UNIVERSITY OF TURKU

Design and Implementation of HD Wireless Video Transmission System Based on Millimeter Wave

Medical Analytics and Health IoT
Master's Degree Programme in Digital Health and Life Sciences
Department of Computing, Faculty of Technology
Master of Science in Technology Thesis

Author:
Weijun Gao

Supervisors:
Tero Koivisto
Arman Anzanpour

May 2023

Master of Science in Technology Thesis
Department of Computing, Faculty of Technology
University of Turku

Subject: Medical Analytics and Health IoT
Programme: Master's Degree Programme in Digital Health and Life Sciences
Author: Weijun Gao
Title: Design and Implementation of HD Wireless Video Transmission System Based on Millimeter Wave
Number of pages: 101 pages
Date: May 2023

Abstract.

With the improvement of optical fiber communication network construction and the improvement of camera technology, the video that the terminal can receive becomes clearer, with resolution up to 4K. Although optical fiber communication has high bandwidth and fast transmission speed, it is not the best solution for indoor short-distance video transmission in terms of cost, laying difficulty and speed.

In this context, this thesis proposes to design and implement a multi-channel wireless HD video transmission system with high transmission performance by using the 60GHz millimeter wave technology, aiming to improve the bandwidth from optical nodes to wireless terminals and improve the quality of video transmission. This thesis mainly covers the following parts:

(1) This thesis implements wireless video transmission algorithm, which is divided into wireless transmission algorithm and video transmission algorithm, such as 64QAM modulation and demodulation algorithm, H.264 video algorithm and YUV420P algorithm.

(2) This thesis designs the hardware of wireless HD video transmission system, including network processing unit (NPU) and millimeter wave module. Millimeter wave module uses RWM6050 baseband chip and TRX-BF01 rf chip. This thesis will design the corresponding hardware circuit based on the above chip, such as 10Gb/s network port, PCIE.

(3) This thesis realizes the software design of wireless HD video transmission system, selects FFmpeg and Nginx to build the sending platform of video transmission system on NPU, and realizes video multiplex transmission with Docker. On the receiving platform of video transmission, FFmpeg and Qt are selected to realize video decoding, and OpenGL is combined to realize video playback.

(4) Finally, the thesis completed the wireless HD video transmission system test, including pressure test, Web test and application scenario test. It has been verified that its HD video wireless transmission system can transmit HD VR video with three-channel bit rate of 1.2GB /s, and its rate can reach up to 3.7GB /s, which meets the research goal.

Keywords: Millimeter-waves, Video transmission system, PCIE, Linux

List of Abbreviations

AAC	Advanced Audio Coding
API	Application Programming Interface
APP	Application
ARM	Advanced RISC Machines
BMAN	Buffer Management
CACBA	Context adaptive binary arithmetic coding
CAD	Computer Aided Design
CPU	Central processing unit
CISC	Complex Instruction Set Computer
DCT	Discrete Cosine Transform
DDR4	Double Data Rate SDRAM 4th
DPAA	Data Path Acceleration Architecture
DTB	Device tree blob
EEPROM	Electrically Erasable Programmable read only memory
EMMC	Embedded Multi Media Card
ESD	Electro-Static discharge
FFmpeg	Fast Forward Mpeg
FLV	Flash video
FMAN	Frame Management
GCC	GNU Compiler Collection
GOP	Group of pictures
GPIO	General-purpose input/output

GPU	Graphics processing unit
HTTP	Hyper Text Transfer Protocol
IDR	Instantaneous decoding refresh
IIC	Inter-Integrated Circuit
I/O	Input and Output
IPC	Inter-Process Communication
JTAG	Joint Test Action Group
JVT	Joint Video Team
LTE	Long Term Evolution
LXC	Linux container
Mbps	Mega bit per second
MMW	Millimeter wave
MPEG	Moving Picture Experts Group
MP4	Moving Picture Experts Group 4
Nginx	Engine x
NPU	Neural-network Processing Unit
NXP	Next Experience
OPTEE	Open-source Portable Trusted Execution Environment OS
OpenGL	Open Graphics Library
OS	Operation system
PCB	Printed Circuit Board
PCIE	Peripheral component interconnect express
PCM	Pulse Code Modulation

PID	Proportion Integral Differential
PS	Parametric Stereo
QMAN	Query Management
QSGMII	Quad-Serial Gigabit Media Independent Interface
RCW	Reset configuration word
RGB	RGB color mode
RGMII	Reduced Gigabit Media Independent Interface
RISC	Reduced Instruction Set Computing
RTMP	Real Time Messaging Protocol
RTSP	Real Time Streaming Protocol
SATA	Serial Advanced Technology Attachment
SBR	Spectral Band Replication
SDHC	High Capacity SD Memory Card
SDL	Simple Direct-Media Layer
SDRAM	Synchronous dynamic random-access memory
SGMII	Serial Gigabit Media Independent Interface
SWF	Shock wave flash
TCP	Transmission Control Protocol
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus
VR	Virtual Reality
VCEG	Video Coding of Expert Group on
VFS	Virtual File System

VXL	Vision-something- Libraries
WIFI	Wireless Fidelity
YUV	Luma- Chrominance
5G	5th Generation Mobile Communication Technology

Table of contents

1 Introduction 1

1.1 Background 1

1.2 Related work 6

1.2.1 Wireless transmission algorithm 6

1.2.2 Video transmission algorithm 10

1.2.3 Transfer protocol 16

1.3 Thesis Work and Contributions 18

1.4 Thesis Structure 19

2 System Design 21

2.1 Hardware platform 21

2.1.1 Baseband & RF module 21

2.1.2 NPU platform 22

2.2 System Architecture 24

2.3 System Design Criteria 24

2.3.1 Publishing platform 25

2.3.2 Pulling platform 25

2.4 Design detail 27

3 Hardware Prototype 28

3.1 Hardware Architecture 28

3.2 Core Board 29

3.3 Mother Board Prototype and Relation Modules 30

3.3.1 Mother Board 30

3.3.2 AQR107 31

3.3.3 PCIE 33

3.3.4 DPAA 35

3.4 MMW Modules 38

3.4.1 Baseband IC 39

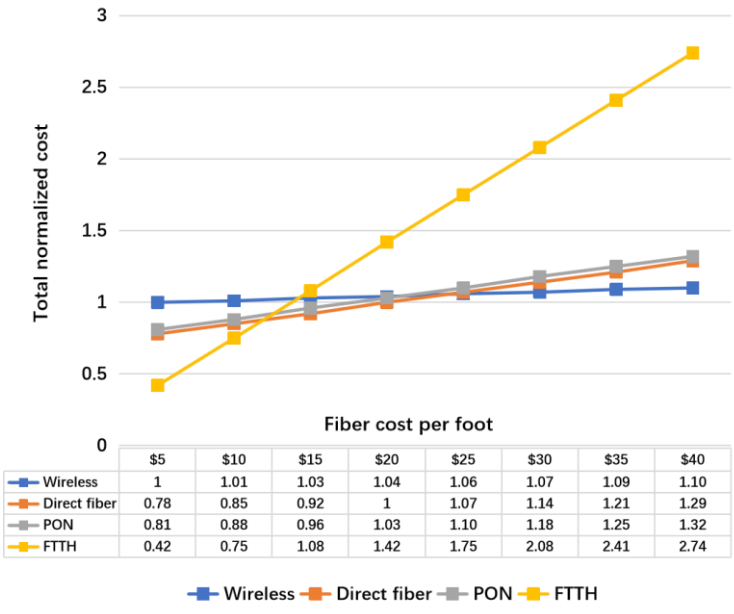
3.4.2 Radio frequency module 39
3.5 Integration of Transmission System 41
4 Software System of Wireless transmission 42
4.1 Preparation of the system 42
4.1.1 U-boot 43
4.1.2 Linux system 46
4.2 Driver and APP 52
4.2.1 Logical of Flow-pushing 52
4.2.2 Video Signal 53
4.2.3 Flow-pushing 54
4.3 Buffering and forwarding 56
4.3.1 Logical of Server running in the system 56
4.3.2 Nginx-RTMP of Server 56
4.3.3 Multiplexing server 58
4.4 MMW driver 60
4.4.1 Logical of Linux driver 61
4.4.2 Mount and Run of MMW 62
4.5 Flow-pulling and rendering 66
4.5.1 Logical of Flow-pulling 66
4.5.2 FFmpeg of Flow-pulling 68
4.5.3 Qt of UI and Audio 69
4.5.4 OpenGL of Rendering 71
4.6 Summarry 73
5 Tests and Results 74
5.1 Channel influence on the transmission system 74
5.2 Environment influence on the transmission system 75
5.3 Pressure Test of Transmission System 77
5.3.1 Speed test of RJ45 79

5.3.2 Speed test of PCIE 79
5.3.3 Speed test of Nginx 81
5.4 HD Video Testing in Transmission System 81
5.4.1 Test result in 1080p video 81
5.4.2 Test result in 2k video 82
5.4.3 Test result in 8k video 83
5.4.4 Test result in 8k 360°VR video 83
5.5 Summary 86
6 Conclusion and Future Research Directions 87
6.1 Conclusion 87
6.2 Outlook 88
6.2.1 Insufficiencies 88
6.2.2 Future Work 89
Reference 91

1 Introduction

1.1 Background

In the era of the Internet of all things, with the variety of mobile terminals, such as smart phones, smart tablets and intelligent cockpit systems, the access of many wireless devices and the corresponding massive data show that the social requirements for network has stepped from the original 100 Mb/s to 1000 Mb/s. With the continuous improvement of optical fiber in communication network construction, in the core backbone network, optical fiber has a transmission rate of up to 400Gb/s [1], but different access technologies from optical nodes to terminals may lead to the reduction of communication quality. By 2021, the network speed requirements of terminal devices are mostly between 350Mb/s and 2Gb/s [2]. The ideal and most appropriate method is to use optical fiber for data transmission and realize optical fiber to the home. According to the research on optical fiber, as shown in Figure 1-1, the laying of optical fiber is complicated and the cost is high due to its physical characteristics. Especially in some areas where the laying of optical fiber is complicated, the construction cost is high and the construction is difficult due to unreasonable planning and construction. And the maintenance costs can't be ignored.



	$5	$10	$15	$20	$25	$30	$35	$40
Wireless	1	1.01	1.03	1.04	1.06	1.07	1.09	1.10
Direct fiber	0.78	0.85	0.92	1	1.07	1.14	1.21	1.29
PON	0.81	0.88	0.96	1.03	1.10	1.18	1.25	1.32
FTTH	0.42	0.75	1.08	1.42	1.75	2.08	2.41	2.74

Figure 1-1: Total normalized cost and fiber cost [2]

Therefore, for such a complex area, wireless access technology has the characteristics

of simple laying and high transmission rate. The mainstream wireless access technologies include LTE signal and millimeter wave signal.

Ikram, Muhammad, Nghia Nguyen-Trong et al. compared the performance of 5G LTE signal and millimeter wave [5]. They believed that the existing wireless communication technologies below 6 GHz, namely 4G and WIFI, were congested and only allowed limited bandwidth. Therefore, only limited data rates can be supported [6]. There is no doubt that under the current wave of 5G, 10Gb/s will be the theoretical maximum speed of 5G. In theory, to achieve a higher transmission rate and bandwidth, its electromagnetic wave should be located at a higher frequency, to open a new working frequency and space. In the study of Lecci, Mattia et al., it was found that millimeter wave has a large amount of undeveloped bandwidth, which can improve the data rate provided to end users [7], thus making it possible to meet the requirements of ultra-high peak throughput (20 Gb/s) and average user experience rate (50-100 Mb/s) of the fifth generation (5G) [8].

Table 1-1: Design details of wireless transmission system [8]

Advanced mobile board-band	2017	2022	Towards 2025
80 percent of sites	150Mbps	350Mbps	600Mbps
20 percent of sites	300Mbps	1-2Gbps	3-5Gbps
Few percent of sites	1Gbps	3-10Gbps	10-20Gbps

Millimeter-wave refers to electromagnetic waves operating at a frequency of 30-300GHz [9] with a wavelength of 1-10mm. Its frequency reaches the level of GHz, which is a transitional electromagnetic wave between microwave and infrared wave. To some extent, it has the advantages of two kinds of things:

Compared with microwave, millimeter wave components are smaller and easier to systematize; Its operating frequency range is 26.5-300GHz [10], up to more than 200 operating frequencies, which is 9 times higher than the sum of all bandwidths from direct current to microwave. Even the effective bandwidth under atmospheric window is 135GHz [11], which is 5 times of that under microwave.

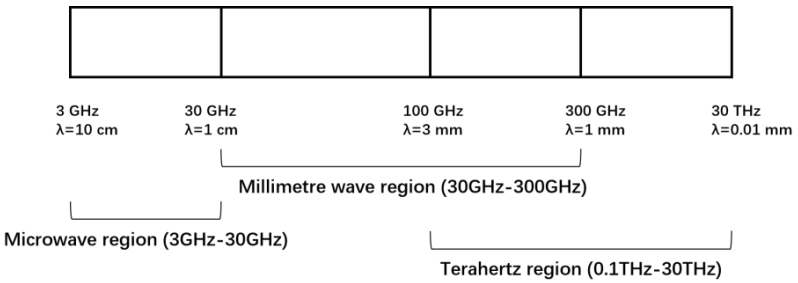


Figure 1-2: Millimeter wave bandwidth distribution [10]

Millimeter wave has a narrower beam, which makes the antenna narrower. Generally, the width is 17° at 10GHz frequency, but only 1.7° at 100GHz frequency, which is enough to distinguish smaller targets [12] or observe target details more clearly. Compared with light wave, millimeter wave not only has the characteristics of atmospheric window [13], but also has several attenuation peaks to realize safe and confidential point-to-point communication due to its minimal attenuation through resonant absorption of gas molecules during propagation.

In addition, the transmission of millimeter-wave is less affected by sandstorms [14]. Even in the bad weather of sand-flying, due to its high frequency, it can penetrate the dust and smoke in the air to achieve all-weather working characteristics in dry areas. Even at the higher levels of scattering produced by explosions or foil, there is only a short period of fading.

Millimeter wave has certain microwave characteristics, such as the detection ability [15]. It can suppress multipath effect and all kinds of chaotic echoes by relying on its broad-spectrum ability of bandwidth. When detecting each other, it can also have enough frequency selection without interfering with each other. More importantly, millimeter wave can obtain a large doppler frequency shift when the target moves in radial direction and has strong enough recognition and detection ability of low speed objects and vibrating objects.

Because of the above broad-spectrum capability and anti-interference, millimeter wave communication basically has no interference source. Several bits of clean electromagnetic spectrum make millimeter wave channel very stable and reliable, and the bit error rate can reach more than 10 magnitude, even no weaker than optical cable transmission.

However, millimeter-wave is not perfect either: in rainy weather, due to the attenuation of millimeter-wave signal [16], the instantaneous intensity of rainfall, as well as the length and shape of raindrops are related. The larger the raindrop, the more severe the attenuation. To solve this problem, it is necessary to design in advance to maintain a

certain level attenuation margin.

Sometimes it's not a bad thing to have a big absorption decay rate on rainy days. This phenomenon will make the point-to-point communication distance very short, which can greatly improve the enemy to steal and interference communication difficulty; And because attenuation peak can be selected for communication, the possibility of MMW being intercepted is greatly reduced in the case of very narrow beam, which ensures the confidentiality of military communication.

According to the research of Chen S, Zhao J et al., they believe that with the progress of science and technology, people's requirements for Internet speed will gradually increase, and more and more new technologies are needed to meet these requirements [17].

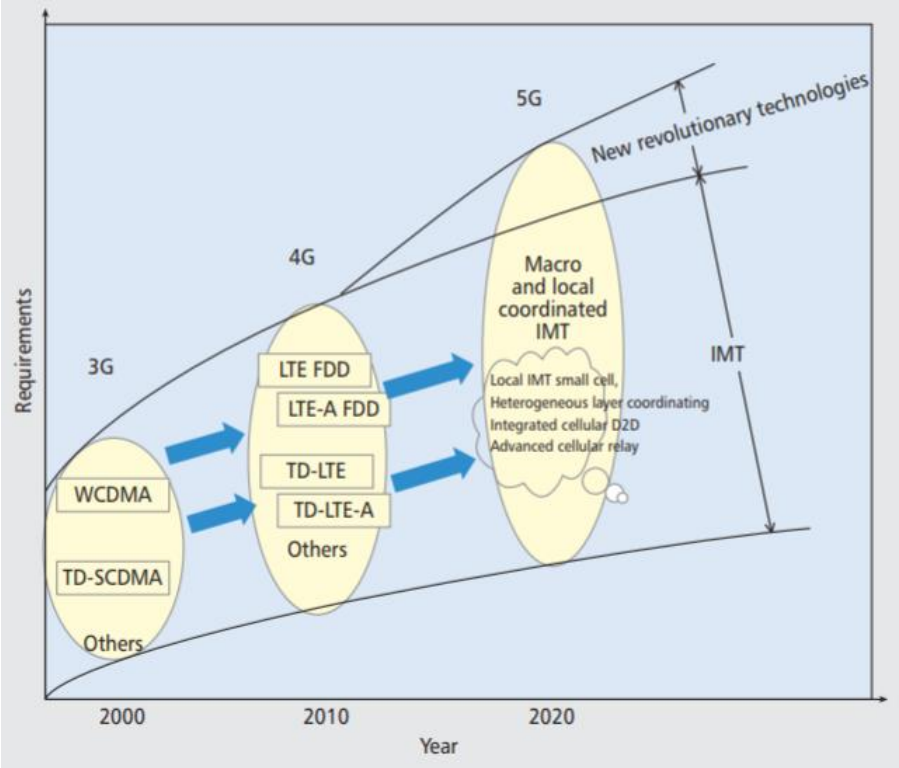


Figure 1-3: Society's changing demand for wireless bandwidth technology [17]

Coaxial cable will gradually decline, and then replaced by microwave, optical fiber, 65% of broadcasting stations will access millimeter wave, millimeter wave will be widely concerned and applied in the future.

In terms of research, Lu Ou, Shaolin Liao, Zheng Qin et al., in 2020, they have realized simple image transmission through millimeter wave. They sampled and compressed images through Hadamard-matrix and sent them, and then the receiver used Fourier spectrum method to reconstruct the images [18]. Verifies the bit error rate and transmission rate of the picture in the transmission. All reach their theoretical speeds.

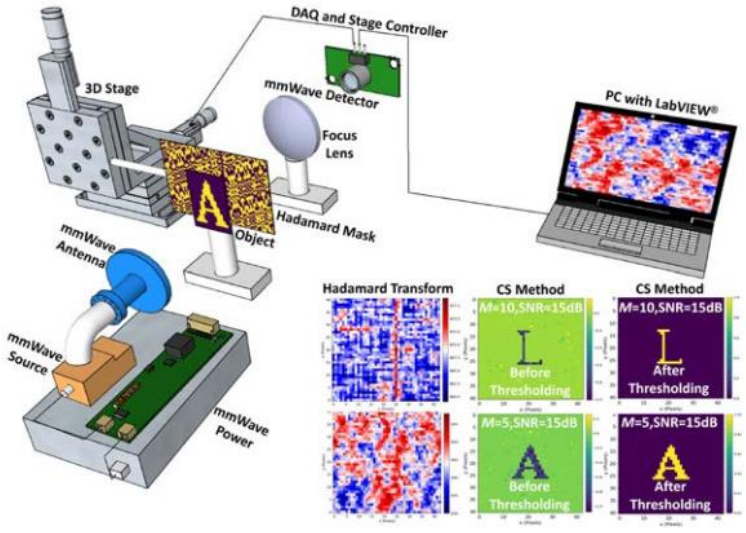


Figure 1-4: Image transmission experiment based on millimeter wave [18]

In a study, D.Ha et al [19]. used millimeter wave technology in the transmitter of FWA system as the wireless access point of the user's home or commercial building to solve the problem of wireless connection between optical nodes and terminal devices. The customer Premise Equipment (CPE) is installed outside the house to receive millimeter-wave signals from the base station. The following Figure shows the specific path.

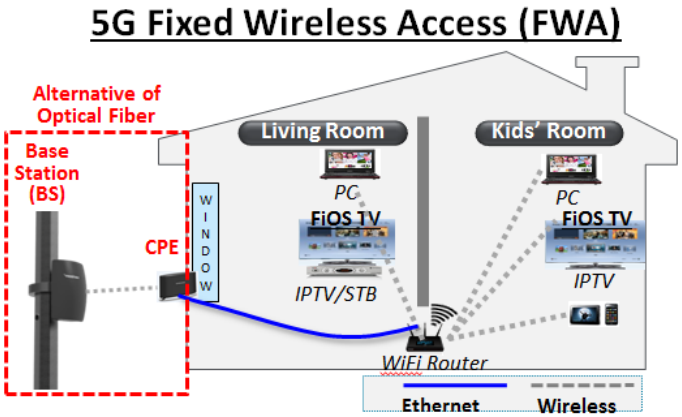


Figure 1-5: FWA application structure [19]

Such researches show the potential of millimeter-wave applications in everyday life, as well as its ability to meet people's demands for high definition image speed and delay. Due to the resolution of problems such as high transmission loss and small coverage of millimeter wave, in the 5G era, users' demand for video transmission, video streaming and video game live broadcasting is increasing, and the requirements for communication bandwidth and speed are getting higher and higher. The huge bandwidth resources of millimeter wave make many new concepts and applications gradually realized, such as millimeter wave radar under autopilot, and millimeter wave communication applied to UAV (Unmanned Aerial Vehicle), etc.

(a) (b)

Figure 1-6: MMW applications include (a) Millimeter wave radar [20] and (b) Millimeter wave VR video [21]

In short, due to MMW's stable transmission and high bit error rate. In the indoor, the interference of rain water is eliminated, and its transmission performance can be maximized. It can not only reduce the multifarious indoor optical fiber network, but also reduce the unaesthetic of indoor network cable winding. For VR users, high-definition VR images can be seen without transmission lines, making users more immersed and greatly improving their VR experience. Therefore, this thesis hopes to realize wireless HD video transmission through millimeter wave.

1.2 Related work

This part mainly introduces the relevant transmission scheme of the wireless HD video transmission system, which includes three parts, namely the wireless transmission algorithm related to the wireless transmission link; Algorithms related to wireless HD video transmission, including video codec and compression algorithms; Network transport protocol.

1.2.1 Wireless transmission algorithm

As a wireless transmission link, it is necessary to select appropriate modulation and demodulation mode and data coding scheme, to achieve lower bit error rate and higher transmission efficiency under the corresponding power consumption. Secondly, to compensate for the transmission distance of millimeter-wave, antenna array can be used to realize MIMO to reduce transmission loss.

64QAM: Quadrature Amplitude Modulation, which has high modulation efficiency and high frequency band utilization rate. It also uses the amplitude and phase of the carrier to transmit bits of information, among which different phase and amplitude

represent different coding symbols. Under the same conditions, 64QAM can accommodate more constellation points for higher bandwidth utilization. According to preliminary experiments, 64QAM compares with other modulation and demodulation methods as table 1-2:

Table 1-2: Rate comparison of modulation and demodulation modes

MCS Index	Modulation	Achievable Rates		
		Full	Half	Quarter
0	DBPSK	28	14	7
1	PI/2 BPSK	385	193	96
2	PI/2 BPSK	770	385	193
3	PI/2 BPSK	963	481	241
4	PI/2 BPSK	1155	578	289
5	PI/2 BPSK	1251	626	313
6	PI/2 QPSK	1540	770	385
7	PI/2 QPSK	1925	963	481
8	PI/2 QPSK	2310	1155	578
9	PI/2 QPSK	2503	1251	626
10	PI/2 16QAM	3080	1540	770
11	PI/2 16QAM	3850	1925	963
12	PI/2 16QAM	-	2310	1155
13	PI/2 16QAM		2502	1251
14	PI/2 64QAM		2888	1444
15	PI/2 64QAM		3465	1733
16	PI/2 64QAM		3754	1877

Compared with QPSK and other modulation and demodulation schemes, its 64QAM can also achieve higher transmission efficiency at half state. Therefore, this thesis mainly uses 64QAM to modulate and demodulate the signal.

MSDU & MPDU: To reduce channel conflict, this thesis adopts frame

aggregation technology, which involves MSDU&MPDU technologies. MSDU technology is to aggregate multiple MSDU buffers and then send them, to reduce the additional information in the MAC head, increase the proportion of effective data, and

thus improve the transmission rate. Its structure is shown in the Figure below:

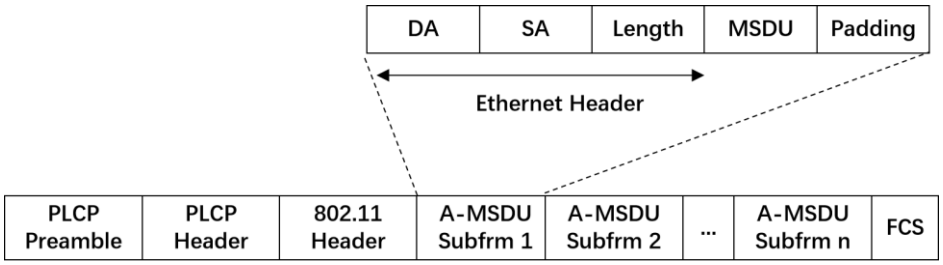


Figure 1-7: MSDU protocol [26]

MPDU also aggregates channel conflict, keeping only the PHY header and reducing the additional information on the PHY layer to reduce the load of the protocol and improve the network throughput to some extent. Its structure is as Figure 1-8：

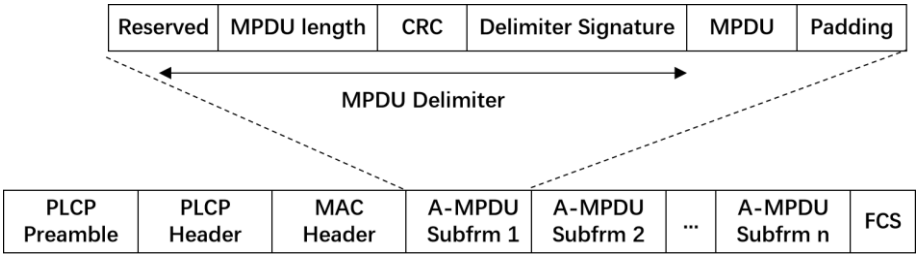


Figure 1-8: MPDU protocol [26]

Block confirmation technology: Block confirmation technology is also used to reduce channel conflicts. Its mechanism is divided into three steps: First, the block confirmation protocol needs to be established through ADDA mechanism, and the sender needs to send certain QoS data information. After receiving all data information and Block-Ack-Req message, the sender can reply to the sender in one time. Finally terminate the block confirmation protocol. The mechanism flow is as Figure 1-9:

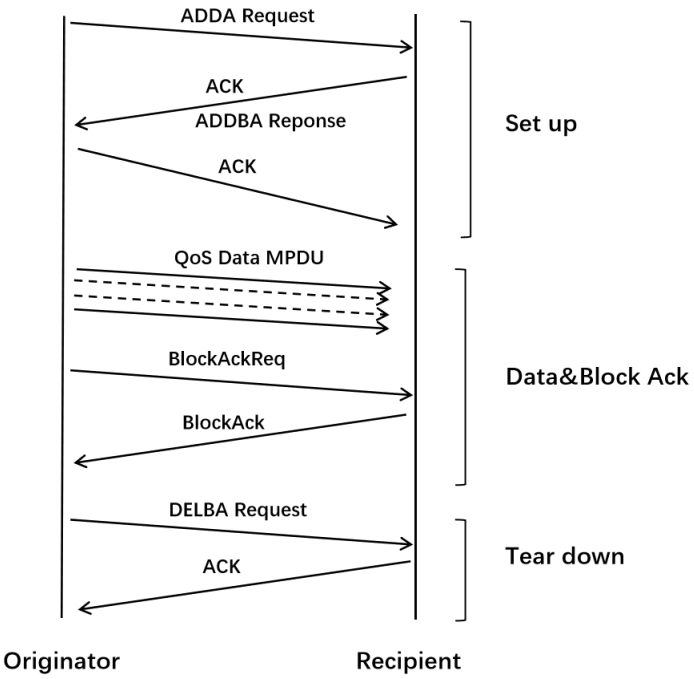
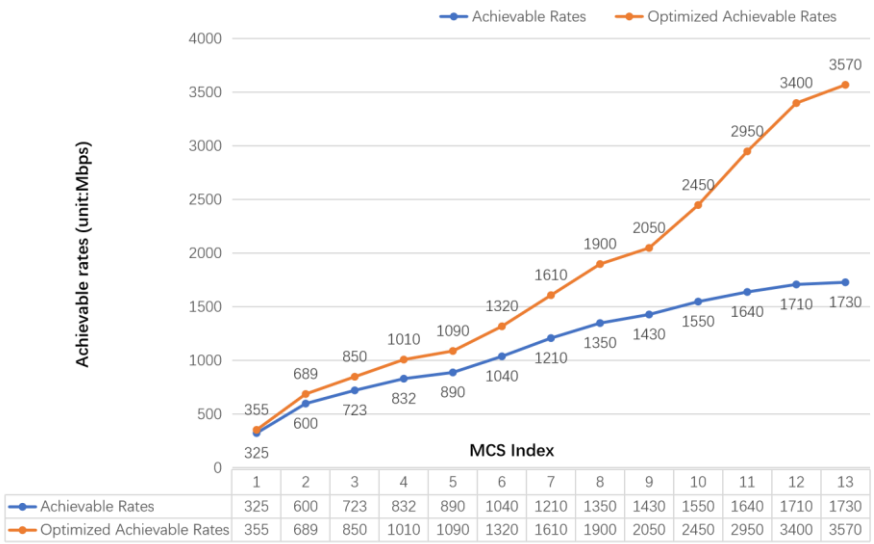


Figure 1-9: Block confirmation mechanism flow [26]

According to the experiment, frame aggregation technology and block confirmation mechanism can effectively improve the video transmission rate, and the experimental results are shown as Figure 1-10. Obviously, with the increase of the number of channels, the optimized transmission efficiency is significantly improved. Compared with the 1.73Gb/s transmission rate, the optimized transmission effect can reach 3.57Gb/s.



	1	2	3	4	5	6	7	8	9	10	11	12	13
Achievable Rates	325	600	723	832	890	1040	1210	1350	1430	1550	1640	1710	1730
Optimized Achievable Rates	355	689	850	1010	1090	1320	1610	1900	2050	2450	2950	3400	3570

Figure 1-10: Optimization experiment comparison

1.2.2 Video transmission algorithm

H.264： It is a video encoding format. It is a highly compressed digital video encoder standard jointly proposed by THE JOINT Video Group (JVT) formed by THE ITU-T Video Coding Expert group (VCEG) and the ISO/IEC Dynamic image Expert group (MEPG).

This code mainly consists of seven parts, namely, block partitioning, vector estimation, mode estimation, DCT and quantization, CBABC, etc., and the logical relationship between them is as Figure 1-11 [24]:

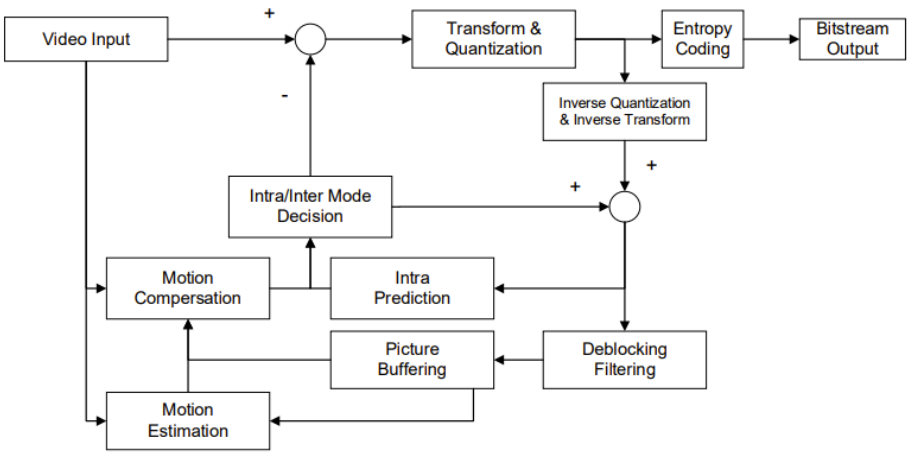


Figure 1-11: H.264 coding process

Research shows that H.264 has better video resolution than MPEG, H.263 and other encoding formats, which can be seen from the following PSNR, H.264 has nearly 1-2dB improvement over the traditional encoding algorithm.

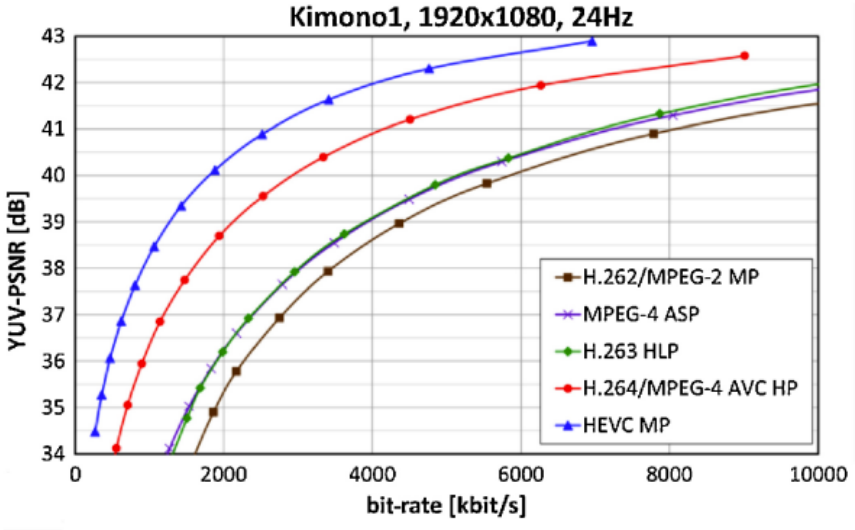


Figure 1-12: Video coding protocol PSNR comparison [24]

Although H.265 has a higher PSNR as shown in the Figure, its corresponding energy consumption is also higher. According to the study, the energy consumption of data transmission using H.265 is 5-8 times that of H.264. This can take up a lot of device resources, with the energy consumption are shown as Figure 1-3:

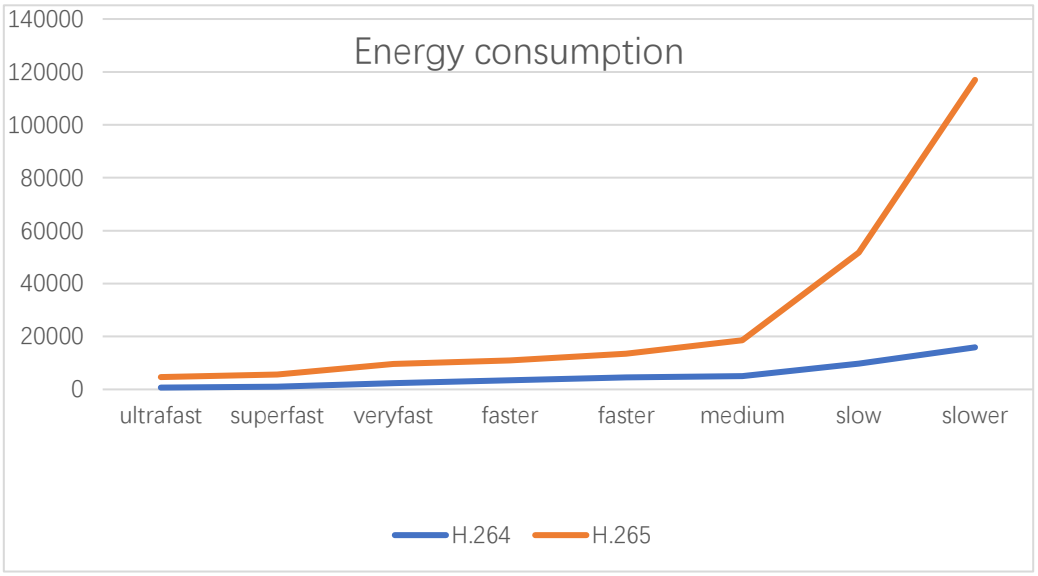


Figure 1-3: Energy comparison of coding protocols

In addition, according to the experiment, H.265 and H.264 encoding methods are used in real-time data transmission in this thesis, and there is a great difference in delay. H.265 will cause a great delay in data transmission and affect the video quality, and the results are shown as Figure 1-4

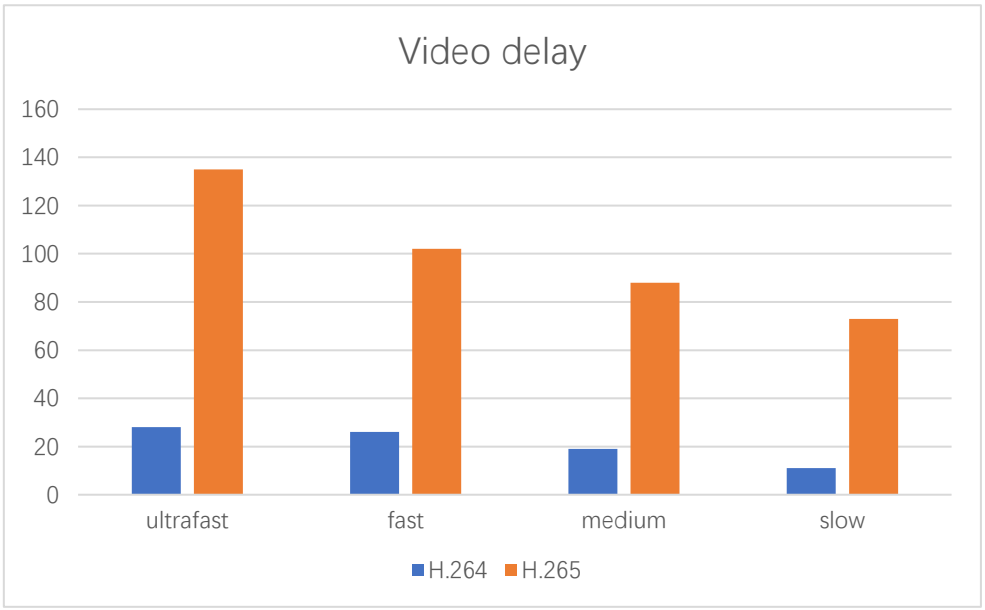


Figure 1-4: Delay comparison of coding protocols

In general, H.264 first takes the received image and uses a 16x16 area as a macro block, then calculates the pixel value of the divided macro block, and so on until all pixels are processed. Then, according to the divided macro block, if the pixel of this macro block is flat enough, the molecular block will not be divided, while if the 16x16 macro block is too complex, the sub-block will be divided [25]:

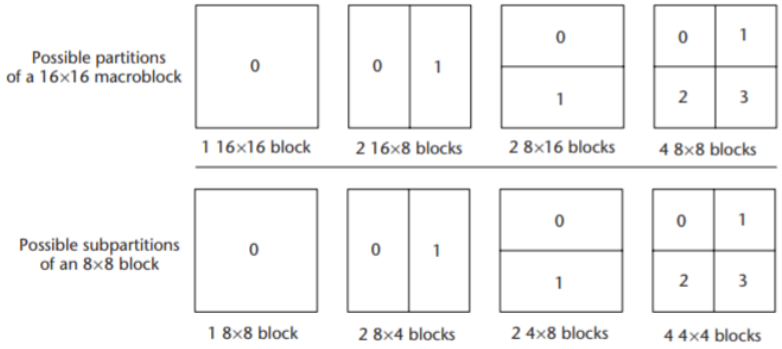


Figure 1-13: Schematic diagram of macroblock partition

Compressed in this way, H.264 can achieve higher compression rates through sub-blocks than MPEG2:

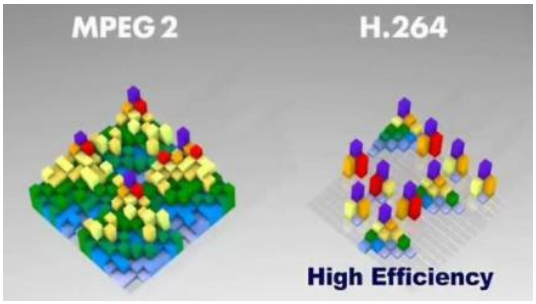


Figure 1-14: Comparison of compression efficiency between H.264 and MPEG2

After the compression of a certain frame is completed, frames can be grouped. Due to the video data transmission, in many cases, the pixels in these pictures change repeatedly with time, that is, there is a large amount of data redundancy in both time and space [26]:

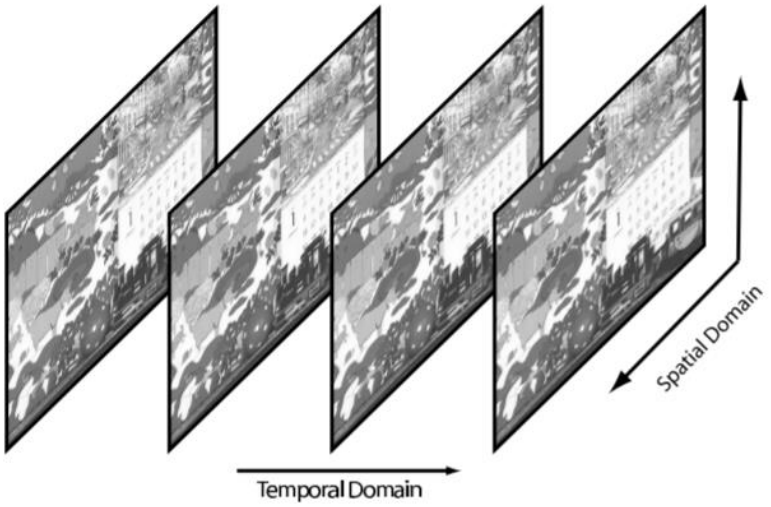


Figure 1-15: Time and space redundancy of frames

For the temporal domain: If the camera captures 30 frames per second, the 30 frames of data are mostly correlated, even the picture doesn't change much. If the camera captures more than 30 frames per second, or even 60 frames per second, then the hundreds of frames of data are closely correlated. For these closely related frames, many times we only need to save a complete frame of data. Other frames can be predicted by certain algorithms or simply write simple data on other frames for corresponding superposition to calculate another frame.

For the spatial domain: This is generally referred to as intra-frame compression. The algorithm mainly provides 9 kinds of macro-block change prediction modes, and then selects the most similar macro-block prediction mode and marks each macro-block with the mode they belong to:

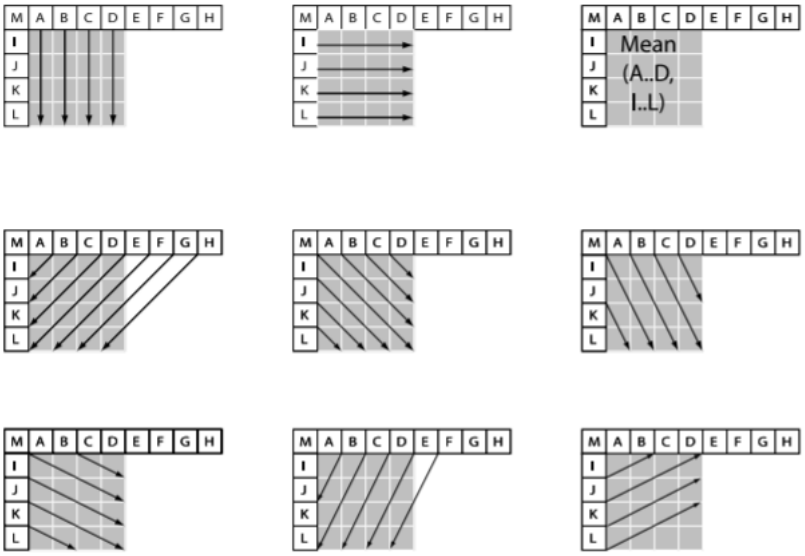


Figure 1-16: Nine budget macro block modes

Then, the predicted image is compared with the original image to obtain the residual value of the original image. When finally displayed, the complete original image can be displayed only by adding the residual value of the prediction model.

AAC： Also known as Advanced Audio Coding, it is an audio coding technology based on MPEG-2 [19], and adds SBR and PS technologies. In theory, AAC not only has a more efficient encoding method, but the sound quality is improved compared to MP3, while the compression size is significantly reduced. However, there is still a loss of sound quality, and since video streaming is mostly video rather than audio, it is sometimes used in the lossless format of PCM.

YUV： YUV is a color-coding method. Compared with RGB [20], when YUV codes photos and videos, it takes human perception into consideration and reduces the width of a certain chroma. Where Y represents brightness and CbCr represents chroma. The advantage of YUV is that even without U and V, it can still display a complete picture, but it gives the impression of a black-and-white picture [21]:

Figure 1-17: YUV in relation to the graph, Y in the upper right

Therefore, we can reduce the color of U and V to reduce the size of each frame. In the format of smaller frames, more pixels and clearer pictures can be transmitted. This frame format is YUV420 [22]:

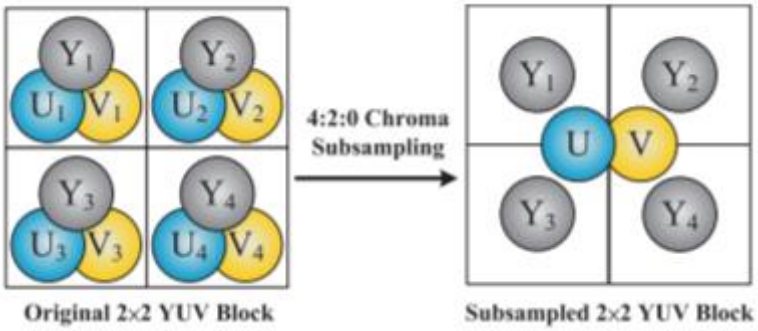


Figure 1-18: The difference between YUV420p and YUV [22]

In the form of storage, each Y will not use their own U and V, but first get a square four Y, U and V respectively, summation and averaged the U and V to obtain one U and V as their U and V. The advantage of this is that every four Y's share one U and V and this is just to reduce a certain degree of bright colors, but saved the space of three quarters. When such data format is obtained, YUV420 has two transmission modes, one is the general form of YUV, that is, according to the format of pixels, each pixel is composed of YUV. In this method, their U and V will be transmitted four times repeatedly, resulting in a waste of resources.

The other one is YUV420P [23], which means that Y is transmitted first, U and V are transmitted last. Even if the bag is lost, it will not affect the integrity of the picture, but only affect the color of the picture which will not affect the user experience too much. But if use YUV420, it will appear sharp picture tearing feeling. Cause a piece of picture to be missing.

1.2.3 Transfer protocol

FLV： Short for Flash Video [29], this package is different from MP4 for it forms very small files, occupies low resources, and loads very fast. This ensures network viewing of video files and effectively solves the problem that SWF files are too large after video files are imported.

In general, FLV files are mainly divided into header and body. The body of the file consists of a series of tags, and each tag includes some fields including previous tag size. The content can be video, audio or even script. It has strong adaptability to RTMP stream. For audio frames, each tag takes up only one byte to store audio information, such as audio encoding type, sampling rate accuracy, etc. The rest is audio data. The same thing with video is that it only takes one byte to store information like frame type and encoding type, and everything else is video data. Compared to MP4's package, FLV has most of the space for valid data, greatly reducing its own redundancy. This is because in the transmission of video and audio, even if the bit error rate is not high enough in the process of FLV transmission, the loss is instantaneous and the feeling is not obvious. However, if the transmission mode of bandwidth expansion is adopted, the transmission bandwidth will always be in a relatively high efficiency state.

RTMP： After ensuring the above data carrier, a reasonable transmission carrier is needed to ensure enough bit error rate, transmission speed and delay. RTMP is a family of protocols based on TCP. It is mainly designed for real-time data communication, mainly used for audio and video and data communication between streaming media/interactive servers of RTMP protocol [30].

Its transmission principle is as Figure 1-19 [31]:

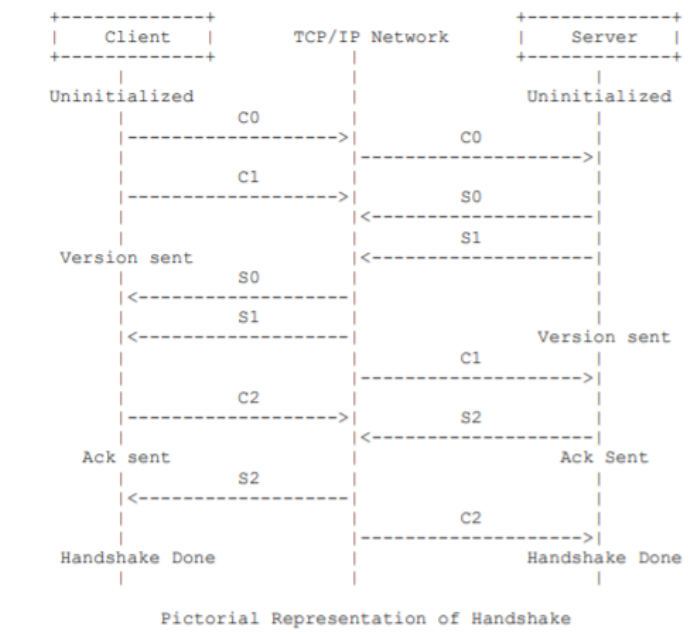


Figure 1-19: RTMP handshake protocol

That is, when the user and the server are enabled, the two sides first reply and quickly send their version numbers without waiting for the reply. After receiving the version number from the other party and confirming that it is correct, it will send its own ACK reply. And complete the handshake process.

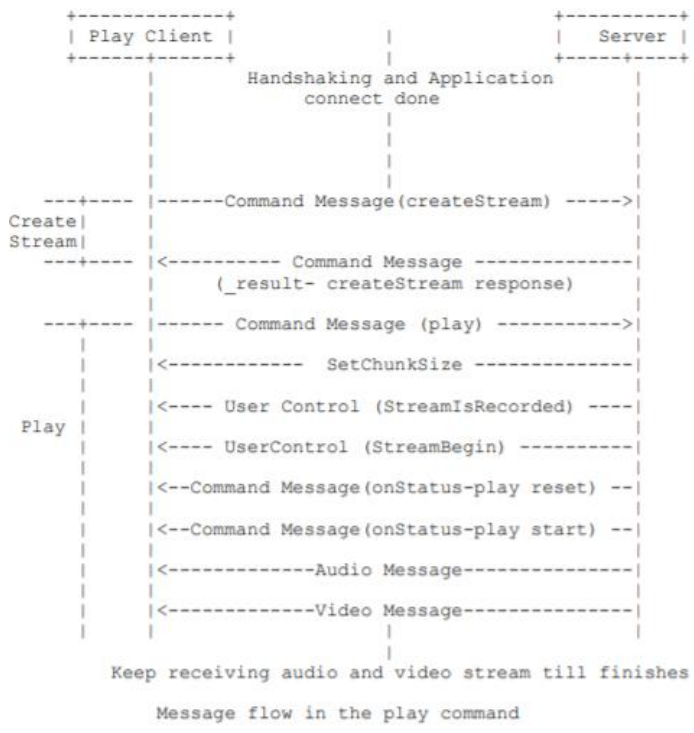


Figure 1-20: RTMP push stream protocol

Once the handshake protocol is complete, the user can ask the server to create its own

flow. After obtaining the server's request permission, the user needs to send the server's request confirmation again until the server hands over control to the user, so that the user can implement the data stream, pushing the data from the file to the server.

1.3 Thesis Work and Contributions

Based on the above preliminary analysis, this thesis hopes to design a wireless HD video transmission system based on millimeter wave through a series of work to solve such or such problems. The wireless transmission system can not only receive 10Gb/s of massive video, but also cache the data to its own multiple servers, so that the corresponding users through millimeter wave, wirelessly access to these high-definition video, to meet the needs of users in this respect.

The main work and contributions of this thesis are as follows:

Design the wireless system transmission architecture based on millimeter wave: Make the corresponding hardware platform comparison. According to the characteristics of wireless transmission system and user requirements, select suitable hardware transmission devices, including network ports, PCIE and NPU, etc. Choose the appropriate software transfer technology, including FFmpeg, OpenCV, OpenGL, etc. Finally, a complete wireless transmission system architecture based on millimeter wave is designed.

Design and implement the hardware prototype of wireless video transmission system: According to the architecture of wireless video transmission system, the circuit of 10 Gigabit network port is mainly designed to receive data to NPU through 10 Gigabit network port. The PCIE circuit transmits control data and video streaming data through the PCIE on the link that the NPU forwards to the MMW module by the server. The temperature module and EEPROM based on IIC driver are designed to measure the temperature of NPU and store u-boot information to ensure that NPU can be started. The USB module circuit and the RS-232 circuit, respectively for NPU kernel image programming and OS operation.

Developed and implemented the software of wireless video transmission system: To realize the wireless video transmission system requires video carrier and transmission protocol. OpenCV is programmed to obtain the relevant HD monitoring video stream, which lays a foundation for FFmpeg conversion. Programming design of

FFmpeg push stream function, including audio and video codec and FLV package and RTMP package, as well as FFmpeg pull stream function, after Qt audio and video output laid a foundation; Rules are also designed for the Nginx-RTMP server to receive push streams and forward them.

The performance of wireless HD video transmission system based on millimeter wave was tested and verified: Two software Iperf and WebBench are transplanted from X86 architecture in this thesis. The maximum transmission rate from ten gigabit network port to NPU, from NPU to millimeter wave module, and through millimeter wave module to receiver is tested by Iperf We used WebBench to test the maximum load of the server Nginx running on NPU, such as the maximum bandwidth rate it can handle, and how many users it can receive.
After the completion of the pressure test, this thesis carried out the test of wireless transmission of video, first carried out 1080P wireless transmission based on RTSP camera. Subsequently, the wireless video stream transmission of normal MP4 1080P, 2K, 8K and even 8K VR is carried out, which verifies the feasibility of wireless HD video transmission on millimeter wave.

1.4 Thesis Structure

This thesis is mainly divided into six chapters, which elaborates the design reasons, design process and system verification of millimeter wave HD wireless video transmission system.

Chapter 1 mainly describes the history of millimeter wave and its performance advantages in indoor wireless transmission. Then, by focusing on millimeter wave, the carrier that can transmit HD video and reasonable transmission mode are selected, as well as the work summary and contribution of this thesis, the feasibility of the HD wireless video transmission system is realized and verified, and the user experience in video is improved.

Chapter 2 mainly describes the hardware and software framework of the system, and analyzes the comprehensive requirements of the product from the perspective of application scenarios. On this basis, the hardware platform is compared, and the appropriate transmission hardware and transmission software are selected. Finally, a more precise hardware scheme and a more logical overall software design are selected

to design a wireless transmission system structure based on millimeter wave covering both hardware and software.

Chapter 3 mainly describes the hardware circuit part of the system, including the selection of core board, performance, circuit and its related design, as well as the main board circuit design and peripheral circuit design. According to the analysis of Chapter2, certain circuit designs are carried out on the RJ45 network port, PCIE, related chips and connection types are selected. The EEPROM circuit and temperature sensing circuit are designed on the IIC driver, which are responsible for the storage of U-boot content and the measurement of NPU temperature. There are also USB module and RS-232 as the module circuit of program and debug respectively, to ensure the normal monitoring and operation of the whole system.

Chapter 4 mainly describes the software part of the system, including the software framework of the system implementation, the process of making and programming the root file system; Push flow related FFmpeg and OpenCV running logic; Forwarding and shunt functions of Nginx server in the system; Docker program for Nginx stable; MMW-driven operation logic flow; FFmpeg push stream part as well as Qt audio, video display and final OpenGL rendering and so on.

Chapter 5 mainly describes the speed test of pressure and video stream test of usage after the completion of the system design. Among them, Iperf and WebBench are used for the pressure test of links and servers. The video stream test uses video streams of various video resolutions to push the stream and test the feasibility of the system in wireless HD transmission.

Chapter 6 is a summary and outlook, mainly analyzing the shortcomings of the system, and hoping to further optimize and improve the hardware and software in the future. Continue to work on wireless transmission.

2 System Design

This chapter selects the hardware platform and designs the system block diagram according to the wireless transmission algorithm and video transmission algorithm. Block diagram is mainly divided into sending platform and receiving platform, covering the corresponding hardware platform, link and software program.

2.1 Hardware platform

The core module of the system is NPU module and baseband module, according to the requirements of the selection of appropriate platform, is the key to the whole system to work with low energy consumption and high efficiency.

2.1.1 Baseband & RF module

To select demodulator chips and RF chips that meet the requirements of 64QAM modulation mode and 10Gb/s transmission rate, RWM6050 chip from IDT company is selected in the baseband module in this thesis, which supports 1.76GBaud symbol rate and 10Gbps transmission rate. Support phased array antenna digital domain beam shaping, IEEE 802.11AD single carrier PHY. The MAC layer supports AES-GCM encryption.

Figure 2-1: RWM6050 baseband IC

TRX BF/01 is used in RF chip. The chip adopts SiGe process and zero if architecture. Supports 16-shot + 16-beam shaped RF millimeter-wave transceivers, and can be extended to 32/48/64 antenna transceiver units. It supports working frequency band 57-71GHz and total output power up to 22dBm. Typical operating voltage 3.1~3.5V. Power consumption in the receiving mode is 3W, and power consumption in the sending mode

is 4W. The phase amplitude adjustment resolution is 5.6 degrees. The chip fully meets IEEE 802.11AD standard.

Figure 2-2: TRX BF06010 RF IC

2.1.2 NPU platform

At present, there are many server hardware platforms that can be used as receiving and pushing streams and forwarding, starting from X86 architecture, ARM, and even the final 51 series can be used as central processors.

X86: has high performance and high-power consumption, in today's ARM architecture performance is gradually shortened environment. Even today's X86 PC, its power consumption is over 30W [33]:

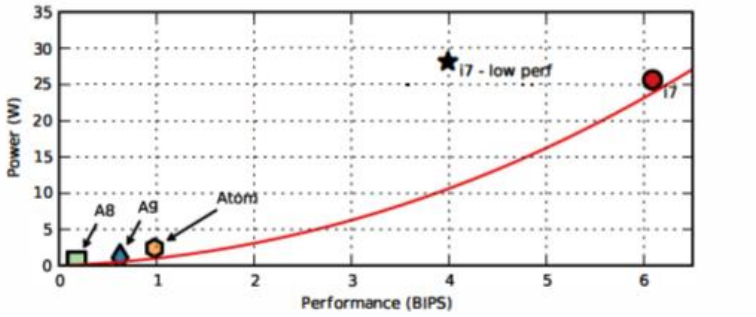


Figure 2-3: Hardware Platform comparison [33]

8051, PIC: 8051 dues to the use of crystal analog clock, 8051 system has high real-time operation, but the reaction speed is too slow when used as OS [34].

PIC is called programmable interrupt controller, which can help the microprocessor handle more interrupts and reduce the processing burden of the CPU, but the clock frequency is lower [35]:

Table 2-1: Hardware Platform comparison [35]

	8051	PIC	ARM
Bus width	8-bit	8/16/32-bit	32/64-bit
Communication Protocols	UART, SPI, IIC	PIC, UART, LIN, Ethernet, SPI, IIS	USART, IIC, SPI, USB, Ethernet, DSP, SAI
Speed	12 Clock cycle	4 Clock cycle	1 Clock cycle
Memory	ROM. SRAM, FLASH	FLASH, SRAM,	FLASH, SDRAM, EEPROM
ISA	CLSC	RISC	RISC
Memory Architecture	Harvard architecture	Von Neumann architecture	Modified Harvard architecture
Power Consumption	Average	Low	Low
Families	8051 variants	PIC16, 17, 18, 32	ARMv4,5,6,7
Cost	Very Low	Average	Low

ARM: also known as Advanced RISC Machines, more than 95% of smart phones and tablets in the world now use ARM's instruction set. Its high performance, low cost and low energy consumption make ARM play a leading role in the CPU fields of smart phones, tablets, embedded control, multimedia digital and so on [36].

Therefore, this work adopts LS1046A, which has high performance in ARM platform, as the transmission system center. LS1046A has quad-core processor, main frequency up to 1.8GHz, ARM Cortex-A72 architecture. Supports eight Gbit Ethernet native ports, up to two XFI (10GbE) ports, PCIe 3.0 (x4), SATA3.0, USB3.0, UART, and IIC ports, and Ubuntu and OpenWrt operating systems. It is a high-performance processor for packets in network applications.

2.2 System Architecture

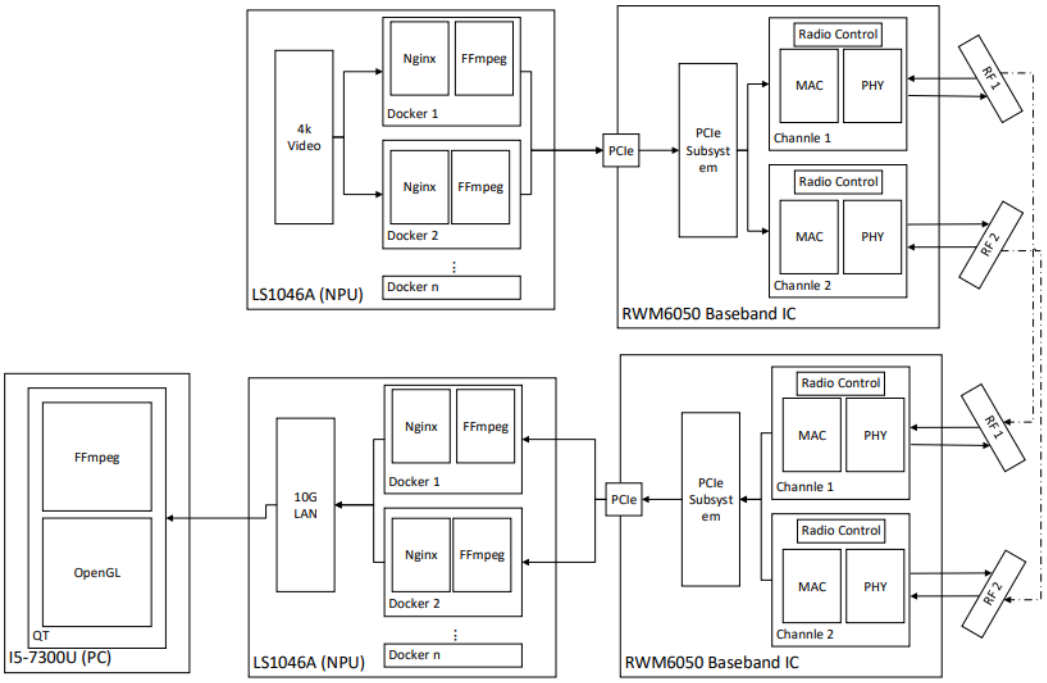


Figure 2-4: Wireless system block diagram

As shown in the Figure, this is the system block diagram of the whole HD wireless video transmission. This block diagram is mainly divided into two platforms, namely, video transmission platform and video receiving and playing platform.

The video transmission platform is responsible for video coding, wireless transmission coding and sending; The video receiving platform is responsible for wireless transmission and decoding, video decoding and the final playback.

2.3 System Design Criteria

The block diagram is mainly divided into publishing platform and pulling Platform, in which the video transmission platform is composed of NPU, Baseband IC and RF modules to achieve wireless transmission of 4K video. Compared with the former, the video receiving platform not only needs NPU, Baseband and RF to receive, but also needs to use 10G LAN and PC to decode and play the video.

2.3.1 Publishing platform

First, NPU will generate 4K video and call FFmpeg software to encode the video. FFmpeg, also known as Fast Forward MPEG will convert the RGB encoding format of NPU 4K video into YUV and stream audio and video into RTMP. Then, H.264 video coding and AAC audio coding are performed on the video [44]. Finally, it is packaged as VLC to transmit streaming video to baseband module through PCIe.

The streaming video will be modulated by the subsystem of the baseband module. The streaming data will be encoded with MCS to reduce the bit error rate, and then the data will be modulated with 64QAM and sent by the RF module.

As NPU needs to realize video multiplexed transmission, multiplexed video coding and sending are required in publishing Platform. To prevent the stability of video transmission and isolate interference between each other, Docker, also known as application container engine, is adopted to solve the complexity of virtualization. Efficiently run the video sending function and save a lot of memory [48]. Docker is mainly composed of four parts, in which the client integrates all runnable and running containers. In this thesis, multiple FFmpeg is placed in multiple Docker containers [50]:

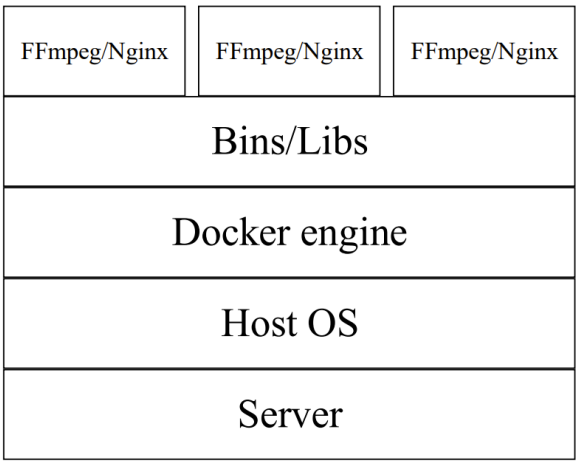


Figure 2-5: Docker runtime architecture

2.3.2 Pulling platform

Pulling platform receives streaming information from publishing platform, which is spectrum shifted by RF module, demodulated by 64QAM in baseband module, decoded by MCS by subsystem and transmitted to NPU via PCIe.

NPU uses Nginx, also known as Engine X. Nginx is a high performance, low memory footprint Web server that provides RTMP forwarding function for received video streaming data. If the RTMP module is equipped, Nginx can support video streaming from FFmpeg [46].

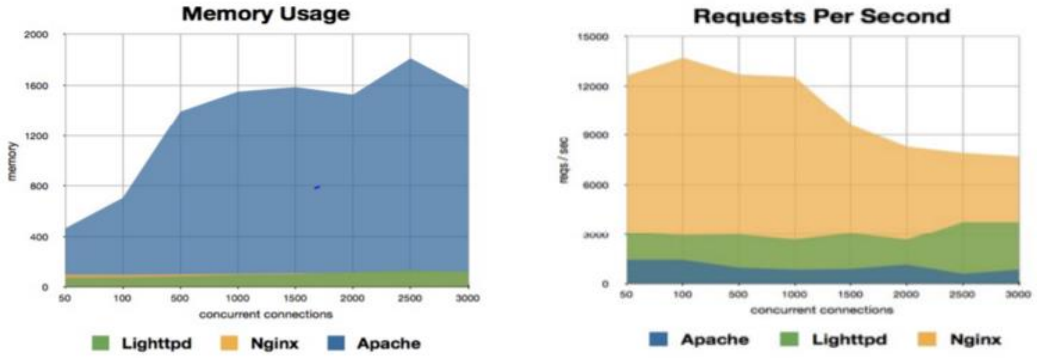


Figure 2-6: Nginx performance comparison [46]

Like publishing Platform, pulling Platform needs to use Docker to receive multi-channel video, which is essentially a mirror image of every Nginx-containing process. Such an NPU can easily run multiple Nginx servers, and when one of the Nginx servers crashes due to heavy load, it does not affect the other container servers.

Since NPU does not have the function of video output, the video is decoded and played on the i5-7300U microcomputer. It mainly uses Qt to develop GUI programs, as well as console tools for audio playback and video rendering. Its cross-platform and good scalability, so that Qt can also call FFmpeg tools for video VLC unpackage, identify RTMP protocol and H.264 decoding, get audio and video timeline. At the end of the audio and video, audio and video and synchronous problem, video playback with Opengl, also known as open graphics library, is mainly used for rendering 2 d and 3 d graphics, cross-platform, each small calculation module units through the GPU graphics pipeline, to carry on the vertex shader, primitive assembly, geometry shader, grating, and the fragment shader, Fast rendering of each pixel. [54].

OpenGL is more compatible than Qt. Many user interface libraries provide OpenGL with the relevant context interface. Therefore, in this thesis, OpenGL is only responsible for video rendering and the final audio and video synchronization issues are handed over to Qt for joint implementation [55].

2.4 Design detail

Finally, the design specifications of the transmission product are as follows: In terms of hardware, a 10-Gigabit network port is required to receive video streams in the form of RTMP or RTSP. The current cost-effective chip is AQR107, that is a network card chip with high cost performance and low power consumption. NPU needs a higher-frequency core processor, needs to have the ability to process images and videos, and the transmission rate of video streams is very high. NPU also needs to have enough performance to process 10Gb/s speeds, so choose NXP's LS1046A. NPU not only has a frequency of 1.2GHz, but also can be overclocked to 2GHz when processing higher-rate video streams, which not only has low power consumption, but also has enough performance [57]. As for the hardware of the video source, we chose the RTSP camera, which is easy to convert from RTMP, and the Hikvision surveillance camera. For PCIE, because the maximum transmission rate of the millimeter wave module is 5Gb/s, and the dual output is used to achieve 10Gb/s speed, PCIE only needs to choose the least slot, PCIEx1, and choose the frequency of PCIE2.0 to meet this transmission without causing performance waste.

In terms of software, first on the video carrier, to compress the video to a certain extent and reduce the burden of MMW bandwidth, we chose H.264 encoding and YUV420p video frame format. Then in video transmission, to reduce the transmission delay and speed up the transmission, we chose the RTMP protocol instead of the HTTP protocol. The software that meets the above two requirements is currently open source and standardized cross-platform FFmpeg. The server chooses a lightweight server, and supports RTMP/RTSP Nginx. Even if it is a lightweight server, with the official test document, because it supports zero copy, the data can be cached without the user mode to make it load. There are 90Gb/s, far exceeding the bandwidth of MMW. For the video source, use OpenCV to obtain the data of the video stream and convert it through FFmpeg. Finally, in terms of video display, to better perform video and audio output and synchronization, we choose Qt with many functions for audio output, audio and video synchronization, and use related OpenGL components for video rendering. The former has certain advantages due to numerous functions and standardization, while the latter is widely used and has high portability.

Table 2-2: Design details of wireless transmission system

Hardware	**Protocol**	**Software**
AQR107 as Etherent Chip	RTMP & RTSP	OpenCV
LS1046A for NPU	YUV420P	FFmpeg
Hikvision RTSP Camera	H.264	Qt
PCIE2.0	AAC	OpenGL
MPU-6050 & TRX BF01 as MMW moduel	PCM	Nginx-RTMP
		Docker

3 Hardware Prototype

3.1 Hardware Architecture

This chapter mainly describes the hardware part of the system architecture, mainly the selection of the core board NPU and the circuit design of the main board. Select the relevant transmission hardware circuit around the main board circuit, including the peripheral transmission circuit and the board-level transmission circuit. Finally, there is the IIC-based U-Boot system storage module on the motherboard. RS-232 and USB modules related to debugging and programming. The specific structure diagram is as Figure 3-1:

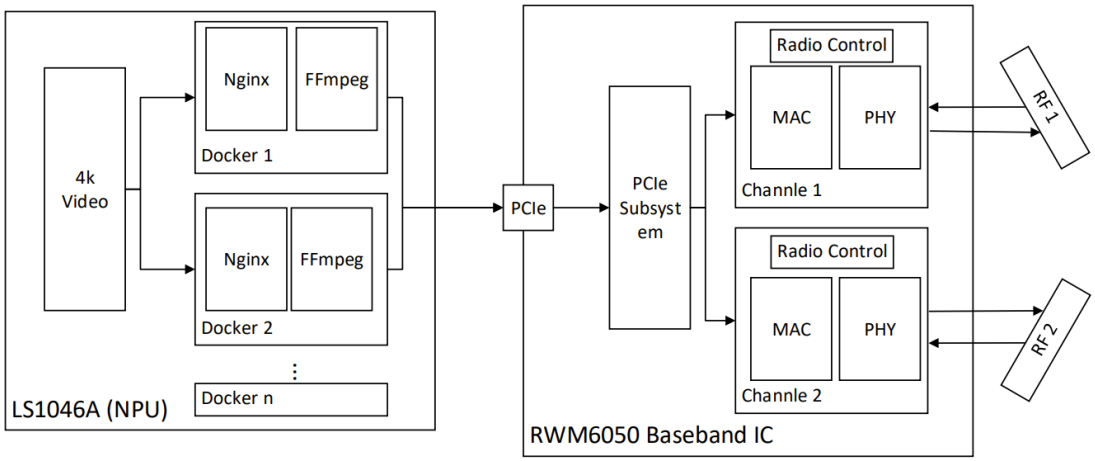


Figure 3-1: Hardware frame diagram of wireless transmission system

The USB module and EEPROM module program the root file system and U-boot to

start the core board, then access the system in the form of command lines through RS-232 to perform related operations.

The home host is connected to the motherboard through the 10-Gigabit Ethernet port AQR107, and forms a high-speed link with the NPU through the DPAA acceleration circuit. Then the NPU is connected to the millimeter wave module through PCIE, and the millimeter wave module modulates the information based on the baseband and transmits radio frequency signals. Finally, the receiver's millimeter wave module will transmit the received signal to the home host via PCIE.

3.2 Core Board

QorlQ LS1046A processor is a 4-core 64-bit ARM processor launched by NXP for embedded networks. Its core is Cortex-A72, with 48KB-I and 32KB-D level one caches, which are used to store data and commonly used RISC commands, respectively. QorlQ LS1046A processor is a 4-core 64-bit ARM processor launched by NXP for embedded network, its core is Cortex-A72, with 48KB-I and 32KB-D level cache, respectively for storing data and common RISC commands; It has 2MB secondary cache and built-in security engine to protect data security; Supports DDR4 SDRAM. DPAA hardware acceleration platform can be selected; You can even do hardware-enhanced virtualization to make the core run like multiple computers. The processor can be multiplexed, with multiple multiplexing schemes, so it has flexible I/O settings and provides broadband performance more than 10Gb/s. Solutions designed to address small scale networking and industrial applications. To realize the economical PCB, the power consumption is low, and a clock design is also adopted. Only 0.9V is required for a normal workload, where the core board and the corresponding NPU are shown in the image [62]:

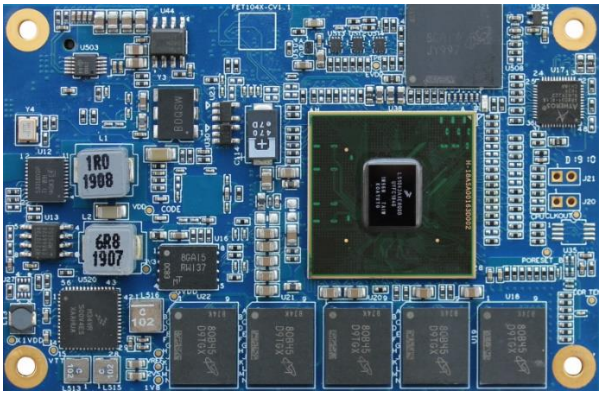


Figure 3-2: LS1046A NPU with core board

Because the NPU chip supports many interfaces, to make all the interfaces can be connected, NXP uses the core board to encapsulate the NPU chip, and the encapsulated pins are output in the form of sockets. To facilitate the developers to have a variety of choices for development, the core board is equipped with a variety of interfaces, can support QSGMII, SGMII, RGMII below 10 gigabit physical network port and 10G XFI physical network port. IIC serial port with associated debugging serial port. SDHC interface for starting the root file system, USB3.0 interface for programming, and multiple PCI 2.0 and SATA3.0 interfaces. PCIE can also adjust the frequency to PCIE3.0 to obtain higher bandwidth. To cope with the forwarding of massive video streams, the lowest frequency of the system is 1.2GHz, and the highest frequency can be overclocking to 1.8GHz. The power supply of the entire core board is only up to 9V-15V.

3.3 Mother Board Prototype and Relation Modules

3.3.1 Mother Board

Because of the slot mode of the core board, the main board circuit designed in this thesis adopts the structure of the core board and the mother board, and is connected through socket. The design of the published baseboard is as Figure 3-3:

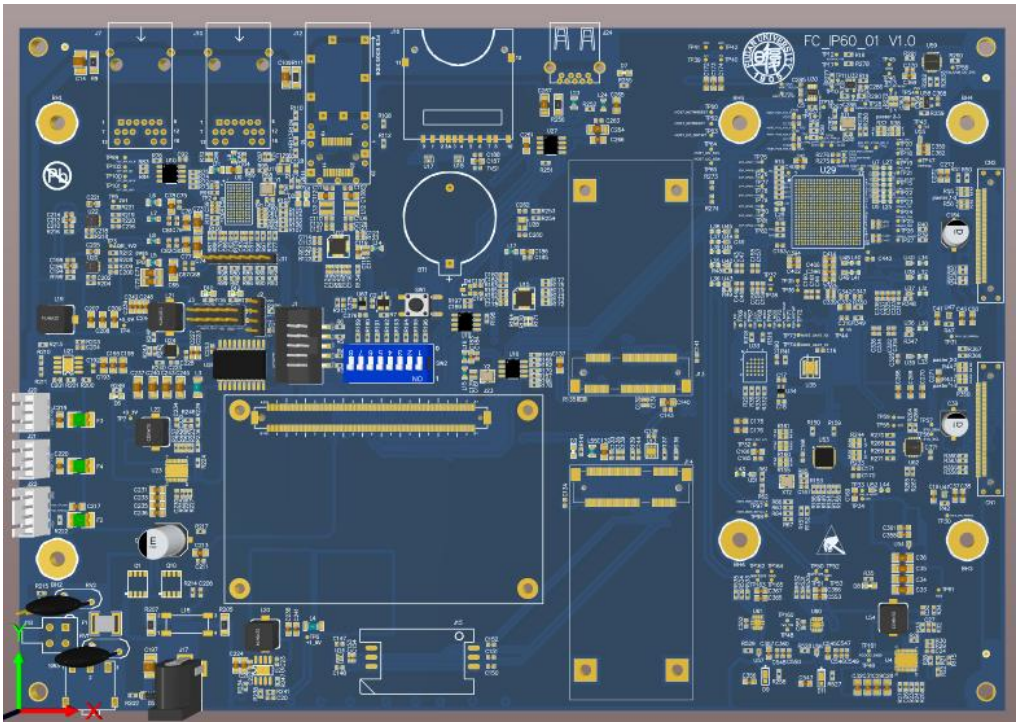


Figure 3-3: LS1046A based motherboard

You can see that there are many modules. However, since wireless video transmission only needs one 10,000M network port, we only retain the circuit of one 10,000M network port. Meanwhile, UART takes up less space in the selection of UART and JTAG. Moreover, if local Linux operation is carried out through the command line, JTAG can input data. However, it is not enough to print and output data, so we choose UART as our debugging serial port, and finally choose RS-232 for the protection of ESD. We embedded PCIE, which was directly led from the core board and connected with the millimeter wave module circuit instead of forming a slot. The EEPROM module is directly connected with the core board based on IIC, as a MEMORY chip for U-boot. USB module is also only designed for the programming of files. The MMW module is connected to the PCIE module borrowed from the core board to ensure that the NPU can use the PCIE to control the startup and working mode of the MMW module and transmit corresponding video stream information.

3.3.2 AQR107

The first receiving link of the transmission system is the network port with ten gigabit bandwidths. Its network port will be connected to AQR107 network port through four pairs of lines, each pair of which is differential to reduce cross talk, and its circuit diagram is shown as Figure 3-4.

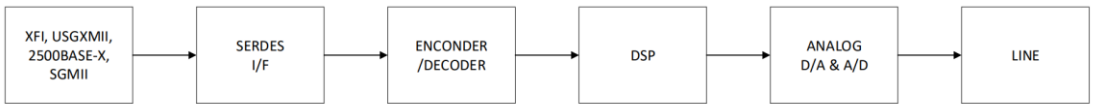


Figure 3-4: AQR107 Block diagram

When AQR107 receives information from the network port, it receives and sends data from the NPU under the condition that its own CLOCK works normally. That is, AQR107 maintains its own CLOCK frequency through the REF CLOCK to synchronize the CLOCK of the NPU. Communicates with the NPU during normal operation.

Because AQR107 is also a small CPU processor, it needs the corresponding FLASH memory to store the relevant processing algorithm and provide instructions to XFI SERDES. And because the processing is ten gigabit band-width, AQR107 also needs the INPUT and output of MDIO interface (serial communication bus management data);

At the same time to deal with some special needs, need JTAG and other AQR107 debugging and so on; Finally, corresponding digital power supply and analog power supply are also needed, not only to supply 3.3V, but also through corresponding level conversion to supply AQR107.

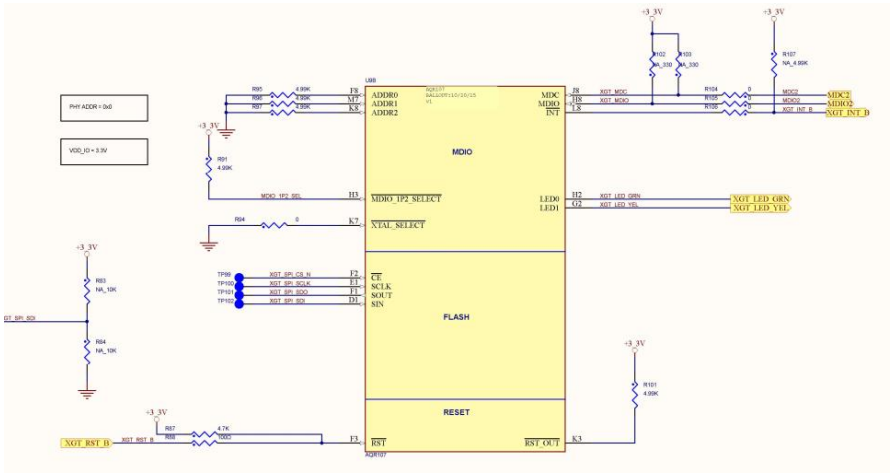


Figure 3-5: AQR107 control part

MDIO is responsible for obtaining the relevant data of XFI SERDES, and relies on the relevant algorithm stored in FLASH for processing and input and output, to ensure that the processing speed of data bandwidth can reach ten thousand megabytes.

In addition, the debugging of JTAG may be used when AQR107 fails. It mainly realizes the view and control of AQR107 through the low rate communication module of SMBUS.

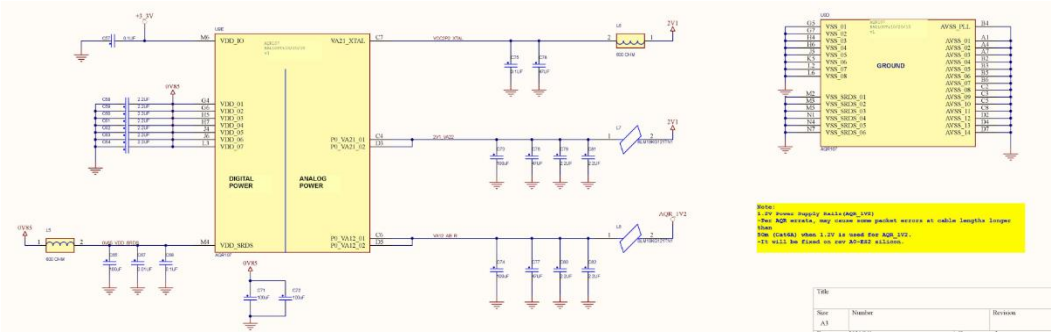


Figure 3-6: AQR107 power supply part

In the end, DIGITAL POWER and ANALOG POWER receive different voltages respectively to maintain the normal operation of AQR107 without interfering with each other to form a relatively clean POWER supply, while GROUND is responsible for providing AQR107 with clean GROUND to reduce cross-talk on other circuits.

3.3.3 PCIE

When the data is forwarded to the MMW module through the 10-gigabit network port and the DPAA of the core board, and then smoothly through the NPU. The related PCIE module is needed, and the general circuit of PCIE X1 is as Figure 3-7:

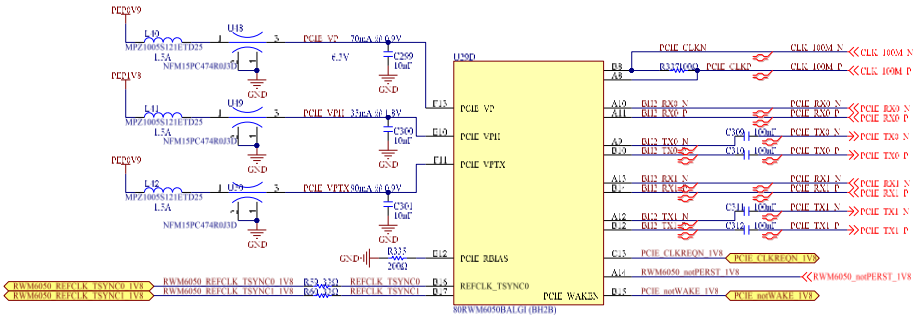


Figure 3-7: PCIE socket circuit diagram

The PCIE mainly uses the following signals to realize transmission:

The two signal power supplies are 12V and 3.3V respectively. 3.3V is the main power supply and the main logic modules used by PCIE devices are powered by 3.3V. But for some special equipment, such as some registers such as STICKY Register, 12V is required.

The second is the PREST signal, which is a global reset signal sent by the NPU. CPUs provide a reset signal for the PCIE device. If the signal is valid, the PCIE device resets. There are two Reset modes: Warm Reset and Cold Reset.

REFCLK+ and REFCLK- signals are used to synchronize the NPU with PCIE. The PCIE requires a reference clock at a frequency of about 100MHz. The NPU provides a reference clock for each such slot to ensure synchronization. And the clock generally has certain requirements, the transmission delay of the two to 2.5ns. In addition, the synchronization mode is divided into two types. Depending on the Common Clock Configuration, the synchronization mode can be synchronous or asynchronous. The asynchronous mode facilitates remote PCIE connections.

WAKE signal. When the PCIE enters the sleep state, the CPU stops supplying power. The PCIE can work properly only when the PCIE sends a WAKE signal to the NPU. This requires a long period of power from the auxiliary power supply of 12V. In the WAKE process, multiple PCIE devices perform bitwise matching and inform the NPU

of the PCIE device to be started. At this point, the NPU provides the PCIE device with a 3.3V voltage and uses the PREST signal to reset the device globally. At this point, the WAKE signal will remain low. Both PREST and WAKE are set to invalid, ending the entire WAKE up process. In addition, PCIE can also be woken up by Beacon, which is a differential signal, but this requires the PCIE device to be in L2 state.

Although SMCLK and SMDAT signals originate from the IIC, they can communicate with PCIE through the IIC bus. That is, SMDAT sends and receives messages from PCIE, and SMCLK provides clock signals for SMDAT. At present, this SMBus protocol has been widely used in Linux. It can be used to communicate with related devices, send control instructions, and obtain the description information of related devices.

PRSNT1 and PRSNT2 signals, which are mainly responsible for the hot swap of PCIE. In the case of Add-in, PRSNT1 and PRSNT2 are directly connected, because they are directly embedded, there is no so-called hot swap problem. However, on the parent board of an NPU, PRSNT1 is generally grounded, and PRSNT2 signals are in a high resistance state through the pull up resistance. When the card slot is inserted into the PCIE device, PRSNT1 will be connected to PRSNT2, and the PCIE device is no longer in a high resistance state. Then, the NPU will obtain whether the PCIE device is inserted through the impedance. Note that the gold finger of the PCIE device corresponding to the two interfaces is relatively short. This advantage is that the NPU is notified of software processing after all links are established. The software notifies the NPU that the device is removed and then disconnects the physical link to implement hot swap.

PETp0 and PETn0 communicate with the NPU, and PERn0 and PERp0 communicate with devices connected to PCIE. The NPU can obtain the physical addresses of PETp0 and PETn0 through memory mapping, and then send data to the PCIE bus through these physical addresses. At this point, the MMW module will also obtain the relevant physical address mapped by PERn0 and PERp0 memory, and receive the relevant data from the physical address.

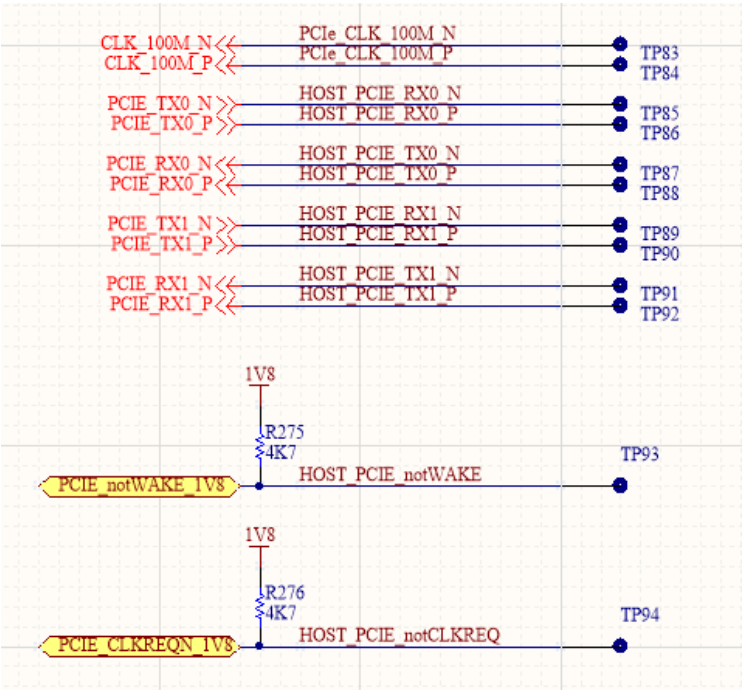


Figure 3-8: PCIE connection diagram

Finally, to reduce the size of the motherboard, this thesis removes the PCIE slot and directly connects the PCIE pins to the pins of the millimeter wave module; Perform the following connection logic to connect the NPU to the MMW through PCIE, so that the NPU can send control commands and related data to the MMW through PCIE.

To be specific, PREST and WAKE are generally invalid during normal operation. REFCLK maintains synchronization between PCIE and the system through the clock, while NPU can send control information to the MMW module through SMCLK and SMDAT. These include increasing REFCLK frequency to increase transmission bandwidth, etc. Finally, PET and PER assist the data transmission between NPU and millimeter wave module along the high-speed link.

3.3.4 DPAA

When the traditional Linux system processes data packets, it generally goes through the kernel mode, which is the working mode of receiving and sending packets in the kernel mode. When the data enters the kernel mode, these data will be handed over to the kernel protocol stack for processing. Although this traditional method allows developers to focus on the upper-level work without having to deal with too many complicated protocols in the user mode. But it brings great trouble to the data

forwarding [58]:

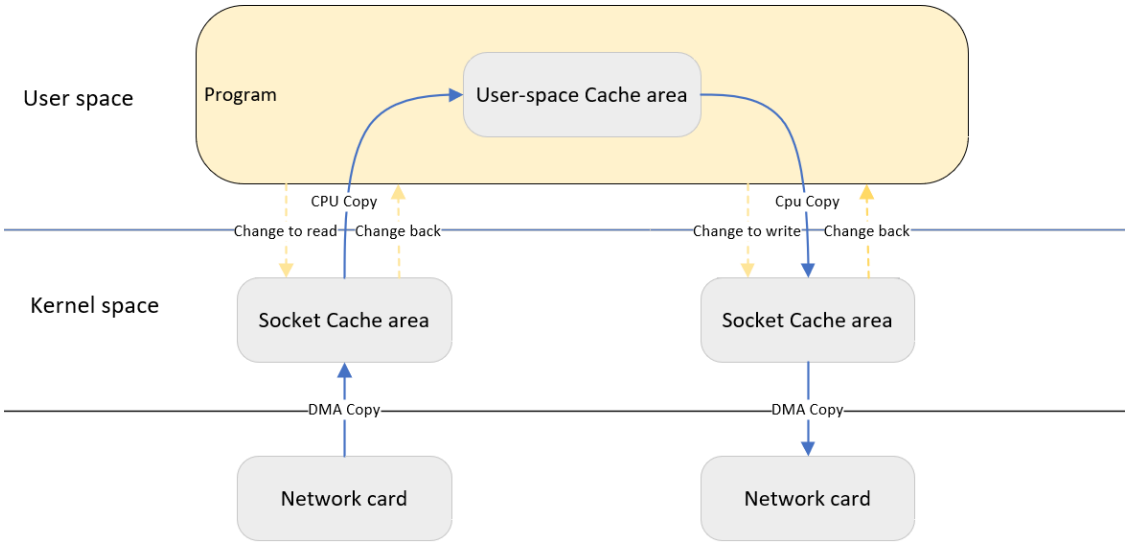


Figure 3-9: Redundant copy processes

When a piece of data is forwarded, it will first copy into the cache area of the kernel mode through the first copy, and then the user mode is always in the read state at this time, once it finds that there is data in the cache area, it will switch to the state of write, and the cache data in kernel area is read into the user cache area. This place not only performs two state transitions, but also performs a CPU copy, allowing the high-speed CPU to perform a low-speed copy.

Obviously, the operating efficiency of the CPU is greatly reduced. When the relevant protocol is used in the kernel mode, it is also necessary to switch between two states and one CPU copy, and finally from the user mode into the socket buffer in the kernel mode. After the encapsulation of the protocol is completed, it will finally go to the forwarding, Network card. Obviously, many copies are made in one forwarding, the user state has also undergone many state transitions, and the operating efficiency of the CPU is also affected.

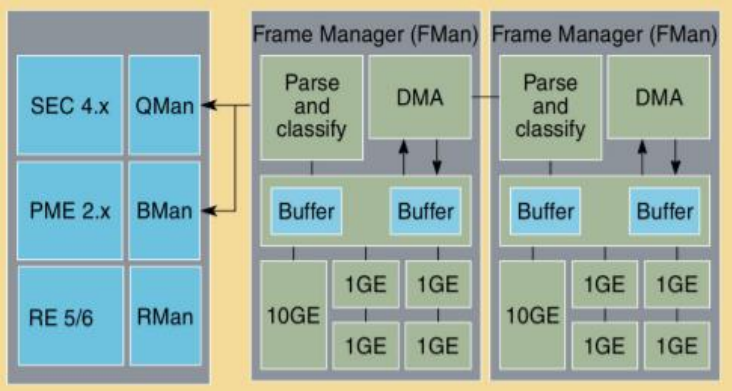


Figure 3-10: DPAA hardware architecture [59]

Therefore, we adopted NXP's DPAA hardware acceleration platform [59], which is mainly composed of three modules, namely the cache management module BMAN, the queue management module QMAN and the data frame management module FMAN. Specifically, when the data comes in from the network card, BMAN in DPAA will optimize the cache area of the kernel mode and perform the management function of the kernel cache pool. It can not only reduce the cache load in the kernel mode, but also replace the cache management of the software. BMAN will also use a free linked list to maintain these caches and release them reasonably when they are not needed. Then QMAN will provide a data queue function between the kernel and the network interface according to the order in which the data is stored in the BMAN, speed up the transmission speed of the network interface, and reduce the number of copies of the CPU that reduce the operating efficiency. QMAN not only has congestion management, priority queuing mechanism, etc., but also has the functions of packet sorting and recovery order, and for some faster data, it also has a simplified low-latency interface and so on. This lays the foundation for FMAN to copy to the buffer area of the Socket. And FMAN is responsible for managing the flow of the package. FMAN through the information exchange with the software, it does not require multiple state transitions in the user state to analyze the package and classify the flow, etc., and can reasonably pass the QMAN data beyond the user state. The copy is directly copied to the corresponding network card to ensure that the data is sent out. And even the copy can be sent directly to the network card through the driver without the CPU [60].

Through Khan A's experimental work [61], they found that the whole data flow direction could be directly controlled by user mode drive without kernel mode copy, which could not only improve the data transmission rate, but also reduce the delay to

some extent. [61]

Table 3-1: Performance comparison between user mode and kernel mode [61]

	User space	Kernel space
Throughput (Mbps)	1152-1808	259-487
Latency (ms)	281-666	431-762

Therefore, in this thesis, DPAA is used to accelerate the hardware circuit of the network port flowing to NPU, to reduce the lack of broadband in the transmission link.

3.4 MMW Modules

Millimeter wave module is the central part of the whole wireless transmission system, which mainly includes baseband modulation and radio frequency. The baseband modulation part is responsible for modulating/demodulation of the video stream from PCIE, and the RF part is responsible for raising the baseband video stream to the millimeter-wave frequency and sending it. The specific framework is as Figure 3-11:

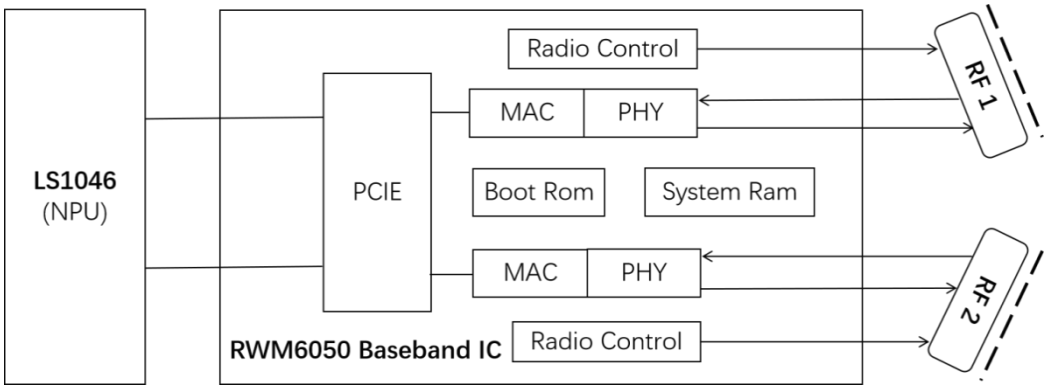


Figure 3-11: The connecting frame of millimeter wave and NPU

That is, the NPU transmits video streams to the baseband chip over the preceding two PCIE circuits. The PCIE transmission rate is 5Gb/s. That is, the two links are 10Gb/s. In the baseband chip, the video stream will be transmitted according to the IP address of PCIE, which is like network protocol and network port. After the modulation, it will be encapsulated in MAC and PHY respectively and then sent to the RF chip to adjust the frequency. Finally, the radio frequency chip will send point-to-point through beam formation and the highest transmission rate that the RF chip can receive is 7Gb/s. Due to the limited PCIE transmission rate, the maximum transmission rate is 10Gb/s.

3.4.1 Baseband IC

The baseband modem selected in this thesis is RWM6050 developed by IDT Company. This chip is a standard system on chip based on IEEE 802.11AD. It supports millimeter signals in the 57-71GHz band and even provides point-to-point and point-to-many wireless transmission modes, suitable for wireless back trip and fixed infrastructure access applications.

The chip adopts dual modems and supports two independent PHY-MAC channels and two corresponding RF channels to achieve parallel processing of RF signals. The internal operation subsystem can configure PCIe transmission, modulation and demodulation, and encoding mode. Its symbol rate support 1.76GBaud, transmission rate support 10Gbps. Digital domain beamforming of phased array antenna is supported. Physical interfaces of baseband chip include PCI Express Gen2 for high-speed data transmission channel, Flash SPI, I2C/SPI/UART, GPIO, JTAG and analog front-end interface connected with RF module. Its specific architecture is shown in the Figure:

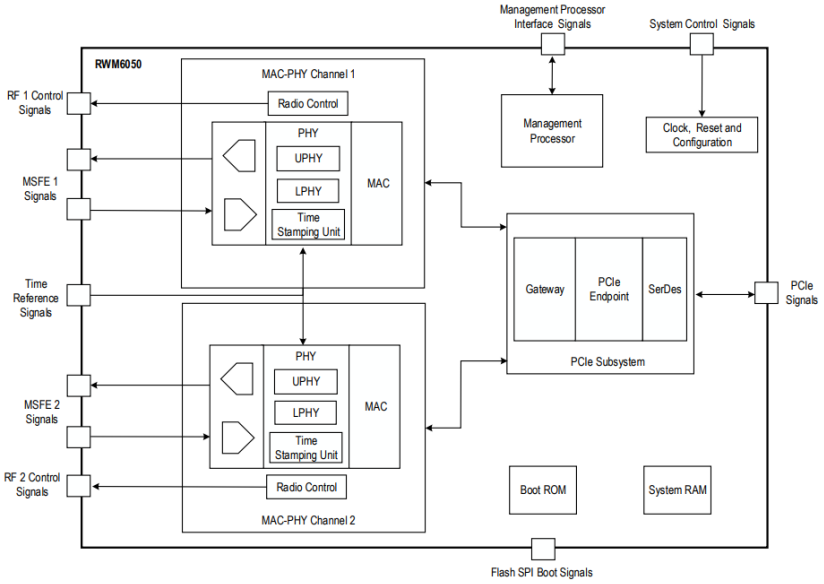


Figure 3-12: Baseband IC block diagram [62]

3.4.2 Radio frequency module

In this thesis, the RF chip adopts BFM06010 chip from SiversIMA Company, which reduces the complexity and difficulty of constructing the whole millimeter-wave system by setting an interface to interact with the baseband module.

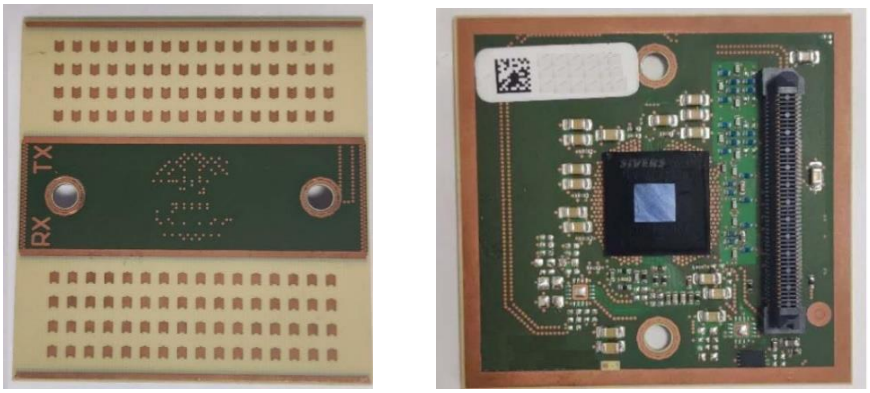


Figure 3-13: BFM06010 RF IC

The Figure shows the RF communication module, which meets ieEE802.11AD standard and supports the working band of 57-71GHz. Supports 64QAM modulation. Integrated low noise amplifier and linear power amplifier, total output power up to 22dBm. Typical operating voltage 3.1~3.5V. Power consumption in the receiving mode is 3W, and power consumption in the sending mode is 4W. The phase amplitude adjustment resolution is 5.6 degrees. The chip fully meets IEEE 802.11AD standard.

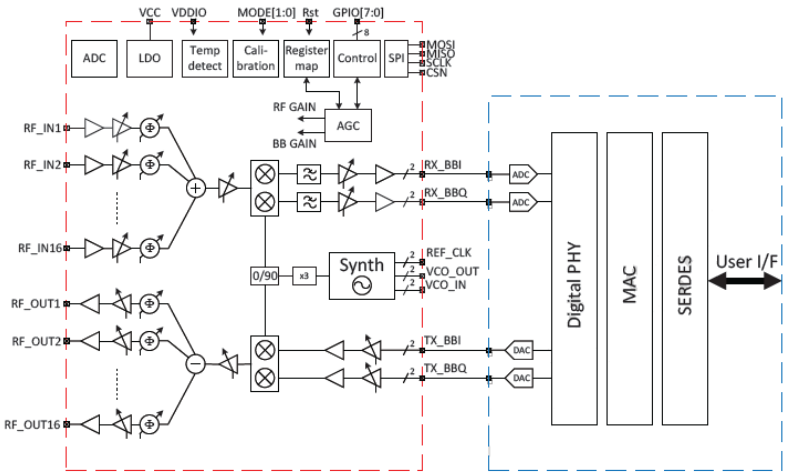


Figure 3-14: RF IC block diagram [62]

The RF module integrates TRXBF/01 millimeter-wave beam-forming IC, loop filter, LDO and antenna array, and has 16 transceiver beam-forming channels that can form a +/-45° steering and thus a 90° beam scanning and communication range. The module uses analog channel filtering to suppress out of band interference to optimize the signal path. AGC and ALC optimization modules are used to optimize the receiving gain and transmission power, and integrated linear power amplifier to achieve 22dBm output power.

3.5 Integration of Transmission System

Finally, combined with the above framework, the prototype architecture of the whole system transmission entity in this thesis is shown in Figure 3-15:

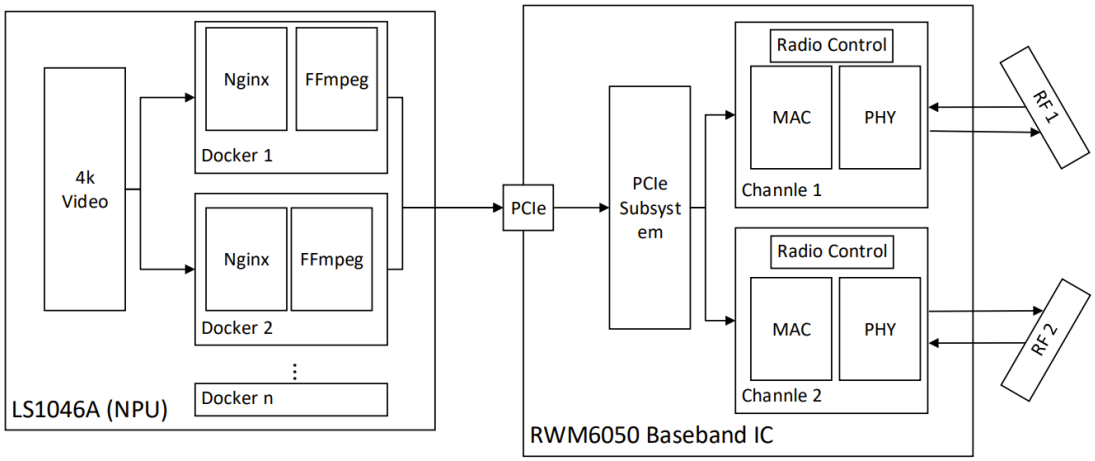


Figure 3-15: Hardware frame diagram of wireless transmission system

The home host is connected to the corresponding mainboard through the network port to realize the push of video stream, and the Linux system can be controlled and observed through RS-232. The mainboard is connected to the NPU of LS1046A through a socket, and the video stream information is quickly sent to the NPU by using DPAA hardware acceleration. Then the motherboard connects to the baseband chip 6050 through PCIE, and finally sends data to another millimeter wave module with the 6050 and rf module vector. The MMW module is connected to the home host through the PCIE slot to receive and output video stream.

.

4 Software System of Wireless transmission

4.1 Preparation of the system

The other part of the wireless transmission system is the software part, which mainly includes two parts. The first part is the startup part of the system, which is mainly related to the boot program and root file system required for NPU startup. The overall process is as Figure 4-1:

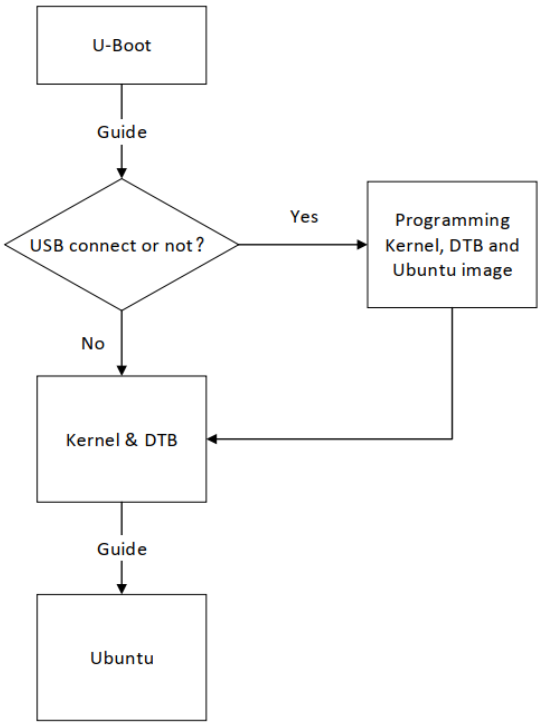


Figure 4-1: System Startup Process

That is, the NPU is started by U-boot and the peripheral circuit self-check is performed to check whether the USB device and its corresponding script are contained. If the requirements are met, the NPU will program the device tree and root file system, as well as the kernel, according to the run script written by the USB device. After the program is complete, the NPU boots up as the kernel and device tree. The kernel initializes the hardware from the device tree and boots up the Ubuntu root file system. When Ubuntu starts up, the system starts up.

The second part is the operation part of the system, which mainly describes the

transmission of video stream on the software under the condition of complete hardware circuit. Including the push stream of video stream, the forwarding on the server and the pull stream of the last video stream, the overall process is as Figure 4-2:

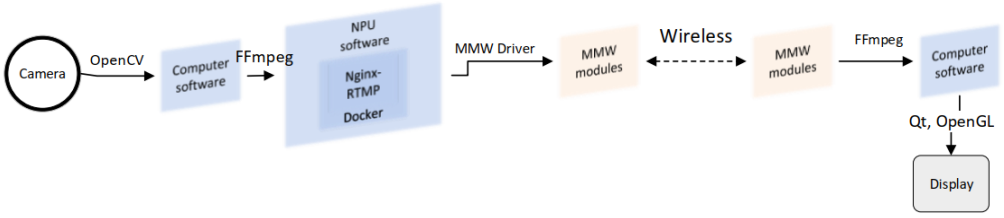


Figure 4-2: Software Framework Process

That is, the home host uses OpenCV to obtain the video stream, hand over FFmpeg for video stream processing and push the stream to the NPU server. NPU caches relevant video data through Nginx and RTMP modules, and relies on Docker to protect the operation of each server from other environmental impacts. And is driven by millimeter-wave to provide service to another home host. The receiver's home host will obtain relevant video data through FFmpeg, which will be decoded and transferred to Qt for processing. Qt relies on OpenGL rendering, combined with FFmpeg data for audio and video synchronization and output.

4.1.1 U-Boot

First, the Boot of any system, no matter Windows or Linux, needs UEFI or BIOS Boot program. As the embedded LS1046A, when it wants to start the Linux system, it generally uses a recognized Boot system program u-boot. For LS1046A system chip, because it has more I/O port, can have rich interface scheme, with DPAA, can over frequency and so on. In addition to the U-Boot, it also requires some additional firmware, including RCW (reset control word), OPTEE (Trust Zone technology developed by ARM), and so on. Therefore, we need to use the relevant toolkit provided by NXP to recompile the software according to our requirements.

Generally, u-boot is mainly used to initialize DDR4. To improve Linux operating efficiency, you can select an appropriate memory running frequency. However, because the program is relatively small, its running speed is only 1-2s even without optimization.

Most U-boot startup is divided into two stages. Stage 1 is mainly assembly code, and

Stage 2 is mainly C language, which can realize more complex codes. The related process is shown as Figure 4-3 [64]:

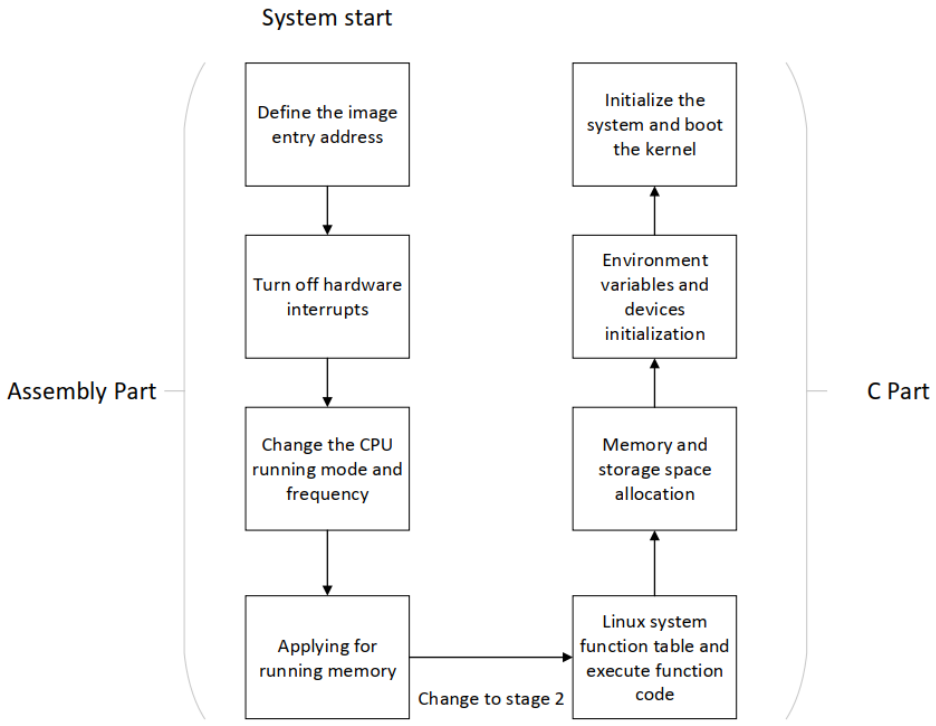


Figure 4-3: U-boot Boot process

For phase 1, defining the image entry is the start of u-boot. Only when you know the image of the Linux system, you can place the image in memory to run and send the corresponding data to the kernel for running. To transfer control to the software level, phase 1 will then turn off various hardware interrupts, Watchdogs, etc., and change the CPU running mode and increase the running frequency. After obtaining enough storage space and u-boot memory space, transfer the image to the memory.

For stage 2, things that can be done in this stage can be much richer, including the Linux system function table and executing function code, running memory space and storage space allocation. Environment variables are set and various peripherals including IIC devices are initialized. Finally, the system initialization, let the Linux system online.

The following Figure shows the operation modes of u-Boot, which are mainly divided into Boot mode and download mode. Boot mode loads the operating system into DDR4 RAM from the corresponding EMMC (solid state storage device) on the NPU and runs the whole process autonomously. This is the most Boot mode of u-Boot. However, if

the EEMC does not have an image, u-boot downloads the kernel and root file system from the USB device through a USB serial port connection, saves the image of the EMMC in the NPU, and then places the image of the EMMC in the memory for Boot.

Considering the performance of the transmission system, we need to multiplex LS1046A with GPIO and improve LS1046A's NPU operating frequency and PCIE operating frequency, to improve the bandwidth of video stream transmission. Therefore, some changes are made on RCW:

```
#include <ls1046a.rcwi>

SYS_PLL_RAT=7
MEM_PLL_RAT=21
CGA_PLL1_RAT=18
CGA_PLL2_RAT=16
SRDS_PRTCL_S1=4160
SRDS_PRTCL_S2=21849
SRDS_PLL_REF_CLK_SEL_S1=1
SRDS_PLL_REF_CLK_SEL_S2=0
SRDS_DIV_PEX_S1=1
SRDS_DIV_PEX_S2=2
DDR_FDBK_MULT=2
DDR_REFCLK_SEL=1
//sd
//PBI_SRC=6
//qspi
PBI_SRC=4
IFC_MODE=37
HWA_CGA_M1_CLK_SEL=6
DRAM_LAT=1
SPI_EXT=1
SPI_BASE=2
UART_BASE=3
UART_EXT=0
IFC_GRP_A_EXT=1
IFC_GRP_E1_EXT=2
IFC_GRP_F_EXT=1
IRQ_OUT=1
IRQ_BASE=128
IRQ_EXT=0
TVDD_VSEL=0
DVDD_VSEL=2
EVDD_VSEL=2
IIC2_EXT=2
IFC_GRP_D_BASE=1
IFC_GRP_E1_BASE=1
SYSCLK_FREQ=600
HWA_CGA_M2_CLK_SEL=1
```

Figure 4-4: RCW Settings

SRDS_DIV_PFX_S1 and S2 are used to control the frequency of PCIE. If the value is 2, the PCIE in channel 2 is 3.0. If the value is 1, the PCIE in channel 1 is 2.0. UART_BASE and UART_EXT are used for LED IO multiplexing. IRQ_BASE and IRQ_EXT together form AWAKE to control the software for PCIE. IFC_GRP_E1_BASE is used to reset all I/O ports to provide software control rights.

After making the changes, this article configures the relevant environment, installs the missing 64 libraries and a suitable GCC cross-compiler, compiles all firmware, adds header information to the U-boot.bin file, validates the relevant image mapping entry, and so on, and finally generates the image.

```
File  Edit  View  Search  Terminal  Help
63+1 records in
63+1 records out
32604 bytes (33 kB, 32 KiB) copied, 0.000282405 s, 115 MB/s
26+1 records in
26+1 records out
13428 bytes (13 kB, 13 KiB) copied, 0.000208416 s, 64.4 MB/s
261+1 records in
261+1 records out
133738 bytes (134 kB, 131 KiB) copied, 0.000790704 s, 169 MB/s
55+1 records in
55+1 records out
28392 bytes (28 kB, 28 KiB) copied, 0.000259357 s, 109 MB/s
 /home/forlinx/work/OK10xx-linux-fs/flexbuild/build/images/firmware_ls1046ardb_
uboot_sdboot_1133_5559.img   [Done]
```

Figure 4-5: U-boot image generation result

4.1.2 Linux system

The startup and normal operation of a Linux system mainly consists of three parts, namely the kernel, the device tree and the root file system Ubuntu. Its overall Linux system startup process is as follows:

First, the first stage of assembly code will be carried out when the Linux kernel is started, which is to find the corresponding processor type and call the corresponding initialization function to establish the page table. Then enter the second phase, which is dominated by C. According to the NPU initialization environment and memory, all physical memory and I/O space are mapped so that users can access the corresponding hardware devices. Initialize interrupt function and system scheduling system, start process management; Create and initialize system caches, including starting virtual file systems; Initialize the memory management and start the corresponding memory management process. Initialize system socket communication; Finally, mount the root file system and start init of the root file system to run relevant scripts [65].

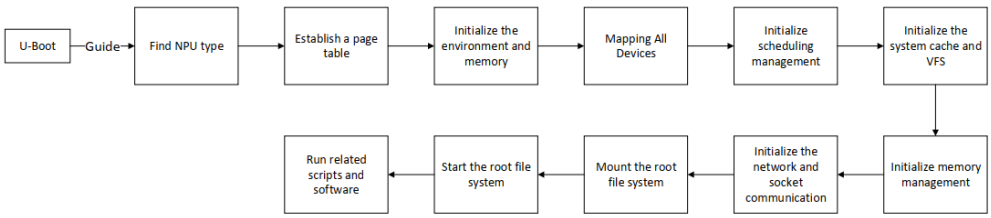


Figure 4-6: Kernel initialization process

The kernel refers to the most basic part of the Linux operating system. It is responsible for managing system processes, device drivers, memory, and network and file systems.

The kernel generally performs the lowest level tasks, controlling the NPU peripherals, processors, memory, hard disks, and other electronic devices. All the superior applications run in an orderly manner, relying on the kernel to coordinate multiple concurrent processes and manage the memory usage between processes. As a result, it provides interfaces to superior software without letting software developers pay too much attention to hardware details. Developers only need to make the software use the relevant kernel functions, the hardware can be flexible control. The kernel also provides applications with secure access to NPU hardware modules, determining how long a program can operate on a piece of hardware. Instead of exposing the entire hardware to the software and letting the software take the lead.

The kernel is divided into four subsystems, namely process management, memory management, file system, device driver and network subsystem. The corresponding control drivers and hardware are shown in Figure 4-7 [66].

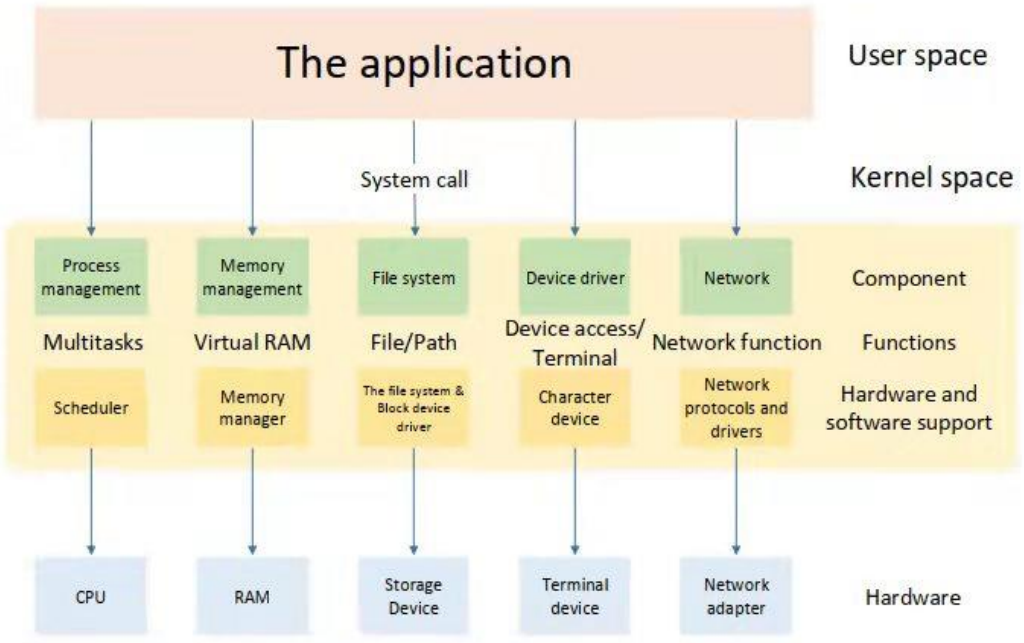


Figure 4-7: Kernel structure

The process management is responsible for the management of CPU resources, it will manage and schedule each process, so that each process has enough time and appropriate way to access the CPU, and exercise certain rights through the CPU; Memory management is to manage the running space of each program, allocate

memory resources reasonably, so that each process can share NPU memory resources safely. Sometimes, such memory resources can even be cross-used by virtual memory mechanism. And when the data in the memory is not needed temporarily, you can put the non-volatile memory in the hard disk first, and then retrieve the memory when it is needed; The VFS is a virtual file system in the kernel that is used to examine peripheral storage devices, such as hard drives, NAND Flash, and input/output UART devices, abstracted into a unified file operation interface to access operations such as open, close, read, and write. During peripheral checks, the Linux kernel accesses the Ubuntu root file system to be run, setting the stage for loading the root file system of the real Ubuntu image. Device drivers are responsible for the management of third-party devices/terminals. Due to the existence of many hardware devices, especially embedded products, their compatibility will be different, resulting in many device drivers. To solve this problem, the kernel unified interfaces for these devices through character devices, reducing development. Network subsystem is responsible for the management system of network equipment, realize diversified network standard, due to network equipment that a driver has more and more large, became an integral part of every NPU, therefore the Linux kernel for network equipment individually designed a subsystem, also designed the corresponding network interface device drivers and unified driver. Then on this basis, the subsystem is specialized, adding two parts of the network protocol, respectively, is to achieve a variety of network transmission protocols, as well as the network protocol shielding hardware devices, to provide the same interface to the upper layer like socket.

As the Performance of Linux system in embedded system is getting better and better, at the same time, to reduce the power consumption of embedded system, various peripheral devices are also abundant. Early Linux kernels sometimes targeted different peripheral circuits, using the same chip and storing vastly different peripheral information. This peripheral detail code is just junk code for the Linux kernel. This makes the maintenance of the native Linux kernel more difficult. Therefore, to strip peripheral information out, Linux uses device trees to store various details of peripherals, including the number and category of CPUs, memory base addresses and sizes, bus, peripheral conditions, attributes of GPIO controller and Clock controller, etc. [67]. Its service flow for the Linux kernel is as Figure 4-8:

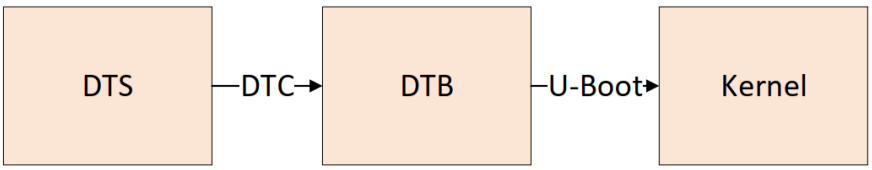


Figure 4-8: Device tree service process

In this thesis, the hardware information on the board level is written in the DTS suffix file, which is saved in a tree structure. To enable the Linux kernel to read these data, the DTC tool is used to convert them into DTB, and the data is placed on the USB hard disk together with the kernel. Then before the Linux kernel is booted, U-boot will be responsible for booting the device tree and the Linux kernel together, and the DTB will be booted to the kernel to ensure the normal operation of the kernel and self-check the peripheral devices. Lays the foundation for starting the Ubuntu root file system.

After the Linux kernel finishes checking itself against the device tree, it loads the Ubuntu file system stored in EMMC, uses the file system as the root file system, and executes initialization scripts such as rcs and inittab, which is the first user-mode process. And initialization of a few applications such as sbin, etc. and bin. These processes are started after the file system is mounted, and the process and memory management allocate resources. Users can access all storage devices through the file system on this interface. You can also use the corresponding commands to view the data and execute the relevant files.

This thesis adopts the DTB of NXP for LS1046A, which uses DPAA hardware circuit, and part of its DPAA configuration is shown as Figure 4-9:

```
fsl,dpaa {
    compatible = "fsl,ls1046a", "fsl,dpaa", "simple-bus";
    ethernet@0 {
        compatible = "fsl,dpa-ethernet-init";
        fsl,bman-buffer-pools = <&bp7 &bp8 &bp9>;
        fsl,qman-frame-queues-rx = <0x50 1 0x51 1>;
        fsl,qman-frame-queues-tx = <0x70 1 0x71 1>;
    };

    ethernet@1 {
        compatible = "fsl,dpa-ethernet-init";
        fsl,bman-buffer-pools = <&bp7 &bp8 &bp9>;
        fsl,qman-frame-queues-rx = <0x52 1 0x53 1>;
        fsl,qman-frame-queues-tx = <0x72 1 0x73 1>;
    };

    ethernet@2 {
        compatible = "fsl,dpa-ethernet-init";
        fsl,bman-buffer-pools = <&bp7 &bp8 &bp9>;
        fsl,qman-frame-queues-rx = <0x54 1 0x55 1>;
        fsl,qman-frame-queues-tx = <0x74 1 0x75 1>;
    };
```

Figure 4-9: Device tree Invoking DPAA

Menu-config is used to configure the kernel, allowing LS1046A to change the CPU running frequency through the command line, and then compile the kernel after the configuration file is generated.

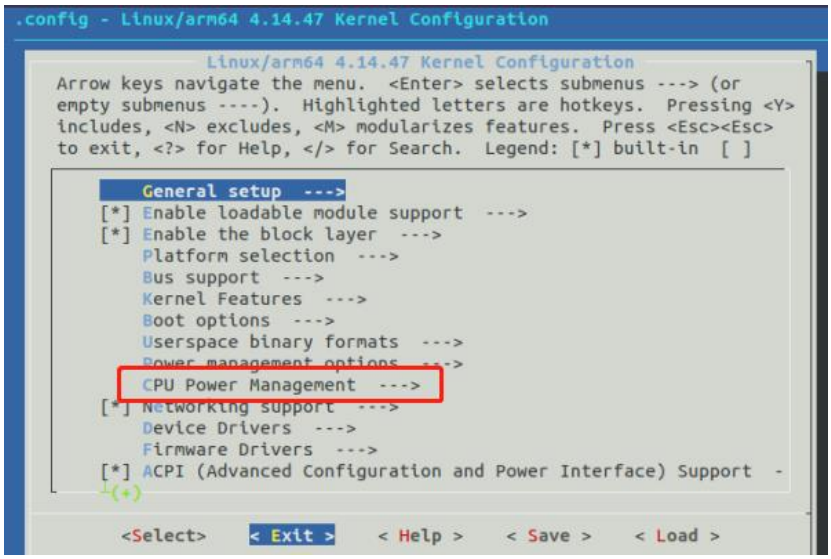


Figure 4-10: Kernel compilation

Finally, configure the root file system, add the corresponding FFmpeg and Nginx transplant files on the original root file system, reduce the tedious degree of transplantation later, and regenerate the Ubuntu root file system image, finally update

the corresponding kernel, DTB and root file system.

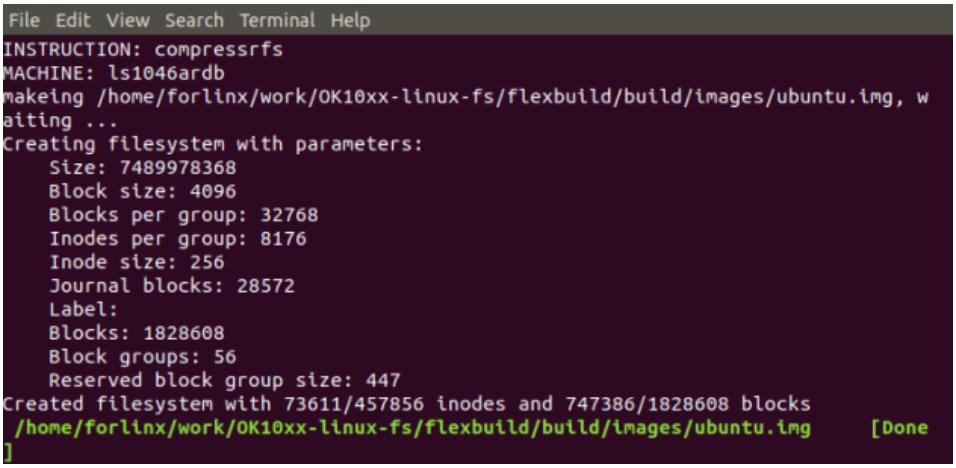


Figure 4-11: Compilation screen for the root file system

Finally, insert the USB hard disk into the USB interface of the mother board, and use the usb_update.itb tool to program related scripts, write the root file system, DTB, and kernel to the NPU.

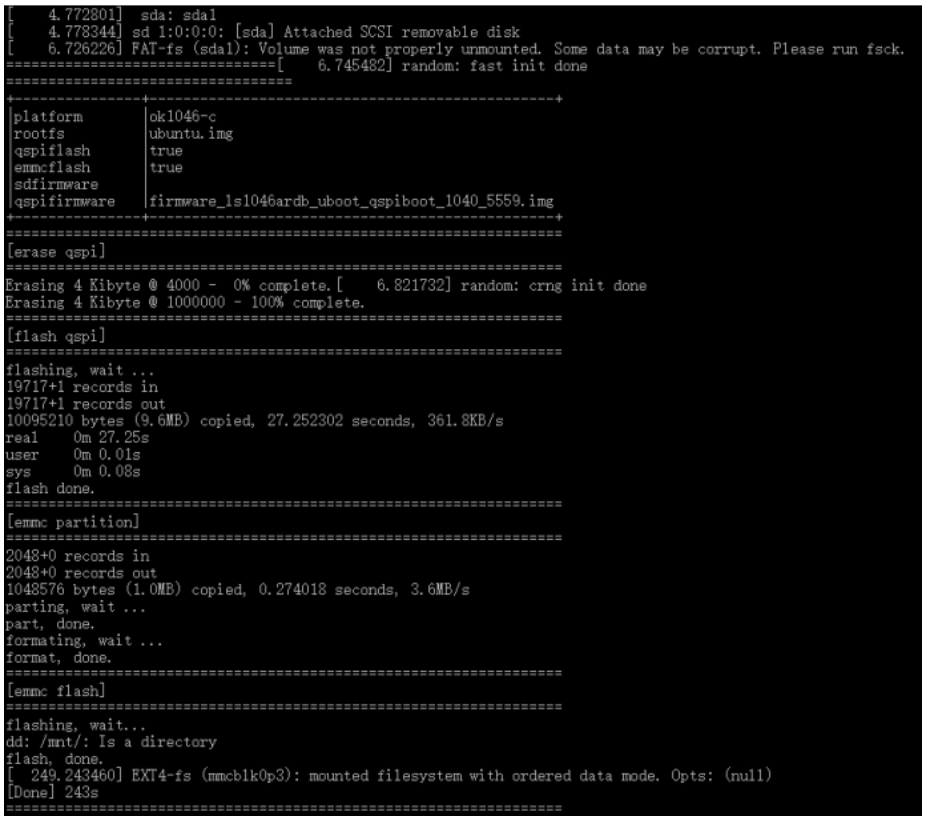


Figure 4-12: Root file system kernel burn

4.2 Driver and APP

Wireless transmission system is mainly divided into three parts, the first part is push current. The push stream includes two schemes: one is to use RTSP video stream for HD surveillance cameras, and the other is to push MP4-type HD video locally. In addition to the different video sources, both need to use FFmpeg and OpenCV for certain data processing, including format conversion, encoding and video stream encapsulation.

4.2.1 Logical of Flow-pushing

This stream push function is deployed on LS1046A to realize the encoding, compression and protocol encapsulation of video stream. Its block diagram is shown as Figure 4-13:

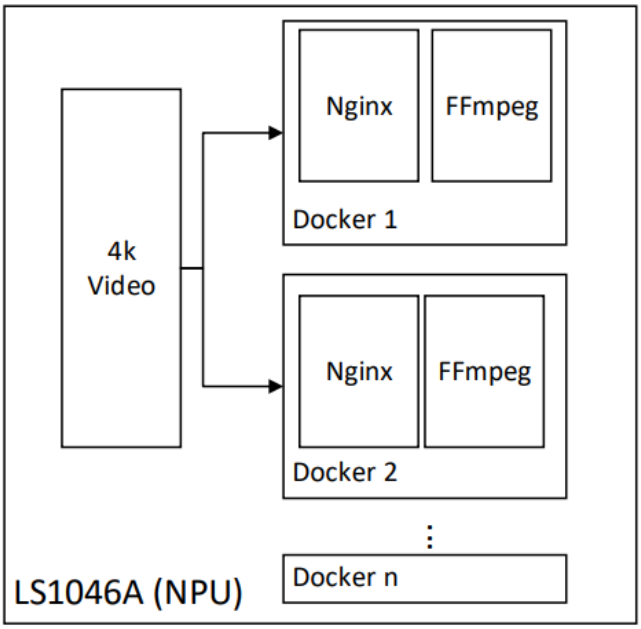


Figure 4-13: Block diagram of push flow

Take RTSP video stream as a video source for example, first collect external audio and video information through HaiKang RTSP camera, and then call OpenCV function to collect audio and video and get frames through OpenCV function, and submit to FFmpeg for corresponding data processing. The specific process is as Figure 4-14:

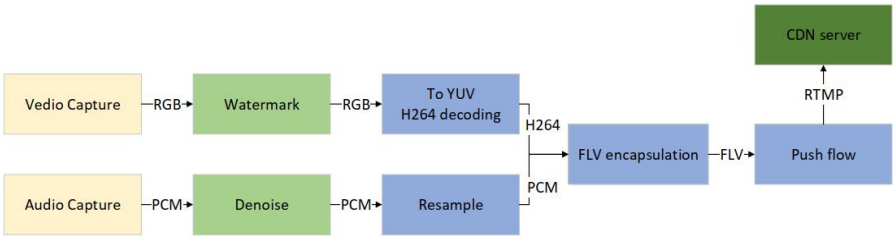


Figure 4-14: Flow chart of push flow

FFmpeg can choose to add watermark and noise reduction for video, noise reduction and change the voice for audio. Then, YUV420p frame format change and H.264 video coding is respectively carried out to transmit a clearer picture with enough bandwidth. Resampling the audio. Finally, the audio and video are encapsulated by FLV, and then the RTMP protocol is encapsulated to achieve the push stream.

Because the video involves two concepts in this process, respectively YUV420p and H.264 coding. FLV encapsulation and RTMP protocol are involved in the process of stream pushing. These concepts are described in detail in Chapter 1 and will not be repeated here.

In general, YUV420p is a compression of the frame format, which can effectively reduce the amount of bandwidth consumed by video streams without significantly degrading the picture quality. H.264 coding is a kind of coding method by reducing the time redundancy and space redundancy brought by the repetition between frames. FLV uses simpler encapsulation and architecture in the form of encapsulation, reducing the bandwidth required for network transmission. All three start in different directions, but the goal is to reduce bandwidth and display clearer video. To ensure real-time transmission, the RTMP protocol reduces the number of handshakes in the form of long connections to ensure that the handshake protocol is not flooded with bandwidth. The next few sections will cover the entire push stream process in more detail from the camera section and the FFmpeg section.

4.2.2 Video Signal

OpenCV as a programmable program to obtain video stream, its specific process is as Figure 4-15:

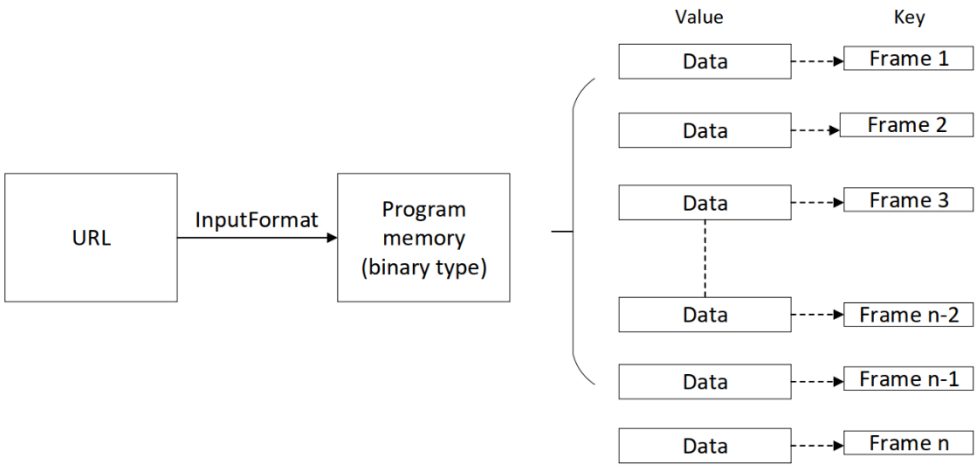


Figure 4-15: Operation flow of OpenCV

OpenCV first identifies whether the URL is a file or an RTSP address, and then it defines an Input-Format to read the video file or stream of the URL and store it in its own runtime memory. The stored information is in binary form. OpenCV also frames these video streams based on the type of file currently in use. Finally, these frames are stored as dictionaries, waiting to be retrieved by FFmpeg. When FFmpeg initialization is complete, the binary data will be retrieved for processing.

4.2.3 Flow-pushing

Before obtaining OpenCV frame data, FFmpeg should carry out corresponding initialization. Only after obtaining enough video stream information, it can carry out reasonable conversion. Its operation logic is as Figure 4-16:

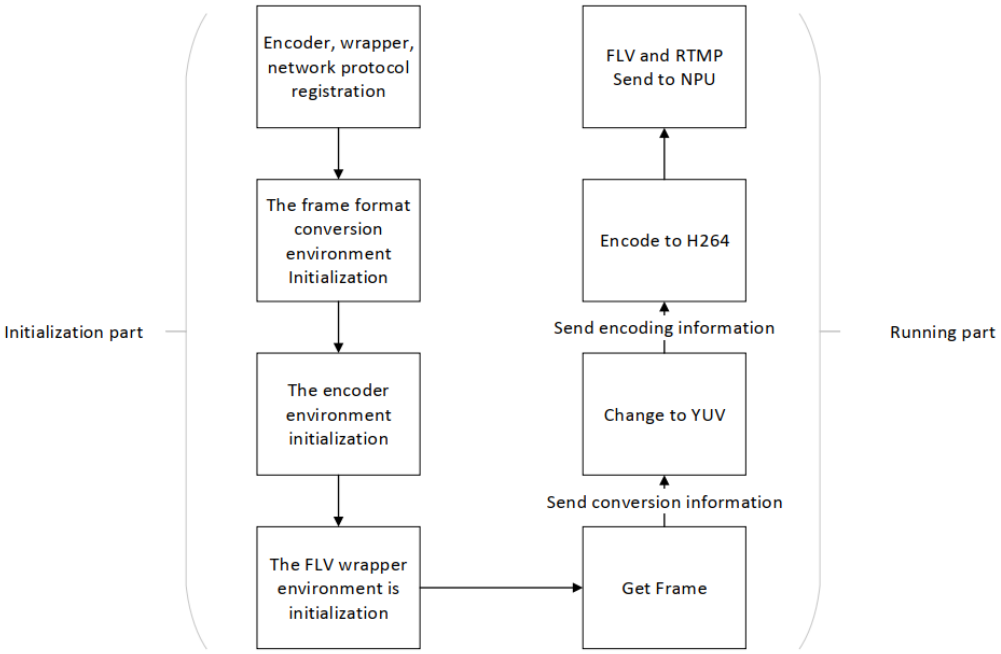


Figure 4-16: FFmpeg running flowchart

First, FFmpeg registers codec, wrapper and network protocol for H.264 coding, FLV packaging and RTMP protocol transmission; According to the video stream data obtained by OpenCV, the frame format conversion environment is initialized to facilitate the conversion of video stream from RGB to YUV420P. For H.264 coding environment initialization, including running memory application, encoding format, bit rate, GOP format and so on; Environment initialization for FLV packaging, including the output wrapper environment and the video stream packaging environment; Finally, determine the structure of the output data and open the RTMP output IO port to complete all initialization.

Then comes the complete data processing: continuously capture video frames from OpenCV, determine the mode of reading video stream, and convert video stream RGB to YUV format using the initialized frame format conversion environment. Then the YUV video frame is sent to the initialized encoder in the form of channel, and the data from the encoder is cached. Finally, the encapsulation environment of FLV is used to encapsulate the cached data and send data through the RTMP output port to complete the push of video stream. The actual operation of the software is shown as Figure 4-17:

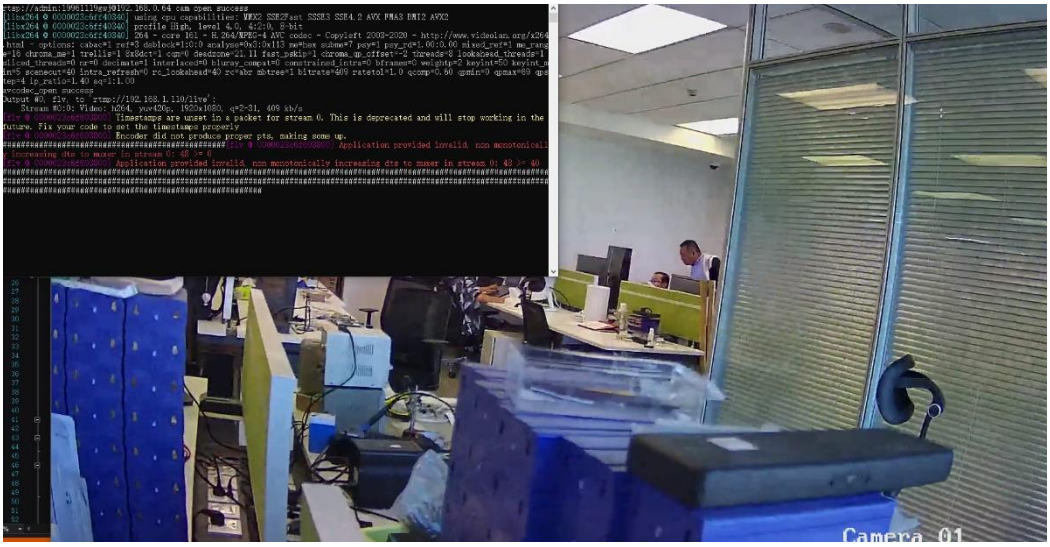


Figure 4-17: FFmepg and OpenCV running effect

4.3 Buffering and forwarding

Nginx is a web server with high stability, low resource consumption and high performance. Its powerful concurrency and low memory occupancy make it suitable for embedded system server. The server is mainly responsible for video stream transfer in HD video wireless transmission system. This chapter will explain the whole process in detail in three sections, including the overall logic, the operating principle of the server and the stability problem.

4.3.1 Logical of Server running in the system

Take THE RTSP video stream as the video source for example, when the video stream is transmitted through the home host along the ten-gigabit network port, the video stream will be accelerated to the NPU where the Nginx server is located due to the hardware of DPAA. At this point, Nginx listens to the RTMP video stream from port 1935 and opens its own port to the network. Another home receiver only needs to know the IP address where Nginx is running to access the corresponding HTTP web page and pull the corresponding video stream.

4.3.2 Nginx-RTMP of Server

The entire Nginx server is modular work, Nginx itself is not much work, every time it receives the corresponding data, it will look up the configuration file, through the

command to start the corresponding module work, so the module is the real worker of Nginx. In the process of RTMP transmission, the RTMP protocol module is responsible for transmission. The transmission process is shown in the Figure:

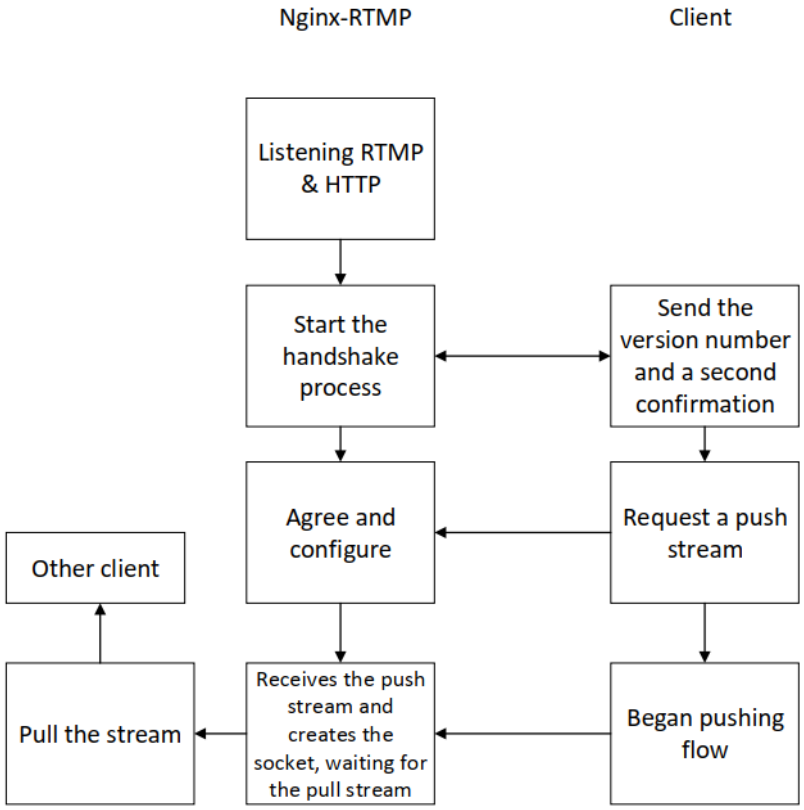


Figure 4-18: Nginx running process

Once Nginx is started. He will always be in a listening state, listening on HTTP and RTMP ports respectively, waiting for the client to connect in poll. When the client is connected, Nginx will start the RTMP module and call the handshake function, and start the RTMP handshake process with the client. The handshake process needs to exchange the version number of the double handshake. Finally, the client confirms the handshake with the server. Then when listening to the client to send data, the server will connect the client to the Nginx through the configuration of the application, and set the corresponding window size and broadband information, ready to receive the push flow information; The client will send the request to create the stream, and the Nginx-RTMP module will respond accordingly, indicating that the request has been received, and the push stream will begin. Nginx-RTMP releases a named stream to the server when the client sends push stream information. Nginx-RTMP creates a new socket with this name

and IP. Other clients can receive and pull corresponding audio and video information according to this name. Finally, Nginx-RTMP successfully pushes data to the corresponding website by opening the channel between the client and the output socket. The running state of Nginx is as Figure 4-19:

Figure 4-19: Running screen of Nginx

4.3.3 Multiplexing server

Because of limited connection server can take, and sometimes may need multiple servers to monitor different port, but if each server running together in a container, it's on the operation of the memory where there will be lots of sharing, is likely when a server fails, the server of the other affected and cannot run normally. To run multiple servers, but also to ensure that the stability of the server is not affected by each other. Docker, an application container engine with high generality and sandbox mechanism, is adopted in this thesis. Its operation mechanism and structure are shown as Figure 4-20:

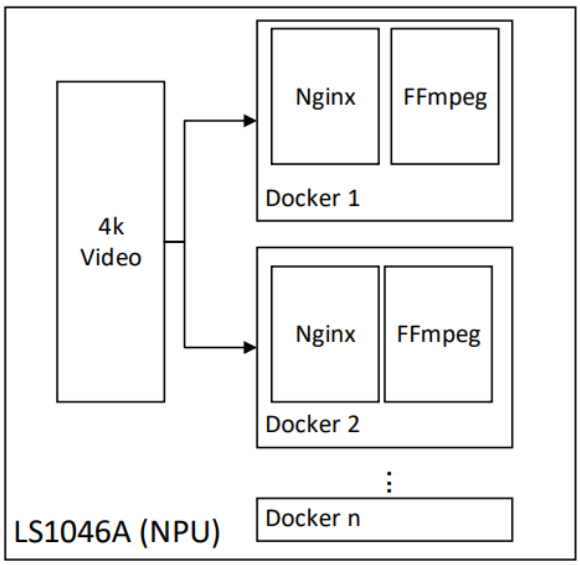


Figure 4-20: Docker architecture

In terms of structure, Docker is a client-server structure. Docker runs on the host Docker daemon through the daemon process. Users can access the corresponding container through the Docker CLI socket connection and configure Nginx accordingly. Rest API is the communication interface provided by Docker to the CLI, which helps users access docker and prevents them from operating docker at will. In addition, Docker CLI is also responsible for managing containers, images, network connections and data. The startup, operation and interface related data of all containers are controlled by CLI. It can be seen from here that Docker is only separated from the user by a layer of API Docker shares the operating system with the host and only has some bin files and lib files alone, which makes the image small, portable and flexible. Instead of creating a new operating system like a virtual machine [68]:

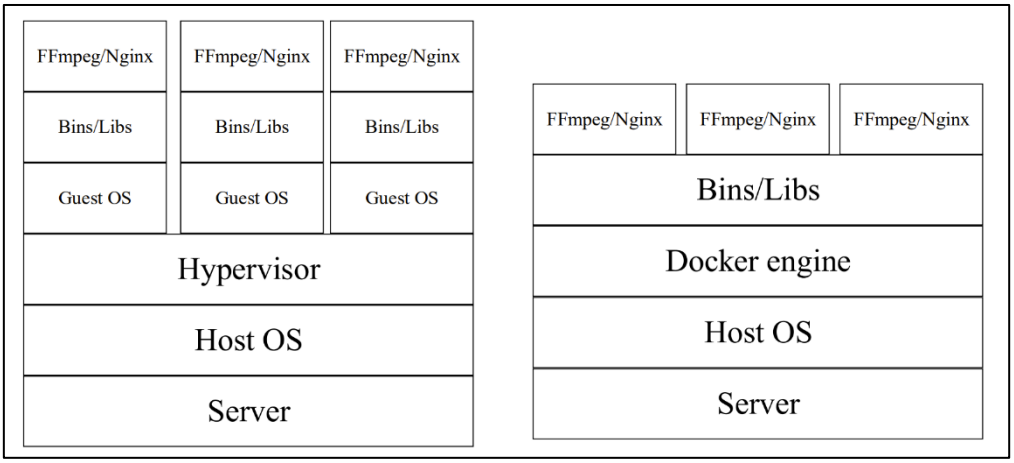


Figure 4-21: Comparison between Docker and other VIRTUAL machine architectures

In terms of mechanism, it is necessary to create multiple Docker containers containing Nginx and transplant them to LS1046A. Currently, there are only these containers on the Docker list, but they are not running yet. Then we can start multiple Dockers by command, and can delete, restart and stop them freely.

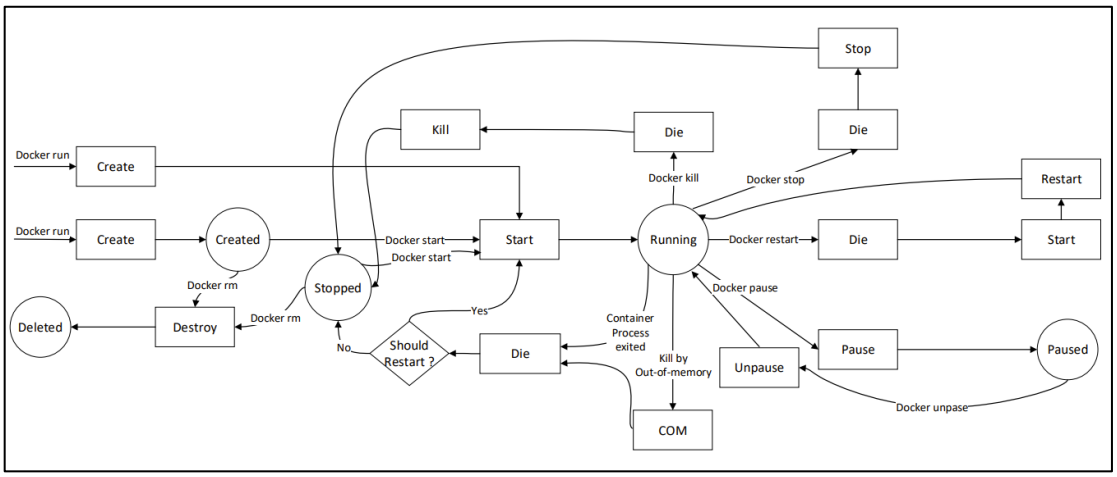


Figure 4-22: Docker running process

When running out of memory or the process is stopped. The Docker master only takes the initiative to close these broken containers to protect the normal operation of other containers.

4.4 MMW driver

Although the MMW link has been implemented on the hardware and the transfer of software such as Nginx server has been configured, the actual operation of the hardware still requires the mount and use of relevant drivers.

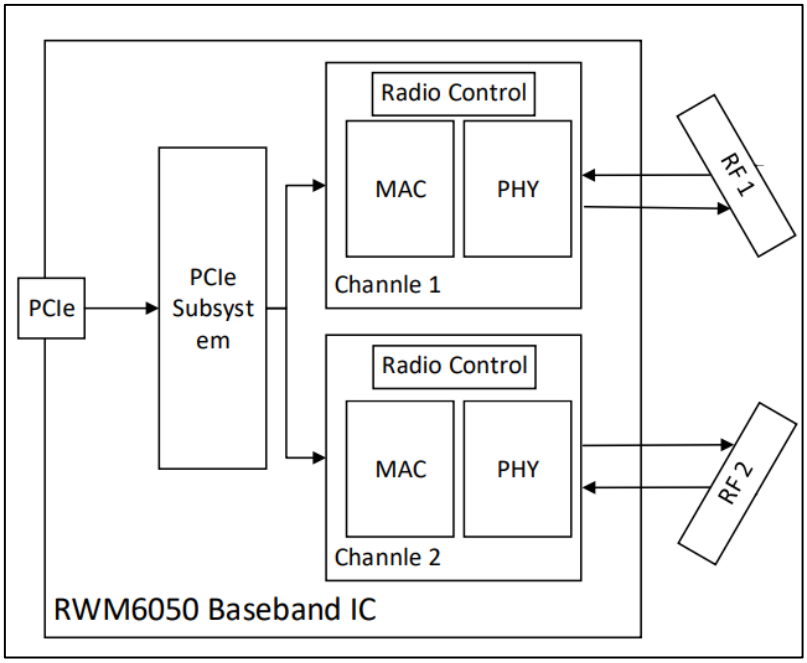


Figure 4-23: Block diagram of MMW module

Due to the current MMW module is not popular, NXP and other companies will not maintain and provide the default driver supporting MMW module. Even if the NPU is connected to the MMW module through PICE, the NPU can neither send commands to control the MMW module nor transmit data to the MMW module because there is no relevant driver. To control the MMW module working mode and network adapter Settings. We need to mount the millimeter wave drive provided by IDT company.

4.4.1 Logical of Linux driver

From the point of view of kernel, millimeter wave module is equivalent to peripheral device module. The module is controlled by the device driver subsystem of the kernel. The smooth use of the application requires the corresponding character device driver, whose detailed driver architecture is shown as Figure 4-24:

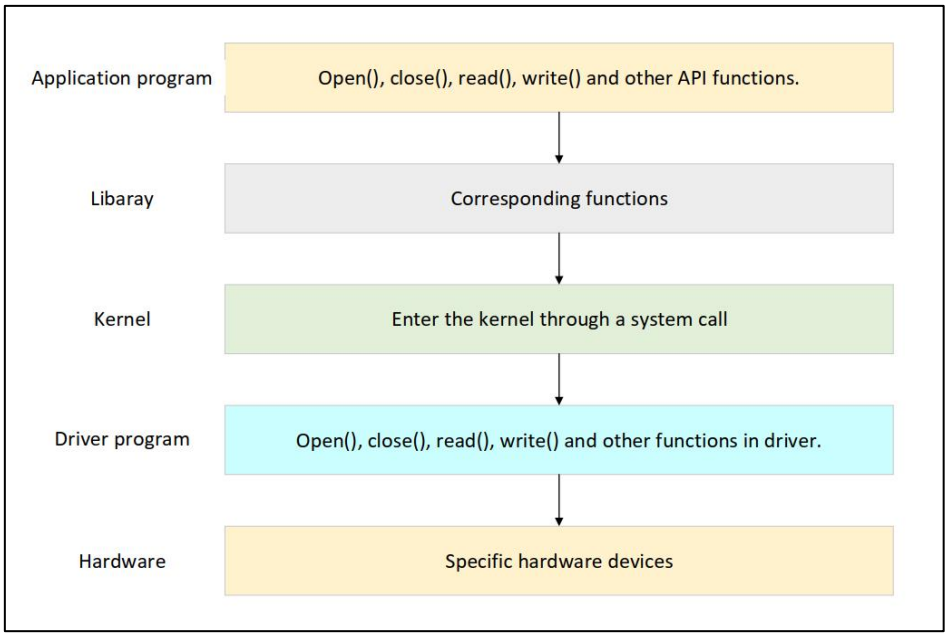


Figure 4-24: Running process of the Linux driver

It can be seen that to reasonably control the specific hardware devices, there are four levels, the first is through the user mode of the application program, by the application program to call the library functions provided by the upper interface such as open(), close() and so on. Once these interfaces are called, the corresponding library functions are used, which have permission to sink into the kernel space to call the drivers mounted in the kernel space and ultimately control the device. Obviously, to keep developers from paying too much attention to kernel scheduling and processing, to keep developers from arbitrarily accessing the kernel, the first three layers in Linux are all unified interfaces, which are abstracted. Only the driver layer performs different drivers for different hardware. So, engineer just need to mount the driver and write the related application using the application API.

4.4.2 Mount and Run of MMW

Drivers can be mounted directly in user mode, or they can be added to the appropriate version when compiling the kernel. The loading details of the driver are as Figure 4-25:

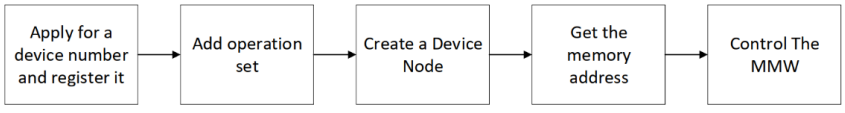


Figure 4-25: Driver mounting process

When the MMW driver starts to mount, the driver will first apply for a device id from the device driver subsystem in the kernel and register it, or apply for a fixed device ID from the system for registration. Then add millimeter wave character device (the operation set of functions communicating with millimeter wave), that is, the function and method of truly specialized operation of millimeter wave module, including communication with millimeter wave, mode change and information transmission, etc., at the same time inform the kernel of the device number and corresponding character device. A device node is created to provide users with access to the device in the form of a folder to learn about the running status and information of the device, and to store corresponding data after obtaining related information from the DTB. You also need to add device information to find millimeter wave modules that meet requirements. Finally, the PCIe address of the device tree is obtained according to the Pin-ctrl subsystem. The Pin-ctrl subsystem is used to configure the PIN according to the relevant information provided by the developer, and can also provide relevant information. Finally, the relevant millimeter wave module can be driven and controlled by character device [69].

It can also directly go to DTB to obtain the corresponding physical address according to relevant parameters, and then conduct memory mapping to obtain the virtual address that the driver can process. Finally, control these related virtual addresses to initialize and run corresponding functions. The mounted node is shown in the Figure:

```
root@localhost:~# lspci
0000:00:00.0 PCI bridge: Freescale Semiconductor Inc Device 81c0 (rev 10)
0001:00:00.0 PCI bridge: Freescale Semiconductor Inc Device 81c0 (rev 10)
0002:00:00.0 PCI bridge: Freescale Semiconductor Inc Device 81c0 (rev 10)
0002:01:00.0 Ethernet controller: Integrated Device Technology, Inc. [IDT] Device 80e8 (rev 02)
0002:01:00.1 Ethernet controller: Integrated Device Technology, Inc. [IDT] Device 80e8 (rev 02)
```

Figure 4-26: PCIE connection

```
root@localhost:/sys/bus/pci/drivers/bh2# ls
0002:01:00.0  0002:01:00.1  bind  module  new_id  remove_id  uevent  unbind
```

Figure 4-27: Node mounted with driver

```
root@localhost:/sys/bus/pci/devices# ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: sit0@NONE: <NOARP> mtu 1480 qdisc noop state DOWN mode DEFAULT group default qlen 1000
    link/sit 0.0.0.0 brd 0.0.0.0
3: wlp2p1s0f0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group default qlen 1000
    link/ether 00:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff
4: wlp2p1s0f1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group default qlen 1000
    link/ether 00:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff
5: lxcbr0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN mode DEFAULT group default qlen 1000
    link/ether 00:16:3e:00:00:00 brd ff:ff:ff:ff:ff:ff
6: virbr0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN mode DEFAULT group default qlen 1000
    link/ether 52:54:00:ef:a1:22 brd ff:ff:ff:ff:ff:ff
7: virbr0-nic: <BROADCAST,MULTICAST> mtu 1500 qdisc pfifo_fast master virbr0 state DOWN mode DEFAULT group default qlen 1000
    link/ether 52:54:00:ef:a1:22 brd ff:ff:ff:ff:ff:ff
8: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN mode DEFAULT group default
    link/ether 02:42:fd:f9:91:e0 brd ff:ff:ff:ff:ff:ff
```

Figure 4-28: The device runs properly after the driver is mounted successfully

After successfully mounting, the MMW modules on both sides can be configured using the corresponding application Settings. The configuration logic is as Figure 4-29:

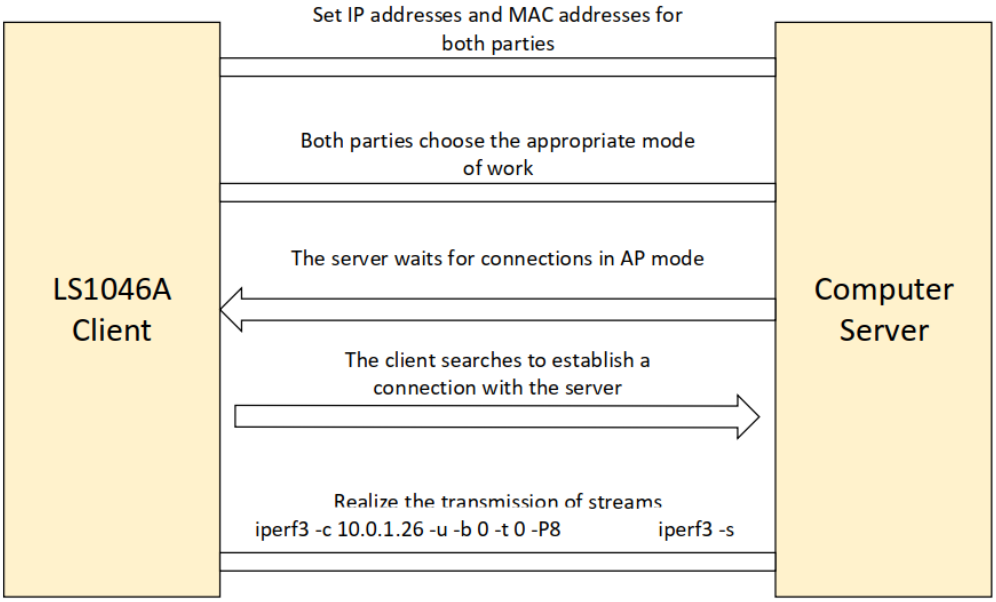


Figure 4-29: MMW program running logic

Since the MMW modules on both sides are connected wirelessly rather than wired, the application uses IP protocol to distinguish them to identify each other. That is, the two parties set different IP addresses and MAC addresses as a network card, and then pull the high level to start the MMW module.

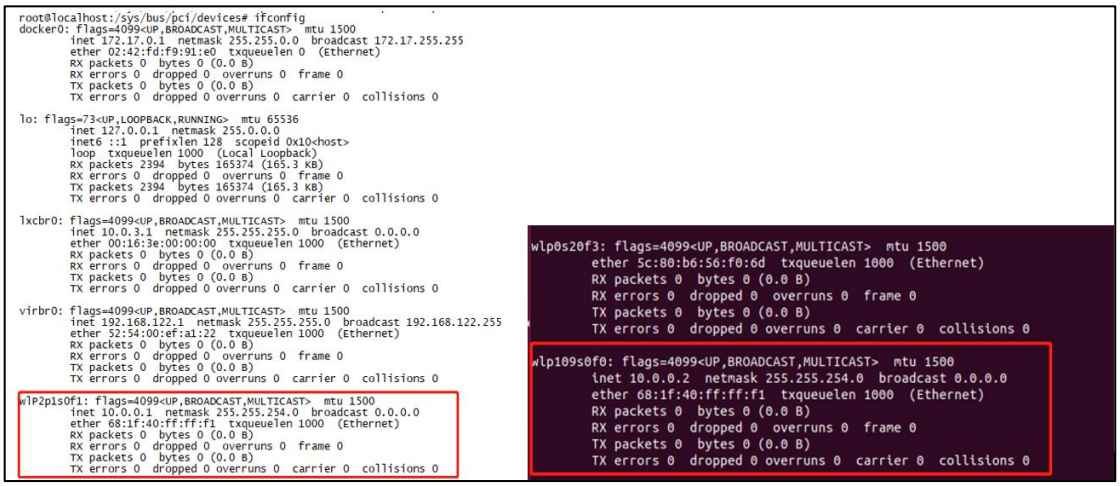


Figure 4-30: Millimeter-wave startup in the form of network adapter

After the two parties start, the RF module is not working and will not emit

electromagnetic waves because there is no corresponding working mode. Therefore, the two sides should choose the appropriate working mode. In this thesis, the millimeter wave module enters the automatic mode and automatically searches for the nearby millimeter wave signal. Rf is already working, but if both parties are sending or receiving, it will not work.

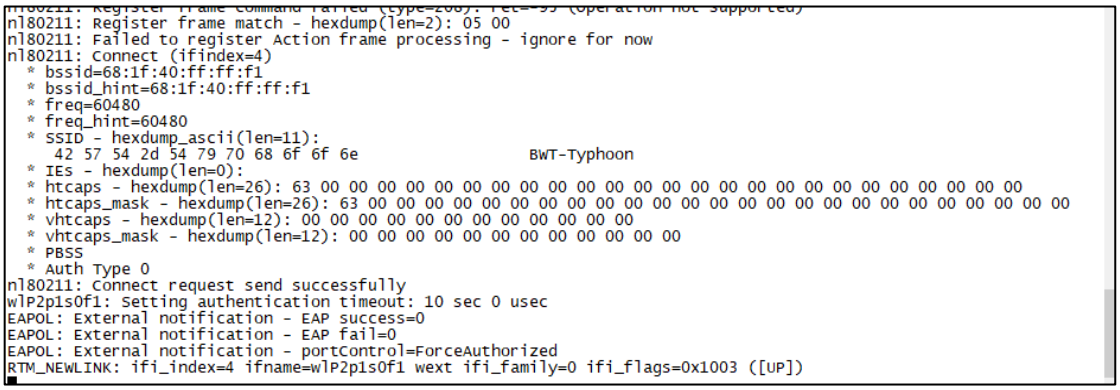


Figure 4-31: Millimeter wave scanning

One side needs to be in AP mode server state, and the other side needs to be in client state. Therefore, the working modes of the two MMW module protocols need to be set. That is, the receiver is in the AP state of the server to receive MMW requests from the corresponding IP address and MAC address. The sender is in the client search state, searches for the MMW corresponding to the IP address and MAC address, and sends a connection request.

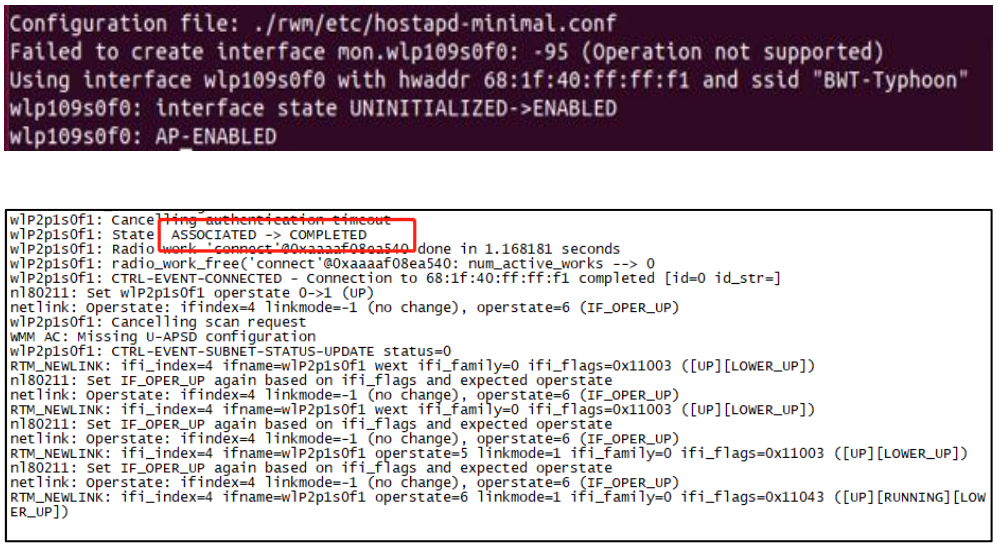


Figure 4-32: Successful connection of millimeter waves

After the connection is successful, the whole link is truly connected. Data transmission

can only be carried out between two millimeter-wave modules, and the successful connection is shown as Figure 4-23:

4.5 Flow-pulling and rendering

When the server has obtained the pushed video stream from the home host and the connection between the receiving host and NPU has been established through wireless millimeter wave. Pull stream can be realized on the software, and the video can be wirelessly pulled to the receiving host. However, even if it is not enough to pull the information to the local, the video stream needs to be unpacked and decapsulated by the protocol, as well as the corresponding audio and video decoding. After decoding, orderly output is also required, rather than unified compression like pushing the stream. Therefore, audio and video output should not only be performed at a normal rate, but also video rendering and audio and video synchronization. This section will explain in detail the specific flow of pull stream, audio and video synchronization and video rendering.

Wherein, the video pull stream and render playback are deployed on I5-7300U PC. QT provides FFmpeg and OpenGL interfaces to coordinate the decoding of FFmpeg video stream and cooperate with OpenGL for video rendering and synchronous playback. Its block diagram is shown as Figure 4-33:

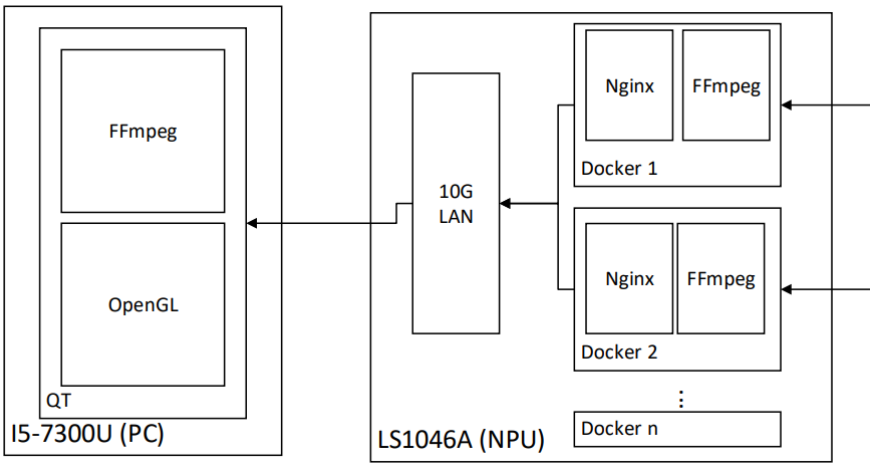


Figure 4-33: Pull flow and render block diagrams

4.5.1 Logical of Flow-pulling

FFmpeg cannot meet all the requirements because it can only pull the video stream data

of the server, including video and audio frames and corresponding attributes. Qt is also required for the integration of each process to reasonably output the data in order, but also need to use OpenGL rendering to display the real picture, so the whole detailed process is shown as Figure 4-34:

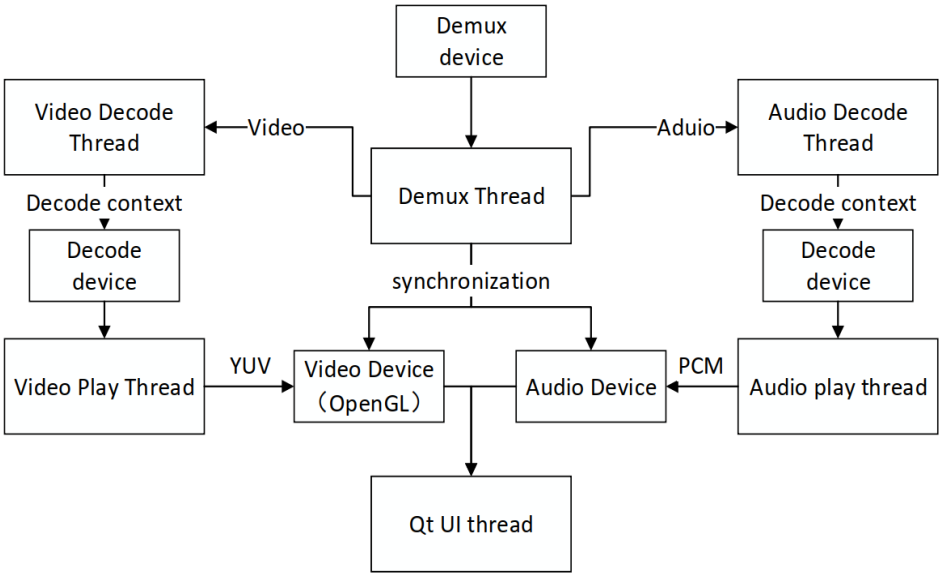


Figure 4-34: Flow chart of a pull flow

As can be seen from the Figure, we run five threads respectively, namely, the unpacking thread, the audio decoding thread, the video decoding thread, the video playing thread and the audio playing thread. First, the unpacking thread is responsible for pulling the stream of the determined server URL, and unpacking the audio and video respectively after obtaining relevant data, and unpacking the RTMP protocol and FLV encapsulation. And the audio and video packages are thrown to the corresponding two threads. The corresponding audio thread decodes the audio package and resamples the corresponding PCM audio to send the corresponding audio frame to the audio player thread. The video decoder thread decodes the H.264 code and sends the video frame to the video player thread for processing. The video player thread called the OpenGL tool, through the tool to render the video stream, the FFmpeg video frame using material extraction and then through Qt display to the home computer. The audio player thread will use Qt's audio tools to write Qt'S IO cache, and Qt will perform the following audio output. All threads are inherited from Qt, and the five threads are managed through Qt.

All of these threads are managed by a total thread. In addition to learning about packages, in order to ensure the synchronization of audio and video output, Qt records

two audio and video timelines to ensure that the audio thread and video thread can be synchronized; If one side when one side of the cache time axis is too fast, then stop sending corresponding frame to the corresponding decoder and cache, that is, control the corresponding decoding thread stop work, do not let it quickly output. When the timelines of the two parties are similar, the corresponding frame codes will continue to be sent. At this point, when the audio and video frames are obtained, the master thread will wake up the audio and video player threads respectively. Let the audio player thread resampling and output; Let the video player thread render and output.

However, due to the use of H.264 encoding, the particularity of B frame and P frame makes the video playing sequence and decoding sequence different, the reasons are as Figure 4-35:

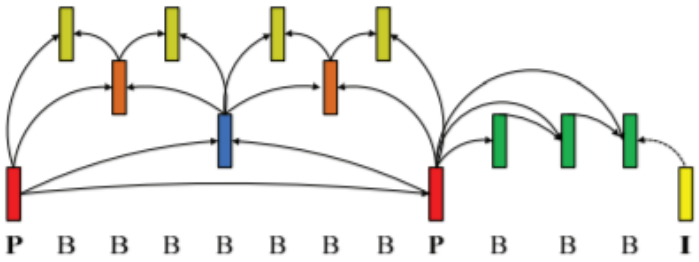


Figure 4-35: SCHEMATIC diagram of GOP decoding

As shown in the Figure, since B frame is the video frame generated for the changes of the previous frame and the next frame, P frame is the video frame generated for the changes of the previous frame. If it is a decoding P frame is fine, just need to decode the previous frame, the next frame for the corresponding processing can be. However, to save more space for video coding, it is better to compare before and after, which leads to multiple B-frames. The decoding of one B frame then requires the decoding of the next frame until the next frame is P frame. Therefore, the decoding sequence is often different from the display sequence. In general, frame I is decoded first, then P frame is decoded by frame I as the reference frame, and then B frame is decoded in sequence with the help of frame I and P. B frame will find out the predicted value and motion vector of pixels relative to frame I and P, and calculate the predicted sum by determining the position of B frame in this frame. To obtains the full B frame [70].

4.5.2 FFmpeg of Flow-pulling

This section describes the concrete implementation of FFmpeg unpacking, audio and

video decoding and simple preprocessing before audio output in the whole process of pulling stream. The specific process is as follows:

First, FFmpeg will start the unpacking thread, initialize the environment, including the initialization of the unpacking, and notify the audio decoder thread and video decoder thread respectively to initialize the environment, send various parameters of audio and video, including video resolution bit rate, audio and video size, etc. After initialization, the entire video stream will be directly received. When receiving a frame, space will be allocated for the frame and stored in the corresponding position. Then rely on the decapsulated environment to read decapsulated FLV package and RTMP protocol, and adjust PTS and DTS to prepare for Qt audio and video synchronization. Then the classification of each frame that has been unsealed is distinguished. If it is an audio frame, the frame is sent to the audio decoding thread. If it is a video frame, the frame is sent to the video decoding thread. At the same time, the time stamp of the audio decoder thread and video decoder thread is always checked, namely DTS. Once the speed of one side is found to be too fast, the data of this side will be buffered and stop sending data to the corresponding decoder thread.

When audio decoding thread receives the audio frame, due to the received from decapsulation passed parameters, it is based on the parameters passed to find their own type of file, create a decoding environment configuration context, and open the decoder environment; Also open the resampling tool, know the output channel, format sample rate and so on to complete initialization. Then, in the form of channel, the obtained audio frame is transmitted to the corresponding audio decoder, and the corresponding frame is received, which lays a foundation for the subsequent audio frame playback.

When the video decoding thread receives the video frame, it also obtains the parameters passed by the wrapper. According to the relevant parameters, the decoder finds the corresponding video code type, creates the corresponding decoder environment, and opens the decoder channel. Later, at runtime, the video frame is also sent to the video decoder through the channel, which lays the foundation for the subsequent video rendering.

4.5.3 Qt of UI and Audio

When FFmpeg decodes audio and video streams, it needs an output control platform,

including the display of pictures and audio protocol processing and output, as well as the synchronization of the two. This section will describe Qt in the UI interface, screen display thread and audio output thread concrete implementation, its specific process is as follows:

First need to implement the relevant UI through Qt, a certain UI layout:

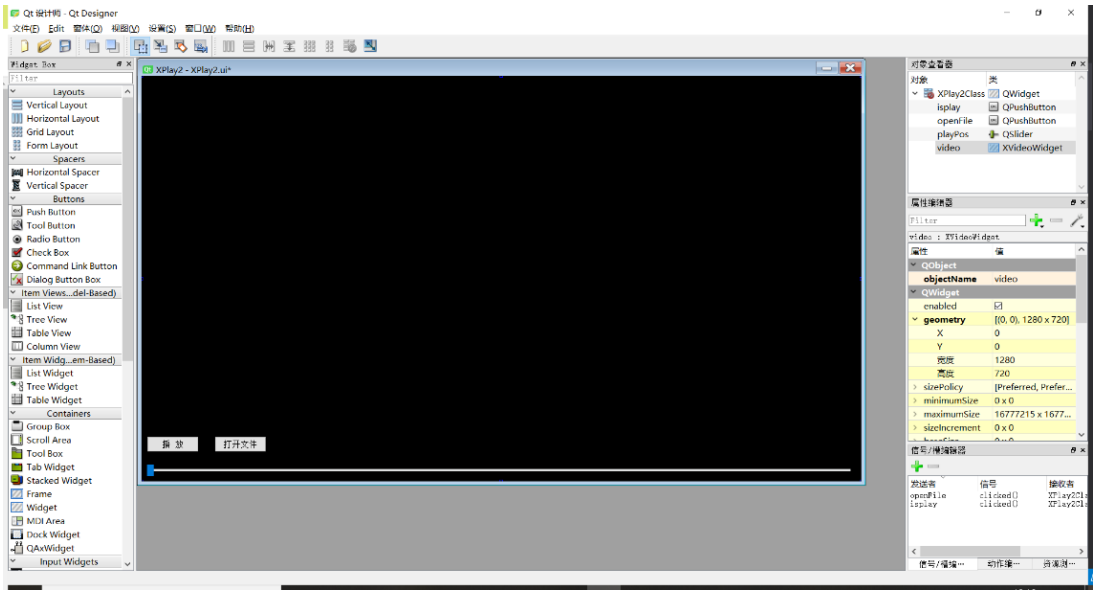


Figure 4-36: Qt design screen

This includes a pause button, a URL/ file open button, a drag bar, and an adequate video display interface. At the same time through the function to achieve pause button, URL open button and so on. The UI interface can then be inherited and launched by the main function as a parent class.

Success starts, because receives the video streaming, decapsulation threads and decoder threads started work, will start the Qt audio playback audio decoder threads, inform the data of the audio frame such as sampling rate, channel and the types of audio PCM, for example, the audio playback thread will initialize the environment, and open the broadcast channel. When the audio decoder thread sends resampled audio to the audio player, the audio player writes the audio frame to the cache and starts playing.

And video decoder, due to the Qt starts successfully, access to the video stream, we get the UI passed video interface, so after receiving data from the video decoding thread, decoder can be directly according to the video frame of data initialization video interface, video player at this time only need to render the video frame on the video

screen.

4.5.4 OpenGL of Rendering

Rendering on Qt video interface, need to use Qt support OpenGL, that is, open graphics library to carry out the corresponding program design, to ensure that when the video frame is transmitted to the video processing thread, you can call the GPU to calculate through material and extract the corresponding data and finally display to the home computer. The specific implementation process is as Figure 4-37 [71]:

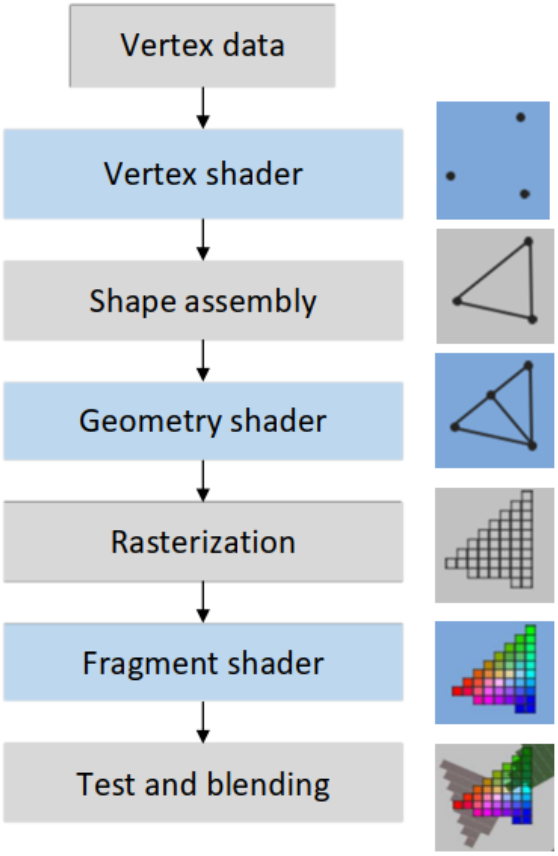


Figure 4-37: OpenGL running process

The first step is to provide Some processing strategies to OpenGL, namely scripts for vertex shaders and scripts for slice shaders [72]. Then OpenGL initialization, the transmission of the vertex script and slice shader script and OpenGL number binding, and passed in the corresponding vertex coordinate data and material coordinate data. You can activate the script at this point. Finally, the address of the entry material of the script is obtained, to facilitate the transfer of value to the material in the future, carry

out convenient color collection, complete the initialization of the two shaders.

Secondly, to obtain the corresponding video frame information from FFmpeg, the corresponding space is also needed to store YUV420p video frame data, which lays a foundation for the subsequent value transmission. In addition, to pass materials to the slice script, the corresponding material areas need to be set up separately. Therefore, the storage area of the original video frame and the area of YUV three-layer material are initialized respectively.

Finally, once running, we bind the material area directly to the original storage area, and then bind the material area to the material entry address of the slice script. Complete data entry.

When a video frame flows into the video player, it is divided into three YUV sections, first stored in the original video frame storage area, and then the vertex shader script will start running. According to the incoming vertex data through the vertex shader script into a coordinate system that OpenGL can process, and in a certain area to draw points. The primitives are drawn according to the coordinate system to form unit shapes such as triangles and rectangles and so on. Rasterization then pixelates these geometric images. Finally, the material script will collect the data of the original video according to the material entry provided by the program, and then run the material script to transform and render the data, and finally display it on the Qt UI video interface.

Among them, since we use YUV420P data frame processing method, and OpenGL only supports RGB rendering, so our slice script needs to perform the corresponding YUV to RGB matrix calculation.

According to the international definition of HD, the relation between RGB and YUV tricolor is [73] :

R = Y + 1.402Pr

G = Y - 0.344Pb - 0.792Pr

B = Y + 1.772Pb

So, the final matrix output looks like this:

$$[R \quad G \quad B] = [Y \quad U \quad V] \begin{bmatrix} 1 & 1 & 1 \\ 0 & -0.34465 & 1.772 \\ 1.40283 & -0.79260 & 0 \end{bmatrix}$$

After this final conversion, OpenGL can render the video smoothly, and the video

player can successfully transfer the video frame to the screen. The running screen of its program is as Figure 4-38:

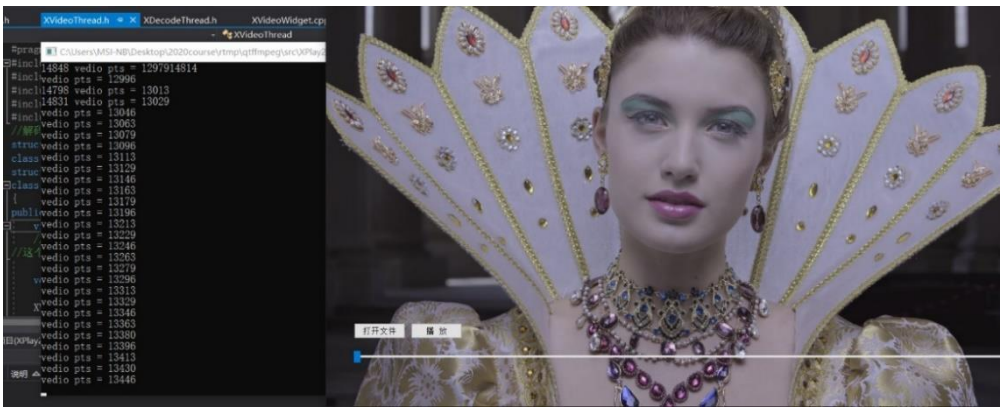


Figure 4-38: Qt running process

4.6 Summary

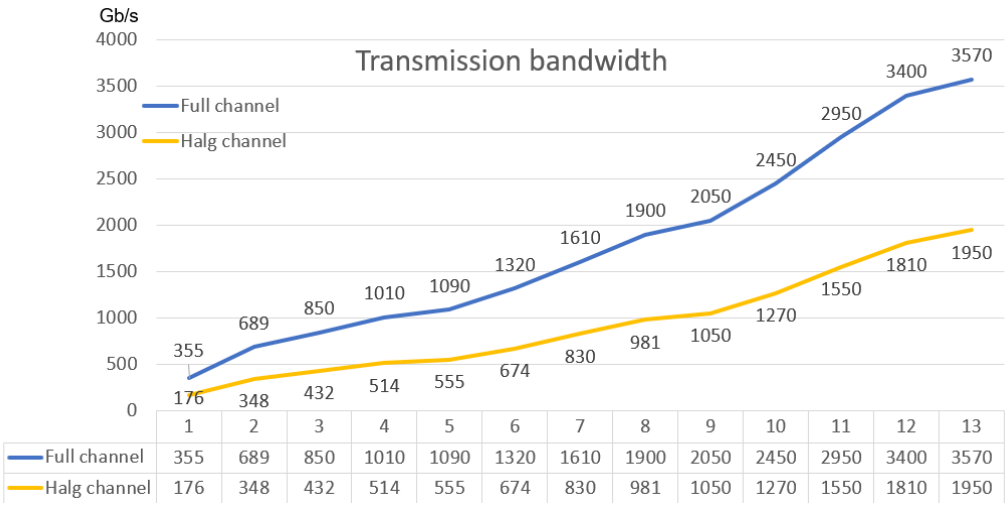
This chapter expounds the operation of Linux system, software flow, server forwarding and software display from four aspects. In Linux system, it mainly introduces the operating principle of boot program, kernel device tree and root file system as well as the related work including mode selection, program transplantation, driver loading and recompiling and burning. The pull flow part of the software mainly explains the operation principle of OpenCV and the software logic design combined with FFmpeg, including the encoding of audio and video and the encapsulation of video stream and the corresponding network protocol. The forwarding part of the server mainly introduces the software migration logic design of the server, the operating principle of the RTMP server and the Docker container containing Nginx that is transplanted to maintain the stability of the server. Millimeter wave drive part is the software support for server forwarding, mainly introduces the operating principle of the drive and the operating logic of millimeter wave drive in wireless transmission system. Finally, there is the pull stream section of the software, which explains the software logic and thread management of FFmpeg and Qt's overall pull stream and playback. Including FFmpeg video pull flow, unsealing and decoding software logic design, Qt UI design, audio playback and video playback based on OpenGL software design. Complete the software design of the whole wireless system and realize the HD video transmission based on the millimeter wave wireless system.

5 Tests and Results

Through the previous hardware and software design, this thesis obtained a complete HD video wireless transmission system based on millimeter wave. Under this transmission system, this chapter conducted pressure test and HD video transmission test respectively. The pressure test is to verify that the transmission rate of the whole millimeter-wave transmission link can reach the acceptable value, iperf3 is used to test the bandwidth rate of all links, and WebBench is used to test the load capacity of Nginx on NPU. The transmission of HD video is to show the feasibility of HD video transmission in millimeter wave wireless transmission system, including the lowest 1080p image quality to VR 360° 8K image quality, and the transmission rate can reach 135Mb/s.

5.1 Channel influence on the transmission system

Because the transmission system has two full-duplex transmission channels, the transmission process can be set as a non-interference half-channel, or full channel. The former reduces signal interference by sharing frequency band resources, but sacrifices part of network throughput. The latter ensures the maximum bandwidth of the transmission system and reduces interference by setting the frequency of the two channels at the maximum degree of separation of their midpoint frequencies. Its half-channel and full-channel transmission performance is shown as Figure 5-1:



	1	2	3	4	5	6	7	8	9	10	11	12	13
Full channel	355	689	850	1010	1090	1320	1610	1900	2050	2450	2950	3400	3570
Halg channel	176	348	432	514	555	674	830	981	1050	1270	1550	1810	1950

Figure 5-1: Transmission rate comparison between full channel and half channel

Due to Full Channel's larger bandwidth and less interference in the indoor environment, the following tests adopt Full Channel's progressive performance on test and video transmission.

5.2 Environment influence on the transmission system

Considering the transmission loss of millimeter wave, this thesis firstly selects different indoor test environments to test the impact of the environment on the transmission system, which can be divided into two environments: open environment and narrow environment. The layout of its transmission is as follows.

(1) Open, fully accessible environment. The transmission system is deployed on a completely open site with no shielding in the middle and visually visible devices at both ends.

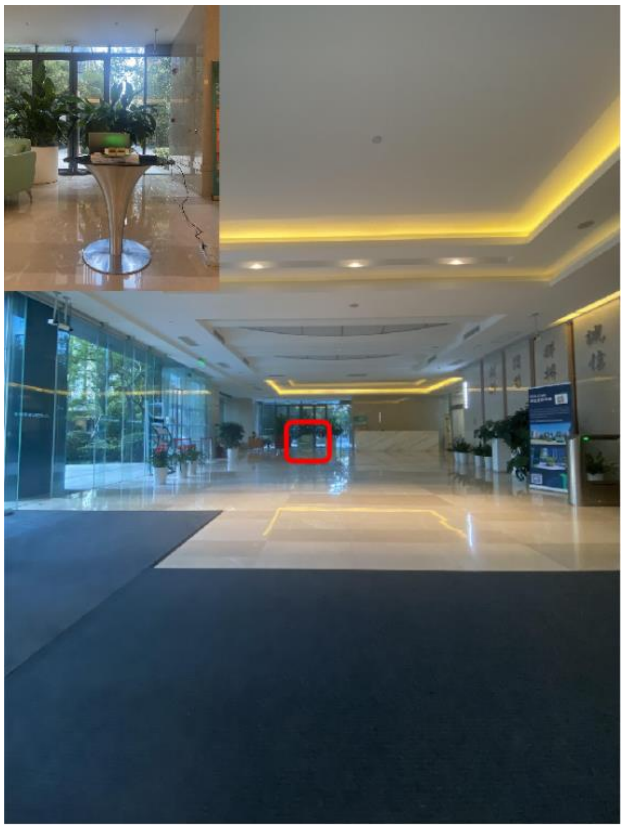


Figure 5-2: Open environment

(2) A small, closed environment. The transmission system is deployed along a narrow corridor, with no shelter in the middle but limited transmission on both sides.

Figure 5-3: Narrow environment

According to the loss ability of millimeter-wave transmission in the atmosphere, the following experimental results are the transmission bandwidth under the guarantee of packet loss rate of 0. And this environment has only two transmission systems, only one signal channel to achieve simple point-to-point transmission. The results of the tests included tests in different environments at different distances. The distance ranges from 5m to 50m (5m, 10m, 15m, 20m, 25m, 30m, 40m, 50m).

First, open indoor scenes get the best performance. When the distance is 25m, it can still reach a large bandwidth. That is, single-channel 1.79Gbps, dual-channel 2.45Gbps, and the packet loss rate is 0. The result is shown below:

```
[ 17]   1.00-2.00   sec  36.7 MBytes   308 Mbits/sec  0.136 ms  0/26578 (0%)
[ 19]   1.00-2.00   sec  36.7 MBytes   308 Mbits/sec  0.023 ms  0/26573 (0%)
[SUM]   1.00-2.00   sec   294 MBytes  2.46 Gbits/sec  0.075 ms  0/212691 (0%)
- - - - - - - - - - - - - - - - - - - - - - - - -
[  5]   2.00-3.00   sec  36.8 MBytes   309 Mbits/sec  0.018 ms  0/26670 (0%)
[  6]   2.00-3.00   sec  36.8 MBytes   309 Mbits/sec  0.023 ms  0/26660 (0%)
[  9]   2.00-3.00   sec  36.8 MBytes   309 Mbits/sec  0.020 ms  0/26664 (0%)
[ 11]   2.00-3.00   sec  36.8 MBytes   309 Mbits/sec  0.021 ms  0/26654 (0%)
[ 13]   2.00-3.00   sec  36.8 MBytes   309 Mbits/sec  0.020 ms  0/26643 (0%)
[ 15]   2.00-3.00   sec  36.8 MBytes   308 Mbits/sec  0.022 ms  0/26628 (0%)
[ 17]   2.00-3.00   sec  36.7 MBytes   308 Mbits/sec  0.025 ms  0/26594 (0%)
[ 19]   2.00-3.00   sec  36.7 MBytes   308 Mbits/sec  0.022 ms  0/26598 (0%)
[SUM]   2.00-3.00   sec   294 MBytes  2.47 Gbits/sec  0.021 ms  0/213111 (0%)
```

Figure 5-4: Transmission rate of 25meter

However, in the narrow corridor, the transmission performance is very unstable when the distance is 15m, and even stable wireless connection cannot be established. Even

if the distance between the two devices is only 5m, the signal is still unstable, indicating that the device must be used in an open area. The transmission bandwidth of 5-50M in the open area is shown as Figure 5-5:

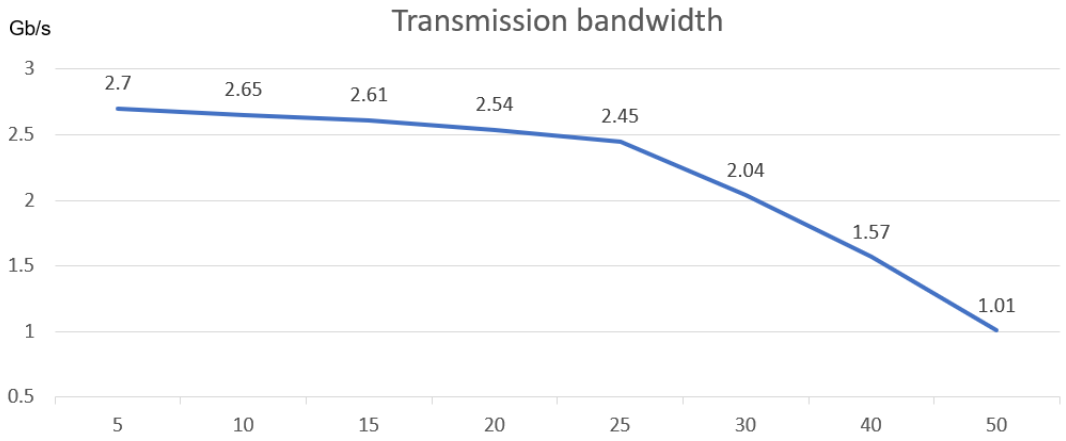


Figure 5-5: Transmission bandwidth in an open environment from 5 to 50 meter

5.3 Pressure Test of Transmission System

For the whole wireless transmission system of millimeter wave, we use iperf3 software to test the overall bandwidth of the link, and use WebBench to test the network load capacity of Nginx. The relevant introduction of the two is as follows:

Iperf3: The software is a network performance testing tool, with test related link TCP and UDP broadband performance, not only that, he can also provide a variety of parameters of the UDP feature performance requirements with testing. In addition to reporting bandwidth, you can also test for jitter and packets, among other things. Iperf3 uses slightly different logic when testing TCP and UDP. For TCP, one party functions as the server and one party functions as the client. After the two parties establish a connection through a three-way handshake, the client sends a packet of appropriate size to the server and records the sending time. The server keeps track of how much data was received and how long it took to complete the connection. At the same time, iperf3 gradually adjusts the TCP window size until it reaches maximum transport performance. It can obtain the maximum bandwidth with the maximum performance through its own data and the time spent; With UDP, the client can determine the rate at which data is sent and the size of packets. Since UDP packets can have unique IDS, the server can calculate the bandwidth and determine the packet loss rate based on this ID number.

You can even calculate delay jitter by getting the client's associated timestamp to get more experimental data.

The code of Iperf3 is based on C++, which is relatively simple in logic. It mainly includes the realization of sending and receiving classes of client and server as well as related information output classes, including bandwidth reporting, packet loss rate and delay jitter, etc. The Client establishes Socket connection through the Client class and sends data through the Speaker class. The Server receives connections through the Server class and receives and processes related data through the Listener class. Because of two-way testing, the connection between client and client is two-way, making it easier for developers to test bandwidth for both sides.

WebBench: WebBench is an excellent performance test tool for the Web on Linux. It focuses on the performance of the server and simulates thousands of people accessing the server. In addition to measuring the network bandwidth, you can also test the maximum amount of access that the server can tolerate. At its core, WebBench, as the parent process, creates multiple child processes, each of which performs virtual loop access to the Web server as required by the tester or by default. The parent process can communicate with the child process through the pipe, and the child process will inform the parent process of the relevant information after a certain number of requests, including the success rate of access and so on. The parent process will extract the information from the pipeline and synthesize it. After completing all the tests, the parent process will display the information to the tester to inform the test result.

In terms of code implementation, this program only has more than 600 lines, most of which are completed by functions. First, the emergence of the parent process requires a series of URL detection and the construction of HTTP request messages for trial connection. When the Web server is reachable, create a child process, and communicate with the child process at any time, for information integration. The child process does a simple loop. That every child has an own timer, before the timer didn't go off, it would have been continuous sends a request message, if no response then disconnects for the next visit request, once the timer goes off, subroutine will run over, and through the channel send access to the parent of success rate and related information.

5.3.1 Speed test of RJ45

For the link of home host and 10 Giga-Bit network port, this thesis installs corresponding Iperf3 test tool in home host and NPU respectively. The bandwidth test for 10 Giga-Bit network port and DPAA hardware circuit is carried out, and the test results are as Figure 5-6:

```
root@localhost:~#  iperf3 -c 192.168.2.181 -i 5 -t 60
Connecting to host 192.168.2.181, port 5201
[  4] local 192.168.2.182 port 38928 connected to 192.168.2.181 port 5201
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-5.00   sec  4.18 GBytes  7.18 Gbits/sec    0    332 KBytes
[  4]   5.00-10.00  sec  4.19 GBytes  7.20 Gbits/sec    0    414 KBytes
[  4]  10.00-15.00  sec  4.23 GBytes  7.26 Gbits/sec    0    414 KBytes
[  4]  15.00-20.00  sec  4.23 GBytes  7.26 Gbits/sec    0    414 KBytes
```

Figure 5-6: Hardware bandwidth of 10 MBIT/s network ports and DPAA

As a 10Gb/s network port, the transmission rate can reach 7.20Gbit/s, ensuring the reliability of the transmission rate on the NPU link at home.

5.3.2 Speed test of PCIE

On the forwarding side, two aspects are mainly measured: one is the processing bandwidth speed of THE NPU and the receiver's home host server software; the other is the hardware link bandwidth of the PCIE and millimeter wave between THE NPU and the receiver. Since the pressure test on the software must be based on the hardware test link, the test on the Iperf3 progressive hardware link is first carried out, and the test results are as Figure 5-7:

```
[  5]  30.00-30.80  sec  13.3 MBytes   139 Mbits/sec  0.057 ms  8191/17848 (46%)
[  6]  30.00-30.80  sec  13.3 MBytes   139 Mbits/sec  0.053 ms  8206/17840 (46%)
[  8]  30.00-30.80  sec  13.3 MBytes   139 Mbits/sec  0.056 ms  8191/17854 (46%)
[ 10]  30.00-30.80  sec  13.3 MBytes   139 Mbits/sec  0.060 ms  8181/17845 (46%)
[ 12]  30.00-30.80  sec  13.3 MBytes   139 Mbits/sec  0.066 ms  8201/17855 (46%)
[ 14]  30.00-30.80  sec  13.3 MBytes   138 Mbits/sec  0.069 ms  8235/17841 (46%)
[ 16]  30.00-30.80  sec  13.3 MBytes   139 Mbits/sec  0.039 ms  8221/17841 (46%)
[ 18]  30.00-30.80  sec  13.3 MBytes   139 Mbits/sec  0.046 ms  8187/17805 (46%)
[SUM]  30.00-30.80  sec   106 MBytes  1.11 Gbits/sec  0.056 ms  65613/142729 (46%)
```

Figure 5-7: CPU 1.0 bandwidth

At the beginning of the test, the transfer rate was only 1.1Gbit/s, which was caused by the PCIE frequency was not high enough:

```
root@localhost:/home/rwm/rwm6050/bin# lspci -n -d 111d:80e8 -vvv |grep -i width
                LnkCap: Port #0, Speed 5GT/s, Width x2, ASPM L0s L1, Exit Latency L0s <512ns, L1 <2us
                LnkSta: Speed 2.5GT/s, Width x1, TrErr- Train- SlotClk- DLActive- BWMgmt- ABWMgmt-
                LnkCap: Port #0, Speed 5GT/s, Width x2, ASPM L0s L1, Exit Latency L0s <512ns, L1 <2us
                LnkSta: Speed 2.5GT/s, Width x1, TrErr- Train- SlotClk- DLActive- BWMgmt- ABWMgmt-
root@localhost:/home/rwm/rwm6050/bin#
```

Figure 5-8: PCIE2.0 Settings

Then increase the PCIE working frequency and adjust the PCIE to 3.0 for downward compatibility:

```
0000:00:00.0 0604: 1957:81c0 (rev 10)
0001:00:00.0 0604: 1957:81c0 (rev 10)
0002:00:00.0 0604: 1957:81c0 (rev 10)
root@localhost:~# lspci -n -d 1957:81c0 -vvv |grep -i width
                LnkCap: Port #0, Speed 8GT/s, Width x1, ASPM L0s, Exit Latency L0s unlimited, L1 unlimited
                LnkSta: Speed 2.5GT/s, Width x1, TrErr- Train- SlotClk+ DLActive- BWMgmt- ABWMgmt-
                LnkCap: Port #0, Speed 8GT/s, Width x1, ASPM L0s, Exit Latency L0s unlimited, L1 unlimited
                LnkSta: Speed 2.5GT/s, Width x1, TrErr- Train- SlotClk+ DLActive- BWMgmt- ABWMgmt-
                LnkCap: Port #0, Speed 8GT/s, Width x1, ASPM L0s, Exit Latency L0s unlimited, L1 unlimited
                LnkSta: Speed 2.5GT/s, Width x1, TrErr- Train- SlotClk+ DLActive- BWMgmt- ABWMgmt-
root@localhost:~#
```

Figure 5-9: PCIE2.0 bandwidth

Its overall transmission bandwidth has been improved to some extent:

```
-----------------------------------------------------------
Accepted connection from 10.0.0.232, port 38864
[  5] local 10.0.0.1 port 5201 connected to 10.0.0.232 port 38866
[ ID] Interval           Transfer     Bandwidth
[  5]   0.00-1.00   sec   247 MBytes  2.07 Gbits/sec
[  5]   1.00-2.00   sec   246 MBytes  2.06 Gbits/sec
[  5]   2.00-3.00   sec   257 MBytes  2.16 Gbits/sec
[  5]   3.00-4.00   sec   240 MBytes  2.01 Gbits/sec
[  5]   4.00-5.00   sec   242 MBytes  2.03 Gbits/sec
[  5]   5.00-6.00   sec   245 MBytes  2.05 Gbits/sec
[  5]   6.00-7.00   sec   253 MBytes  2.12 Gbits/sec
[  5]   7.00-8.00   sec   241 MBytes  2.02 Gbits/sec
[  5]   8.00-9.00   sec   243 MBytes  2.04 Gbits/sec
[  5]   9.00-10.00  sec   258 MBytes  2.17 Gbits/sec
[  5]  10.00-10.00  sec   512 KBytes  1.61 Gbits/sec
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth
```

Figure 5-10: PCIE2.0 bandwidth after increasing the frequency

However, there is still a gap between the overall speed and the upper limit. Therefore, this thesis improves the running frequency of CPU and speeds up the processing speed of packets. The results are as Figure 5-11:

```
- - - - - - - - - - - - - - - - - - - - - - - - -
[  4]  49.00-50.00  sec  38.1 MBytes   320 Mbits/sec  4880
[  6]  49.00-50.00  sec  38.1 MBytes   320 Mbits/sec  4880
[  8]  49.00-50.00  sec  38.1 MBytes   320 Mbits/sec  4880
[ 10]  49.00-50.00  sec  38.1 MBytes   320 Mbits/sec  4880
[ 12]  49.00-50.00  sec  38.1 MBytes   320 Mbits/sec  4880
[ 14]  49.00-50.00  sec  38.1 MBytes   320 Mbits/sec  4880
[ 16]  49.00-50.00  sec  38.1 MBytes   320 Mbits/sec  4880
[ 18]  49.00-50.00  sec  38.1 MBytes   320 Mbits/sec  4880
[SUM]  49.00-50.00  sec   305 MBytes  2.56 Gbits/sec  39040
```

Figure 5-11: Rate after increasing the frequency

As you can see, the transmission rate can reach 2.5Gbit/s. Since the transmission adopts

the form of dual antenna, the transmission rate can reach 5.0Gbit/s if the transmission is carried out in two channels.

5.3.3 Speed test of Nginx

After the pressure test on all hardware is completed, the pressure test of server transfer is carried out. In this thesis, WebBench test software is installed on the home machine receiving millimeter-wave module to test the pressure of Nginx server running on NPU, and the test results are as Figure 5-12:

```
rwm@NUC232:~/rwm/webbench-1.5$ ./webbench -c 20 -t 20 http://10.0.0.1:54656/WeCh
at_20200316173606.mp4
Webbench - Simple Web Benchmark 1.5
Copyright (c) Radim Kolar 1997-2004, GPL Open Source Software.

Benchmarking: GET http://10.0.0.1:54656/WeChat_20200316173606.mp4
20 clients, running 20 sec.

Speed=672 pages/min, 51181500 bytes/sec.
Requests: 224 susceed, 0 failed.
```

Figure 5-12: Nginx stress test

The bandwidth speed that the server can process meets basic requirements. There is enough bandwidth in both software and hardware.

5.4 HD Video Testing in Transmission System

After the pressure test proves that the bandwidth of the whole wireless transmission system meets the requirements, the real application scenario test is needed. The tests in this section cover 1080p general video transmission to 8K video transmission, demonstrating the feasibility of HD wireless video and surveillance using MMW. At the same time, to show the feasibility of VR in millimeter wave, 8K 360° VR video resources are added.

5.4.1 Test result in 1080p video

The following Figure shows the relevant images and data when transmitting 1080p

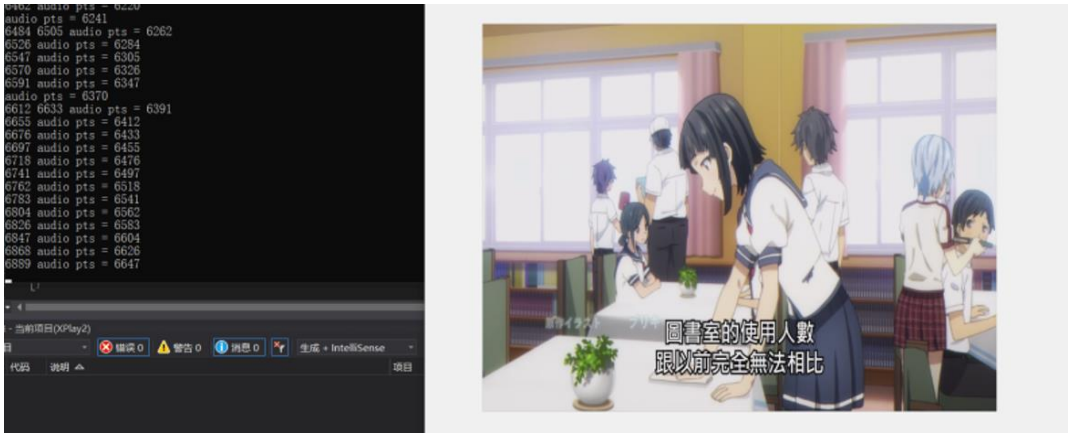


Figure 5-13: Received 1080p screen

The corresponding video data and transmission rate are as Figure 5-14:

```
Metadata:
  major_brand     : isom
  minor_version   : 512
  compatible_brands: isomiso2avc1mp41
  encoder         : Lavf58.29.100
Duration: 00:24:00.15, start: 0.000000, bitrate: 1261 kb/s
  Stream #0:0(und): Video: h264 (High) (avc1 / 0x31637661), yuv420p, 1920x1080 [SAR 1:1 DAR 16:9], 1062 kb/s, 23.98 fps
, 23.98 tbr, 11988 tbn, 47.95 tbc (default)
  Metadata:
    handler_name    : VideoHandler
  Stream #0:1(und): Audio: aac (LC) (mp4a / 0x6134706D), 48000 Hz, stereo, fltp, 192 kb/s (default)
  Metadata:
    handler_name    : SoundHandler
```

Figure 5-14: Received data of 1080p screen

5.4.2 Test result in 2k video

Similarly, when transferring 2K, the relevant images and data did not lag.

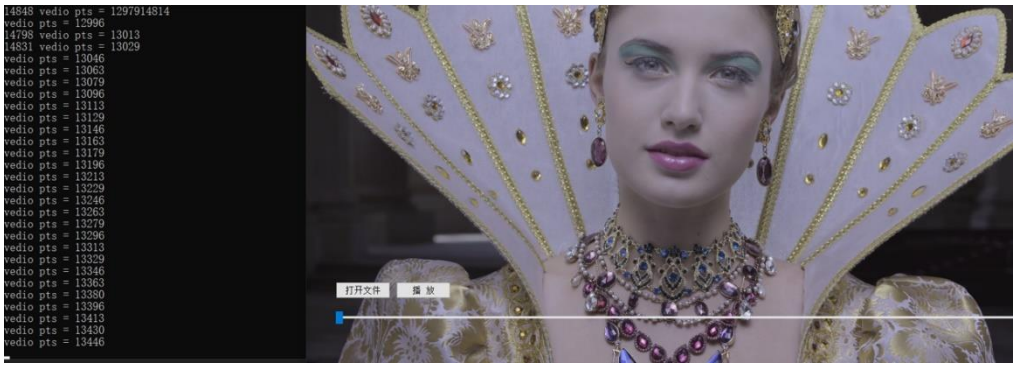


Figure 5-15: Received 2K picture

The related video transmission data and rate are shown as Figure 5-16:

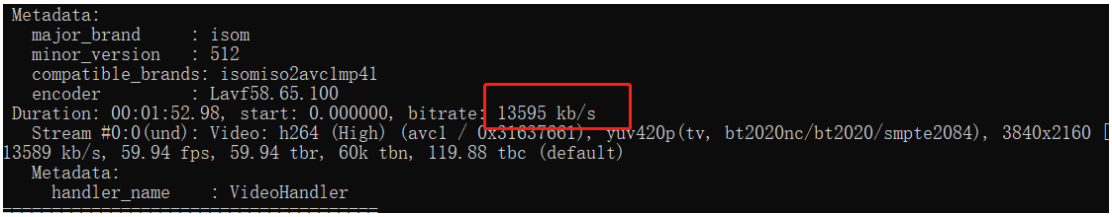


Figure 5-16: Received data of 2K picture

5.4.3 Test result in 8k video

When transmitting 8K video, the screen is as Figure 5-17:

Figure 5-17: Received 8K picture

The relevant transmission rates and data are as Figure 5-18:

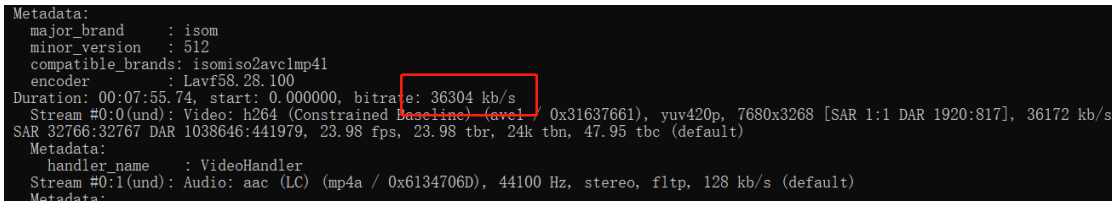


Figure 5-18: Received data of 8K screen

5.4.4 Test result in 8k 360°VR video

When transferring VR 8K video, the graphics display was a little strange because there was no VR device:

Figure 5-19: Received 8K VR screen

But it has a transmission rate of 137Mb/s:

```
  Metadata:
    major_brand     : isom
    minor_version   : 512
    compatible_brands: isomiso2avc1mp41
    encoder         : Lavf58.28.100
  Duration: 00:02:09.99, start: 0.000000, bitrate: 137035 kb/s
    Stream #0:0(und): Video: h264 (Constrained Baseline) (avc1 / 0x31637661), yuv420p, 8192x4096 [SAR 1:1 DAR 2:1], 1369
31 kb/s, 24 fps, 24 tbr, 12288 tbn, 48 tbc (default)
    Metadata:
      handler_name    : VideoHandler
    Side data:
      stereo3d: 2D
      spherical: equirectangular (0.000000/0.000000/0.000000)
    Stream #0:1(und): Audio: aac (LC) (mp4a / 0x6134706D), 44100 Hz, stereo, fltp, 128 kb/s (default)
    Metadata:
      handler_name    : SoundHandler
=====================================
```

Figure 5-20: Received data of 8K VR screen

To test the larger bandwidth of the data, we improved the bit rate of the video. By receiving picture quality information, the bit rate was increased to about 1Gb/s. The data can be transmitted normally, but the picture quality does not change much.

```
[libx264 @ 0000021d2587dc00] 264 - core 161 - H.264/MPEG-4 AVC codec - Copyleft 2003-2020 - http://www.videolan.org/x264.html - options: cabac=
1 ref=3 deblock=1:0:0 analyse=0x3:0x113 me=hex subme=7 psy=1 psy_rd=1.00:0.00 mixed_ref=1 me_range=16 chroma_me=1 trellis=1 8x8dct=1 cqm=0 dead
zone=21,11 fast_pskip=1 chroma_qp_offset=-2 threads=18 lookahead_threads=3 sliced_threads=0 nr=0 decimate=1 interlaced=0 bluray_compat=0 constr
ained_intra=0 bframes=3 b_pyramid=2 b_adapt=1 b_bias=0 direct=1 weightb=1 open_gop=0 weightp=2 keyint=250 keyint_min=24 scenecut=40 intra_refre
sh=0 rc_lookahead=40 rc=abr mbtree=1 bitrate=1049739 ratetol=1.0 qcomp=0.60 qpmin=0 qpmax=69 qpstep=4 ip_ratio=1.40 aq=1:1.00
Output #0, mp4, to 'output.mp4':
  Metadata:
    major_brand     : isom
    minor_version   : 512
    compatible_brands: isomiso2avc1mp41
    encoder         : Lavf58.45.100
    Stream #0:0(und): Video: h264 (libx264) (avc1 / 0x31637661), yuv420p, 8192x4096 [SAR 1:1 DAR 2:1], q=-1--1, 1049739 kb/s, 24 fps, 12288 tbn
, 24 tbc (default)
    Metadata:
      handler_name    : VideoHandler
      encoder         : Lavc58.91.100 libx264
    Side data:
      cpb: bitrate max/min/avg: 0/0/1049739000 buffer size: 0 vbv_delay: N/A
      stereo3d: 2D
      spherical: equirectangular (0.000000/0.000000/0.000000)
    Stream #0:1(und): Audio: aac (LC) (mp4a / 0x6134706D), 44100 Hz, stereo, fltp, 128 kb/s (default)
    Metadata:
      handler_name    : SoundHandler
      encoder         : Lavc58.91.100 aac
frame=  802 fps=1.2 q=5.0 size= 4136448kB time=00:00:33.85 bitrate=1000918.3kbits/s speed=0.0508x
```

Figure 5-21: Received data of 8K-higher rate VR screen

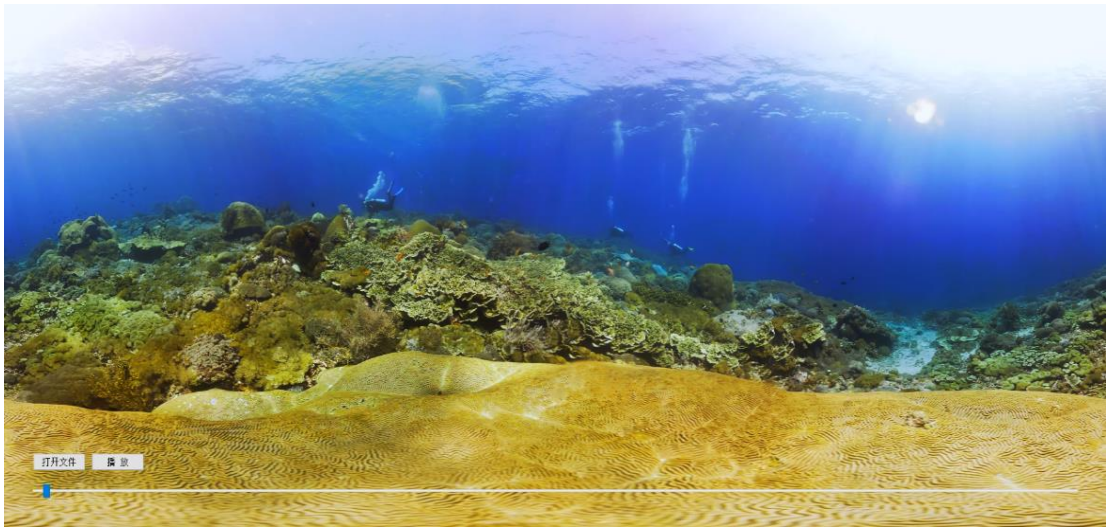


Figure 5-22: Received 8K-higher rate VR screen

To make full use of the transmission bandwidth of the system, docker was used as a container in the experiment to support multichannel transmission, and the three-channel video encoding and sending program was run in NPU, so that the bandwidth used by the video reached 3.1GB /s. The data results are shown as Figure 5-23:

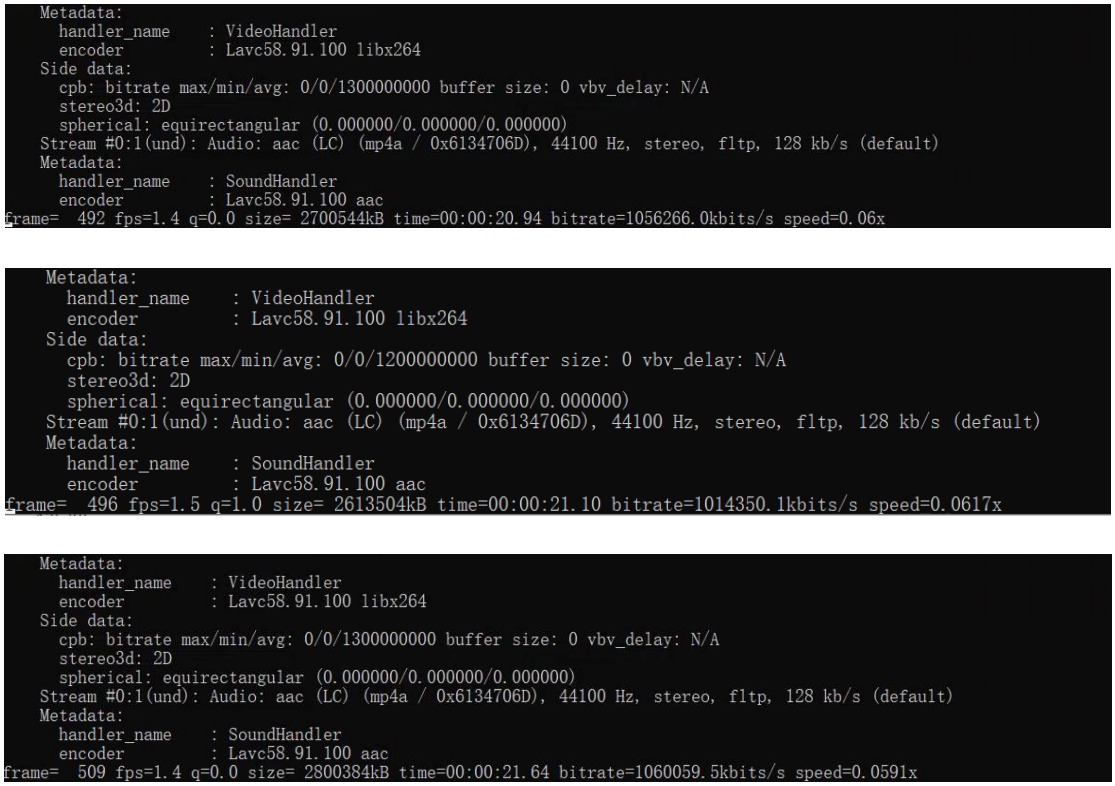


Figure 5-23: channels 8k video transmission rate

Below are the transmission rates for each video stream. As the quality of the video

continues to rise, so does the corresponding bit rate. The system can still transmit video at a rate of 1Gb/s.

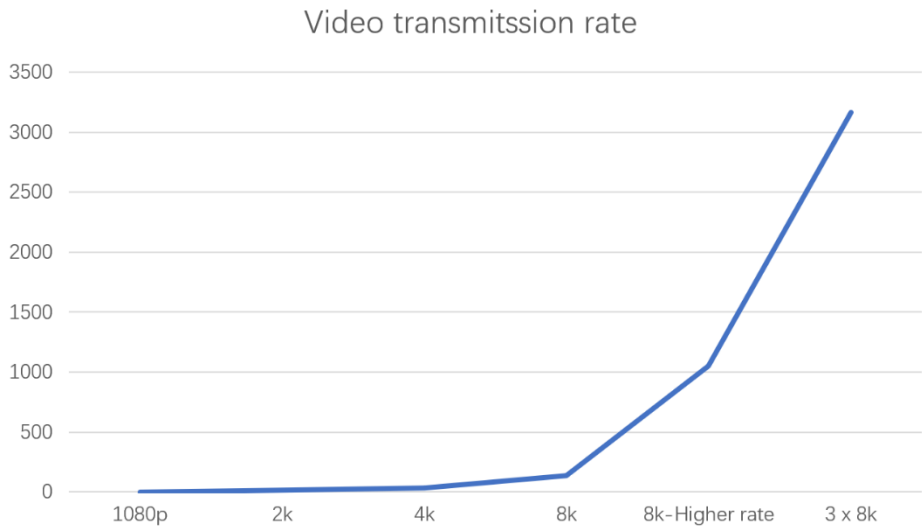


Figure 5-24: Video transmission rate

5.5 Summary

Through the above pressure test and actual test, this thesis verifies the bandwidth of the wireless transmission system based on millimeter wave. At the same time, it also indicates that the WIGI era is coming when we believe that this kind of transmission mode will gradually become a solution for the increasing life requirements of the public, with certain commercial value and the ability to compete with WIFI.

6 Conclusion and Future Research Directions

6.1 Conclusion

In this thesis, a wireless transmission system based on millimeter wave is designed and implemented to meet people's increasing demand for high-definition video. Its main work is divided into four parts:

The first part is the architecture design, according to the requirements of high-definition video and the size of the system components and so on, the corresponding hardware equipment and software programs are selected. The overall wireless transmission system architecture is preliminarily designed. The hardware part uses 10 gigabit network port and PCIE to speed up data transmission. LS1046A is selected as the NPU to process the transfer data. Using 6050 baseband chip and rf chip as millimeter wave module. The software part uses FFmpeg to process video stream, OpenCV to obtain visual frequency, Nginx server is responsible for data transfer, and Qt and OpenGL are responsible for data output.

The second part is the hardware circuit, according to the choice of 10 gigabit network port, PCIE and DPAA corresponding circuit design, to ensure that the transmission link in the hardware can have enough bandwidth. To ensure the start of the system, the EEPROM based on IIC is used to design the circuit and save the boot program. USB3.0 is selected to ensure the root file system and kernel program. The circuit of RS-232 is designed to control LS1046A and print and output data. Finally, the corresponding millimeter wave module is connected to realize the whole link.

The third part is the software part. According to the requirements of the second chapter, this thesis changed the PCIE frequency of u-boot, changed the CPU frequency of the kernel, transplanted the relevant Nginx program code to the root file system, and added the millimeter-wave driver to the Ubuntu image, compiled and burned. Then in the push stream, transfer and pull stream three parts respectively using OpenCV to obtain video stream, FFmpeg encoding and packaging; Nginx-RTMP is used for transfer, and Docker is used to preserve its stability; The device relates to millimeter wave drive. With FFmpeg data pull flow, unsealing and decoding, use origin Qt for audio output, with OpenGL for video output and at last again Qt for audio and video synchronization

The fourth part is the test part, including stress test and application scenario test. The stress test tests the bandwidth of the hardware link through Iperf3 and the load capacity of the Web server through WebBench. Application scenarios are gradually tested in the resolution from low to high, indicating the feasibility of the system in the transmission of HD video.

6.2 Outlook

According to the short distance millimeter wave communication technology, this thesis analyzes the wireless transmission codec algorithm, and video coding standards, according to the transmission bandwidth, algorithm complexity and throughput are discussed and compromise. Then in the hardware aspect based on millimeter wave module, NPU and other dedicated chips to build millimeter wave wireless HD video transmission prototype system, in the software development including FFmpeg, Nginx and OpenGL and so on, and the use of container design to support multi-channel transmission. Finally, the software and hardware are integrated and validated. With PCIE and 10 gigabit network port transfer rates, Ubuntu is good enough for servers to use the kernel and zero copy technology to forward data to users faster, but there are still some shortcomings.

6.2.1 Insufficiencies

First of all, the millimeter-wave module problem, the radio frequency chip adopted this time is a double channel transmission, although it has the requirement of 10Gb/s in essence, due to the limited transmission rate of the radio frequency chip, compared with a single channel can achieve 10Gb/s, this implementation method is still a little insufficient.

The second is the bandwidth problem. Even with ten gigabit band-width, the transmission rate does not reach 10Gb/s completely, and can only reach 70% efficiency at most. This may be because the operating frequency of NPU is not enough, and the DPAA hardware accelerated architecture has experienced some copy problems from kernel to user mode.

Finally, there is the problem of video stream forwarding. Although the video

stream can be forwarded through the server, the server forwarding is not the fastest way to forward. If an NPU is directly used as a Layer 2 forwarding switch, it can theoretically forward video streams faster. Due to the Linux kernel and root file system mechanism, data sent from a network port needs to be copied for multiple times. As a result, the forwarding speed cannot be kept up. Even in the case of zero copy technique commonly used today, also sent an order of magnitude. So, if you want to use Ubuntu as a switch, you need to use DPDK technology. At present, DPDK technology can directly transfer the processing of these data from kernel mode to user mode, which significantly solves the problem of copy. In this thesis, the millimeter wave module and driver do not support DPDK operation, so we can only use DPAA and Nginx to realize the forwarding of video stream.

6.2.2 Future Work

The above three problems can be integrated into two problems. For millimeter wave, it is necessary to develop or adopt rf chips and baseband chips with higher performance, including the design of high-speed circuit boards and the selection of chips. Increasingly, the performance of a two-channel implementation can be achieved with a single channel.

In view of NPU bandwidth and video stream transfer problems, in fact, whether DPAA, zero copy technology, NPU operation frequency increase or as a switch for forwarding they still must go through the kernel state and then through the user state, resulting in multiple copy resource waste. Therefore, DPDK technology can completely bypass the kernel state and hand over to the user state to deal with the related protocol stack and data forwarding. DPDK is a data plane development kit, which is mainly used for fast packet processing and provides corresponding network drivers. Greatly improves data processing performance and throughput. Packets are generally processed in the form of polling rather than interrupts between kernel-mode user states. The network protocol is processed directly by the user mode software without interrupting CPU, and the data is directly forwarded by the related software and driver of DPDK.

In this thesis, the future work of DPDK is to try to perfect the millimeter wave drive, transplant the millimeter wave drive to the DPDK tool platform, and directly control the operation mode of the millimeter wave module and the data transmission with ten

gigabit network port in the user state through DPDK, to realize the fast data forwarding.

In addition, can be seen from the above experiments, the attenuation of transmission performance in narrow area is large, this is because caused by the multipath effect, while the millimeter-wave modules have been adopted by the MIMO and beam informs the way to deal with the attenuation problem, but the future can be in millimeter wave module on flexible use of and development problem of the beam informs, So that it has a certain environmental adaptability, in the narrow environment can flexibly change the transmission mode.

Finally, wireless transmission system still has transmission instability. When the problem of bandwidth decrease caused by people or other environmental changes, software can be developed in the future to adapt to the situation from reducing video frame rate and video resolution to ensure that the delay does not fluctuate greatly.

Reference

[1] El Hattachi, Rachid, and Javan Erfanian. "Ngmn 5g white thesis." NGMN Alliance, February (2015)

[2] Zhou, Hongyan, et al. "A bend-insensitive ultra low loss and large Aeff fibre for long haul transmission." Optical Fiber Communication Conference. Optical Society of America, 2016

[3] Hu, Fenghe, et al. "Cellular-connected wireless virtual reality: Requirements, challenges, and solutions." IEEE Communications Magazine 58.5 (2020): 105-111.

[4] Thakshanth Uthayakumar. “Microwave Backhaul in 5G Networks.” Telecommunications System & Management (2018): 2167-0919

[5] Ikram, Muhammad, Nghia Nguyen-Trong, and Amin Abbosh. "Common-Aperture Sub-6 GHz and Millimeter-wave 5G Antenna System." IEEE Access (2020)

[6] T. S. Rappaport, S. Sun, R. Mayzus, H. Zhao, Y. Azar, K. Wang, G. N. Wong, J. K. Schulz, M. Samimi, and F. Gutierrez, ‘‘Millimeter wave mobile communications for 5G cellular: It will work!’’ IEEE Access, vol. 1, pp. 335–349, 2013.

[7] Lecci, Mattia, et al. "Simplified Ray Tracing for the Millimeter Wave Channel: A Performance Evaluation." arXiv preprint arXiv:2002.09179 (2020).

[8] Series, M. "IMT Vision–Framework and overall objectives of the future development of IMT for 2020 and beyond." Recommendation ITU 2083 (2015).

[9] Singh N, Shukla A. A Review on Progress and Future Trends for Wireless Network for Communication System[M]//Advances in Communication, Devices and Networking. Springer, Singapore, 2022: 445-453.

[10] Kleyn A W, Luo W, Yin B. Millimeter Wave Communication Technology[J]. 6G Wireless Communications and Mobile Networking, 2021: 23..

[11] Thomas, Michael E. "Infrared-And Millimeter-Wavelength Absorption In The Atmospheric Windows By Water Vapor And Nitrogen: Measurements And Models." *Optical, Infrared, Millimeter Wave Propagation Engineering*. Vol. 926. International Society for Optics and Photonics, 1988.

[12] John D K, Ronald J M. Antennas: for all applications[J]. Mc Graw Hill, 2002.

[13] Cao Y, Tang Y. Design of a Millimeter-wave Slotted Array Antenna with Miniaturized Dimension and Low Side-lobes[C]//2021 IEEE MTT-S International

Wireless Symposium (IWS). IEEE, 2021: 1-3.

[14] Tareq Q, Ragheb A M, Esmail M A, et al. Performance of Injection-Locked Quantum-Dash MMW Source Under Clear and Dusty Weather Conditions[J]. IEEE Photonics Journal, 2021, 13(3): 1-9.

[15] Ogawa M, Natsume K. Development of Automotive Millimeter Wave Radars[J]. IEICE Communications Society GLOBAL NEWSLETTER Vol. 45, No., 2021: 7.

[16] Elmutasim I E, MoHD I I. Examination rain and fog attenuation for path loss prediction in millimeter wave range[C]//Proceedings of the 11th National Technical Seminar on Unmanned System Technology 2019. Springer, Singapore, 2021: 935-946.

[17] Chen S, Zhao J. The requirements, challenges, and technologies for 5G of terrestrial mobile telecommunication[J]. IEEE communications magazine, 2014, 52(5): 36-43.

[18] Ou, Lu, et al. "Millimeter Wave Wireless Hadamard Image Transmission for MIMO enabled 5G and Beyond." IEEE Wireless Communications (2020).

[19] Pullin G, Treviranus J, Patel R, et al. Designing interaction, voice, and inclusion in AAC research[J]. Augmentative and Alternative Communication, 2017, 33(3): 139-148.

[20] Podpora M, Korbas G P, Kawala-Janik A. YUV vs RGB-Choosing a Color Space for Human-Machine Interaction[C]//FedCSIS (Position Thesiss). 2014: 29-34.

[21] Pokorny P. Lossy Compression in the Chroma Subsampling Process[J]. Wseas Transactıons On Computers, 2016, 15.

[22] Lin C H, Chung K L, Yu C W. Novel chroma subsampling strategy based on mathematical optimization for compressing mosaic videos with arbitrary RGB color filter arrays in H. 264/AVC and HEVC[J]. IEEE Transactions on Circuits and Systems for Video Technology, 2015, 26(9): 1722-1733.

[23] Safin R, Garipova E, Lavrenov R, et al. Hardware and software video encoding comparison[C]//2020 59th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE). IEEE, 2020: 924-929.

[24] Kwon S, Tamhankar A, Rao K R. Overview of H. 264/MPEG-4 part 10[J]. Journal of Visual Communication and Image Representation, 2006, 17(2): 186-216.）

[25] Kalva H. The H. 264 video coding standard[J]. IEEE multimedia, 2006, 13(04): 86-90.

[26] Igarta M. A study of MPEG-2 and H. 264 video coding[J]. MSECE, Purdue

University, 2004.

[27] Zhang Y, Mei S, Chen Q, et al. A novel image/video coding method based on compressed sensing theory[C]//2008 IEEE International Conference on Acoustics, Speech and Signal Processing. IEEE, 2008: 1361-1364.

[28] Tabash F K, Izharuddin M. Efficient encryption technique for H. 264/AVC videos based on CABAC and logistic map[J]. Multimedia Tools and Applications, 2019, 78(6): 7365-7379.

[29] Mozo A J, Obien M E, Rigor C J, et al. Video steganography using flash video (FLV)[C]//2009 IEEE Instrumentation and Measurement Technology Conference. IEEE, 2009: 822-827.

[30] Nurrohman A, Abdurohman M. High Performance Streaming Based on H.264 and Real Time Messaging Protocol (RTMP)[C]//2018 6th International Conference on Information and Communication Technology (ICoICT). IEEE, 2018: 174-177.

[31] Parmar H, Thornburgh M. Real-time messaging protocol (RTMP) specification[J]. Adobe specifications, December, 2012.

[32] Singh T, Rangarajan S, John D, et al. 2.1 Zen 2: The AMD 7nm Energy-Efficient High-Performance x86-64 Microprocessor Core[C]//2020 IEEE International Solid-State Circuits Conference-(ISSCC). IEEE, 2020: 42-44.

[33] Blem E, Menon J, Sankaralingam K. A detailed analysis of contemporary arm and x86 architectures[J]. UW-Madison Technical Report, 2013..

[34] Nayyar A. An Encyclopedia Coverage of Compiler's, Programmer's & Simulator's for 8051, PIC, AVR, ARM, Arduino Embedded Technologies[J]. International Journal of Reconfigurable and Embedded Systems, 2016, 5(1).

[35] Unhale N S, Bhawarkar N B, Patil A, et al. GLOBAL JOURNAL OF ENGINEERING SCIENCE AND RESEARCHES[J].

[36] "MCU Market on Migration Path to 32-bit and ARM-based Devices: 32-bit tops in sales; 16-bit leads in unit shipments". IC Insights. 25 April 2013. Retrieved 1 July 2014.

[37] Ismail N A, Idrus S M, Butt R A, et al. Performance evaluation of comprehensive bandwidth utilizationfor 10-gigabit passive optical network[J]. Bulletin of Electrical Engineering and Informatics, 2019, 8(3): 1047-1052.

[38] Page S, Uznanski S, Todd B, et al. JACoW: FGC3. 2: A New Generation of Embedded Controls Computer for Power Converters at CERN[J]. 2019.

[39] Neugebauer R, Antichi G, Zazo J F, et al. Understanding PCIe performance for end

host networking[C]//Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication. 2018: 327-341.

[40] Kaliaskarov N, Ivel V, Gerasimova Y, et al. Development of a Distributed Wireless Wi-Fi System for Monitoring the Technical Condition of Remote Objects[J]. Eastern-European Journal of Enterprise Technologies, 2020, 5(9): 107.

[41] Sivers I M A. Adopting the 64 to 71 GHz Band for Fixed Wireless Applications[J]. Microwave J., 2019.

[42] Escrivá D M, Joshi P, Mendonça V G, et al. Building Computer Vision Projects with OpenCV 4 and C++: Implement complex computer vision algorithms and explore deep learning and face detection[M]. Packt Publishing Ltd, 2019.

[43] Sinha U. OpenCV vs VXL vs LTI: Performance Test[J]. 2010.

[44] She B, Wang Q, Zhong X, et al. The Design and Implementation of Campus Network Streaming Media Live Video On-Demand System Based on Nginx and FFmpeg[C]//Journal of Physics: Conference Series. IOP Publishing, 2020, 1631(1): 012158.

[45] Newmarch J. Ffmpeg/libav[M]//Linux Sound Programming. Apress, Berkeley, CA, 2017: 227-234.

[46] DeJonghe D. Nginx CookBook[M]. O'Reilly Media, 2020.

[47] Jader O H, Zeebaree S R, Zebari R R. A state of art survey for web server performance measurement and load balancing mechanisms[J]. International Journal of Scientific & Technology Research, 2019, 8(12): 535-543.

[48] Nickoloff J, Kuenzli S. Docker in action[M]. Simon and Schuster, 2019.

[49] Noyes, Katherine (August 1, 2013). "Docker: A 'Shipping Container' for Linux Code". Linux.com. Archived from the original on August 8, 2013. Retrieved August 9, 2013.

[50] Simonsson J, Zhang L, Morin B, et al. Observability and chaos engineering on system calls for containerized applications in docker[J]. Future Generation Computer Systems, 2021, 122: 117-129.

[51] Blanchette J, Summerfield M. C++ GUI programming with Qt 4[M]. Prentice Hall Professional, 2006.

[52] Cheng Y, Liu Q, Zhao C, et al. Design and implementation of multimedia format converter based on FFmpeg[M]//Software Engineering and Knowledge Engineering: Theory and Practice. Springer, Berlin, Heidelberg, 2012: 857-865.

[53] Gois J P, Batagelo H C. Interactive graphics applications with opengl shading

language and qt[C]//2012 25th SIBGRAPI Conference on Graphics, Patterns and Images Tutorials. IEEE, 2012: 1-20.

[54] Shreiner D, Bill The Khronos OpenGL ARB Working Group. OpenGL programming guide: the official guide to learning OpenGL, versions 3.0 and 3.1[M]. Pearson Education, 2009.

[55] "Khronos OpenGL Registry". Khronos Group. Retrieved July 31, 2017.

[56] Smith, Ryan (June 5, 2018). "Apple Deprecates OpenGL Across All OSes; Urges Developers to use Metal". www.anandtech.com. Purch. Retrieved June 5, 2018.

[57] Freescale Drives Embedded Multicore Innovation with New QorIQ Advanced Multiprocessing Series". Freescale. 2011-06-21.

[58] Tianhua L, Hongfeng Z, Guiran C, et al. The design and implementation of zero-copy for linux[C]//2008 Eighth International Conference on Intelligent Systems Design and Applications. IEEE, 2008, 1: 121-126.

[59] Dubiel M, Zięcik P. FreeBSD on Freescale QorIQ Data Path Acceleration Architecture Devices[J].

[60] Siu S. An Introduction to the QorIQ™ Data Path Acceleration Architecture (DPAA)[J]. Jun, 2010, 23: 1-21.

[61] Khan A. Evaluation and Implementation of Linux User-space Fast Path Technologies: Linux, User-space[J]. 2013.

[62] Holdings A R M. ARM Cortex-A72[J].

[63] Qin Y, Mo L, Chen C. Design of Data-Driven Visualization Teaching System for Preschool Basketball Courses[C]//International Conference on E-Learning, E-Education, and Online Training. Springer, Cham, 2020: 292-302.

[64] Liu Lei, ZHANG Fengli, Qin Zhiguang. Building embedded Linux Bootloader Based on U-boot [J]. Application Research of Computers, 2007, 24(12): 238-240.）

[65] Love R. Linux kernel development[M]. Pearson Education, 2010.

[66] Bowman I. Conceptual architecture of the linux kernel[J]. URL: http://plg. uwaterloo. ca/itbowman/CS746G/a1, 1998.

[67] Likely G, Boyer J. A symphony of flavours: Using the device tree to describe embedded hardware[C]//Proceedings of the Linux Symposium. 2008, 2: 27-37.

[68] Liu D, Zhao L. The research and implementation of cloud computing platform based on docker[C]//2014 11th international computer conference on wavelet actiev media technology and information processing (ICCWAMTIP). IEEE, 2014: 475-478.

[69] Beyer D, Petrenko A K. Linux driver verification[C]//International Symposium On Leveraging Applications of Formal Methods, Verification and Validation. Springer, Berlin, Heidelberg, 2012: 1-6.

[70] Zatt B, Porto M, Scharcanski J, et al. Gop structure adaptive to the video content for efficient H. 264/AVC encoding[C]//2010 IEEE International Conference on Image Processing. IEEE, 2010: 3053-3056.

[71] Govil-Pai S. Principles of Computer Graphics: Theory and Practice Using OpenGL and Maya®[M]. Springer Science & Business Media, 2004.

[72] Rost R J, Licea-Kane B, Ginsburg D, et al. OpenGL shading language[M]. Pearson Education, 2009.

[73] Reddy K S P, Raju K N. Performance of Full Reference Video Frames quality evaluation metrics by using First order and Second order Edge detection operators[J].