without loss of generality, assume c_0 is monic. So, the predictor format is:

$$\hat{y}(n|n-1) = \varphi^T(n)C \tag{3.20}$$

where

$$\varphi(n) = [-y(n-1), -y(n-2), \ldots, -y(t-N), x(n-1), x(n-2), \ldots, x(n-M)]^T \tag{3.21}$$

and,

$$C = [C_F C_B]^T \tag{3.22}$$

The least square solution of linear regression is:

$$\hat{C} = \Gamma^{-1}\theta \tag{3.23}$$

with:

$$\Gamma = \begin{pmatrix} A_{N\times N} & B_{N\times M} \\ B^*_{M\times N} & I_{M\times M} \end{pmatrix}, \tag{3.24}$$

where

$$A_{ij} = R_{yy}(i-j) = \frac{1}{N}\sum_{n=1}^{N} y(n-i)\,y(n-j), \tag{3.25}$$

and

$$B_{ij} = R_{xy}(i-j) = \frac{1}{N}\sum_{n=1}^{N} x(n-i)\,y(n-j), \tag{3.26}$$

and

$$\theta = [D_N \quad E_M]^T \tag{3.27}$$

with

$$D_i = R_{yy}[i], \quad E_i = R_{yx}[i], \tag{3.27}$$

where R_{yy} and R_{xy} are auto-correlation and cross correlation functions. We can use channel impulse response to calculate coefficients directly

$$A_{ij} = \sum_n h_n h^*_{n+j-i} \tag{3.28}$$

$$B_{ij} = h^{*}_{-i+j}. \tag{3.29}$$

Estimation of channel impulse response can help in choosing optimum window location. As explained before, c_B is not necessarily causal, therefore dely parameter plays an important role in the performance of the equalizer. Other important parameters are the lengths of the two filters c_F and c_B.

3.3.1 Delay Parameter

Unlike classic decision feedback equalizers, an ARMA equalizer can have a feedback filter with non-causal transfer function i.e. the general error term is:

$$c_F * Y_k - c_B * X_{k-d}. \tag{3.30}$$

Delay unit *d* has significant effect on overall performance. The number of taps *M* of filter is approximately the same as the prefix length and number of taps for the forward filter $N \leq M$. The delay unit should be chosen properly to capture the most significant window of the impulse response. Brute force trial and error is one option. A more intelligent technique can use the estimate of the impulse response to pick a proper starting point where the energy of the impulse response is above a threshold. The threshold should be the ratio of tap power to overall window power. Since the length of the window function and forward equalizer should be as short as possible, proper choice of starting point is of significant importance.

In other words, in an ARMA model the number of zeroes is not assigned properly.

3.3.1 AR Approximation of ARMA Model

Direct solution of Equation 3.23 requires a non-Toeplitz matrix inversion and is numerically sensitive and intensive. One technique to alleviate the numerical difficulty of least square solution is by using techniques for AR approximation of the channel response ARMA model, which results in a numerically attractive Toeplitz correlation matrix. A description of embedding techniques is presented in the Appendix. In the following, the procedure for this analysis is briefly described.

By using embedding techniques, as explained in the Appendix, the ARMA model of Equation 3.18 is approximated by a two-channel AR model.

If the covariance matrix is available the problem can be reduced to solution of a normal (Yule-Walker) equation. The multi-channel version of the Levinson algorithm is a computationally efficient solution for the two- channel AR model at hand. As a reference, the recursive technique is presented here:

Initialization:

$$C_1^f[1] = -R^{-1}[0]R[1]$$
$$C_1^b[1] = -R^{-1}[0]R[-1] \tag{3.31}$$

where

$$C_i^f[j] = \begin{bmatrix} -a_i^j & b_i^j \\ 0 & 0 \end{bmatrix} \tag{3.32}$$

is the j-th forward prediction coefficient of two-channel model at i-th iteration and $C_i^b[j]$ is the j-th backward prediction coefficient at i-th iteration and

$$R[i] = \begin{bmatrix} R_{yy}[i] & R_{yx}[i] \\ R_{xy}[i] & R_{xx}[i] \end{bmatrix} \tag{3.33}$$

If the input signal x is white then

$$R[i] = \begin{bmatrix} R_{yy}[i] & R_{yx}[i] \\ 0 & 0 \end{bmatrix} \tag{3.34}$$

with

$$R[0] = \begin{bmatrix} R_{yy}[0] & R_{yx}[0] \\ R_{xy}[0] & I \end{bmatrix} \tag{3.35}$$

Recursion:

$$C_i^f[i] = K_i^f$$
$$C_k^f[i] = C_{k-1}^f + K_k^f[i]C_{k-1}^b[k-i]$$
$$C_i^b[i] = K_i^b$$
$$C_k^b[i] = C_{k-1}^b + K_k^b[i]C_{k-1}^f[k-i]$$

where the reflection coefficients are,

$$K_k^f = -\Delta_k^{f^T} \Sigma_{k-1}^{b^{-T}}$$
$$K_k^b = -\Delta_k^{f^T} \Sigma_{k-1}^{b^{-T}}$$
$$\Delta_k^f = \sum_{i=0}^{k-1} R[k-i]C_{k-1}^{f^T}[i].$$

Figure 6.3 shows the structure of an equalizer using a lattice filter. This structure represents the case of equal number of poles and zeros. The extension to unequal number of poles and zeroes is straightforward and explained in [5].

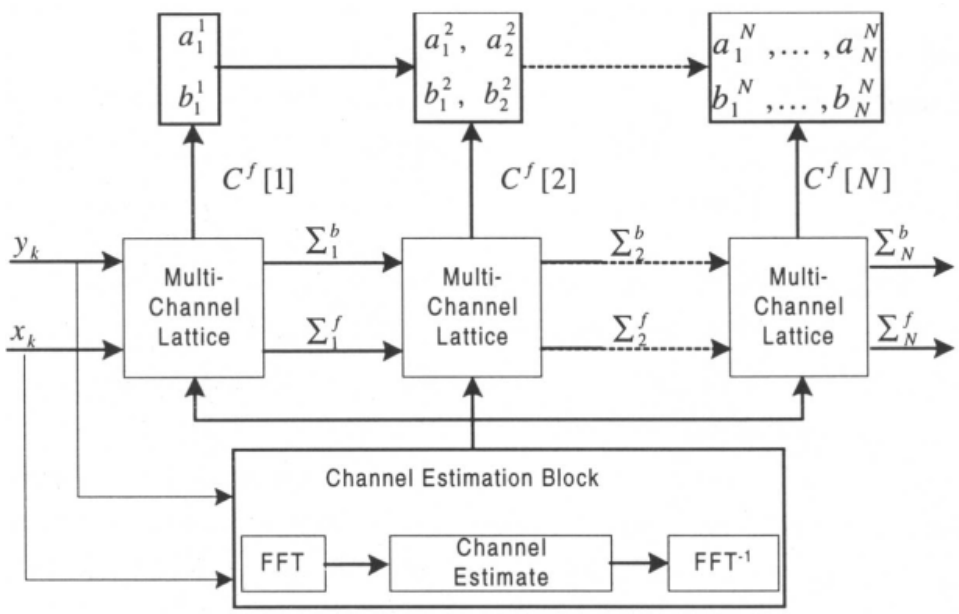


FIGURE 6.3. EQUALIZATION USING ORDER UPDATE. [1]

The above approach is convenient for batch or frame based processing where the entire frame is buffered and processed for equalization. Instead of inverting the large matrix of Equation 6.13 for example, this structure can be used in the training mode of an ADSL receiver. This technique can be implemented iteratively in time and order when the covariance is not knownin advance. The time/order updating algorithm increases the order while updating the coefficients of the equalizer. In some cases, we can preserve the order and continue updating the parameters using adaptive techniques. This technique is useful when the delay profile of the channel does not change significantly but some slow adaptation improves the performance.

3.4-Frequency Domain Equalization

Once it is assured that orthogonality of the sub-carriers is maintained, possibly through the use of the cyclic prefix and time domain equalization as previously discussed, then the final frequency domain equalization of an OFDM signal is an extremely simple process. This is certainly one of the key advantages of OFDM.

After demodulation, the sub-carriers will be subjected to different losses and phase shifts, but there will be no interaction among them. Frequency domain equalization therefore consists solely of separate adjustments of sub- carrier gain and phase, or equivalently of adjusting the individual decisionregions. For the case where the constellations consist of equal amplitude points, as in PSK, this equalization becomes even simpler in that only phase need be corrected for each sub-carrier, because amplitude has no effect on decisions.

A simplified picture of the place of frequency domain equalization is shown in Figure 3.4, where the equalizer consists of the set of complex multipliers, {A}, one for each sub-carrier.

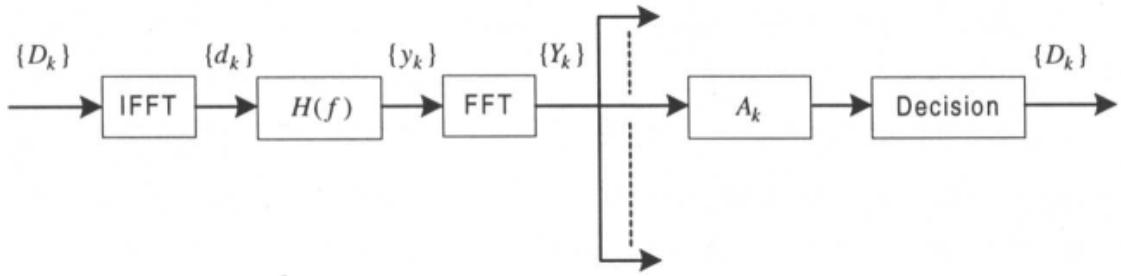


FIGURE 3.4. AN OFDM SYSTEM WITH FREQUENCY DOMAIN EQUALIZATION. [1]

Here the linear channel transfer function $H(f)$ includes the channel, the transmit and receive filters, and any time domain equalization if present. $H(f)$ is assumed bandlimited to less than N/T for a complex channel.

Cyclic extension is not shown although it is almost certain to be present. The following analysis assumes that both amplitude and phase need be corrected, and that equalization consists of multiplying each demodulated component by a quantity such that a fixed set of decision regions may be used.

The signal presented to the demodulator is

$$y(t) = \sum_{n=0}^{N-1} d_n h(t - n\frac{T}{N}). \tag{3.36}$$

The kth output of the demodulator is then

$$Y_k(f) = D_k H_k \quad , \quad k = 0,1,\ldots N-1,$$

where the H_k are samples of $H(f)$

$$H_k = H(\frac{k}{T}). \tag{3.37}$$

Thus each output is equal to its associated input data symbol multiplied by a complex quantity which differs among the outputs, but are uncoupled at its simplest then consists of setting the multipliers to $1/H_k$ for each non-zero channel.

The above approach is optimum in every sense under high signal-to- noise conditions. It also produces minimum probability of error at any noise level, and is an unbiased estimator of the input data D_k. However if the criterion to be optimized is minimum mean-square error particularly, then the optimum multipliers are modified to

$$A_k = \frac{1}{H_k} \frac{1}{1 + \frac{\sigma_k^2}{|D_k H_k|^2}}, \tag{3.38}$$

where σ_k^2 is the noise power in the demodulated sub-channel. However this value produces a biased estimator and does not minimize error probability.

As a practical implementation issue, for variable amplitude constellations, it is frequently

desirable to have a fixed grid of decision regions. The s can then be scaled in amplitude such that the separation of constellation points is constant.

Since a shift in timing phase τ is equivalent to a phase shift frequency domain equalization readily corrects for such timing shift.

$$H_k = e^{j2\pi\frac{k}{T}\tau} \tag{3.39}$$

In principle frequency domain equalization could be employed when orthogonality is lost due to interference among OFDM symbols. In this case, rather than a simple multiplier per sub-channel, a matrix multiplication would be required. This approach is bound to require more computational load than the combination of time and frequency domain equalizers.

During system initialization, any time domain equalizer must be adjusted before frequency domain equalization is performed. Then any periodic test signal with full frequency content, such as a repeated segment of a PN sequence without cyclic prefix, may be used to adapt the frequency domain equalizer. The demodulator outputs should be averaged over many periods to minimize the effects of noise, and the reciprocal of these measured values used to set the multipliers. During steady state data operation adaptation may be performed using a single variable LMS algorithm as shown below.

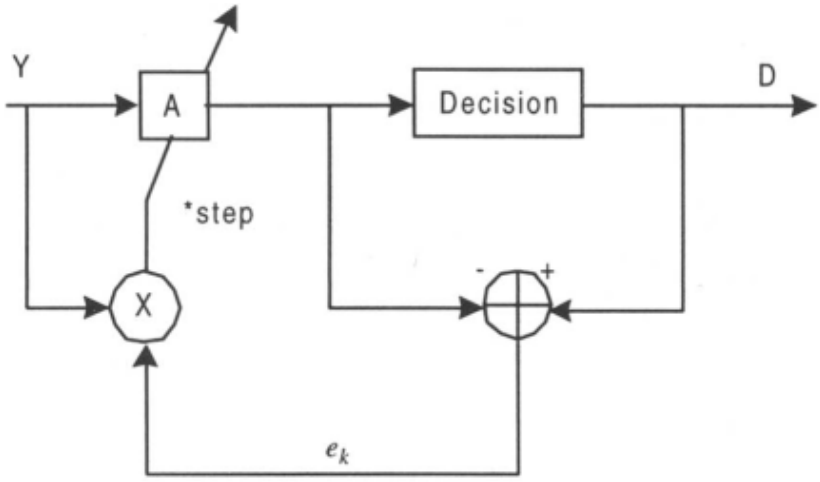


FIGURE 3.5. LMS ADAPTATION OF A FREQUENCY DOMAIN EQUALIZER MULTIPLIER.

This LMS algorithm will adapt to the values given by Equation 3.38 above.

An interesting interpretation of time domain and frequency domain equalizer can be obtained by studying their role in channel distortion compensation. As discussed before, a typical channel model for DMT and channels with long impulse response is an ARMA model of the form 3.18

Time domain equalization shortens the impulse response to a tolerable level for an OFDM system. Mathematically, it's equivalent to compensating the AR part of the channel impulse response $1 / (1 + B(z))$. So after successful time domain equalization, the equivalent impulse response of

channel is reduced to a FIR filter of short duration $A(z)$. Since it does not violate orthogonality of sub-carriers, we can remove its effect after the FFT by frequency domain equalization. The above process is shown in Figure 3.6.

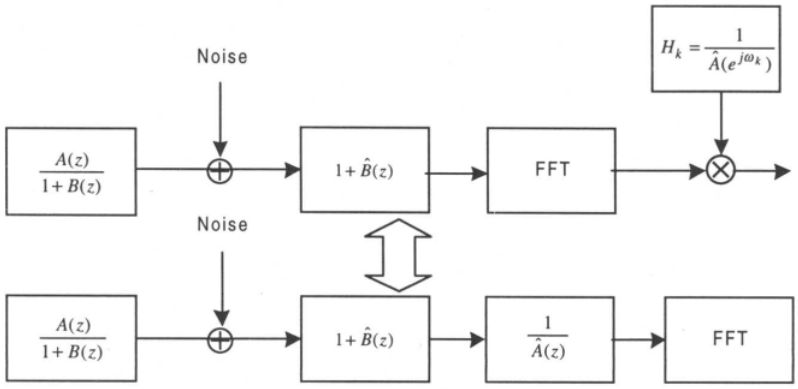


FIGURE 3.6. TIME AND FREQUENCY DOMAIN EQUALIZATION. [1]

3.5 Echo Cancellation

In wireline applications, it is usually highly desirable to communicate full duplex (simultaneous transmission in both directions) over a single wire- pair. A classic sub-system known as the hybrid provides partial separation, but is not sufficient for high performance digital applications.

Two possible techniques are frequency division and time division duplexing. In the former, different frequency bands are used in each direction, with analog filtering to provide separation. In the latter, alternating time slots are used to transmit alternately in each direction. For symmetric communication, that is equal rates for the two directions, both of these techniques require at least double the bandwidth as unidirectional transmission, which is a very severe performance penalty.

Far preferable is the technique of echo cancellation, where at each end the leakage of transmitted signal into the co-located receiver is cancelled by creating and subtracting a replica of that leakage. The precision required is quite high, as illustrated in the example of a bad case wireline system shown in Figure 3.7.

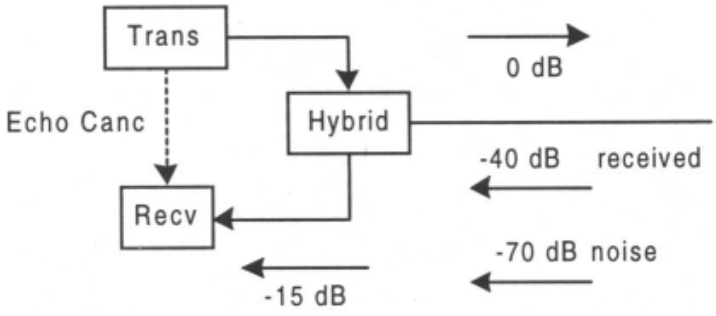


FIGURE 3.7. AN EXAMPLE SYSTEM ILLUSTRATING ECHO CANCELLATION REQUIREMENT. [1]

In this example, the loss of the line is 40 dB. while the hybrid provides only 15 dB of loss. The leakage of the transmitted signal, or "echo", is therefore 25 dB higher than the desired received signal. One consequence is that the level which the front end of the receiver, in particular the A/D converter, must accommodate is dominated by the echo rather than the received signal. Of more interest here is the high degree of echo cancellation required. Assume, as in the figure, that a signal-to noise ratio of 30 dB is required, and that uncancelled echo can be treated as noise. If we use the reasonable design rule that the uncancelled echo should be at least 10 dB below the line noise, thereby increasing the noise by less than 0.4 dB, the level of the uncancelled echo must be below -80 dB. Because the hybrid provides only 15 dB of reduction, the echo canceller must provide an additional 65 dB of rejection. This requirement is far more severe than that of equalization, so that a much greater arithmetic precision and number of taps is required.

In principle, the same form of echo canceller used in single carrier systems could be used in OFDM. This is shown in Figure 3.8.

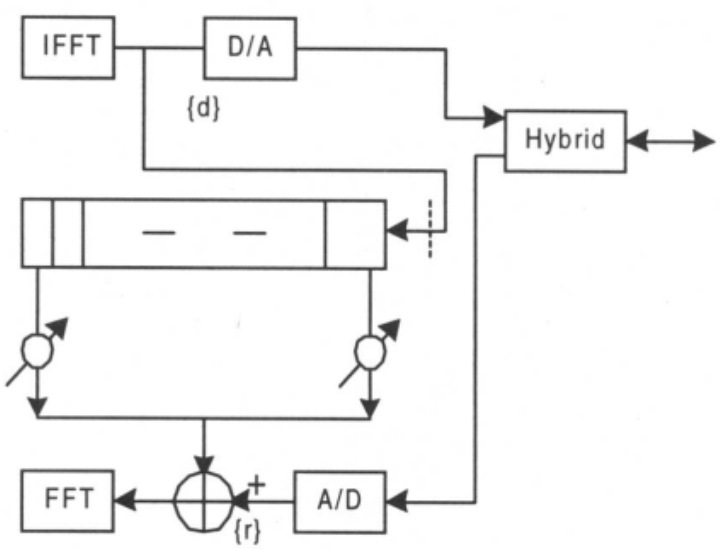


FIGURE 3.8. A BASIC ECHO CANCELLER. [1]

Residual echo is minimized when the tap coefficients of the echo canceller are equal to the corresponding samples of the echo path response $p_k = p(kT)$, where the sampled echo is

$$r_n = r(nT) = \sum_k d_k p_{n-k}. \tag{3.40}$$

Adaptation is similar to that of equalization. Residual error results from echo samples beyond the time span of the canceller, and also from finite precision in signal sampling and in the coefficients. Unlike the case of single carrier equalization, the inputs to the canceller are samples of signals quantized by an A/D converter as opposed to exact data symbols. In one variation [6], adaptation is performed in the frequency domain, followed by an IFFT and time domain cancellation.

In any variation, time domain cancellation requires extensive computation, reducing a prime advantage of OFDM. Another possibility is frequency domain cancellation. If the echo of the

transmitted OFDM signal maintains orthogonality, then echo cancellation may be performed very easily after demodulation in the receiver. An advantage of this approach is the use of exact data values as inputs to the canceller rather than quantized signal samples. Furthermore, orthogonality leads to simple adaptation. The canceller consists of one complex multiplication per sub-carrier, as in the frequency domain equalizer.

In order for this technique to be valid, there must be no inter-symbol interference among OFDM signals. Otherwise a matrix multiplication is required rather than a single multiply per sub-carrier [7]. In another approach described in Reference [8], the time domain equalization is modified so as to shorten both the channel response and the echo response to a time span less than the duration of the cyclic prefix. The resulting equalizer and echo canceller can then both be performed quite simply in the frequency domain as shown in Figure 3.9.

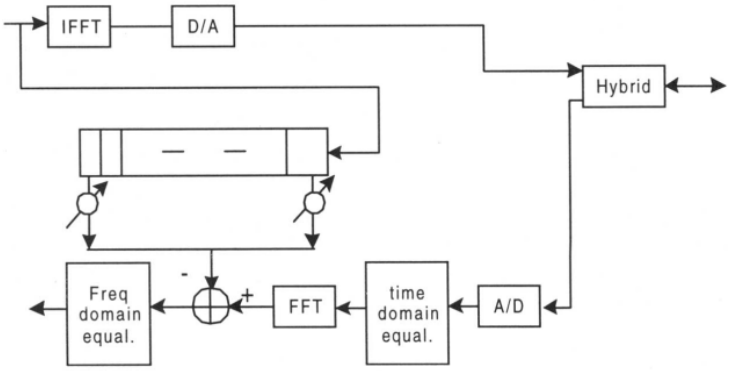


FIGURE 3.9. FREQUENCY DOMAIN ECHO CANCELLATION. [1]

Techniques have been proposed for echo cancellation in OFDM that combine time and frequency domain components in ways to reduce total required computation [8, 10]. Shown below is the approach of Reference [8] for a symmetric system.

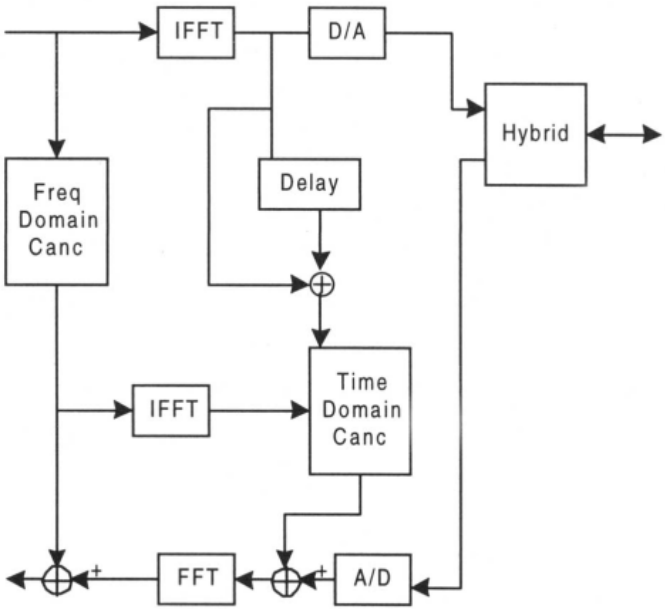


FIGURE 3.10. COMBINATION ECHO CANCELLATION FOR SYMMETRIC TRANSMISSION.

The time domain canceller is a short one if the echo response is not much longer than the duration of the cyclic prefix. Its functions are to undo the effect of cyclic extension, and to cancel the overlap from the previous symbol. The frequency domain canceller can now deal with orthogonal symbols as before.

In asymmetric applications such as ADSL, the penalty for using frequency division duplexing is not as great because the bandwidth used in one direction is relatively small. However for optimum performance, the use of overlapping bands and echo cancellation is employed. Shown in Figure 3.11 and Figure 3.12 are modifications of the previous structure [8] when the same OFDM symbol rate but different numbers of sub-carriers are used in each direction, with overlap of the frequencies of some of the sub-carriers.

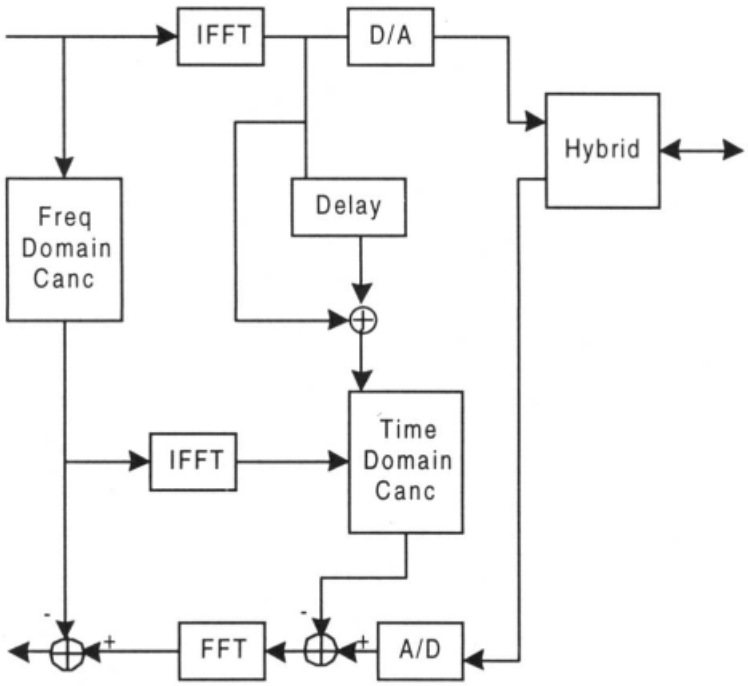


FIGURE 3.11. ECHO CANCELLATION WHERE THE TRANSMIT RATE IS LOWER THAN RECEIVE RATE. [1]

Figure 3.11 illustrates a case in which the transmit rate is lower than the receive rate, as in the remote terminal of ADSL. Because the transmit spectrum is smaller than the receive spectrum, it is replicated so that the number of sub-carriers is increased to equal that of the received signal. For the time domain portion of the echo canceller, the transmit signal is interpolated to the higher rate by inserting the appropriate number of zeroes between the time samples.

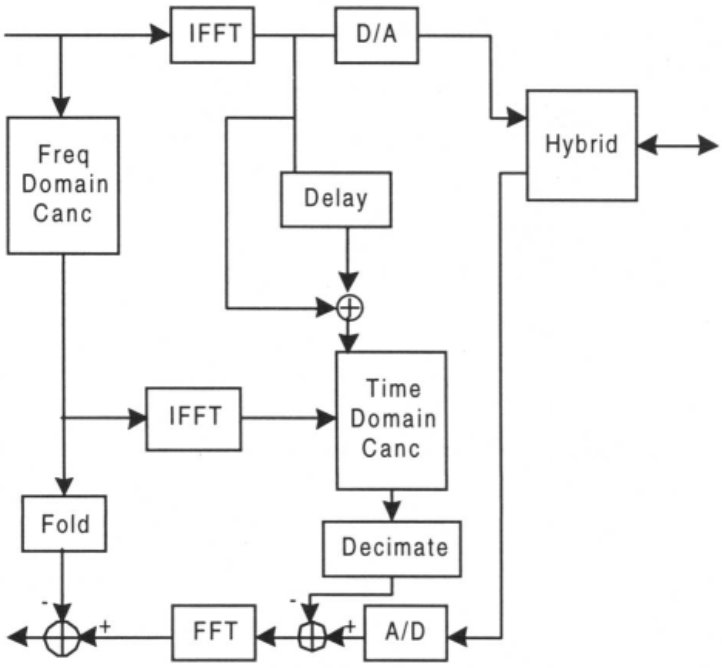


FIGURE 3.12. ECHO CANCELLATION WHERE THE RECEIVE RATE IS LOWER THAN TRANSMIT RATE. [1]

The converse case, in which the transmit rate is higher than the receive rate, as in the central office terminal of an ADSL link, is shown Figure 3.12. The size of the input to the frequency domain canceller is reduced by folding subsets of the OFDM symbol on top of each other and adding them to produce a number of sub-carriers equal to the number received. The time domain sampling rate is reduced by computing the short echo at further spaced points.

Initial adaptation, for any echo cancellation structure, may be performed at the same time and with the same training signals as the equalization at the other end. Having the remote transmitter silent improves this initial adaptation.

DESCRIPTION OF 802.11 STANDARD
4-1 Characteristics of wireless LANs

Wireless LAN (WLAN) uses over the air infrared light waves, radio waves, electron-magnetic waves instead of copper wires or optical fiber as the transmission medium. WLAN provides the same features and benefits of a tradition LAN, such as Ethernet. Table 1 lists the comparison of the three kinds of standard.

Table 1 Comparison of IEEE802.11a、 b、 g [11]

	802.11b	**802.11a**	**802.11g**
Spectrum	2.4~2.4835 GHz	5.15~5.825 GHz	2.4 GHz
Max Data Rates	11Mbps	54Mbps	54Mbps
Transmission Range	Typical: 100M Max to 300M (Need Repeater)	About 100M, shorter Than 802.11b Transmission range	About 100M
Advantages	Base physical specification. To provide longer distance transmission	More bandwidth to support user's connection	Compatible with 802.11b and data rates grow up to 54Mbps
Disadvantages	High power consumption	Spectrum in some countries is not open to public	Spectrum 2.4 G is crowded

4.1.2- IEEE 802.11 Overview

- ### *4.1.2.A-IEEE 802 Concepts*

The IEEE 802 family consists of a series of specifications for local area network (LAN) technology. Figure. 4.2 shows the relationship between various components of the 802 family and the layer they place in the OSI (Open system Interconnect, and which is shown in Figure. 4.1) model. IEEE 802 specification focuses on the two lowest layers of the OSI model because they incorporate both physical and data link component and the data link layer is partitioned into the logical link control (LLC) and the media access control (MAC). All 802 networks have both a MAC and a physical component. The MAC component is a set of rules to determine how to access the medium and send the data, and the PHY handles the details of transmission and reception.

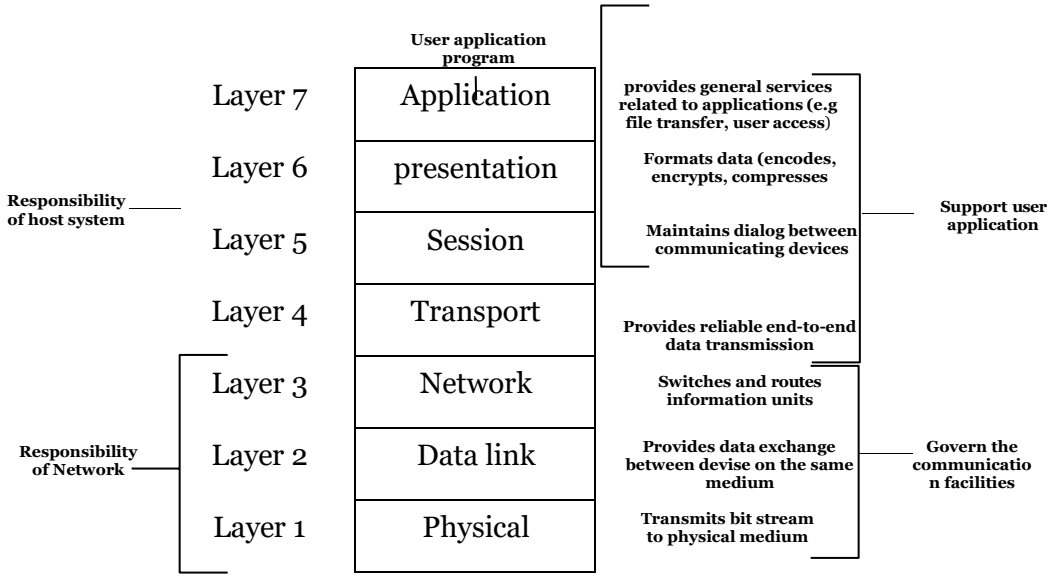


FIGURE.4 1 THE SEVEN-LAYER OSI REFERENCE MODEL [11]

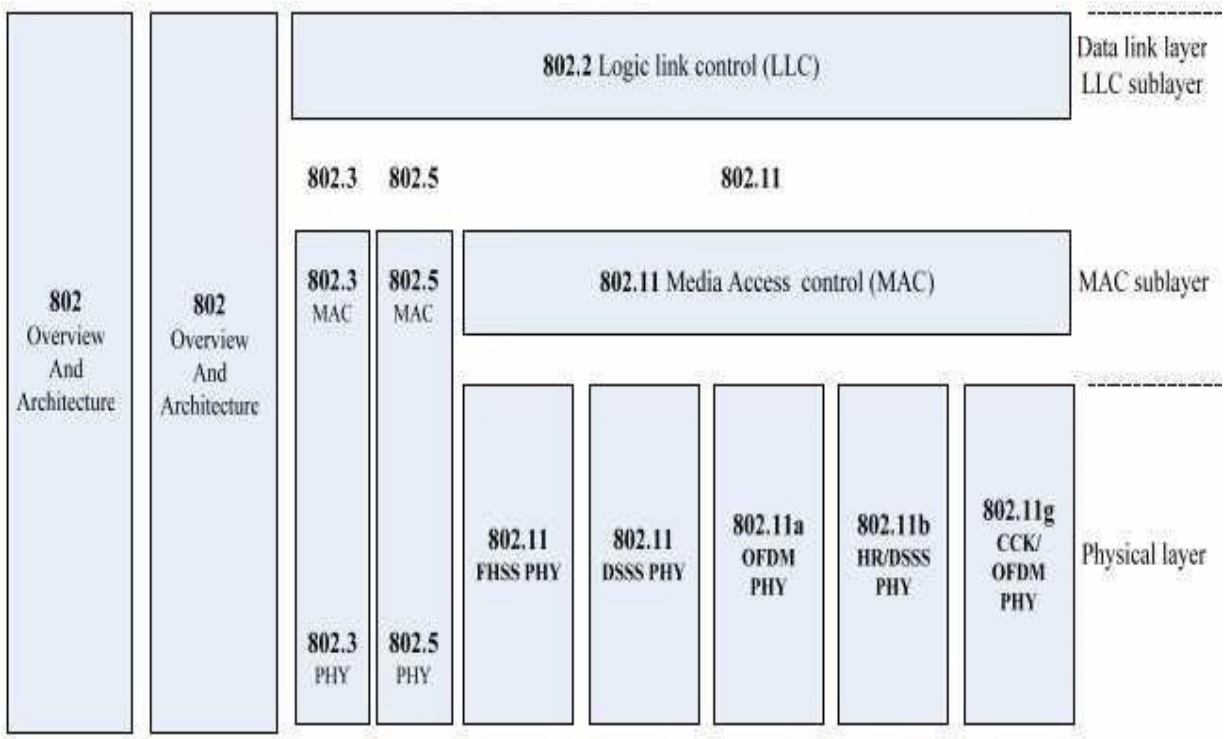


FIGURE.4 2 IEEE802 FAMILIES AND ITS RELATION TO THE OSI MODEL [10]

Individual specifications in the 802 series are identified by the other number. For Example, the management features for 802 networks are specified in 802.1, and 802.2 specifies the logical link control (LLC) layer. The 802.3 is the specification for a Carrier

Sense Multiple Access network with Collision Detection (CSMA/CD). The 802.5 is the Token Ring specification. And the 802.11 incorporate a number of additional features into the MAC to allow for mobile network access.

- ***4.1.2.B- 802.11 Physical Components***

There are four major physical components to form the 802.11 networks, and they are summarized in Figure.4.3. These four components are Distribution system (DS), Access points (AP), Wireless medium and Station (STA).

B.1- Distribution System

When several access points are connected to form a large coverage area, they must communicate with each other to track the movements of mobile station. The distribution system is the logical component of 802.11 used to forward frames to their destination. A distribution system medium is the backbone network used to transmit frames between access points. Therefore it is often called the backbone network.

B.2- Access points

The frame used in 802.11 must be covered to another form for delivery to other types of network. Access points perform the wireless-to-wired bridging function. Access points can perform a number of other functions, but bridging is by far the most important.

B.3- Wireless medium

The 802.11 standard transmits frames from one station to others with the wireless medium. There are several physical layers defined; the architecture allows multiple physical layers to support the 802.11 MAC. Initially, two radio frequencies (RF) (DSSS/FHSS) and one infrared physical layer were standardized, though the RF layers have been proved for more and more popular applications.

B.4- Station

The purpose of building network is to transmit data between stations.

The station is a computing device with wireless network interfaces. Usually, stations are battery-operated laptop or handheld computers. However, it is not necessary that stations and it must be portable computing device. In some environments, wireless networking is used to avoid pulling cables, and desktops are connected by wireless LANs.

- ***4.1.2.C- 802.11 Network Topologies***

The basic service set (BSS) is the basic building block of 802.11 network. It consists of a group of stations that communicate with each other. BSSs come in two flavors, both of which are Infrastructure BSS and Independent BSS. The service coverage of BSSs can be created in small office and homes but they can't provide network coverage to larger area. 802.11 allow linking several BSSs into an extended

service set (ESS) to create wireless networks of arbitrarily large size. An ESS is created by chaining BSSs together with a backbone network. 802.11 does not specify a particular backbone technology. It just requires that the backbone provide a specified set of services. Figure.4.3, the ESS is the union of the two BSSs (That provided that all the access points are configured to be part of the same ESS).

C.1- Independent Networks

Figure. 4.4 illustrates an Independent BSS (IBSS), in which the station in an IBSS communicates directly with each other. Therefore, the stations must be within direct communication range. The smallest possible 802.11 network is an IBSS with two stations. Usually, IBSS are set up for a specific purpose and for a short period of time with a small number of stations. One common usage is to create a

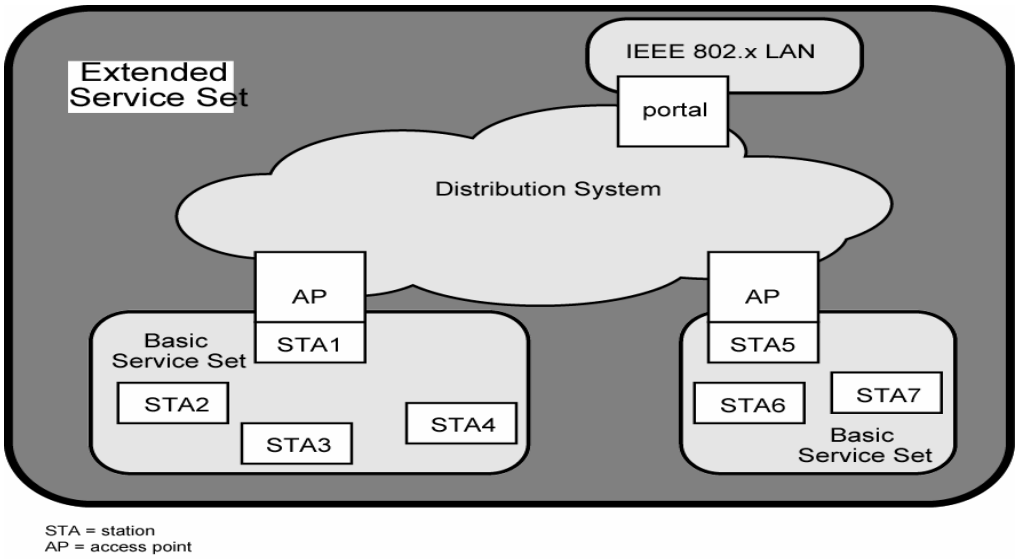


FIGURE. 4.3 COMPONENTS OF 802.11 LANS [11]

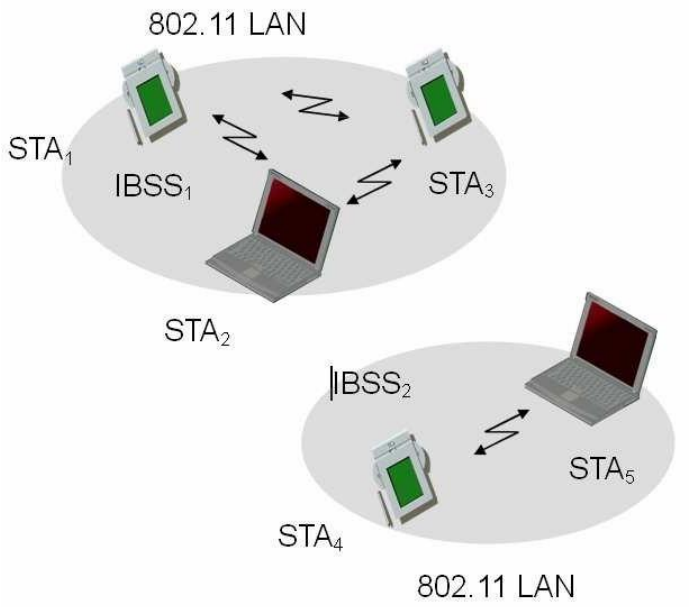


FIGURE. 4.4 INDEPENDENT BSS (AD HOC) [11]

short-lived network to support a single meeting in a conference room. As the meeting beginning, the meeting members create an IBSS to share data, and the IBSS is dissolved at the end of the meeting. Due to their short duration, and special purpose, IBSSs are sometimes referred to as ad hoc BSSs or ad hoc networks.

C.2- Infrastructure Networks

Figure.4.5 illustrates an Infrastructure BSS (IBSS is not short for infrastructure BSS) which should include at least one Access point. Access points are used for all communications in infrastructure networks, including communication between mobile nodes in the same area. If one mobile station in an infrastructure BSS needs to communicate with a second mobile station, the communication must take two steps: first, original mobile station transfers the frame to the access point and second, the access point transfers the frame to the destination station. Since all communications need to relay through an access point, the basic service area corresponding to an infrastructure BSS is defined by the points in which transmissions from the access point can be received and joined.

Although the multi-step transmission consumes more transmission than a directed frame from the sender to the receiver, it has two major advantages: one advantage is that an infrastructure BSS is defined by the distance from the access point. All mobile stations are required to be within to reach the area of the access point, but no restriction is placed on the distance between mobile stations themselves. The advantage of allowing mobile stations direct communication each other would save transmission capacity but at the cost of

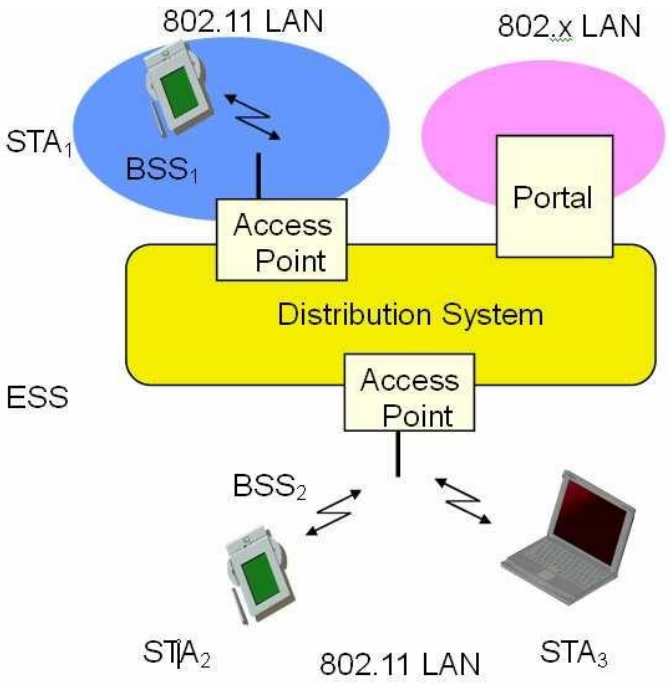


FIGURE.4.5 INFRASTRUCTURE BSS [11]

increased physical layer complexity because mobile stations would need to maintain neighbor relationships with all other mobile stations within the service area; The other advantage is that the effect of the access point in infrastructure networks is to assist with stations attempting to save power.

The access point can record that which station enters a power-saving mode and buffer frame for it. Battery-operated stations can turn the wireless transceiver off and power it up only to transmit and retrieve buffered frames from the access point.

4.1.3- MAC Layer Function Description

The key to the IEEE 802.11 specification is the MAC, rides on the physical to controls the data which transmits on air. It is in charge of the core framing operations and the interaction with a wired network backbone.

- #### 4.1.3.A- MAC Layer Operation

The MAC sub-layer uses one of two methods to gain access to the network. The one method is the distributed coordination function (DCF). Using the DCF, all the stations contended for the channels for each packet transmission. The other method is the PCF (point coordination function), which uses a centralized decision maker, such as an AP, to provide contention-free frame transfers. Figure.4.6 illustrates the architecture of these two coordination functions with the MAC sub-layer. The DCF resides on the top of the physical layer and the PCF is implemented on top of the DCF. Both the DCF and the PCF can operate concurrently in the same BSS to provide alternating contention and contention-free transmission periods. In an ad hoc network the stations use only the DCF. In other networks, stations can operate using just the DCF or a coexisting combination of the DCF and PCF.

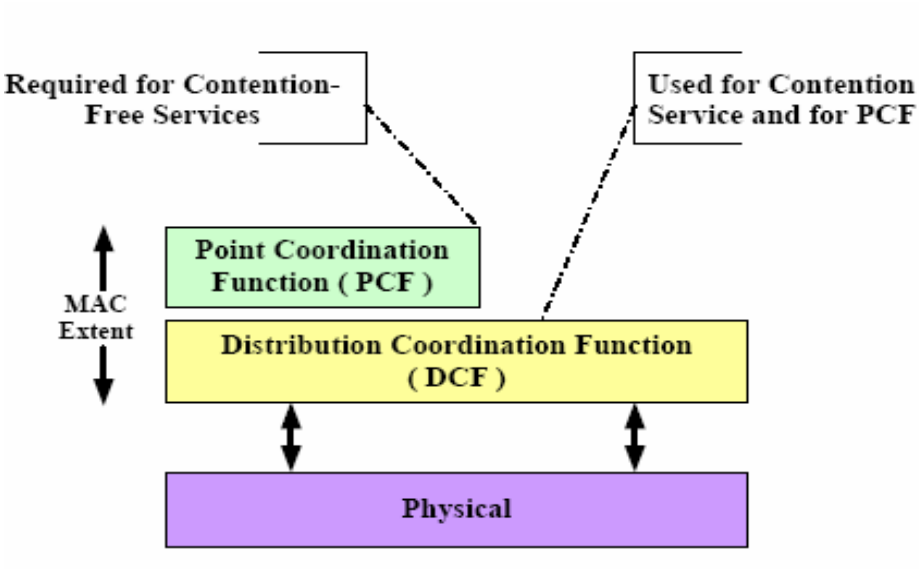


FIGURE.4.6 MAC ARCHITECTURE [11]

A.1- Fragmentation/Defragmentation

The process of partitioning a MAC service data unit (MSDU) into smaller MAC level frames, MAC protocol data units (MPDUs), is called fragmentation. Fragmentation creates MPDUs smaller than the original MSDU length to increase reliability, by increasing the probability of successful transmission of the MSDU in cases where channel characteristics limit reception reliability for longer frames. Fragmentation is accomplished at each immediate transmitter.

The process of recombining MPDUs into a single MSDU is defined as defragmentation. Defragmentation is accomplished at each immediate recipient. When a directed MSDU is received from the LLC with a length greater than a Fragmentation Threshold, the MSDU shall be fragmented. The MSDU is divided into MPDUs. Each fragment is a frame no larger than a Fragmentation Threshold. A brief illustration of fragmentation is shown in Figure.4. 7 and a MSDU is fragmentation to four MPDUs.

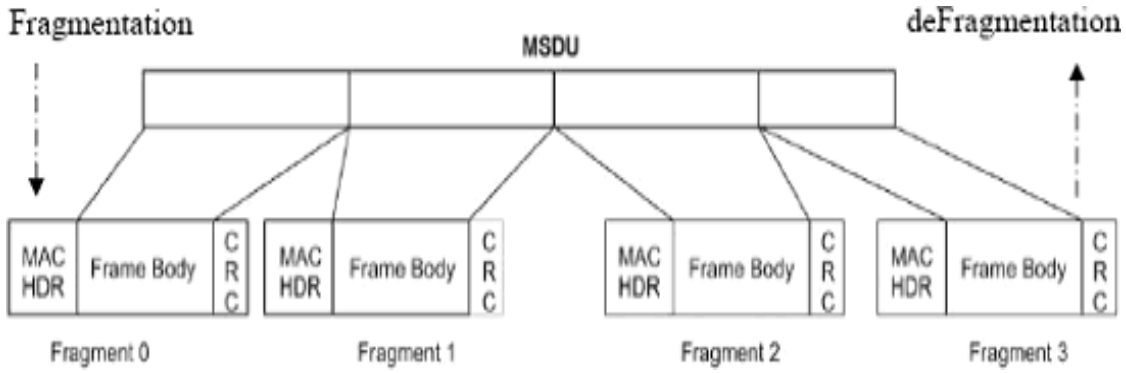


FIGURE.4.7 FRAGMENTATION/DEFRAGMENTATION [11]

A.2- Carrier-Sensing Function and Network Allocation Vector

Carrier sensing is used to determine if the medium is available. There are two types of carrier-sensing function in 802.11: the physical carrier-sensing and virtual carrier-sensing function. If the carrier-sensing function indicates that the medium is busy, the MAC reports and indicates this to higher layer.

Physical carrier-sensing functions are provided by the physical layer in question and depend on the medium and modulation used. It is difficult (or, more to the point, expensive) to build physical carrier-sensing hardware for RF-based media, because transceivers can transmit and receive simultaneously only if they incorporate expensive electronics. Virtual carrier-sensing is provided by the Network Allocation Vector (NAV). Most 802.11 frames carry a duration field, which can be used to reserve the medium for a fixed time period. The NAV is a timer that indicates the amount of time the medium will be reserved. Stations set the NAV to time for which they expect to use the medium, including any frames necessary to complete the current operation. Other stations count down from the NAV to 0. When the NAV is nonzero, the virtual carrier-sensing function

indicates that the medium is busy; when the NAV reaches 0, the virtual carrier-sensing function indicates that the medium is idle [2] [3].

4.2-MAC Receiver Design Architecture and Implementation

In this chapter we describe how to implementation of MAC receiver in this thesis. We only realize MAC receiver part. Figure.4.9 illustrates system architecture. We try to divide MAC receiver part to 4 parts. Those are RX MAIN module, CRC32 module, Scan and synchronization module and defragmented module.

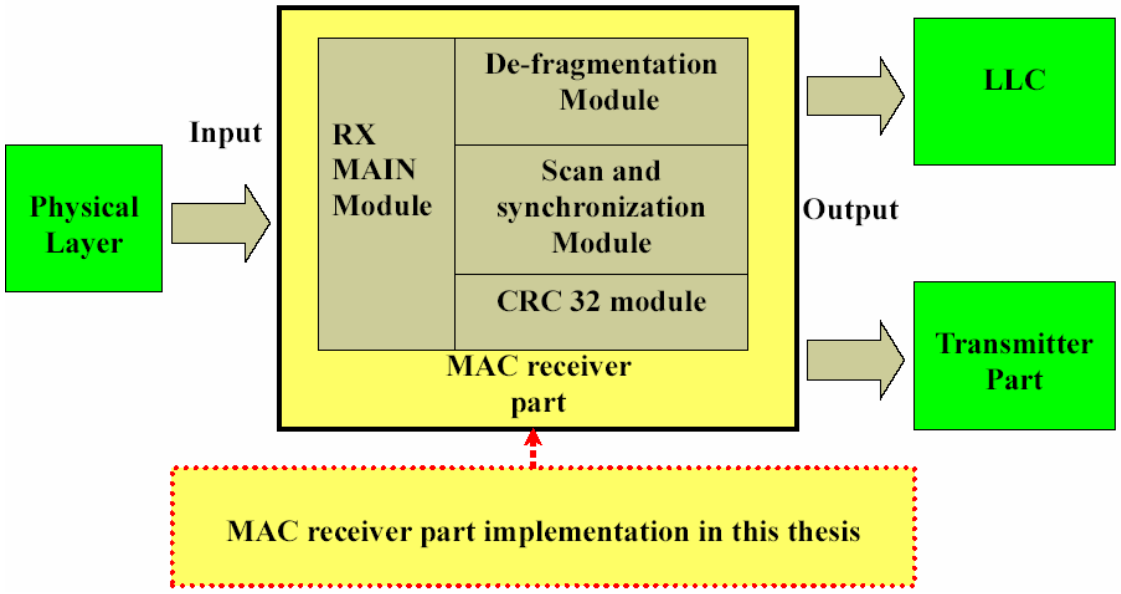


FIGURE. 4.9 SYSTEM ARCHITECTURE [11]

The 2-1 section, we describe our design flow and tool. Other sections will have more detail description for each module [9].

4.2.1- Design flow and tools

Design flow is shown Figure 4.10. Firstly, we create MAC receiver function specification in our thesis and base on 802.11 b/g specification. We define the MAC system specifics and functions module.

Second, we start to create each function module by VHDL language. Then we create behaviour module and generate test bench as input on Xilinx ISE9.1i. We simulate each module by ModelSim SE6.0c program and get simulation results. If simulation results meet our specific and requirement we can start to implement to FPGA (hardware circuit). After synthesize and fit out design on Xilinx ISE9.1i program we implement to FPGA and use LA to check results. After each module validates done we combine all function modules and validate MAC receiver total function at last.

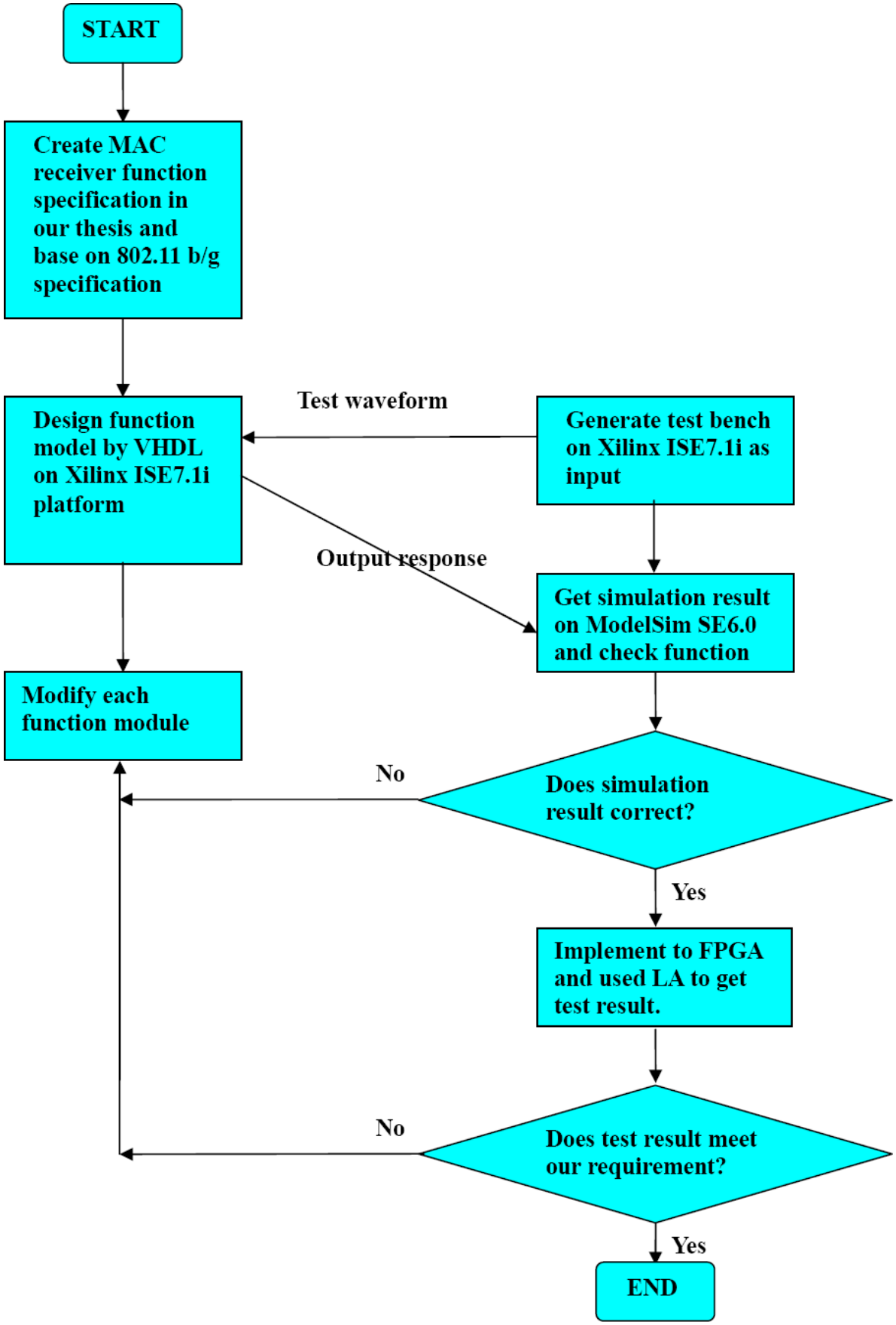


FIGURE. 4.10 DESIGN FLOW CHART

4.2.2 Specifications of MAC Receiver each Module

- ***4.2.2.1-Receiver Module***

The receiver part module is in charge for coordination others modules. It receives MPDUs from PLCP and decodes packages. The specification is shown in Figure.4.11 and shows that we want to realize in thesis. Table 8 list MAC receiver part interface signals. The following notations are used to describe the signal type:

I Input signal

O Output signal

I/O Bi-directional Input / Output signal

Table 8 Receiver part interface signals

NAME	TYPE	DESCRIPTION
Gen PRFrame	O	Notifies transmitter send ProbeRequest frame
GenACKFrame	O	Notifies transmitter send acknowledge (ACK) frame
GenCTSFrame	O	Notifies transmitter send Clear To Send (CTS) frame
Receive ACK	O	Notifies transmitter that receiver got ACK
Receive RTS	O	Notifies transmitter that receiver got RTS frame
Data Ready	O	Notifies LLC layer that some valid date are ready to be transmitter
Out RoLLC	O	Data signals to LLC layer
RX_Start	1	Physical layer notifies receiver start to receive data.
PHYData	1	Data signals from physical layer.

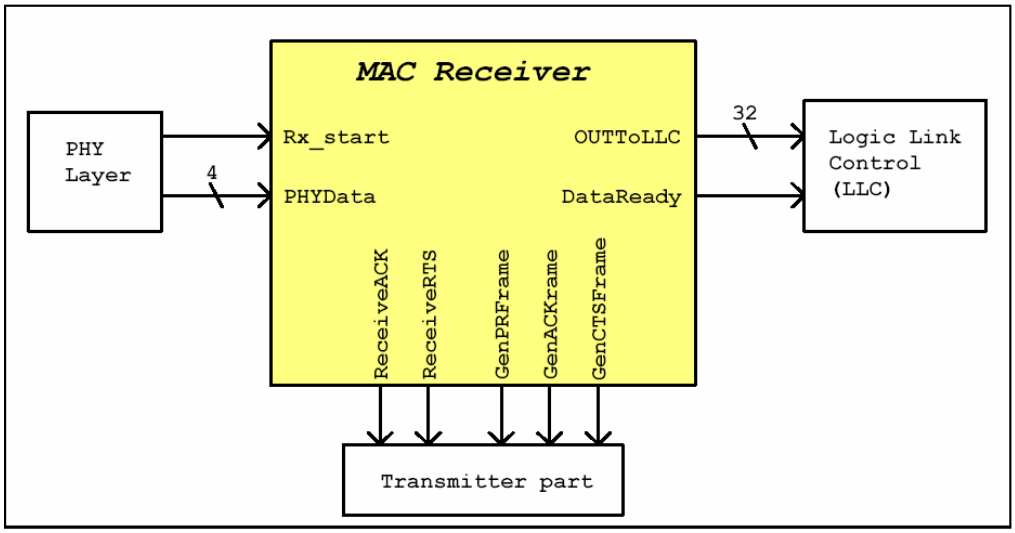


FIGURE. 4.11 RECEIVER PART SPECIFICATION [11]

The physical layer splits to two components in 802.11, the PLCP (Physical Layer Convergence Procedure) and PMD (Physical Medium Dependent). We assume the data processing was done by PLCP layer in this thesis so we don't need to parse frame from PHY layer.

The PLCP is the bridge between MAC and radio transmission, it translates MPDUs to PMD frames. The PMD is responsible for transmitting any bits it receives from the PLCP to the wireless medium by using antenna. There are some modulation modes based on data rates in 802.11(a, b, g). The major efforts are almost in the PLCP and PMD layer so we did not consider more any physical layer problem in this thesis.

- ***4.2.2.2- CRC32 Module***

The CRC32 module is used for judgment the frame is valid or not. The data input composes of three parts, MAC header + Frame body + FCS. FCS value is calculated based on CRC-32 algorithm. CRC-32 is cyclic redundancy code and 32 represent the length of checksum in bits. It uses following standard generator polynomial:

$$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x + 1$$

Calculating FCS, the frame is represented (original frame, before adding the FCS) as a polynomial, M(x), then multiplies M(x) by X32 , and divides the result by a standard generator polynomial G(x) (all of procedure is completed by using Modulo-2 operation). The resulting remainder of the above operation is FCS. The FCS is appended to the original frame to form coded frame which will be transmitted. The whole of the above operation can be mathematically represented as

CRC= Remainder of $(M(x)*X^{32})/G(x)$)

Figure 4.12 shows example of CRC generation. We set M(x) =03(h) and start to calculate the CRC value. Then get the full frame value="30D4326D9(h)".

The Table 9 shows CRC32 module interface signals and Figure.4.13 shows CRC32 Validation Function Specification that we define.

M(x)=0011(b)=3(h)

CRC=0D4326D9(h)

Full Frame =30D4326D9(h)

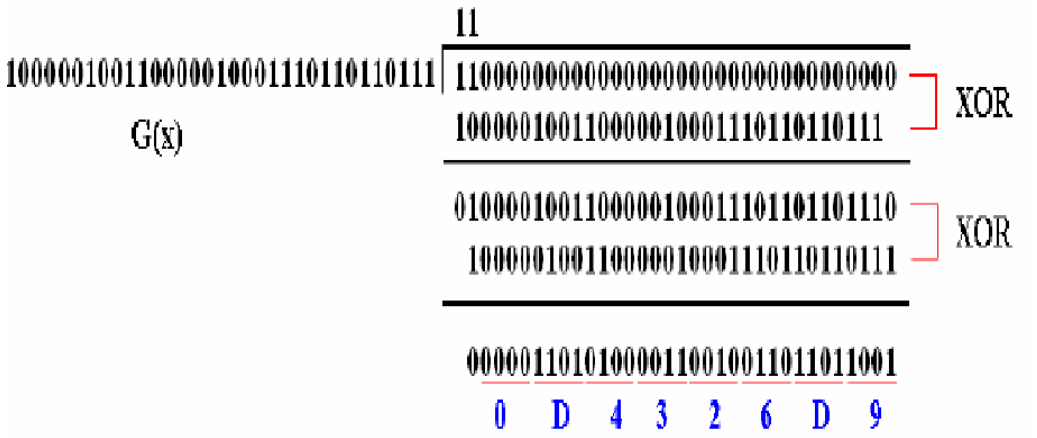


FIGURE. 4.12 EXAMPLE OF CRC GENERATION

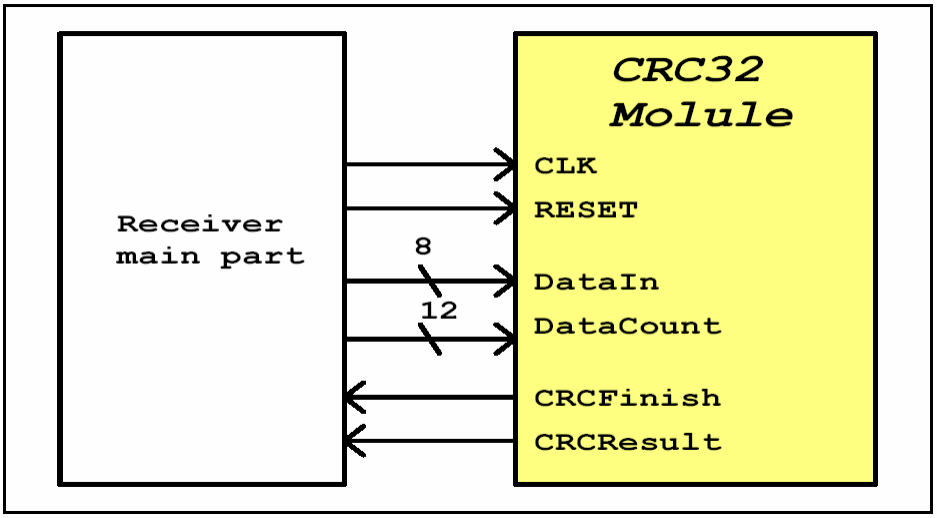


FIGURE. 4.13 CRC32 VALIDATION FUNCTION SPECIFICATION [11]

The single-bit data input (serial) calculation of CRC-32 is implemented with a linear feedback shift register (LFSR). The CRC-32 LFSR is illustrated in Figure.4.14 (register bits "3" through "25"are left out of the Figure to simplify the drawing). Presetting the flip-flops to 0xFFFFFFFF is equivalent to complementing the first 32-bits of the data stream. For the first 13 cycles, the right-most XOR gate in the Figure is an inverter. The XOR of any data with a binary "1" result in the complement of the original data. We used the reference module as a component and implemented to our CRC32 module.

Table 9 CRC32 module interface signals

NAME	TYPE	DESCRIPTION
DAT IN	I	8 bit data from host to calculate.
DATA COUNT	I	Data size crc32, maximal calculated data size =212=4096 bytes
CRC FINISH	O	Indicated CRC32 finished all data process.
CARC RESULT	O	Indicated CRC32 checked result by this module.
CLK	I	System common clock.
RESET	I	System common reset.

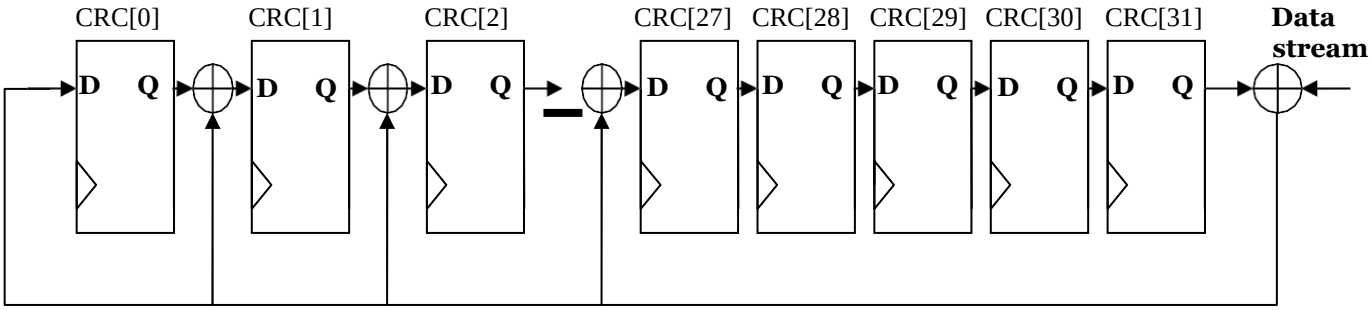


FIGURE. 4.14 LINEAR FEEDBACK SHIFT REGISTER IMPLEMENTATION OF CRC-32 [11]

Figure.4.15 shows the flow chart of CRC32 module that we modified from the reference code. We illustrate brief CRC32 flow chart as below. When we receive MPDUs from PLCP we will calculated MAC header + Frame body to generate new FCS. Then we compared new FCS to old FCS that received from PLCP. When two value are the same it means no error during received.

The CRCResult signal will output “high” and inform receiver main module. The MPDU is correct. When two value are different it means there are some errors during received. The CRCResult signal will output “low” and inform receiver main module. The MPDU is wrong and discard it [10].

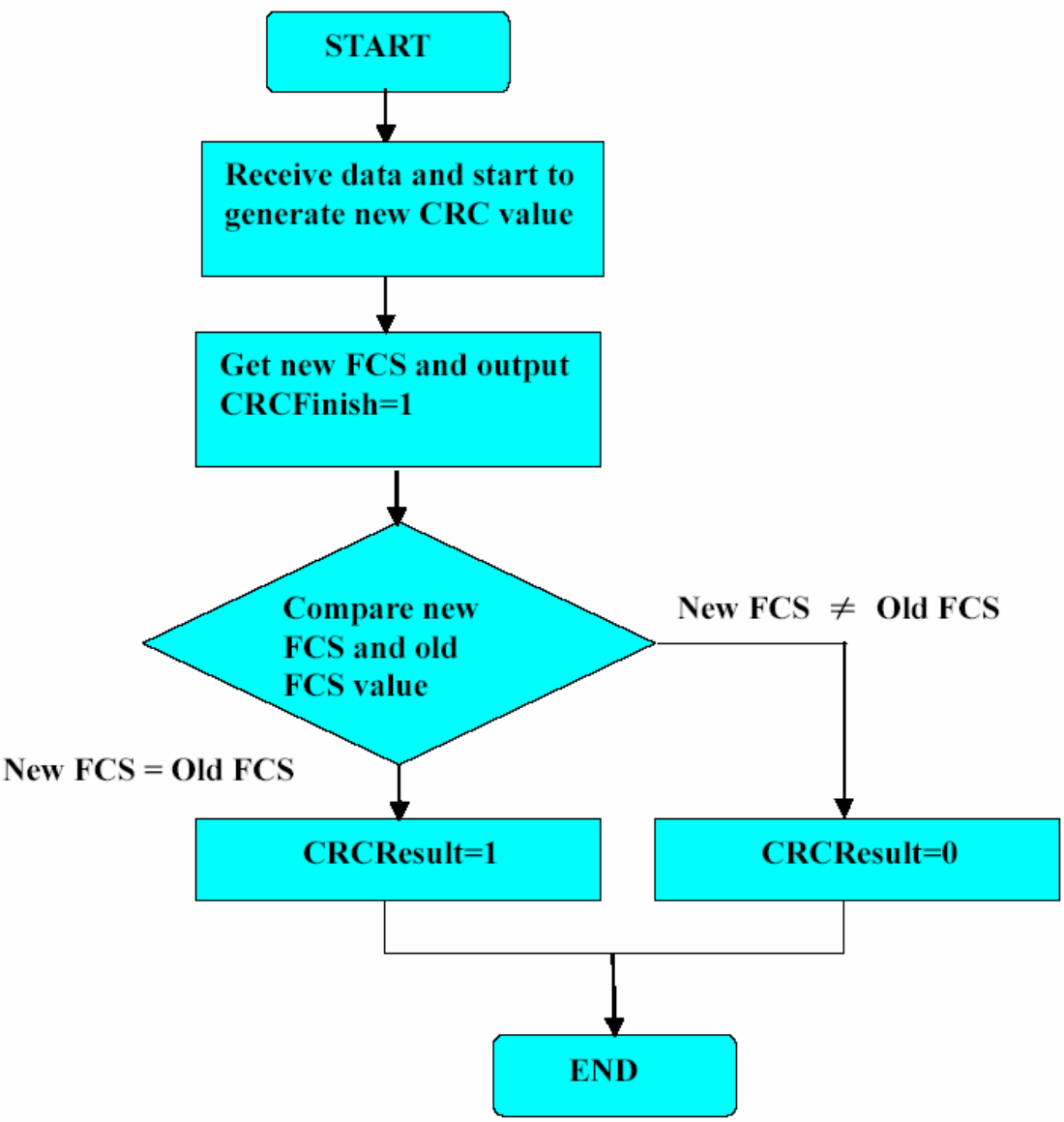


FIGURE.4.15 FLOW CHART OF CRC32 MODULE

- ***4.2.2.3- Scan and Synchronization Module***

STAs must scan firstly, join BSS and start to synchronization. Scan and synchronization is very important in receiver part. STAs shall do synchronization with a BSS before any actions. First we must scan all network environment to check any AP existed or not (If-BSS), and then update TSF by user select. Figure.4.16 shows the block diagram of this function.

We only used active scanning in this thesis. Table 10 list signals of the scan and synchronization module that we define specifications for our MAC design.

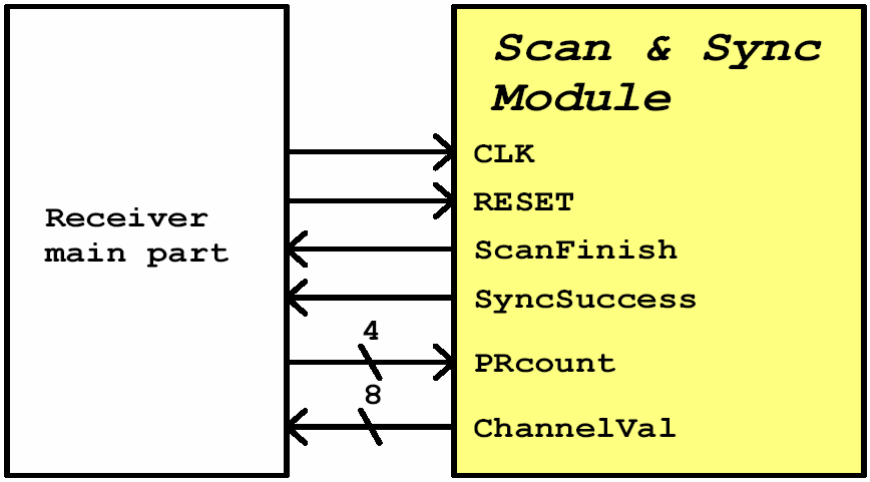


FIGURE.4.16 ACTIVE SCANNING FUNCTION BLOCK DIAGRAM [11]

Table 10 Scan and synchronization module interface signals

NAME	TYPE	DESCRIPTION
SCAN finish	O	Indicates host that all channels has been scanned completely.
Gen pr frame	O	notifies TX part to transmit probe request frame
Pr count	I	Probes response count from host.
Channel Val	O	sets channel value to PHY
Syn_Success	O	Indicates synchronization success, TSF is update by user select.
CLK	I	System common clock.
reset	I	System common reset

Scan and Synchronization flow chart is shown in Figure. 4.17. The module sets the ChannelVal to inform which channel is going to scan. Either ProbeDelay expired or an incoming frame detected, start the DCF access mechanism and TSF function firstly. Second send a ProbeRequest frame and then starts the probe timer. Wait until MinChannelTime, if the medium never busy, change to next channel, and otherwise wait until MaxChannelTime. In the same time process any ProbeResponse frames.

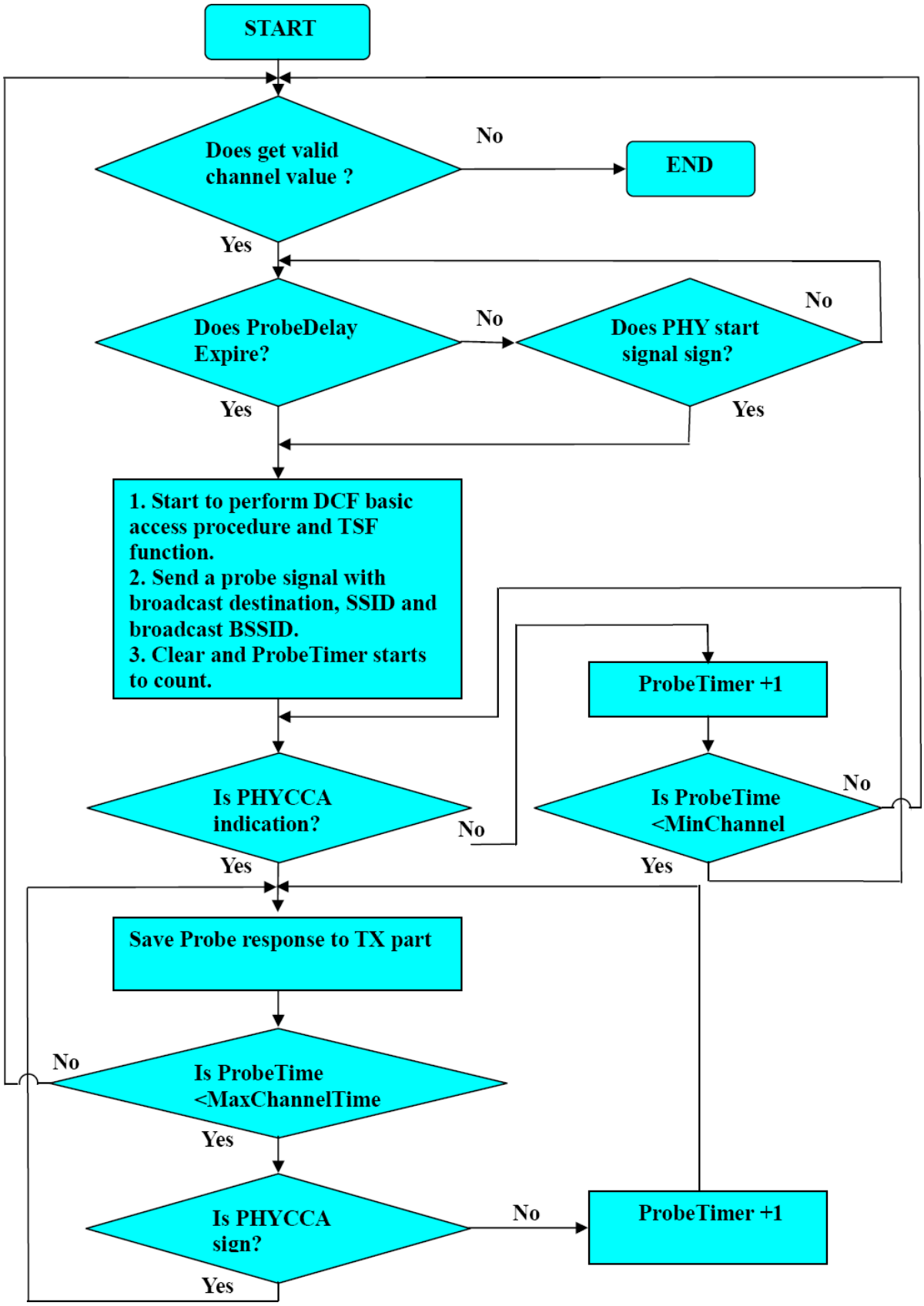


FIGURE. 4.17 SCAN AND SYNCHRONIZATION FLOW CHART

4.2.2.4 Defragmentation Module

The Defragmentation module processes the frame which is fragmented in the transmitter end. After main module check More Fragments bit in MAC header, if More Fragments bit was set to 1, this module saves incoming frame body and sequence control, then reassembly them after all fragments of MSDU received successfully. Figure.4.18 shows defragmentation function block diagram. Table 11 list signals of the module.

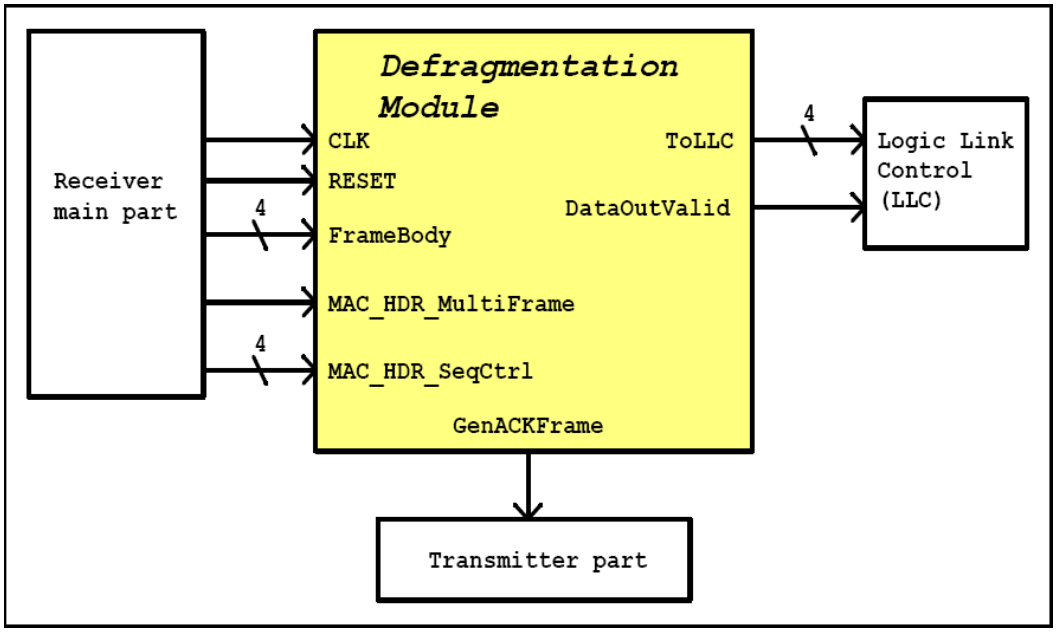


FIGURE.4.18 DEFRAGMENTATION BLOCK DIAGRAM

Table 11 Defragmentation interface signals

Name	Type	Description
ToLLC	O	After data defragmentation complete, send data out to LLC through these signals.
MAC_HDR_MultiFrame	O	Multi-frame indicator.
Mac_HDR_SeqCtrl	I	MAC header information contains sequence control field.
DataOutValid	O	Indicate defragmentation done, data are valid.
FrameBody		Frame input from main module.
GenACKFrame	O	Notifies transmitter send ACK frame.
CLK	I	System common clock.
RESET	I	System common reset.

Figure.4.19 shows the Defragmentation module flow chart. If the More Fragments bit is clear in the MAC_HDR_Multiframe, then reassembly the data in the buffer and sent to LLC according to previous data information. If the More Fragments

bit is set, then sort the data in buffer to check. Is it duplicate frame or not? Then discard duplicate frame or save new frame data to buffer.

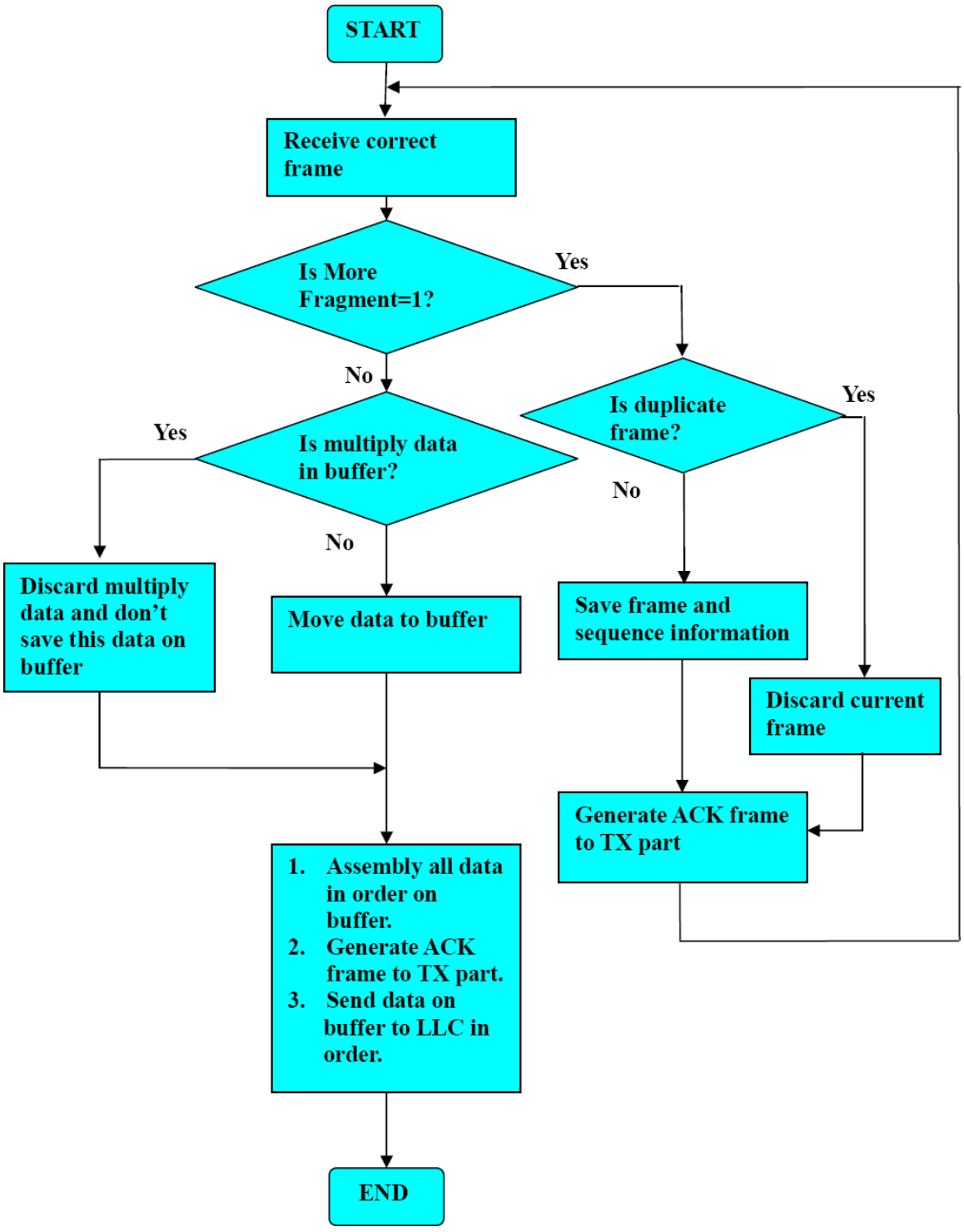


FIGURE. 4.19 DEFRAGMENTATION FLOW CHART

- ***4.2.2.5 Functional Verify Procedure of Receiver Module***

In order to verify our designs, we will verify all modules' functions by the flow chart which is shown in Figure.4.20. We combined all modules as a MAC receiver part. First we perform the synchronization module to find the AP, then assume one large MSDU was segmented to three small frames transmitted in transmitter, so defragmenter module assemble these frames in receiver, then sent to upper layer, data link layer [11] .

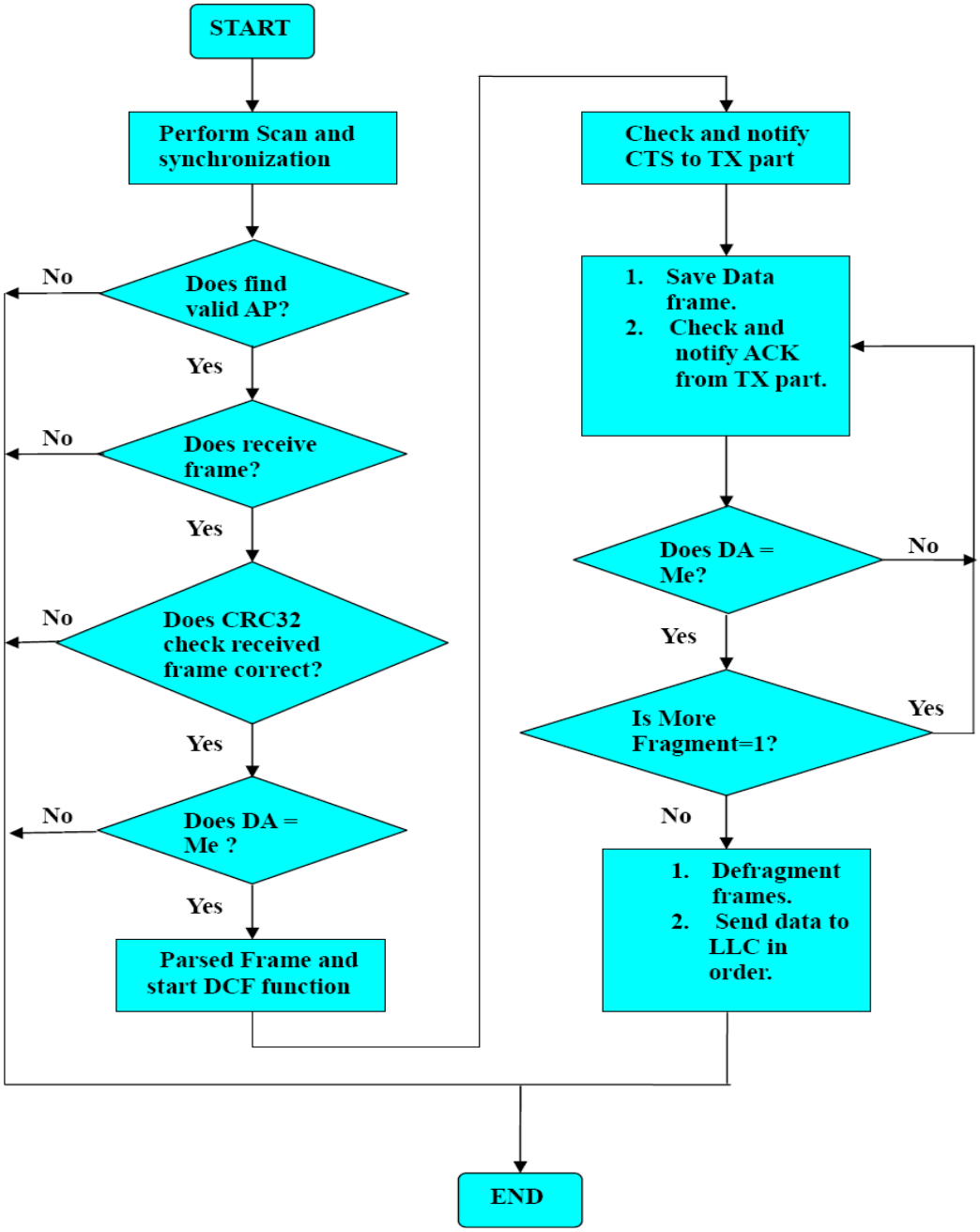


FIGURE. 4.20 FULL FUNCTION FLOW CHART

SIMULATION AND VERIFY RESULTS
4-3 Simulation and Results

We are using ModelSim software to simulate for each module separately, and then do a full function simulation based on our design.

4-3-1 CRC32 function

Because the CRC32 generate module we refer to exist code on Xilinx support Web site. First we must validate CRC32 generation function. Is the module workable? We input one data (“03h”) as Figure. 4.12. The CRC result is “0D4326D9h”. Figure. 4.21 shows simulation result and simulation result is 0D4326D9h” as well

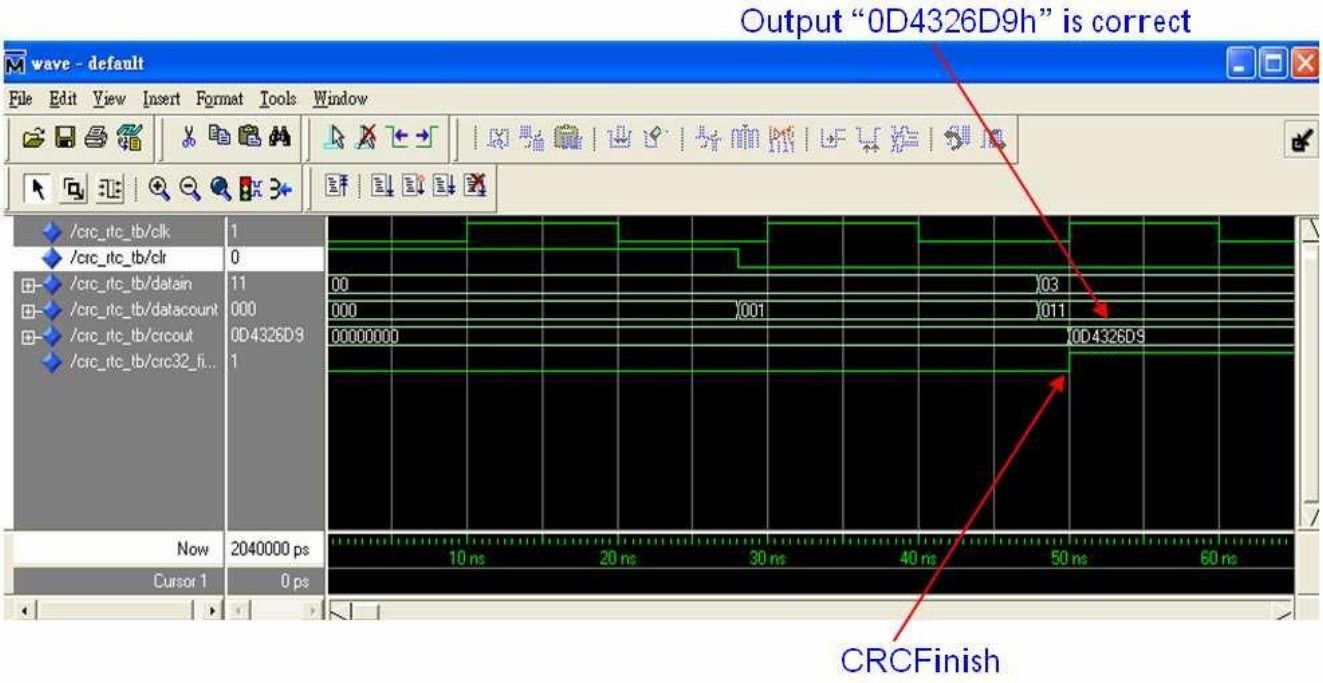


FIGURE. 4.21 CRC32 GENERATION FUNCTION VALIDATION

Than we verify frame integrity after received. We assume a RTS frame is received from RX module. We shown in Table 12 and its CRC32 value is as below list:

11111001010000100011011111110011 (F94237F3)

After CRCfinish signal go to high we check CRCResult signal behavior.If CRCResult signal goes to high the result is corrected. Please check Figure. 4.22. We assume get a correct data frame and the CRCResult signal goes to high.

Please check Figure. 4.23. We assume get a corrupt RTS frame and the CRCResult signal still keeps low. It means the CRC32 function is workable and the result is correct.

Table 12 RTS frame data

Frame Control	Duration	RA	TA	FCS
2D00	E610	AABBCCDDEEFF	665544332211	F94237F3

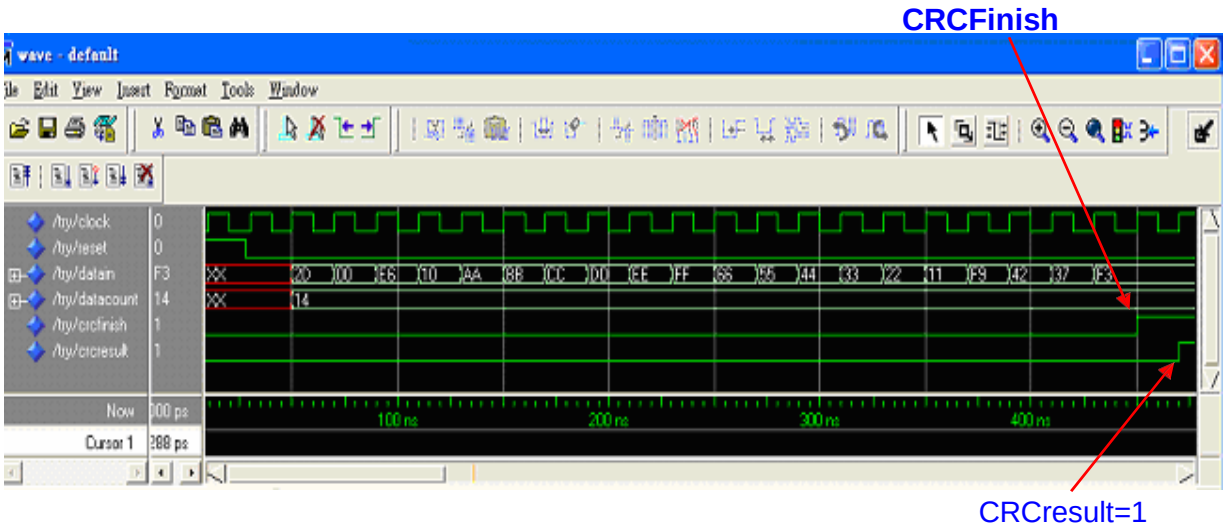


FIGURE. 4.22 CRC32 GOT A CORRECT RTS FRAME

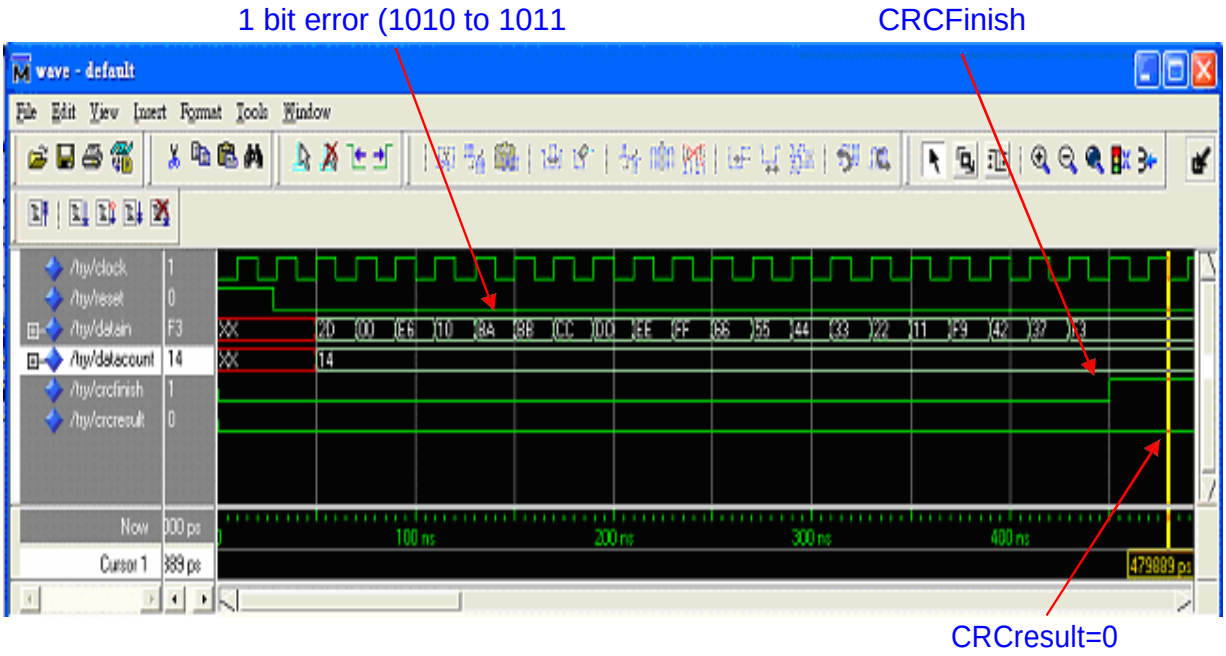


FIGURE. 4.23 CRC32 GOT A CORRUPT RTS FRAME

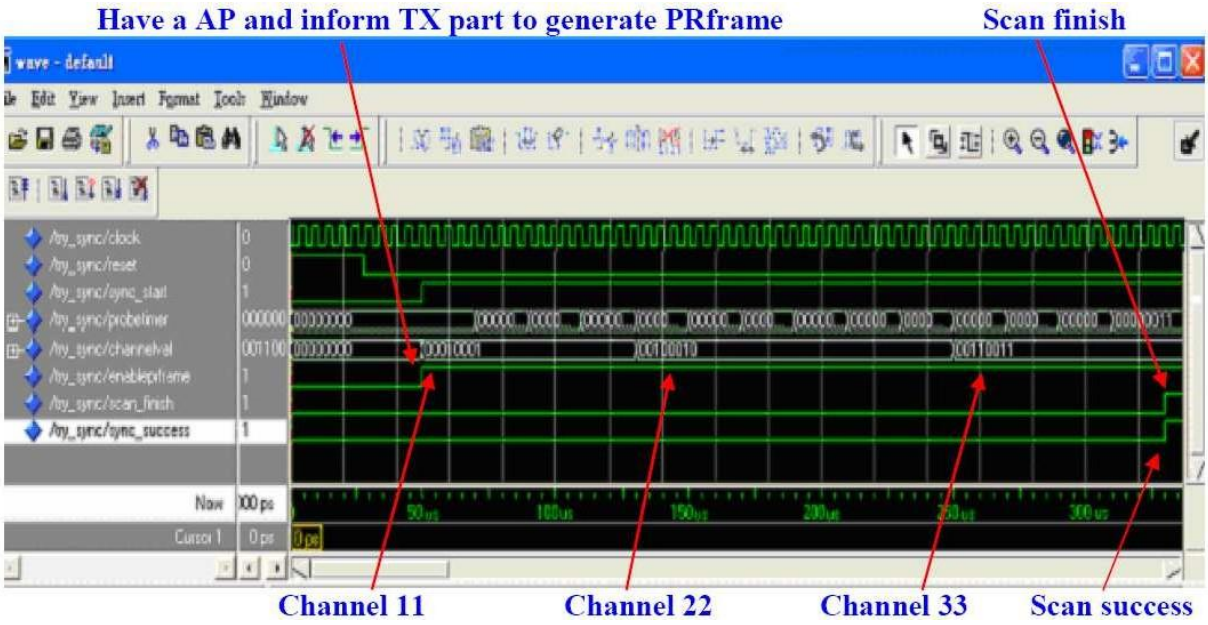


FIGURE. 4.24 ACTIVE SCANNING SUCCESS

4-3-2 Scan and synchronization function

We validated scan and synchronization function in the same way as CRC32 function. We assume the MinChannelTime is 100us, MaxChannelTime is 150us, and there are three channels (0x11,0x22,0x33) need to be scanned. In The Figure.4.24 shows the active scanning function is performed successfully [13].

The Figure.4.25 shows scan failure symptom.If there is not any ProbeResponse frame in these three channels before reach the MinChannelTime the scanfinish signal will output “low”. It means that scan AP fail.

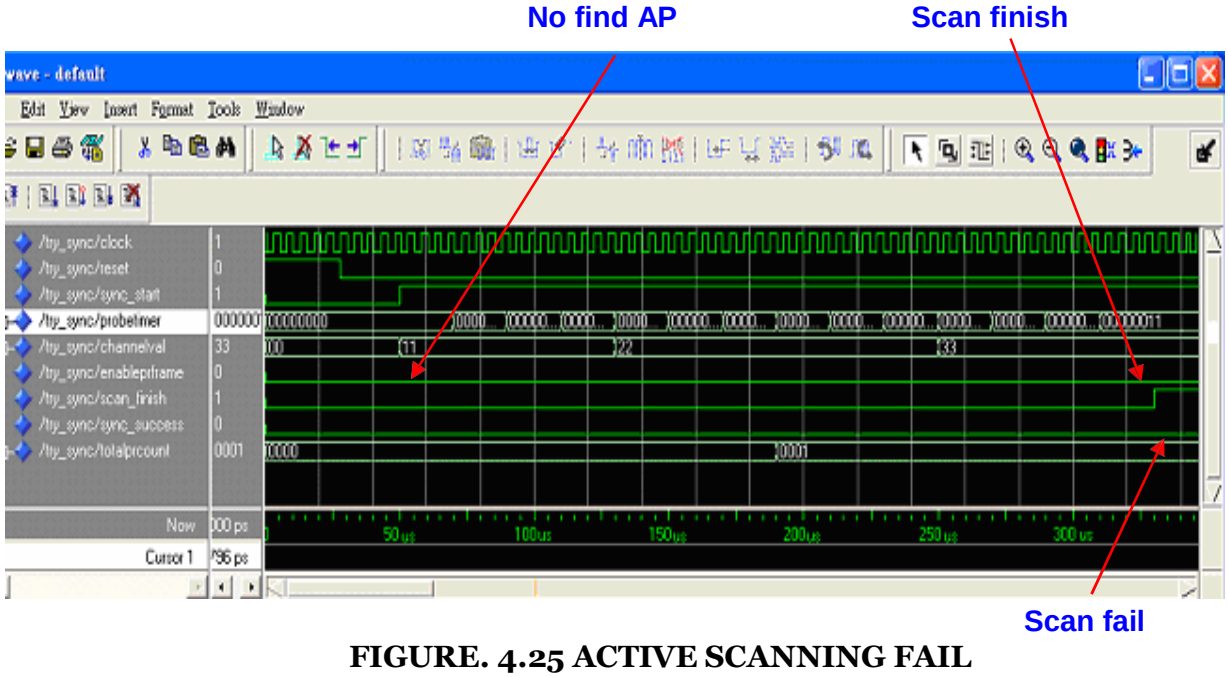


FIGURE. 4.25 ACTIVE SCANNING FAIL

4-3-3 Defragmentation function

Fragmentation is a MAC-level proposition that essentially tries to minimize the risk of suffering a frame error due to a difficult terrain presented by the PHY. When MSDU is grater than fragment threshold, transmitter will fragment MSDU into small frames, receiver should defragment (assembly) these frames by their sequence in MAC header. We simulate the four cases to validate defragmentation function [14].

Case 1: Only one data frame and MAC_HDR_Multiframe= 0 (More Fragments = 0) received. So defragmentation module can pass frame body to LLC directly. Defragmentation module informs TX part to generate ACK frame. Figure.4.26 shows simulation result.

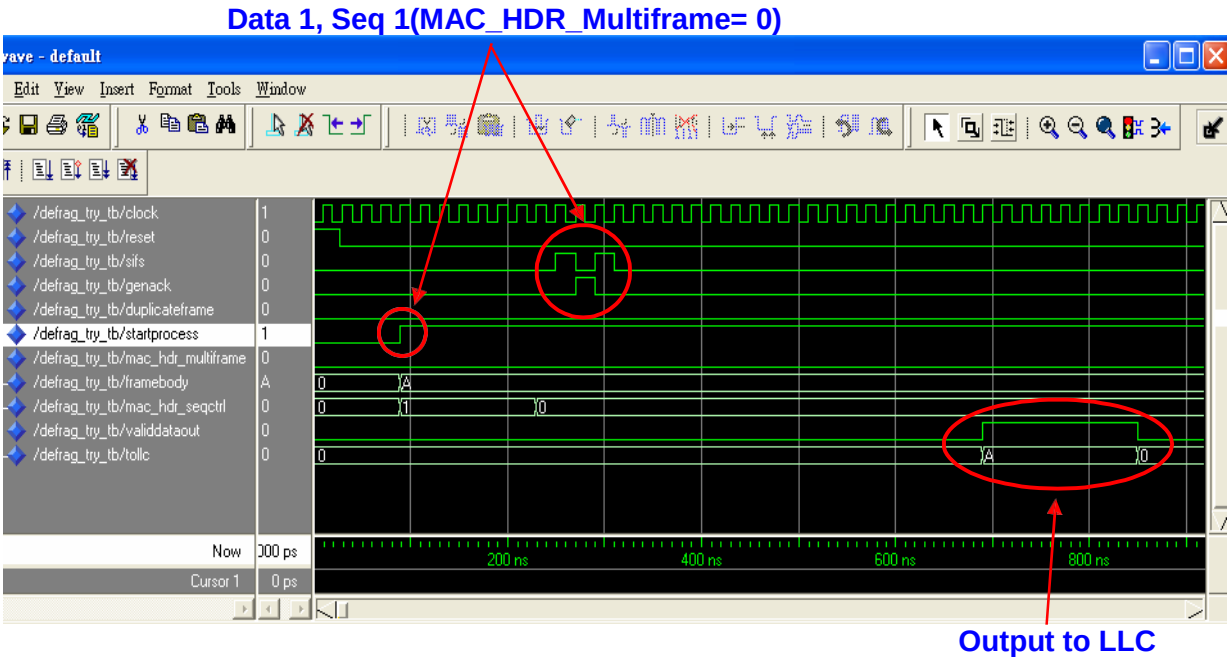


FIGURE. 4.26 DEFRAGMENTATION (SINGLE FRAME)

Case 2: There are three data frames received with normal sequence and no duplicate frame detected.

Table 13 list three data with its sequence number. We input these data in order to defragmentation module which we design. Figure.4.27 shows the simulation result as we expect. There are three data that are received and output to LLC in order.

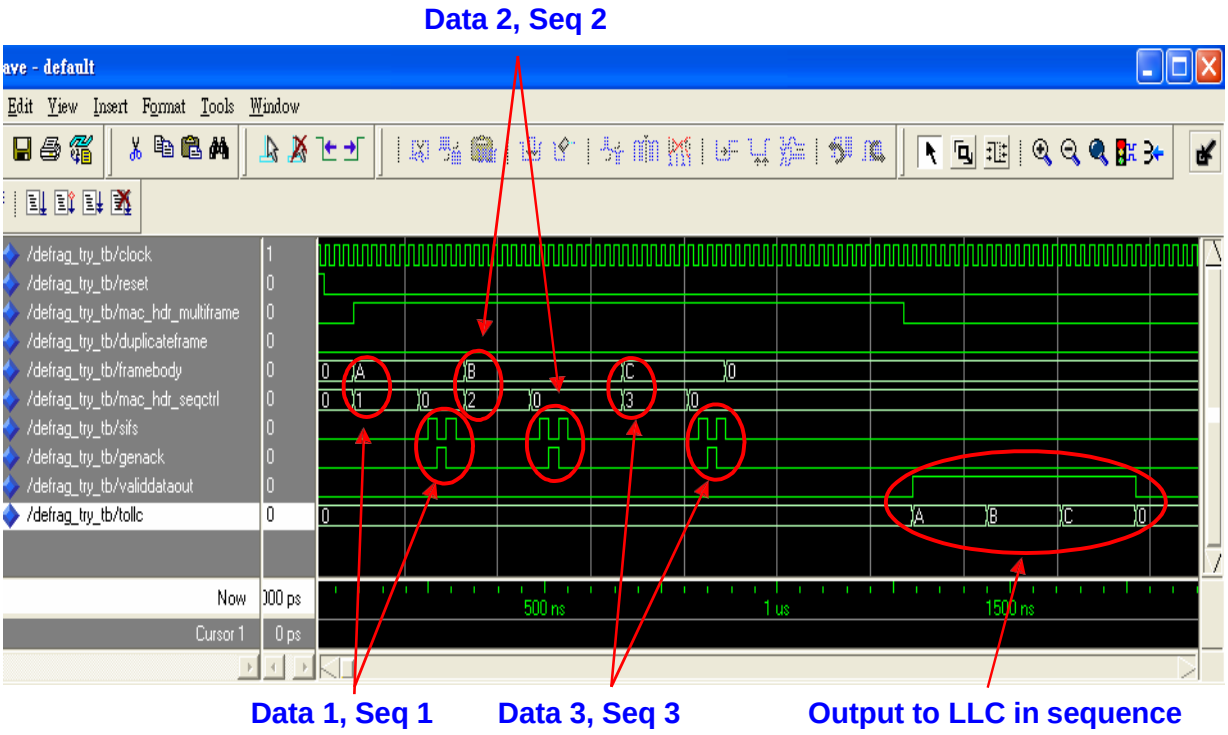


FIGURE. 4.27 DEFRAGMENTATION (MULTIPLY FRAMES WITH NORMAL SEQUENCE)

Table 13 Three data with sequence number

	Data1	Date2	Data3
Data	AAAAAAAA	BBBBBBBB	CCCCCCCC
Sequence number	01	02	03

Case 3: There are three data frames received deranged sequence and no duplicate frame detected.

The “frame one” sequence number is one, “frame two” sequence number is three, “frame three” sequence number is two. There are three data that are received but are not in sequence number. Defragmentation module will rearrange sequence and output data in sequence to LLC output. Figure. 4.28 shows the simulation result as we expect and data output to LLC are correct.

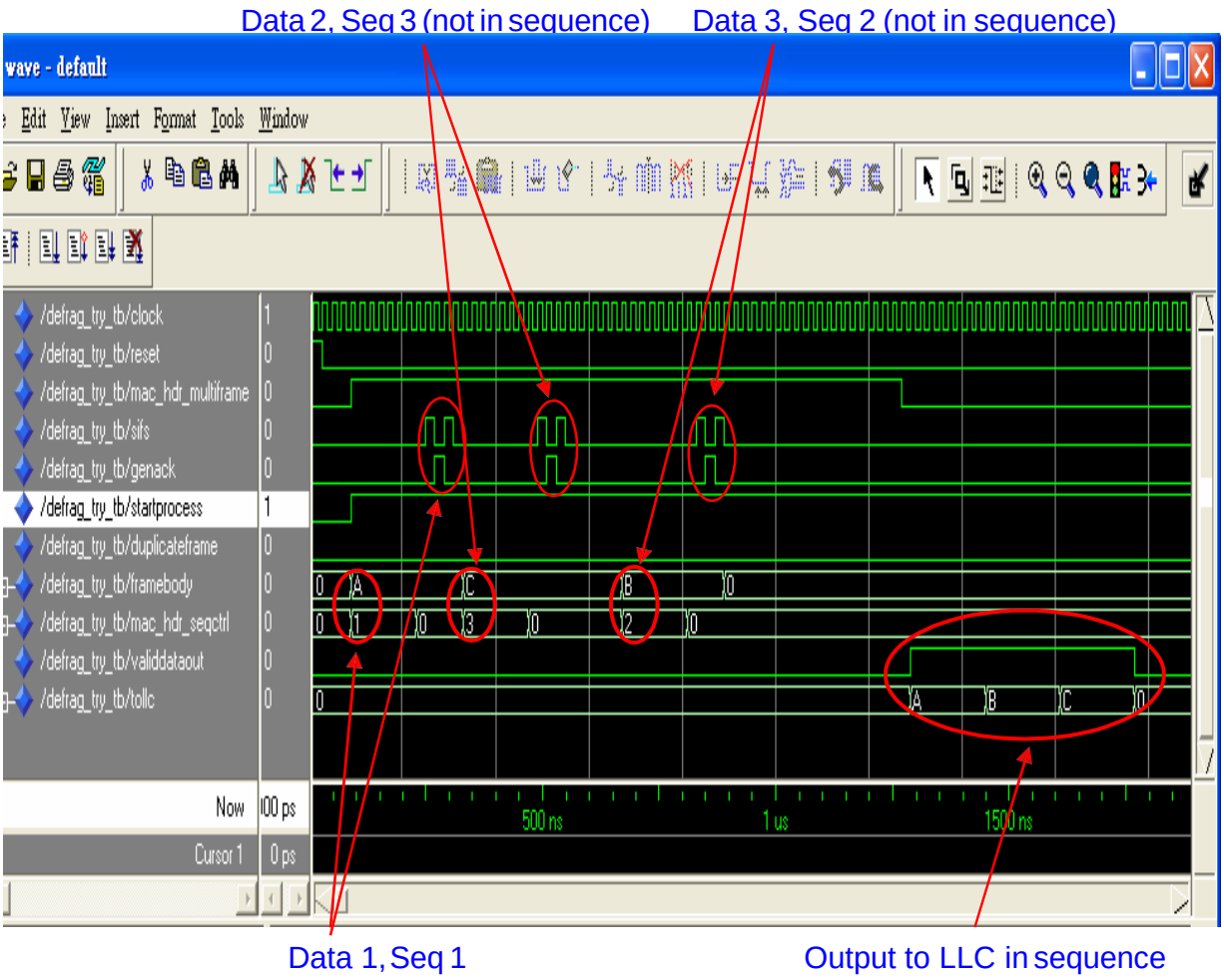


FIGURE. 4.28 DEFRAGMENTATION (MULTIPLY FRAMES WITH DERANGED SEQUENCE)

Case 4: There are three data frames received, deranged sequence and one duplicate frame detected.

The “data frame one” sequence number is one. The “data frame two” has the same sequence number and data as data frame one. Defragmented module will discard duplicate frame two and output correct data of buffer in sequence to LLC. Figure. 4.29 shows the simulation result. Because data frame two is discarded defragmentation module only send two data to LLC.

Through four case simulation results we can make sure our design that meets our requirement.

4-3-4 Full function verification

We set one situation as Figure. 4.20 to verify the full function of our design.

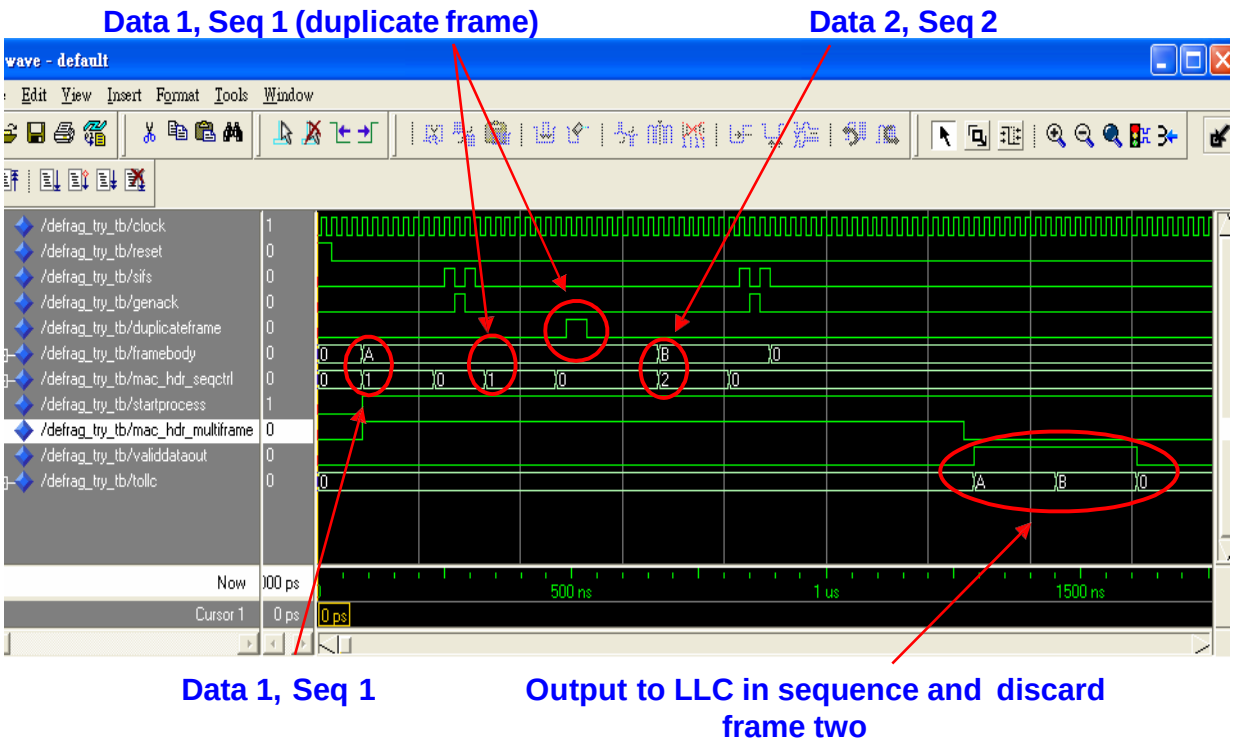


FIGURE. 4.29 DEFRAGMENTATION (MULTIPLY FRAMES WITH ONE DUPLICATE FRAME)

We assume there are four data frames that were received from PHY layer. We list on Table 14. First, the module scans the AP in the three channels and then starts to receive the data frames. Finally reassembly the frame body and sent out to LLC. Figure. 4.30 shows the total simulation result and Figure.4.31 shows frame body data to LLC in order. The result is correct.

Table 14 Four data frames from PLCP layer

No	Frame Control	Duration	Address1	Address2	Address3	Sequence Control	Frame Body	FCS
1	2081H	C400H	AABBCCDDEEFFH	001122334455H	001122334455H	0001H	35724163H	9964104AH
2	2081H	C400H	AABBCCDDEEFFH	001122334455H	001122334455H	0002H	254468DBH	74EFF292H
3	2081H	C400H	AABBCCDDEEFFH	001122334455H	001122334455H	0003H	33224567H	F6100549H
4	20A1H	C400H	AABBCCDDEEFFH	001122334455H	001122334455H	0004H	ABAC3421H	6CD0D5A5H

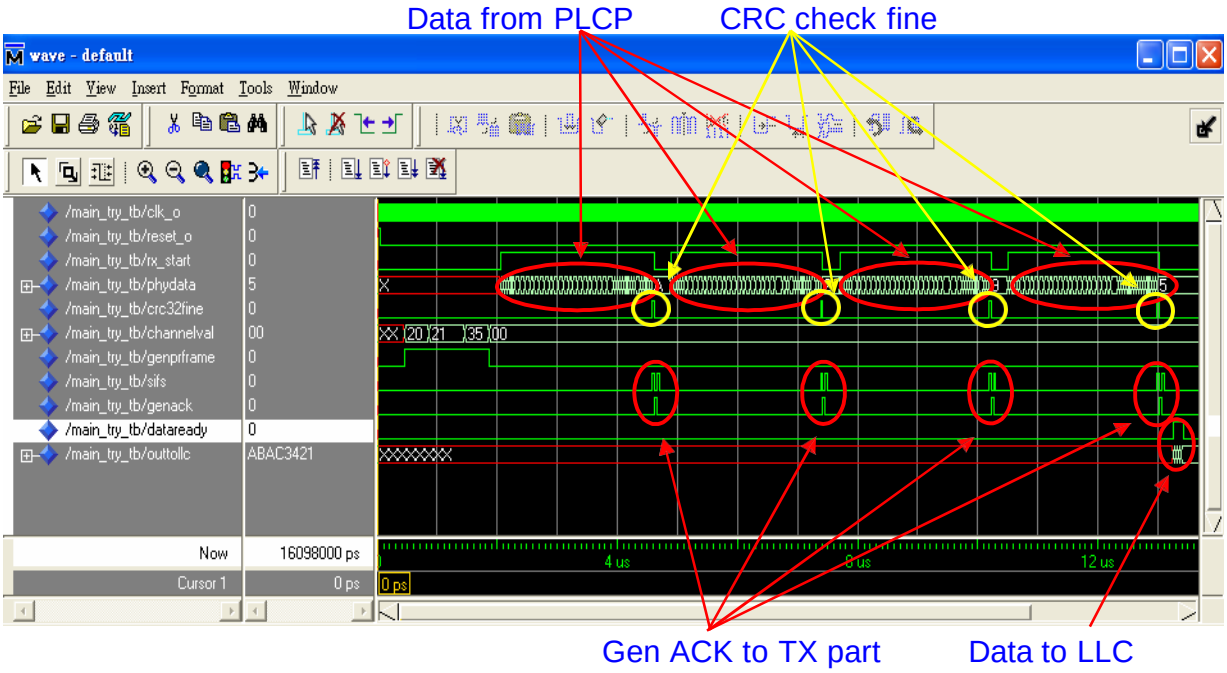


FIGURE. 4.30 FULL FUNCTION SIMULATED RESULT

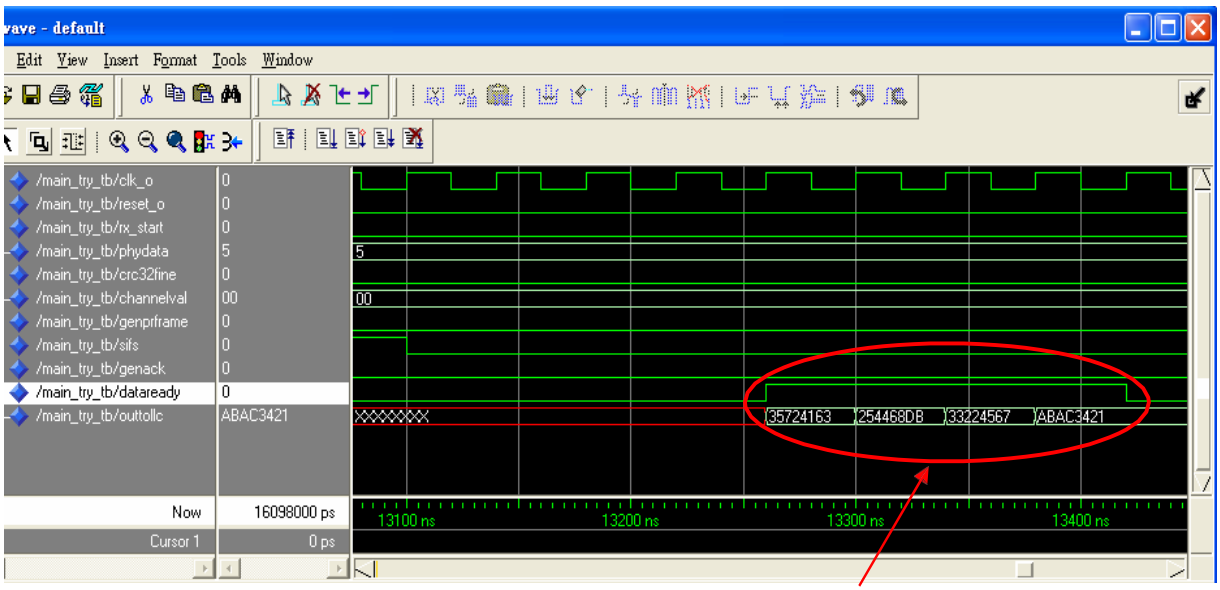


FIGURE. 4.31 FULL FUNCTION SIMULATED RESULT (TO LLC DATA)

4-4 MAC Design Method Comparison

We make a MAC design method comparison table of Table 15. There are three populate ways to design MAC. These are CPU-Base, FPGA and vender total solution.

Table 15 MAC design method comparison table

Method	Development time	Design flexibility	Design mobility	Speed/Power consume	Cost
CPU-Base(Intersil 384, ARM…)	Short	High	Medium	Low / High	High
FPGA	Long	High	High	High/Low	Low
Totalsolution (BMC4318, RTL8186…)	Medium	Low	Low	High/ Low	Medium

For the design flexibility, if you need other features that want toimplement in your design you can use reserved I/O to design. That will be better than using total solution.

For design mobility, we used VHDL language to design and can implement to all FPGA vendor. We can select a suitable one based on our function or cost requirement.

For speed and power consumed FPGA has nice performance than CPU-Base design [17].

For cost, because more and more people used FPGA to design network protocol communication products, FPGA cost is cheaper than before. Some chips only need 2US dollars.

Though we use FPGA to design will need more development time. We can separate our design into several modules and make various people to validate individual module in the same time. That can decrease development time. Because of many good features we use FPGA to design MAC receiver.

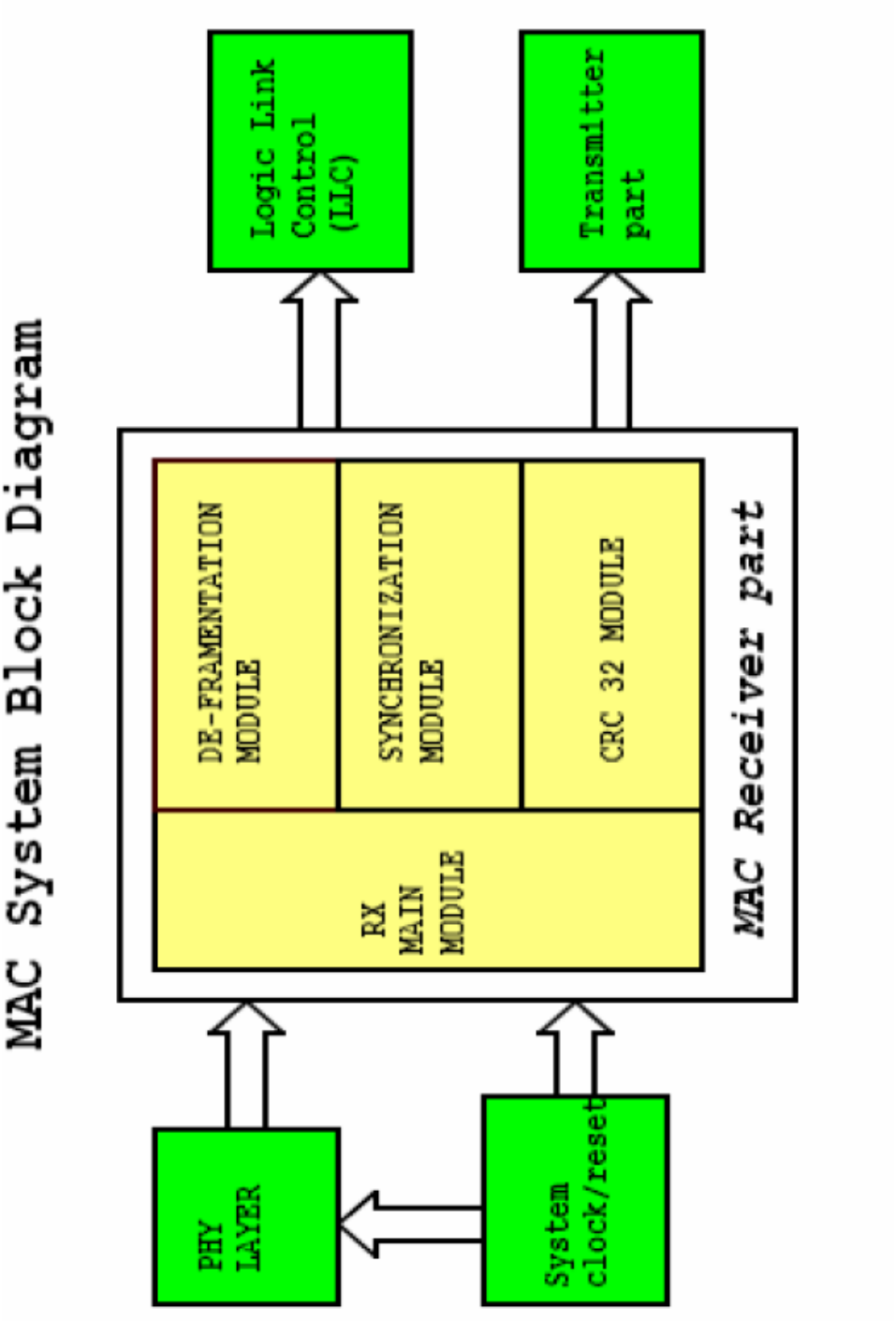
Most people design FPGA by using Verilog language. We try to design by using VHDL language. Both are IEEE standards and are supported by all the major Electronic Design Automation (EDA) vendors. Both can be used for designing Application-Specific Integrated Circuits (ASICs) and simulating systems. However, VHDL support for system level modeling and simulation is far more comprehensive than Verilog. We compare and contrast individual aspects of the two languages of Table 16 [18] [19].

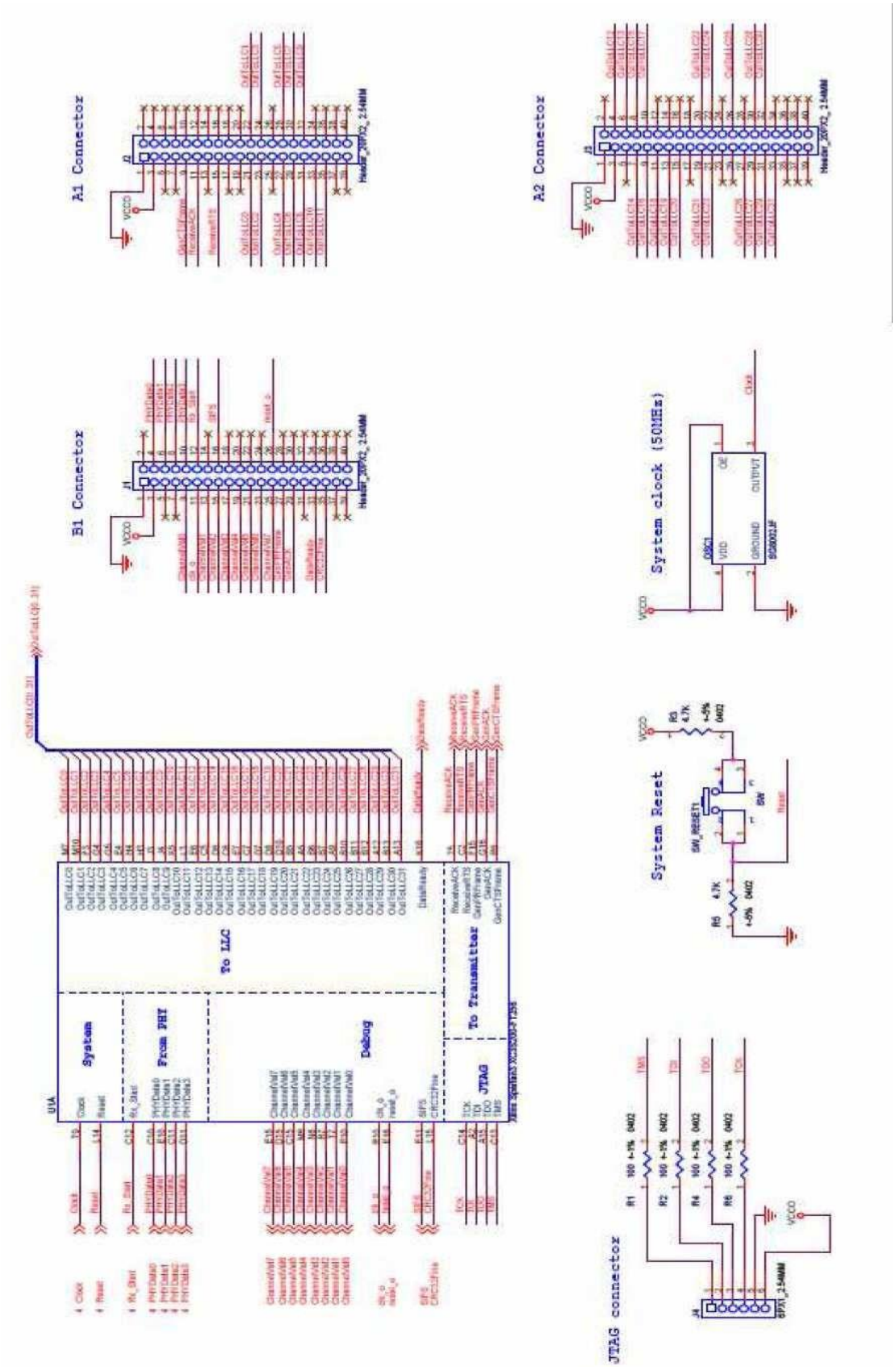
Table 16 VHDL/Verilog compared & contrasted [8]

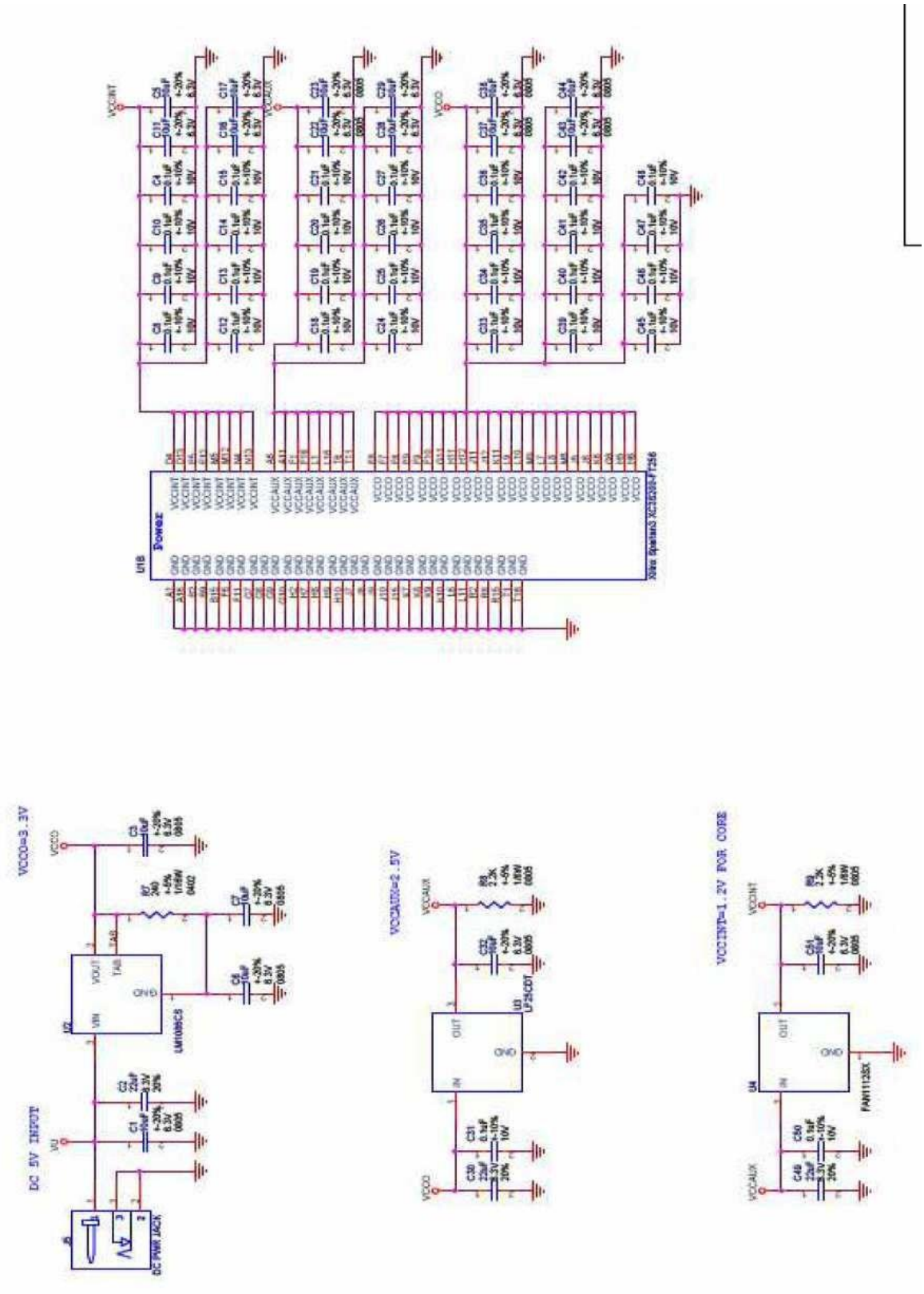
Property	**Design language**	**Compared & Contrasted**
Compilation	VHDL	Multiple design-units (entity/architecture pairs), that reside in the same system file, may be separately compiled if so desired. However, it is good design practice to keep each design unit in its' own system file in which case separate compilation should not be an issue.
	Verilog	The Verilog language is still rooted in it's native interpretative mode. Compilation is a means of speeding up simulation, but has not changed the original nature of the language. As a result care must be taken with both the compilation order of code written in a single file and the compilation order of multiple files. Simulation results can change by simply changing the order of compilation.
Data types	VHDL	A multitude of language or user defined data types can be used.
	Verilog	Compared to VHDL, Verilog data types a re very simple, easy to use and very much geared towards modeling hardware structure as opposed to abstract hardware modeling.
Design reusability	VHDL	Procedures and functions may be placed in a package so that they are avail able to any design-unit that wishes to use them.
	Verilog	There is no concept of packages in Verilog. Functions and procedures used within a model must be defined in the module. To make functions and procedures generally accessible from different module statements the functions and procedures must be placed in a separate system file and included using the `include compiler directive.
Easiest toLearn	VHDL	VHDL may seem less intuitive at first for two primary reasons. First, it is very strongly typed; a feature that makes it robust and powerful for the advanced user after a longer learning phase. Second, there are many ways to model the same circuit, specially those with large hierarchical structures.
	Verilog	Verilog is probably the easiest to grasp and understand.
Libraries	VHDL	A library is a store for compiled entities, architectures, packages and configurations. Useful for managing multiple design projects.
	Verilog	There is no concept of a library in Verilog. This is due to it's origins as an interpretive language.
Managing large designs	VHDL	Configuration, generate, generic and package statements all help manage large design structures.
	Verilog	There are no statements in Verilog that help manage large designs.
Readability	VHDL	VHDL is a concise and verbose language; its roots are based on Ada.

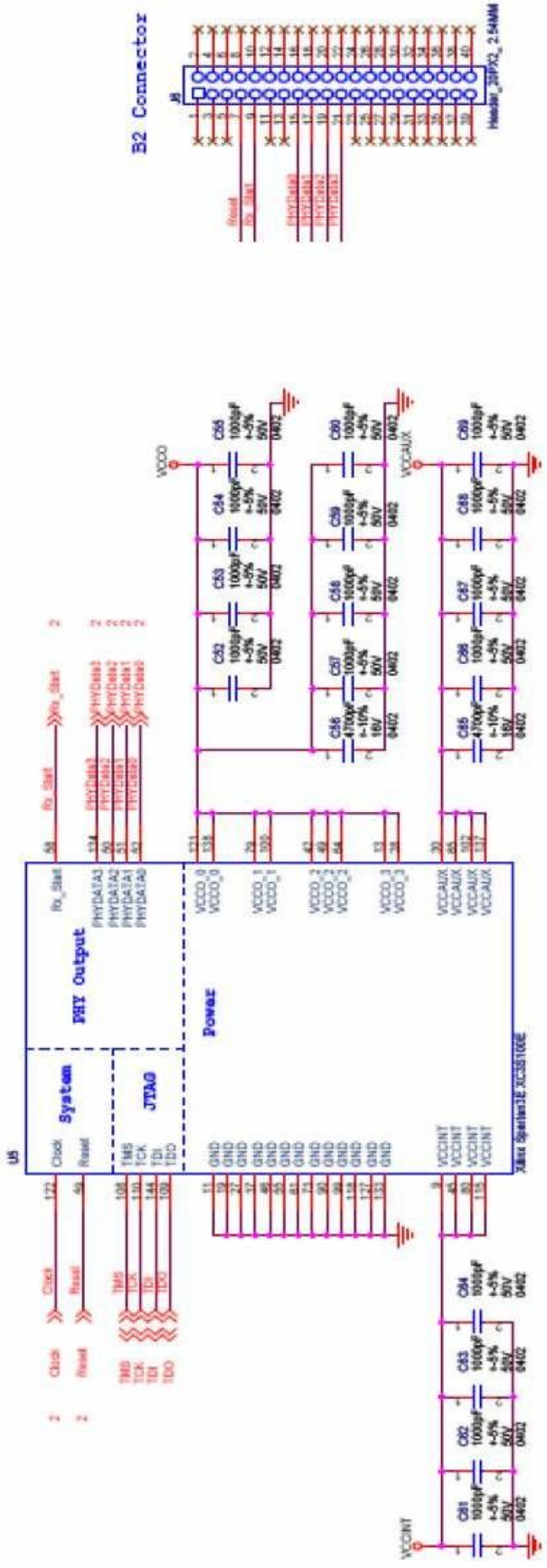
	Verilog	Verilog is more like C because its' constructs are based approximately 50% on C and 50% on Ada.
Structural replication	VHDL	The generate statement replicates a number of instances of the same design-unit or some sub part of a design, and connects it appropriately.
	Verilog	There is no equivalent to the generate statement in Verilog.
Verboseness	VHDL	Because VHDL is a very strongly typed language models must be coded precisely with defined and matching data types. This may be considered an advantage or disadvantage. However, it does mean models are often more verbose, and the code often longer, than its Verilog equivalent.
	Verilog	Unused bits will be automatically optimized away during the synthesis process. This has the advantage of not needing to model quite as explicitly as in VHDL, but does mean unintended modeling errors will not be identified by an analyzer.

VHDL requires longer learning and is not so amenable to quick-and-dirty coding but some properties for compilation, design reusability, libraries, managing large designs and structural replication are better than using Verilog to design FPGA. That is why we choose VHDL to design MAC receiver part of FPGA in the thesis.


MAC System Block Diagram
Logic Link Control (LLC)
Transmitter part
DE-FRAMENTATION MODULE
SYNCHRONIZATION MODULE
CRC 32 MODULE
RX MAIN MODULE
MAC Receiver part
PHY LAYER
System clock/reset


A1 Connector
A2 Connector
B1 Connector
System clock (50MHz)
System Reset
JTAG connector
To LLC
From PHY
To Transmitter
System
Debug
JTAG


VCCINT
VCCAUX
VCCO
U18
Power
Xilinx Spartan3 XC3S400A-FT256
VCCO=3.3V
DC 5V INPUT
DC PWR JACK
LM1086CS
VCCAUX=2.5V
LP2985IDT
VCCINT=1.2V FOR CORE
FAN1112SX


B2 Connector
J8
Header_20PX2_2.54MM
Rxsel
Rx_Start
PHYData0
PHYData1
PHYData2
PHYData3
U5
PHY Output
System
JTAG
Power
Clock
Rxsel
TMS
TCK
TDI
TDO
VCCO
VCCAUX
VCCINT
GND
PHYDATA3
PHYDATA2
PHYDATA1
PHYDATA0
Xilinx Spartan3E XC3S1600E

CONCLUSION

OFDM technique has been widely implemented in high-speed digital communication to increase the robustness against frequency selective fading or narrowband interface. The major advantage of OFDM is the ability to enhance the basic signal using approaches that can overcome channel impairments.

In digital communication, binary information is converted by means of a modulator into a continuous-time signal which is sent over the transmission channel. A digital receiver is to extract the information sequence from a discrete signal obtained after sampling and quantizing the distorted signal presented to the demodulator. At the receiver, accurate timing recovery is critical to obtained performance close to that of the optimal receiver timing in a data receiver must be synchronized to the symbols of the incoming data signals

We implement IEEE 802.11 MAC receive controller by FPGA method with VHDL langrage in this work and base on IEEE 802.11b/g specification. For the architecture, it can realize the DCF function of receiver part and ensure those functions are workable. We can use those modules and implement in other vendor's FPGA. For fulfilled that cost down and customize requirement or other similar application.

This work in an opportunities to takes part of wireless system, this simulation show the importance of technique OFDM to provide immunities against

VHDL is managing large designs better than Verilog. VHDL is a very strongly typed language models must be coded precisely with defined and matching data types. When we combine many people's modules of one design for FPAG, we can reduce compatible issue and development time. Compared software simulation results to hardware test results and we can find some timing difference. Designer needs to notice this issue. The difference is caused by synthesizer when implemented to FPGA. Logic gates will have delay time. Difference pin assign to FPGA will impact synthesized results as well.

In the future, we plan to analysis the performance and cost of the proposed MAC protocols. And we will try to implement the protocols to our developed hardware, the mobile sensor nodes can be taken into consideration and the issue of schedule maintenance must be solved. Thus, we can plan to design a more efficient MAC protocol.

Index

```
PROGRAMM
--      AccelDSP 10.1. build 1134 production, compiled 2008-02-12
--
--    THIS IS UNPUBLISHED, LICENSED SOFTWARE THAT IS THE CONFIDENTIAL
--       AND PROPRIETARY PROPERTY OF XILINX OR ITS LICENSORS
--
--      Copyright(c) Xilinx, Inc., 2000-2008, All Rights Reserved.
--    Reproduction or reuse, in any form, without the explicit written
--         consent of Xilinx, Inc., is strictly prohibited.
--
--  User: soufi_aek
--  Machine: SOUFI (i686, Windows XP Service Pack 2, 5.01.2600)
--  Date: Tue Nov 25 21:25:46 2008
--
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;


entity iir_filter is
        port (
                sample : in signed( 8 downto 0 );
                y : out signed( 30 downto 0 );
                ac_InputAvail : in std_logic;
                ac_OutputAvail : out std_logic;
                Reset : in std_logic;
                Clock : in std_logic
        );
end iir_filter;


architecture RTL of iir_filter is

        signal signal_y : signed( 30 downto 0 ) := ( others => '0' );
        signal signal_ac_OutputAvail : std_logic := '0';

        signal ac_InputAvail_internal : std_logic;
        signal ac_is_busy : boolean;
        constant ac_scalar_coeff_a_0 : signed( 9 downto 0 ) := "1111100110";
        constant ac_scalar_coeff_a_1 : signed( 9 downto 0 ) := "0000110011";
        constant ac_scalar_coeff_b_0 : signed( 9 downto 0 ) := "0000110011";
        signal ac_scalar_x_n_0_reg : signed( 8 downto 0 );
        signal ac_scalar_x_n_1_reg : signed( 8 downto 0 );
        signal ac_scalar_x_n_2 : signed( 8 downto 0 );
        signal ac_scalar_y_n_0 : signed( 20 downto 0 );
        signal ac_scalar_y_n_1 : signed( 20 downto 0 );
        signal mtimes_001_accum_2 : signed( 20 downto 0 );
        signal mtimes_002_accum_3 : signed( 32 downto 0 );
        signal proceed_var : boolean;
        signal sample_internal : signed( 8 downto 0 );
        signal ac_shared_adder_1_result : signed( 20 downto 0 );
        signal ac_shared_multiply_3_result : signed( 30 downto 0 );
        type STATE_VAR_DEF_iir_filter_process_state is
        (
                sm0_1,
                sm0_0
        );
```

```
	signal iir_filter_process_state : STATE_VAR_DEF_iir_filter_process_state;
begin
	y <= signal_y;
	ac_OutputAvail <= signal_ac_OutputAvail;

	proceed_var <= ((ac_InputAvail_internal = '1') or ac_is_busy);

	ac_shared_multiply_3_process_combinational_1:
	process ( iir_filter_process_state, ac_scalar_y_n_0, ac_scalar_y_n_1 ) is
		variable ac_shared_multiply_3_lhs : signed( 9 downto 0 );
		variable ac_shared_multiply_3_rhs : signed( 20 downto 0 );
	begin

		case iir_filter_process_state is

			when sm0_1 =>
				ac_shared_multiply_3_lhs := ac_scalar_coeff_a_0;

			when sm0_0 =>
				ac_shared_multiply_3_lhs := ac_scalar_coeff_a_1;

			when others =>
				ac_shared_multiply_3_lhs := "0000000000";

		end case; --iir_filter_process_state


		case iir_filter_process_state is

			when sm0_1 =>
				ac_shared_multiply_3_rhs := ac_scalar_y_n_0;

			when sm0_0 =>
				ac_shared_multiply_3_rhs := ac_scalar_y_n_1;

			when others =>
				ac_shared_multiply_3_rhs := "00000000000000000000";

		end case; --iir_filter_process_state

		ac_shared_multiply_3_result <= (ac_shared_multiply_3_lhs *
ac_shared_multiply_3_rhs);

	end process ac_shared_multiply_3_process_combinational_1;


	ac_shared_adder_1_process_combinational_0:
	process ( ac_scalar_x_n_0_reg, ac_scalar_x_n_2, iir_filter_process_state, sample_internal,
mtimes_001_accum_2 ) is
		variable ac_scalar_x_n_0 : signed( 8 downto 0 );
		variable ac_scalar_x_n_1 : signed( 8 downto 0 );
		variable ac_shared_adder_1_lhs : signed( 20 downto 0 );
		variable ac_shared_adder_1_rhs : signed( 20 downto 0 );
		variable ac_tmp_3 : signed( 16 downto 0 );
		variable ac_tmp_4 : signed( 16 downto 0 );
		variable ac_tmp_5 : signed( 10 downto 0 );
		variable mtimes_001_accum_3 : signed( 20 downto 0 );
		variable mtimes_001_prod1_1 : signed( 18 downto 0 );
		variable mtimes_001_prod1_2 : signed( 18 downto 0 );
```

```
		variable mtimes_001_prod1_3 : signed( 18 downto 0 );
	begin
		mtimes_001_prod1_3 := "0000000000000000000";
		mtimes_001_prod1_2 := "0000000000000000000";
		mtimes_001_prod1_1 := "0000000000000000000";
		mtimes_001_accum_3 := "00000000000000000000";
		ac_tmp_5 := "00000000000";
		ac_tmp_4 := "00000000000000000";
		ac_tmp_3 := "00000000000000000";
		ac_scalar_x_n_1 := "000000000";
		ac_scalar_x_n_0 := "000000000";
		ac_scalar_x_n_1 := ac_scalar_x_n_0_reg;
		ac_tmp_4 := (ac_scalar_x_n_1 & "00000000");
		ac_tmp_3 := SHIFT_RIGHT( ac_tmp_4, 1 );
		mtimes_001_prod1_2 := (ac_tmp_3(16) & (ac_tmp_3(16) & ac_tmp_3));
		ac_tmp_5 := ("000000000" - (ac_scalar_x_n_2(8) & (ac_scalar_x_n_2(8) &
ac_scalar_x_n_2)));
		mtimes_001_prod1_1 := (ac_tmp_5 & "00000000");

		case iir_filter_process_state is

			when sm0_1 =>
				ac_shared_adder_1_lhs := (mtimes_001_prod1_2(18) &
(mtimes_001_prod1_2(18) & mtimes_001_prod1_2));

			when sm0_0 =>
				ac_shared_adder_1_lhs := (mtimes_001_prod1_1(18) &
(mtimes_001_prod1_1(18) & mtimes_001_prod1_1));

			when others =>
				ac_shared_adder_1_lhs := "00000000000000000000";

		end case; --iir_filter_process_state

		ac_scalar_x_n_0 := sample_internal;
		mtimes_001_prod1_3 := (ac_scalar_coeff_b_0 * ac_scalar_x_n_0);
		mtimes_001_accum_3 := (mtimes_001_prod1_3(18) & (mtimes_001_prod1_3(18) &
mtimes_001_prod1_3));

		case iir_filter_process_state is

			when sm0_1 =>
				ac_shared_adder_1_rhs := mtimes_001_accum_3;

			when sm0_0 =>
				ac_shared_adder_1_rhs := mtimes_001_accum_2;

			when others =>
				ac_shared_adder_1_rhs := "00000000000000000000";

		end case; --iir_filter_process_state

		ac_shared_adder_1_result <= (ac_shared_adder_1_lhs + ac_shared_adder_1_rhs);

	end process ac_shared_adder_1_process_combinational_0;


	iir_filter_process_clocked:
	process ( Clock ) is
		variable ac_tmp_18 : signed( 8 downto 0 );
```

```
		variable ac_tmp_19 : signed( 8 downto 0 );
		variable ac_tmp_20 : signed( 30 downto 0 );
		variable ac_tmp_21 : signed( 27 downto 0 );
		variable ac_tmp_22 : signed( 30 downto 0 );
		variable ac_tmp_23 : signed( 19 downto 0 );
		variable ac_tmp_26 : signed( 20 downto 0 );
		variable ac_tmp_27 : signed( 32 downto 0 );
		variable ac_tmp_28 : signed( 22 downto 0 );
		variable mtimes_002_accum_1_0 : signed( 32 downto 0 );
		variable mtimes_002_prod1_2_0 : signed( 30 downto 0 );
		variable mtimes_002_prod1_3_0 : signed( 30 downto 0 );
		variable ac_tmp_24 : signed( 30 downto 0 );
	begin
		if (rising_edge(Clock)) then
			if ((Reset = '1')) then
				signal_ac_OutputAvail <= ('0');
				ac_is_busy <= false;
				ac_scalar_x_n_0_reg <= "000000000";
				ac_scalar_x_n_1_reg <= "000000000";
				ac_scalar_x_n_2 <= "000000000";
				ac_scalar_y_n_0 <= "00000000000000000000";
				ac_scalar_y_n_1 <= "00000000000000000000";
				mtimes_001_accum_2 <= "00000000000000000000";
				mtimes_002_accum_3 <= "00000000000000000000000000000000";
				signal_y <= "00000000000000000000000000000000";
				iir_filter_process_state <= sm0_1;
			else

				case iir_filter_process_state is

					when sm0_1 =>
						if (proceed_var) then
							mtimes_002_prod1_3_0 :=
ac_shared_multiply_3_result;
							ac_tmp_19 := ac_scalar_x_n_0_reg;
							ac_tmp_18 := sample_internal;
							ac_scalar_x_n_2 <= ac_scalar_x_n_1_reg;
							mtimes_002_accum_3 <=
(mtimes_002_prod1_3_0(30) & (mtimes_002_prod1_3_0(30) & mtimes_002_prod1_3_0));
							mtimes_001_accum_2 <=
ac_shared_adder_1_result;
							iir_filter_process_state <= sm0_0;
							ac_scalar_x_n_1_reg <= ac_tmp_19;
							ac_scalar_x_n_0_reg <= ac_tmp_18;
							signal_ac_OutputAvail <= '0';
							ac_is_busy <= true;
						else
							signal_ac_OutputAvail <= '0';
						end if;

					when sm0_0 =>
						ac_tmp_26 := ac_shared_adder_1_result;
						ac_tmp_23 := ac_tmp_26( 19 downto 0 );
						mtimes_002_prod1_2_0 := ac_shared_multiply_3_result;
						mtimes_002_accum_1_0 := ((mtimes_002_prod1_2_0(30)
& (mtimes_002_prod1_2_0(30) & mtimes_002_prod1_2_0)) + mtimes_002_accum_3);
						ac_tmp_22 := ((ac_tmp_23(19) & ac_tmp_23) &
"0000000000");
						ac_tmp_27 := mtimes_002_accum_1_0;
						ac_tmp_21 := ac_tmp_27( 27 downto 0 );
```

```
                                                ac_tmp_20 := (ac_tmp_22 - (ac_tmp_21(27) &
(ac_tmp_21(27) & (ac_tmp_21(27) & ac_tmp_21))));
                                                signal_y <= ac_tmp_20;
                                                ac_scalar_y_n_1 <= ac_scalar_y_n_0;
                                                ac_tmp_24 := ac_tmp_20;
                                                ac_tmp_28 := ac_tmp_24( 30 downto 8 );
                                                ac_scalar_y_n_0 <= ac_tmp_28( 20 downto 0 );
                                                iir_filter_process_state <= sm0_1;
                                                signal_ac_OutputAvail <= '1';
                                                ac_is_busy <= false;

                                        end case; --iir_filter_process_state

                                end if;
                        end if;

        end process iir_filter_process_clocked;


        ac_register_inputs:
        process ( Clock ) is
        begin
                if (rising_edge(Clock)) then
                        if ((Reset = '1')) then
                                sample_internal <= (others => '0');
                                ac_InputAvail_internal <= ('0');
                        else
                                sample_internal <= sample;
                                ac_InputAvail_internal <= ac_InputAvail;
                        end if;
                end if;

        end process ac_register_inputs;

end RTL;
```

References

1- Multi-Carrier Digital Communications: Theory and Applications of OFDM Ahmad R. S. Bahai and Burton R. Saltzberg

2- Principles of Digital Transmission: With Wireless Applications Sergio Benedetto and Ezio Biglieri

3- Simulation of Communication Systems, 2nd Edition: Methodology, Modeling, and Techniques Michel C. Jeruchim, Philip Balaban, and K. Sam Shanmugan

4- Rabiner, L.R., Gold B. *Theory and Application of Digital Signal Processing*. Englewood Cliffs, NJ: Prentice Hall, 1975.

5- Adaptive Techniques for Multiuser OFDM Thesis submitted by Eric Phillip LAWREY BE in December 2001

6- "Digital Video Broadcasting", ETSI, Online: http://www.etsi.org/broadcast/dvb.htm, October 2000

7- ETSI TS 101 475 V1.1.1, "Broadband Radio Access Networks (BRAN); HIPERLAN Type 2; Physical (PHY) layer", April 2000, Online: http://www.etsi.org.

8- Programmable Logic Design Quick Start Hand Book By Karen Parnell & Nick Mehta.

9- **Advanced FPGA Design Architecture**, Implementation, and Optimization by Steve Kilts

10- A Study of OFDM for timing Based on IEEE 802. 11a, graduate student: Wang Shun instructs professor: Dr. Wu Zhongshi Republic of China nine ten seven in July

11- FPGA IMPLEMENTATION OF IEEE 802.11 b/g MAC LAYER RECEIVER PART Graduate student: Guo Junhong instructs professor: Cai Mujin, China on July 10, 2007