九十七年度
低軌道衛星通訊基頻接收器軟體設計
研究委託研究計畫期末報告

中文計畫名稱：低軌道衛星通訊基頻接收器軟體設計研究

英文計畫名稱: Software Design Research for the Baseband Receiver of Low-orbit Satellite Communications

申請機構：台灣大學

執行單位：電信工程學研究所

主　持　人：電信工程學研究所 陳光禎教授

共同主持人：電信工程學研究所 林茂昭教授

中　華　民　國　９　８　年　１　１　月　３　０　日

Contents

Abstract

In order to build up own design capability for LEO remote-sensing satellite communication receiver capability including hardware and software implementation, we proceed the design of baseband receiver for Foremosa-2 front-end Mira board. We proceed from entire system specification, interface (ECL/LVDS, time code, ..), to achieve high-rate data acquisition from baseband receiver. Based on Formosa-2 satellite system requirements, we will try to deliver the baseband design of receiver in this project, from architecture, specifications, block diagram, implementation design, MATLAB/C implementation, to simulations (in floating-point). It is expected to establish Taiwan’s own design capability in LEO satellite communication receiver.

1 Goal of project

The purpose of this project is to design a satellite communication baseband receiver to meet the system requirement of Formosa-2 satellite system, to reach the purpose of self-control key technology. It is a low-orbital satellite with average height 891km. The communication system parameters include (D)QPSK with concatenated codes to support up to 120M bps over 120M Hz bandwidth. In this proposed project, we shall develop own technology to design LEO satellite baseband receiver and its simulations. The key to success of system integration like this project would be the confirmation of system specification and interface functions. Based on our extensive hand-on experience in satellite communications (PI has worked on INMARSAT, DVB-S, etc. different kinds of satellite communication systems), we will work closely with NSPO to inverse the need of this LEO satellite communication link requirements. Bases on this step, the baseband receiver shall include ECL/LVDS Interface, high rate acquisition and time code interface. The functions of baseband design include frame-sync, unscrambling and RS decoding, which shall be facilitated by block/module design concept. We can therefore complete the design of baseband receiver and evaluate real implementation.

2 Introduction to LEO satellite

2.1 Overview of Low earth orbit satellite

The first artificial satellite has launched on October 4 by Soviet Union in 1957 for 51 years ago. In recent years, low earth orbit (LEO) satellite communication system has become more and more popular. The characteristics of LEO satellite are listed in the following [1]:

- Orbit height: 800-1500 Km height
- Period to encircle the earth: 90~120 mins
- Motion period LEO satellite in sky: 15~20 mins
- Ground antenna is needed to track satellite when satellite is appearing in the sky

Compared to Medium Earth Orbit (MEO) satellite and Geostationary Orbit satellite (GEO), the advantage of LEO satellite are

- Less radiated power from satellite to ground station
- Low propagation delay (< 20 ms)

The disadvantage of LEO compared to MEO and GEO is:

- Higher Doppler shift (about hundreds of kHz)
- Low lifetime because of gravity between LEO and earth is larger than GEO and MEO.

Another characteristic of LEO channel is that with high elevation angle observed from ground base station, the channel between satellite and ground station can be model almost AWGN channel. Similarly, the channel can be model as multipath channel at low elevation angle. The boundary between low and high elevation angle is not obvious and it depends on the true situation.

In the following section, we briefly review the characteristics of our Formosa II satellite which is also a LEO satellite communication system.

2.2 Formosa II LEO satellite

Formosa II is a Low earth orbit (LEO) satellite communication system. It is a second our own artificial satellite in Taiwan. Formosa II is launched at Vandenberg of America on May 21 in 2004 [2]. The major mission of Formosa II is telemetry (image transmission). We can see the appearance of Formosa II from Fig. 2.1 and Fig. 2.2.

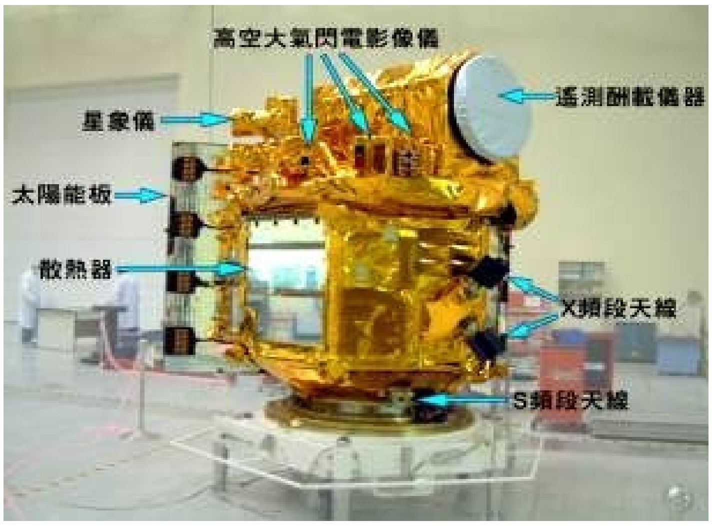


Fig. 2.1 Appearance of Formosa II (1) [2]

Fig. 2.2 Appearance of Formosa II (2) [2]

Although some parameter maybe list before, we still arrange some basic parameters in the Table 2.1 for convenient reading. There are some other basic properties of Formosa II lists as follows.

Type	Science satellite
Orbit height	891 km
Height	2.4 m
Weight	760 kg
Period to encircle the earth	103 mins
Appear time above Taiwan	2 times. 1st about at am 9:40 2nd about at pm 9:40
Resolution for telemetry to the surface of earth	Black and white image: 2m Color image: 5m
Mission lifetime	5 years

Table 2.1 Characteristics of Formosa II [2]

After brief introduction, we should go to more detail in LEO satellite receiver design in the following chapter which is the major concern of this project.

3 Receiver system architecture

3.1 Overview

In this chapter, we will present the conventional LEO communication receiver with reference of Kongsberg MEOS capture system and other general basic design guideline [6]. We introduce the whole system architecture mainly based on the Kongsberg MEOS capture operational and maintenance manual [7] and an additional remark which is from general basic design guideline will be given if the operational manual doesn’t describe very well.

Many important components of receiver are introduced briefly in following sections. Although RF front end is not our main topic of this project, Section 3.2 also gives a basic concept about three traditional RF down-conversion architectures. We will pay more attention on Section 3.3, which concerns about the principle of baseband receiver design. Some mathematical representation and theoretical introduction will also cover in this chapter.

3.2 RF front end

3.2.1 RF down-conversion architecture

There are three important types which are often used in the real life front-ends architecture. That are super-heterodyne, direct conversion and low IF architecture [4]. Three architectures each has its own characteristics and application and the more detail is in the following section.

3.2.1.1 Super-heterodyne architecture

The feature of super-heterodyne architecture is that it needs two times to down-convert the received RF signal to baseband signal [4]. The block diagram of super-heterodyne architecture is in Fig 3.1.

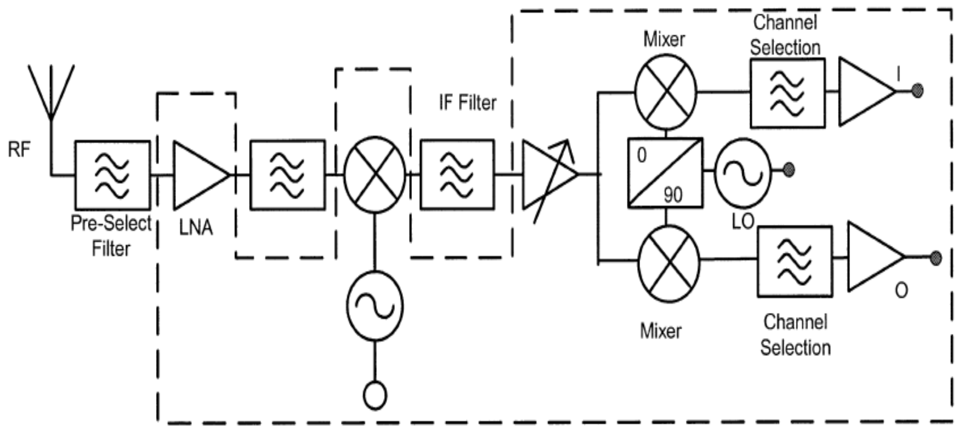


Fig. 3.1 Super-heterodyne architecture [5]

The advantage of this architecture is good selectivity for channel because the high quality IF filter can be constructed to reduce the adjacent channel interference and free from DC offset. The disadvantage of super-heterodyne architecture may be the lager complexity than direct conversion architecture due to high quality requirement of IF filter and less suitable for integration because of the many chip connections to external lumped elements.

3.2.1.2 Direct conversion architecture

The feature of direct conversion architecture is that it down convert RF signal at one time [4]. The block diagram can be seen in Fig 3.2.

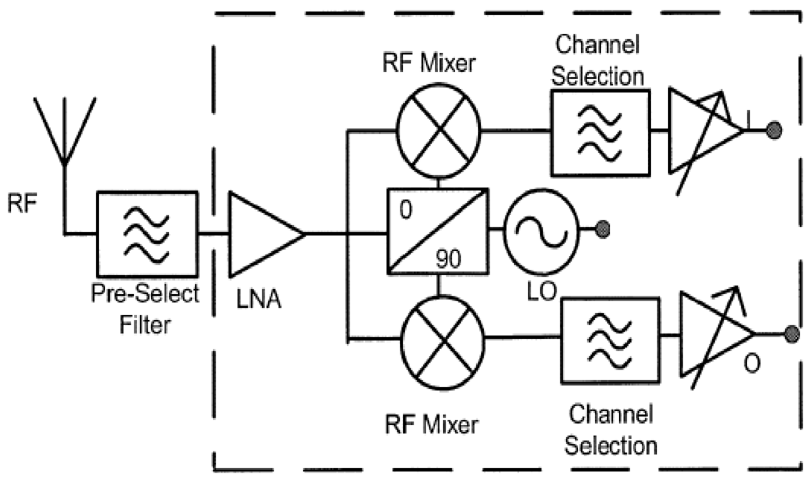


Fig. 3.2 Direct conversion architecture [5]

The advantage of direct conversion architecture is that it greatly simplifies the complexity by removing IF receiving chain and use single down-conversion section. By direct conversion architecture, several hardware-driven tasks could become software-driven, which make it easy for the goal to software defined radio. The disadvantage of direct conversion architecture is that direct conversion receiver suffer from DC offsets, self mixing, LO leakage, flicker noise and IQ imbalance, etc. The more consideration for compensating the above mentioned distortion than SHR architecture and the design requirement is also tougher than SHR.

3.2.1.3 Low IF architecture

The feature of low IF architecture is mostly like direct-conversion architecture except the down-convert signal is not DC signal but a low IF signal [4]. The block diagram is the same as Direct conversion architecture in Fig. 3.2. The only different portion is the output of mixer which is an intermediate frequency (IF) signal.

The complexity of low-IF architecture lies between the direct-conversion architecture and super- heterodyne architecture. This architecture reserve more advantage of direct conversion architecture but inject the image signal issue which also existed in super- heterodyne architecture. The remedy is the same as super-heterodyne architecture that image-rejection filter is put in front of mixer, which the output of mixer may not be distorted severely by image signal.

In following section, we will continue briefly introducing the function of some key components of three architectures.

3.2.2 Pre-select filter [4]

In the Fig. 3.1 and Fig. 3.2, we can see that there is a pre-select filter after the transmitted band-pass signal is received from antenna. Pre-select filter is a band-pass filter and its function is to filter other band noise which we don't desire. It is an indispensable element to RF front end when we want to choose the desired frequency band signal. Much outer band noise can be filtered by this band-pass filter.

3.2.3 Low noise amplifier (LNA) [4]

When the band-pass signal passed through pre-select filter, the next stage is the low noise amplifier (LNA). It is another key function of RF front end circuit that the Signal to Noise ratio (SNR) of received signal can be enhanced and the noise of all the subsequent stages is reduced by the gain of the LNA. Thus, it is necessary for an

LNA to increase the desired signal power with little noise and distortion so that the SNR of this signal can be utilize in the later stages of the system.

3.2.4 Local Oscillator (LO)[4]

Local oscillator is the heart of all RF front end circuit. Without local oscillator, the high frequency signal can not be generated to down-convert the received RF signal. In general, the pure oscillator only supplies one high frequency signal with high precision. By cooperation with frequency synthesizer which has Phase lock loop (PLL) and frequency divider, LO can provide many different frequency carrier in order to perform multiple choices of channel selection.

3.2.5 Mixer [4]

In spite of what architecture is chosen, the mixer which performs the multiplication of two signals always exists in Fig. 3.1 and Fig. 3.2. The operation of mixer uses Prosthaphaeresis formula to perform signal down-conversion. In usual, a low pass filter with desired bandwidth is following the mixer in order to filter out the desired signal.

3.3 Baseband receiver

3.3.1 Overview

After RF front end processing, the received signal is down-converted to baseband. There are still many works that are needed to deal with. Several compensations of non-ideal effect of channel are performed in baseband processing chain. According to [3], we can divide the baseband receiver into two categories.

One is inner receiver whose major design rule is to make channel effect at the output of inner receiver could be more like the AWGN channel. Section 3.3.2 will introduce more detail about inner receiver. The other is outer receiver whose main goal is to perform channel decoding and source decoding with respect to the signal which is from the output of inner receiver. Section 3.3.3 will also introduce the detail of outer receiver.

Kongsberg's High Rate Demodulator (HRD) is our major reference for inner receiver introduction, and Front end processor (FEP) is our main reference for portion of outer receiver which does not include source decoder [6]-[7]. Some important components which is not presented in [6]-[7] will be introduced in this chapter.

3.3.2 Inner receiver

The inner receiver block diagram recorded in MEOS capture operational manual is shown in Fig. 3.3. The components of the block diagram are AGC, RSSI, ADC, Carrier recovery, Timing recovery, Equalizer and Channel estimation, etc. All of the blocks have their own role and application in the entire system, and the description will be given in each section.

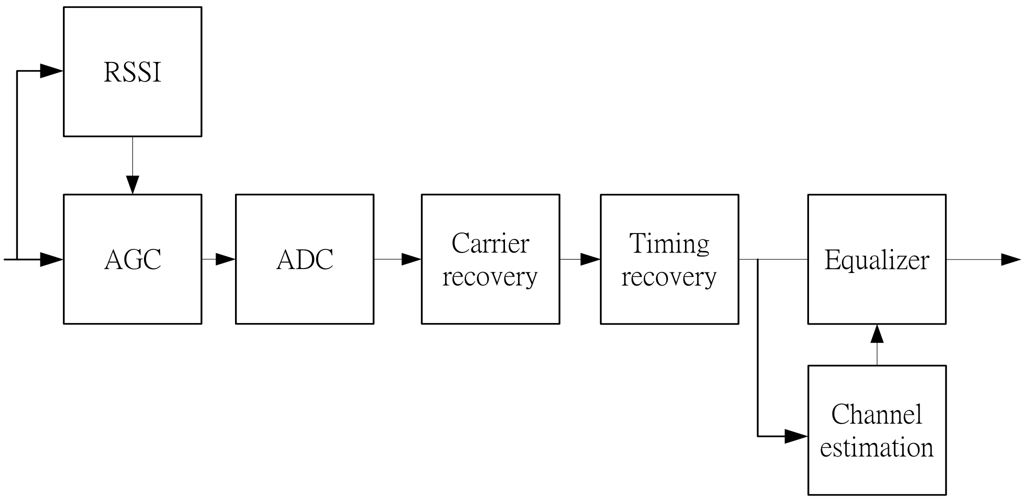


Fig. 3.3 Inner receiver block diagram of Kogsberg Capture system [6],[7]

3.3.2.1 Automatic gain control (AGC) [4],[6],[7]

The reason why we need automatic control is that AGC can maintain the input dynamic range in certain level. What an important thing for AGC that it makes

Analog to digital converter (ADC) design because the higher dynamic range is, the more difficult for ADC design. AGC can also help the received signal avoid fluctuation of amplitude which is good properties for phase-shift keying modulation.

3.3.2.2 Received signal strength indicator (RSSI) [4],[6],[7]

The function of Received signal strength indicator (RSSI) is mainly to measure the total power in front of AGC. RSSI also provide an indication for AGC that AGC can depend on the indication to adjust the amplifier gain to satisfy the need in practice. When we detect the higher value of signal strength above some threshold, RSSI give information to AGC and it can adjust the amplifier gain smaller. In reverse, if the lower value of signal strength is detected, AGC will adjust the amplifier gain larger.

3.3.2.3 Analog to Digital Converter (ADC) [7],[8]

In recent years, with the progress of VLSI technique, the computation time of DSP processor or FPGA become more and more short. It is an important role for ADC to convert analog signal into digital signal and DSP processor can perform processing. Sampler and quantization is two key function of ADC. By determining sampling rate and quantization bits, analog signal could be transformed into more sampling values with determined bits to represent them by ADC.

3.3.2.4 Matched filter [7],[8]

In communication theory, the real life communication channel is often band-limited and the transmitted signal will be distorted in time domain when passing through such a band-limited channel. This situation will lead to inter-symbol interference (ISI) problem. Matched filter is a good way to overcome this problem. By putting pulse-shapping filter in transmitter side and matched filter in receiver side, we can reduce the ISI which caused by the effect of band-limited channel.

3.3.2.5 Carrier synchronization [3],[8]

Consider the general case of the received baseband signal x(n) as follows,

$$x(n) = e^{j(2\pi vn+\theta)} \sum_{k=1}^{L-1} h(k)a(n-k-\varepsilon) + w(n)$$

where v is the normalized carrier-frequency offset (CFO) and θ is the phase offset. h(k) means the channel taps corresponds to channel impulse response and L stands for total channel taps number. $a(n-k)$ is the information signal which is usually represented by complex number if QPSK modulation is adopted. ε is the normalized timing offset with respect to modulation symbol time. w(n) is an AWGN channel that all RF front end thermal noise can be modeled by AWGN.

The goal of carrier synchronization is to recovery the carrier-frequency offset and phase offset which caused by channel or other hardware non-idealities. The one of major reason of phase offset is caused by oscillator phase noise. The general remedy is to use pilot-aided phase estimation algorithm to compensate this non-idealities. The major cause of CFO is distorted by the relative speed between transmitter and receiver, and the mismatch of the Tx and Rx local oscillators. Without this compensation, the system performance could degrade severely by the existence of CFO and phase offset. Especially for LEO satellite, as result of high speed of motion (maybe several ten thousand km per hour), the Doppler shift is larger than any other terrestrial cellular communication system. Therefore, the CFO compensation is the majorly important issue in LEO satellite receiver design.

3.3.2.6 Timing synchronization [3],[8]

As seen in the upper equation, timing synchronization is try to estimate the timing offset ε and compensate the offset in order to make sure the signal is sampled at the best sampling time. That is to say, sample the received signal at the maximum eye open position. The reason why the timing synchronization is required is that the effects of propagation delay and sampling clock offset due to real life non-idealities make us have to do compensation for the timing error. Without this compensation, the input to the decision device could not be maximum signal to noise ratio (SNR) at that situation and system performance will be degraded.

3.3.2.7 Channel estimation [3],[8]

After synchronization procedure is over, the signal would enter into next stage and go to cancel out another impairment due to multipath channel propagation. The procedure should divide into two steps. The first step is trying to estimate the multipath channel L taps, that is h(k), k=0….L-1. The second step is to utilize the equalizer to compensate the effect which caused by each channel tap. The function of channel estimation is reproduction of multipath channel taps in some constant period which might be a frame period. In other words, there is an assumption of channel to be static in some constant period. Although this is not the truth in real life, this method is a good way in implementation issue.

3.3.2.8 Equalizer [3],[8]

The function of equalizer takes over the unfinished work of compensation of Channel effect. It performs compensation by designing the tap-delay line filter with the inverse channel response or channel state information (CSI) as the coefficients of filter. After this operation, the signal is going to the decision device to identify which information is transmitted and recovery them into bit sequence. Therefore, the input signal of outer receiver is always bit sequence.

3.3.3 Outer receiver

In this section, the outer receiver block diagram recorded in MEOS capture operational manual is shown in Fig. 3.3. The components of the block diagram are Differential decoder, Viterbi decoder, PCM decoder, Frame synchronization, Cyclic Redundancy Check, R-S decoder and Source decoder. All of the blocks have their own role and application in the entire system, and the description will be given in each section.

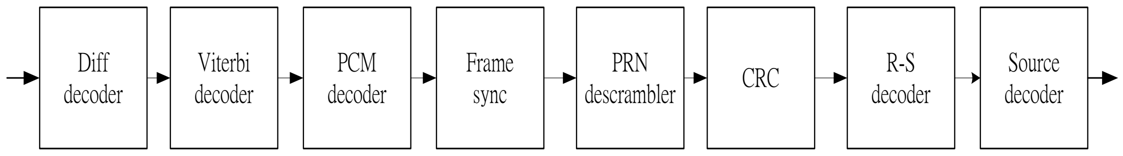


Fig. 3.4 Outer receiver structure of MEOS Capture system [7]

3.3.3.1 Differential decoder [6],[7],[9]

In digital communications, differential coding is a technique which used to tell unambiguous signal from reception by using some types of modulation. The differential encoder is to make the transmitted data depend not only on current bit (or symbol), but also from the previous one. Then, two bits are combined into a symbol and encode or decode in use of the previous symbol relationship with present symbol. The original information could be extracted by comparing the difference of two successively received signals in the differential decoder.

3.3.3.2 Viterbi decoder [8],[9]

Viterbi dcoder use the famous Viterbi algorithm to decode any encoder which can be represented by finite state machine. A convolutional code encoder consists of several shift registers and XOR operations on the input and output from certain registers, as depicted in Fig. 3.5. A code generator is defined by its code rate and constraint length (defined as number of shift registers plus one). For simplicity, we consider code rate to be 1/2 only. As we have a code generator, then a series of encoded bits can be generated. In addition, the optimal choice of code generators for different constraint length and even code rate is provided in [7],[9].

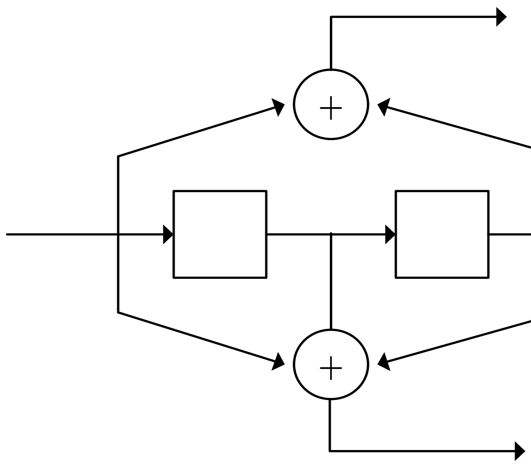


Fig. 3.5   Code rate 1/2 of CC Encoder with code generator [9]

Convolution code is the well-known channel coding which can decode by this method. Although Viterbi algorithm has the more consumption of resource compared to other decoding method, it performs maximum likelihood (ML) decoding which have better performance than others The ML-based Viterbi algorithm is used for decoding. The decoder has a total of 2^(number of shift registers) states, see Fig. 3.6.

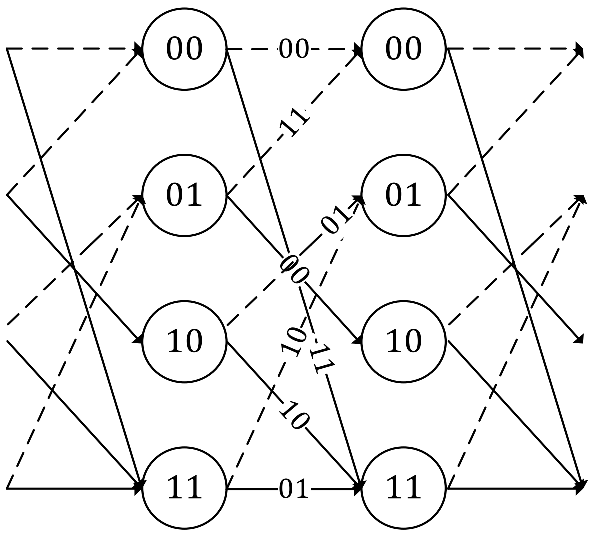


Fig. 3.6 Decoder state diagram for Viterbi algorithm [9]

On each state transition, it corresponds to a sequence of output values. Note that the output values are binary for hard decision, or with higher quantization for soft decision. As a codeword is received, then it is compared to each of the state transitions at each time interval. The hamming distance between the received codeword and transition outputs at each interval is recorded as a branch metric (BM). For each decoding path, it corresponds to a path metric (PM) which is the accumulation of BM along the path. In addition, at each time interval, PM of the paths which transit to a common state are added with current BMs respectively, compared, and the one with the smallest consequent PM is selected as the best. Note that at each time instant, only a path is stored for each state. Consequently, the path at the end with the smallest PM is determined as the decoded output. The more information can be found in [6],[7],[9].

3.3.3.3 PCM decoder [6],[7],[8]

Pulse code modulation decoder operates in terms of one bit duration as a unit. It can change some other type line coding waveform into the only waveform of NRZ-L in this MEOS capture system. That is, the input of PCM decoder waveform can be NRZ-L, NRZ-M, NRZ-S and output waveform of PCM decoder should be always NRZ-L. This function is also optional in Kongsberg FEP board and absent in IO 21000 board. Hence, when we close this function, the input and output format should be the same. Another reason for using this function is to ensure that the input signal format of frame synchronization should be the same.

3.3.3.4 Frame synchronization [7],[8],[10]

Since there are many sensors in satellite, each sensor would like transmitting the measurement signal. The way how to integrate the sensors' information efficiently is a big problem. The more efficient method is to design a frame structure, and combine every sensor's measurement signal into a frame. Hence, the multiplexing technique play an important role in satellite communication system and the way how to de-multiplex the combined signal information is an important thing.

In order to make sure the exactly extraction of information of each sensor, frame synchronization is necessary for our baseband receiver design. False frame synchronization could make every individual sensor information incorrectly acquired by receiver and let all sequence of frame be scrambled. Consequently, in order to avoid false synchronization, an algorithm of frame synchronization which uses the synchronization pattern to recognize if the frame is correctly identified is needed.

In Kongsberg FEP, the synchronization pattern is a pre-determined pattern according to Consultive Committee for Space Data System (CCSDS) standard [10]. It is shown as follows:

0x1acffc1d

Fig. 3.7 Synchronization pattern in CCSDS standard [10]

We can see in Fig. 3.7 that a 32-bit synchronization pattern is represented by hexadecimal. More specific algorithm for MEOS Capture system will be described in detail in Chapter 4.

3.3.3.5 PRN Descrambler [6],[7],[8,[9],[10]

The purpose of Scrambler in transmitter is used to ensure that the data has sufficient bit transition density which could help timing recovery by capturing the zero cross point and could not complicate the timing recovery circuit implementation. The operation of descrambler is the same as scrambler. They always make input bit sequence have exclusive-or operation with the sequence generated by pseudo random number (PRN) generator. Pseudo random number (PRN) sequence could be determined by its own specific generator.

For example, the PRN sequence implementation by a generator polynomial

$$g(x) = x^{15} + x^{14} + 1$$

is shown in Fig 3.8.

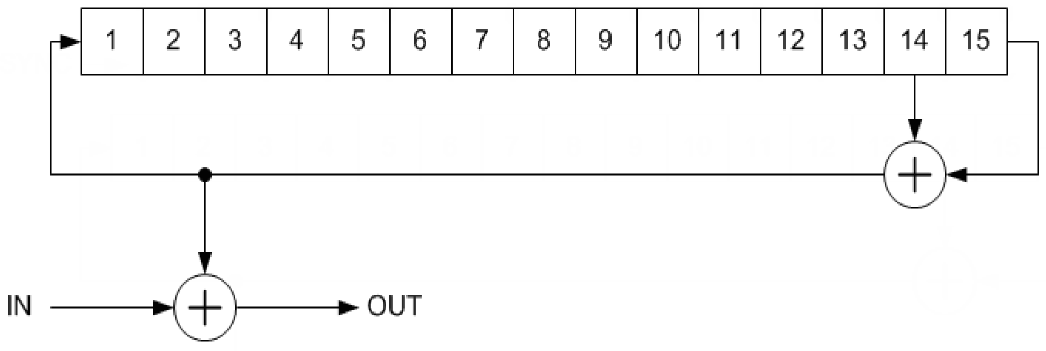


Fig. 3.8 PRN descrambler generator [10]

In Kongsberg FEP, the generator polynomial of descrambler is chosen according to Consultive Committee for Space Data System (CCSDS) standard [7],[9]. The specific generator is shown as follows:

$$g(x) = x^8 + x^7 + x^5 + x^3 + 1$$

where means that there are nine registers in generator of PRN and every input bit performs exclusive-or with every output bit of PRN generator and the more detail is shown in Fig. 3.9. In Fig. 3.9, the abbreviated term ASM means attached sync marker which is also named synchronization pattern.

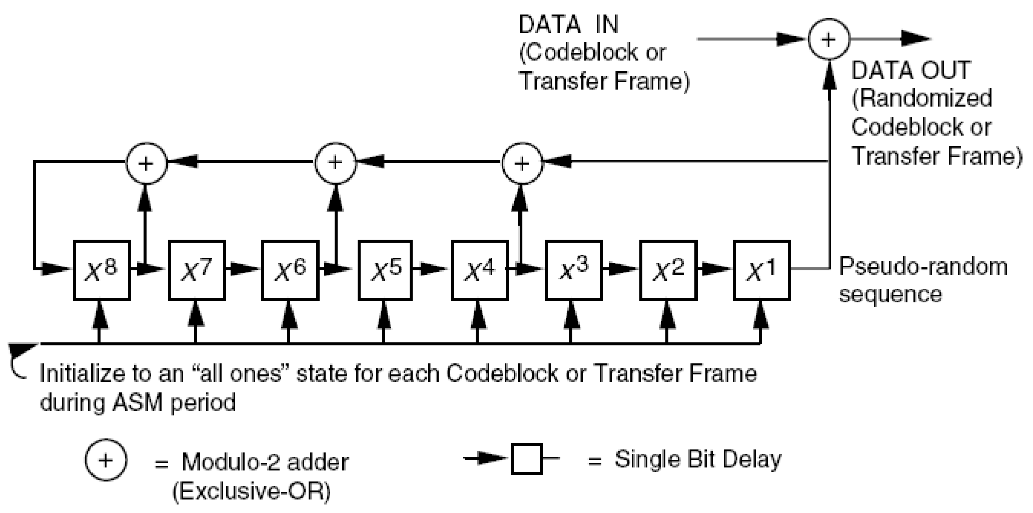


Fig. 3.9 PRN descrambler logic diagram [10]

3.3.3.6 Cyclic Redundancy Checker (CRC) [6],[7],[8],[9],[10]

The CRC general rule is to calculate the remainder of the input data frame by the generator polynomial in the finite field operation and put the remainder into the frame tail and then transmit the frame to the receiver. The remainder of divided input data frame is called CRC checksum. On the transmitter side, the calculated CRC checksum appended to the data end of frame and transmit it to the receiver. After pass through the channel, the CRC checker performs calculation again with a frame including data and checksum in the receiver side. If the remainder is not zero, we infer that there must be something wrong among the frame bits. There is something worth noting that CRC checker could only detect limited errors of the received frame and can’t correct the error of the frame.

In Kongsberg FEP, the generator polynomial of descrambler is chosen according to Consultive Committee for Space Data System (CCSDS) standard [10]. It is shown as follows:

$$g(x) = x^8 + x^7 + x^5 + x^3 + 1$$

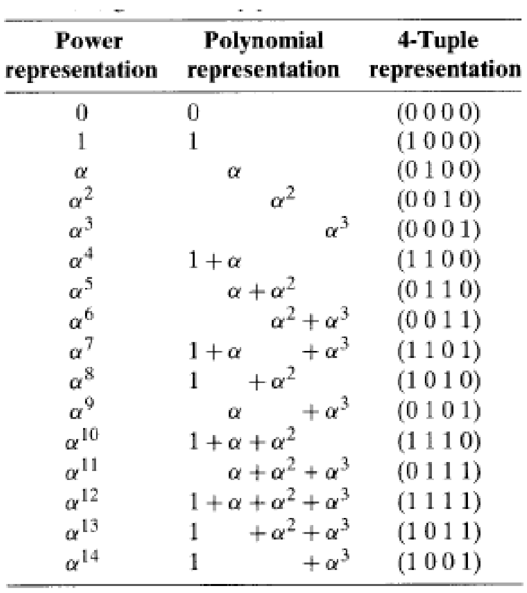
A 16-bit cyclic redundancy checksum is calculated by g(x) in finite field GF(2) operation. More detail will be described in the chapter 4.

3.3.3.7 Reed-Solomon decoder

The Reed-Solomon (R-S) code belongs to a class of non-binary cyclic codes for which each codeword has consecutive roots so that algebraic encoding and decoding procedures can be implemented. Since the R-S code can correct multi-bit symbol errors, it is useful in correct a combination of random errors and burst errors. Hence, R-S codes are popular in increasing the reliability of wireless communications and memory systems.

➢ Finite Field

Each symbol of an R-S code is an element of a finite field. Each finite field is of the form GF(p^m),which p is a prime. Every finite field has size which is a power of a prime. Finite fields are also called Galois Fields. Suppose that there are q element in GF(q). Let α be a primitive element in GF(q), then the q elements in GF(q) are 0, α^0, α^1, $\alpha^2 \ldots \alpha^{q-2}$, and$\alpha^{q-1}=1$. The finite field can be represented in various forms, such as the power representation, the polynomial representation and the vector representation, etc.



Power representation	Polynomial representation	4-Tuple representation
0	0	(0 0 0 0)
1	1	(1 0 0 0)
α	α	(0 1 0 0)
α^2	α^2	(0 0 1 0)
α^3	α^3	(0 0 0 1)
α^4	$1+\alpha$	(1 1 0 0)
α^5	$\alpha+\alpha^2$	(0 1 1 0)
α^6	$\alpha^2+\alpha^3$	(0 0 1 1)
α^7	$1+\alpha+\alpha^3$	(1 1 0 1)
α^8	$1+\alpha^2$	(1 0 1 0)
α^9	$\alpha+\alpha^3$	(0 1 0 1)
α^{10}	$1+\alpha+\alpha^2$	(1 1 1 0)
α^{11}	$\alpha+\alpha^2+\alpha^3$	(0 1 1 1)
α^{12}	$1+\alpha+\alpha^2+\alpha^3$	(1 1 1 1)
α^{13}	$1+\alpha^2+\alpha^3$	(1 0 1 1)
α^{14}	$1+\alpha^3$	(1 0 0 1)

Table 3.1

	1	**α^1**	**α^2**	**α^3**	**α^4**

$\alpha^{-}=0$	0	0	0	0	--
$\alpha^{0}=1$	1	0	0	0	--
α^{1}	0	1	0	0	--
α^{2}	0	0	1	0	--
α^{3}	0	0	0	1	--
α^{4}	0	0	0	0	1
	1	1	0	0	--
α^{14}	1	0	0	1	--
α^{15}	0	1	0	0	1
	1	0	0	0	--

Table 3.2

Table 3.1 illustrates the example of GF(2^4). The field elements are generated by the primitive polynomial p(X)=1+X+X^4. Table 3.2 shows the steps to construct the 4-tuple representation from the power representation of elements of GF(2^4) in Table2. Since P(α)= 1+α+α^4=0, we can exchange α^4 with (1+α). Using Table 3.2, α^n can be represented by $a_0 1+a_1\alpha^1+a_2\alpha^2+a_3\alpha^3$ $a_0 \sim a_3 \in \{0,1\}$

- **Example 1: :($\alpha^{13}+\alpha^5$)*α^8**

$(\alpha^{13}+\alpha^{5})*\alpha^{8} = (1011+0110)*\alpha^{8}=(1101)*\alpha^{8}=(1+\alpha+\alpha^{3})*(1+\alpha^{2})$

$=(1+\alpha+\alpha^{3}+\alpha^{2}+\alpha^{3}+\alpha^{5})=(1+\alpha+\alpha^{2}+\alpha^{5})$----------------(1)

$=(1+\alpha+\alpha^{2}+(\alpha+\alpha^{2}))=1$------------------------------------ (2)

From Example1, multiplication and addition calculation are showed. Addition calculation only needs a XOR unit, while multiplication is more complex. The design of multiplication in GF(2^8) is showed in Fig. 3.10.

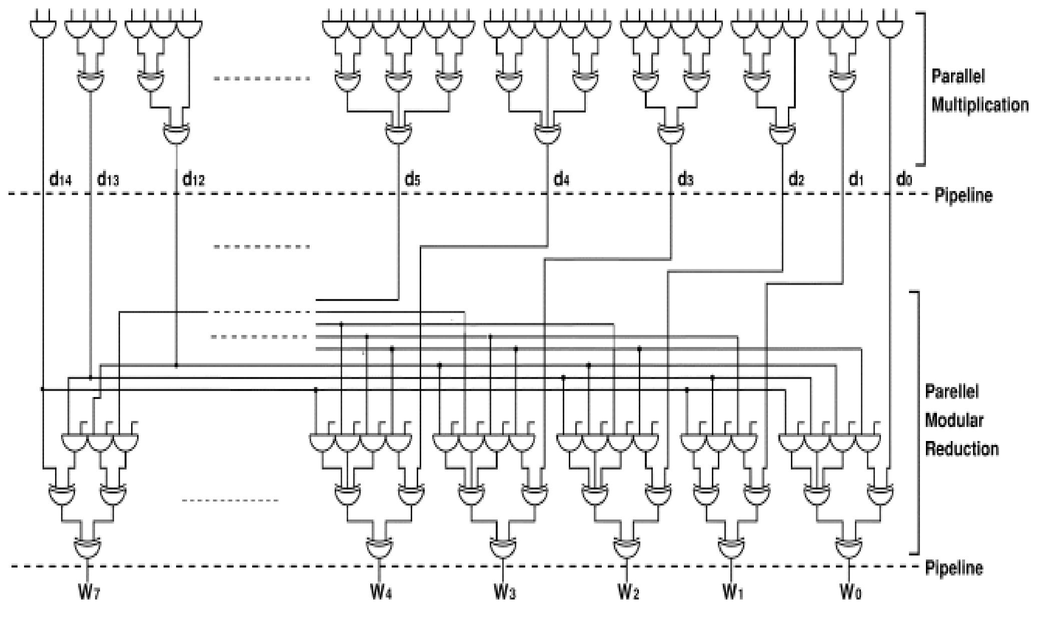


Fig. 3.10

The parallel multiplication part does the same thing in (1). In the arithmetic operation of a and b in GF(2^8), we set a and b to be in the form of $a_7\alpha^7+a_6\alpha^6+\ldots+a_0$ and $b_7\alpha^7+b_6\alpha^6+\ldots+b_0$ respectively, where a_n and $b_n \in \{0,1\}$. The product of a and b, i.e., **d**=a*b must be the form $d_{14}\alpha^{14}+d_{13}\alpha^{13}+\ldots+d_0$. For the arithmetic operation in

GF(2^8), $d_{14}\alpha^{14}+d_{13}\alpha^{13}+\dots+d_8\alpha^8$ must be converted to the linear combination of $1\sim\alpha^7$. This is the work of the part of Parallel Modular Reduction

- **Encoder of the Reed-Solomon code**

The generator polynomial of a t-error-correcting RS code contains t consecutive roots. Hence, consider an RS code with generator polynomial in the following form,

$$\mathbf{g(x) = (X-\alpha)(X-\alpha^2)\dots(X-\alpha^{2t})}$$

$$\mathbf{= g_0+g_1x+g_2x^2+\dots+g_{2t}x^{2t}} \qquad g_i \in GF(q) \quad \text{for} 0 \le i < 2t$$

A t-error-correcting R-S code with symbols from GF(q) has the following parameters

Block length: $n = q - 1$,
Number of parity-check symbols: $n - k = 2t$,
Dimension: $k = q - 1 - 2t$,
Minimum distance: $d_{min} = 2t + 1$.

Considering a RS(255,223) code over GF(2^8) with $n=2^8-1=255$

$t=16$

$k=255-2*16=223$

$dmin=33$

The RS(255,223) code uses 8 bits to represent each symbol. Hence, this code in binary form is a (2240,1784) code. This code has the capability of correcting 16 symbols.

- **Decoder of Reed-Solomon code**

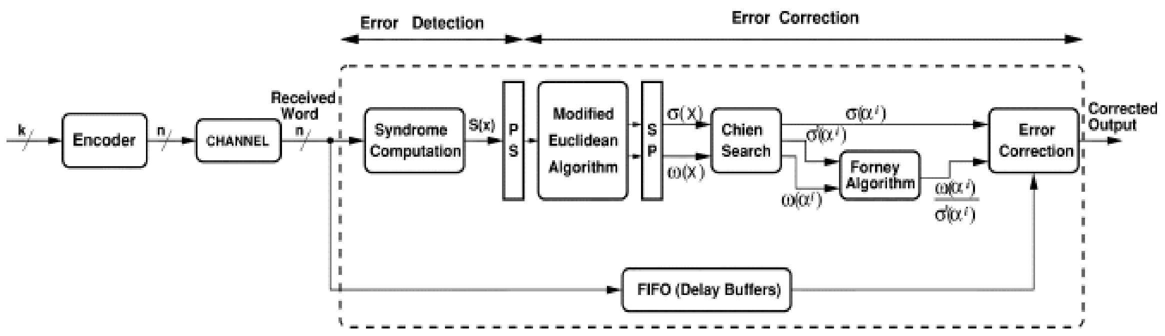


Fig. 3.11

Fig. 3.11 shows the block diagram of a decoder for the R-S code. The functional blocks are respectively described as follows.

- **Syndrome computation:**

The syndrome is a 2t-tuple: $(S_1,S_2,\ldots S_{2t})$, with $S_i=r(\alpha^i)$ for $1\leqq i\leqq 2t$. And we define the syndrome polynomial S(X) as:

$S(X)=S_1+S_2X+\ldots+S_{2t}X^{2t-1}$.

Let $v(X)=v_0+v_1x+\ldots+v_{n-1}X^{n-1}$ be the transmitted polynomial,

$r(X)=r_0+r_1x+\ldots+r_{n-1}X^{n-1}$ be the received polynomial

$e(X)=e_0+e_1x+\ldots+e_{n-1}X^{n-1}$ be the error pattern polynomial which added by channel

Since $r(X)=v(X)+e(X)$, we have $S_i=v(\alpha^i)+e(\alpha^i)= e(\alpha^i)$. Then, we obtain the following set of equation:

$$S_1=\alpha^{j1}+\alpha^{j2}+\ldots+\alpha^{jv}$$

$$S_2=\alpha^{2j1}+\alpha^{2j2}+\ldots+\alpha^{2jv}$$

$$\vdots \quad \vdots \quad \vdots \quad \vdots$$

$$S_{2t}=\alpha^{2tj1}+\alpha^{2tj2}+\ldots+\alpha^{2tjv}$$

For $1\leqq i\leqq v$ let $\beta_i=e_{ji}$. Then, the above equations can be simplified as follows:

$$S_1=\beta_1+\beta_2+\ldots+\beta_v$$

$$S_1=\beta_1^2+\beta_2^2+\ldots+\beta_v^2$$

$$\vdots \quad \vdots \quad \vdots \quad \vdots$$

$$S_1=\beta_1^{2t}+\beta_2^{2t}+\ldots+\beta_v^{2t}$$

These equations are known as power-sum symmetric functions. If $S(X)$ is the zero polynomial, it indicates the codeword is error-free. Hence, $r(X)=v(X)$.

- **Modified Euclidean (ME) Algorithm:**

The error-location polynomial is defined as $\sigma(X)=(1-\beta_1X)(1-\beta_2X)..(1-\beta_vX)$

$=\sigma_0+\sigma_1X+\ldots+\sigma_vX^v$

where $\sigma_0=1$. Consider the product $\sigma(X)S(X)$, i.e.,

$\sigma(X)S(X) = (\sigma_0+\sigma_1X+\ldots+\sigma_vX^v)(S_1+S_2X+\ldots+S_{2t}X^{2t-1})$

$$=S_1+(S_2+\sigma_1 S_1)X+\ldots(S_{2t}+\sigma_1 S_{2t-1}+\ldots+\sigma_vS_{2t-v})X^{2t}+\ldots$$

Only the coefficients of the first 2t terms are known. Let $[\sigma(X)S(X)]_{2t}$ denote the first 2t terms of $\sigma(X)S(X)$, then $\sigma(X)S(X) - [\sigma(X)S(X)]_{2t}$ is divisible by X^{2t}. Then, we have,

$$[\sigma(X)S(X)]_{2t}\equiv\sigma(X)S(X) \bmod X^{2t}$$

$$\text{Let } \omega_0(X)= [\sigma(X)S(X)]_{2t}$$

$$\text{Then } \omega_0(X)\equiv\sigma(X)S(X) \bmod X^{2t} \qquad (3)$$

This equation is called the **key equation.** We can use Euclidean algorithm to solve the key equation. For VLSI, there is a more efficient algorithm called Modified Euclidean algorithm (ME) which are shown as follows:

The initial condition of the algorithm is

$$R_0(x) = x^{2t},\ Q_0(x) = S(x),\ L_0(x) = 0,\ U_0(x) = 1.$$

In the i-th iteration,

$$\begin{aligned}
R_i(x) &= [\sigma_{i-1}b_{i-1}R_{i-1}(x) + \overline{\sigma}_{i-1}a_{i-1}Q_{i-1}(x)] \\
&\quad - x^{|l_{i-1}|}[\sigma_{i-1}a_{i-1}Q_{i-1}(x) + \overline{\sigma}_{i-1}b_{i-1}R_{i-1}(x)] \\
Q_i(x) &= \sigma_{i-1}Q_{i-1}(x) + \overline{\sigma}_{i-1}R_{i-1}(x) \\
L_i(x) &= [\sigma_{i-1}b_{i-1}L_{i-1}(x) + \overline{\sigma}_{i-1}a_{i-1}U_{i-1}(x)] \\
&\quad - x^{|l_{i-1}|}[\sigma_{i-1}a_{i-1}U_{i-1}(x) + \overline{\sigma}_{i-1}b_{i-1}L_{i-1}(x)] \\
U_i(x) &= \sigma_{i-1}U_{i-1}(x) + \overline{\sigma}_{i-1}L_{i-1}(x)
\end{aligned}$$

$$l_{i-1} = \deg(R_{i-1}(x)) - \deg(Q_{i-1}(x))$$

$$\sigma_{i-1} = \begin{cases} 1, & \text{if } l_{i-1} \geq 0 \\ 0, & \text{if } l_{i-1} < 0 \end{cases}.$$

where a_{i-1} and b_{i-1} are the leading coefficients of $R_{i-1}(X)$ and $Q_{i-1}(X)$ respectively. The algorithm stops when $\deg(R_{i-1}(x))<t$. If the stop condition is satisfied, then

$\sigma(X)=Li(x) \quad \omega_0(X)=Ri(X)$

- **Chien Search:**

Chien search is the algorithm to find the root of $\sigma(X)$. Every nonzero element β of $GF(2^m)$ are generated in sequence$1,\alpha,\alpha^2$.., and the inverse of which is substituted into $\sigma(X)$ to see whether $\sigma(\beta^{-1})$ is zero or not. After all the β^{-1} is tested, the error locations are known.

- **Forney Algorithm:**

RS code is a non-binary error-correcting code. In the decoding, we need to recover both the error locations and the error magnitudes. The algorithm to calculate the magnitude is as follow.

$$e_{n-x}(X)=\frac{-\omega(a^{-(n-x)})}{\sigma'(a^{-(n-x)})}=\frac{-\omega(a^{x})}{\sigma'(a^{x})},$$

where $\sigma'(X)$ is the derivative of $\sigma(X)$. Once the error locations and error magnitudes are known, the estimated codeword is generated.

The introduction of algorithm of Reed-Solomon code which is mentioned before is useful preparation for our Reed-Solomon encoder and decoder hardware implementation. The more detail is discussed in chapter 5.

3.3.3.8 Source decoder [8]

When the signal has passed through inner receiver and outer receiver, the remaining work is how to make the compression of original information go back to the raw data bits. The block among the entire system to do this work is called source decoder. In practical, in order to reduce transmitting information load, the transmitter side always use source coding technique to compress data in lossless or loss way. Source decoder in receiver side could decompress the data and recover the raw data, and then give them to the upper layer for further application.

3.4 Summary

In this chapter, we briefly review the whole communication system of LEO satellite by referencing book and MEOS capture operational manual. To know this basic concept of entire system can help us to do analysis of whole system structure.

We will focus on IO 21000 board of MOES capture system in this project. More discussion about interface analysis of IO 21000 board is given in chapter.

4 Interface analysis of IO 21000 board

4.1 Overview

Our optimal goal of this project is trying to reproduce the whole IO 21000 board and able to produce a new board to replace the original. The first important thing we should do is to analyze what the interface signals are and how they flow among each components of the board. In addition to interface analysis, we must study the function of the IO 21000 board at the same time. Four major function of IO 21000 board are Frame sync, Pseudo Random Number (PRN) generator, Cyclic redundancy checker

(CRC), and Reed-Solomon decoder. Compared to other three functions, the complexity of Reed-Solomon decoder is harder. Hence, we study and implement Reed-Solomon decoder in FPGA board to simulate its behavior on IO 21000 board.

In this chapter, we focus on the interface analysis of IO 21000 board and the further study and other related issues of Reed-Solomon decoder implementation is left for chapter 5.

4.2 Introduction to IO21000 board

Before we go to further study for the IO 21000 board, the role of the IO 21000 board in the whole LEO satellite system is needed to understand. Generally speaking, a simplified LEO satellite receiver system block diagram could be shown in the following Fig 4.1. Only important and representative block are listed in this figure.

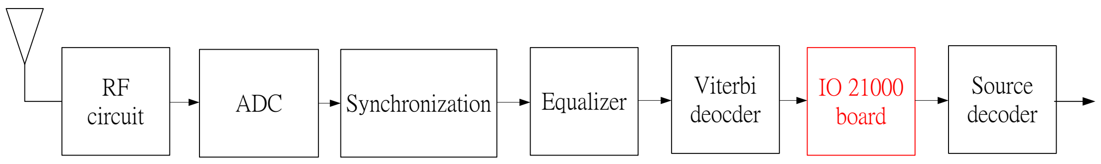


Fig. 4.1 The role of IO 21OOO board in MEOS capture system

When the signal is transmitted from the satellite to the ground station, it will pass many blocks in the receiver side. After antenna, the first block is RF circuit whose function is to down-convert the signal into the baseband. Then, the baseband signal gets into ADC and the digital form signal is generated at ADC output. The task of Synchronization block is to let the output signal of this block free from the influence of phase offset, timing offset and frequency offset. Then, the compensated signal enters equalizer which eliminates the channel distortion effect such as multipath fading. The output of equalizer gets into the well-known Viterbi decoder which always used to decode the signal encoded by convolutional code. No matter the Viterbi-decoder is soft-decision or hard-decision, the output are bit-level signal. That is to say, the input of IO 21000 board is digital signal which carries information "1" or "0". Intuitively, the output of IO 21000 board are also digital signal and the source decoder is not included in the function of this board. The functions of IO 21000 board are:

- Frame synchronization
- PRN descrambler
- CRC checker
- Reed-Solomon decoder

Kongsberg is a satellite communication company in Norway whose main product is to provide a total solution of the satellite communication receiver. In this project, IO 21000 board is one components of Kongsberg total LEO satellite receiver solution. The hardware figure of IO 21000 board can be shown in Fig. 4.2. Two major interface that Differential ECL interface and PCI-X interface are easily observed in the Fig. 4.2. Many hardware components such as Xilinx FPGA processor, Intel IO processor, Xilinx programmable ROM, IDT FIFO buffer, etc, are also the critical points for our interface analysis of our project. More descriptions and details are given in the following section.

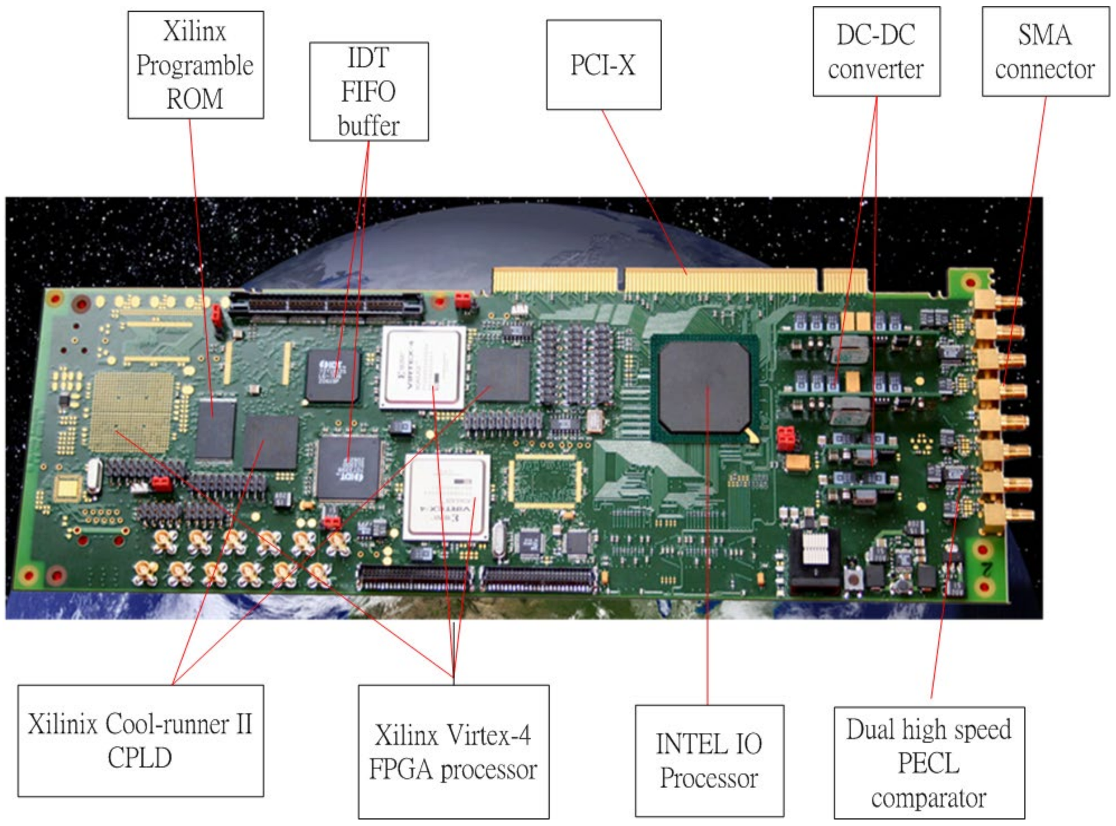

Xilinx Programble ROM
IDT FIFO buffer
PCI-X
DC-DC converter
SMA connector
Xilinix Cool-runner II CPLD
Xilinx Virtex-4 FPGA processor
INTEL IO Processor
Dual high speed PECL comparator

Fig. 4.2 Kongsberg IO 21000 board [6]

4.2.1 CCSDS [10],[19]

CCSDS is an abbreviation of Consultative committee for space data systems which is a committee that establish the space transmission standards. CCSDS publishes several technical recommendations on a data format and transmission methods for telemetry and tele-command. Three major standards, Packet telemetry recommendation, tele-command recommendation and advanced orbit systems (AOS) recommendation will be introduced in the following.

Packet telemetry recommendation mainly concerns the transmission of remote measurement signal. Standard data unit is defined for transferring telemetry from spacecraft to the ground station. Two data units for sending telemetry are defined as follows:

- Source packet
  - This format is set up for upper such as application, transport, network layer of the system.
- Transfer frame
  - This format is dedicated for Data link , Physical layer and RF link of the system.

Tele-command recommendation describes the transmission of remote control signal. Standard data unit is defined for sending commands from ground station to spacecraft. Two data units for sending telemetry are defined as follows:

- ■ Layer 3 defines a standard packet:
    - ◆ TC packet
        - ● Used for packing a command for an onboard application
        - ● Identical to the source packet defined in packet telemetry
- ■ Layer 2 defines a data unit:
    - ◆ TC transfer frame

Adavanced-orbiting-system (AOS) recommendation describes the transmission of remote signal among from ground station and spacecraft. Standard data unit is defined for exchanging commands and telemetry between ground stations and advanced space systems like the international space station. Two data units for sending telemetry are defined as follows:

- ■ CCSDS Packet
- ■ VCDU
    - ◆ Similar to Transfer frame defined in Packet Telemetry but different in some way to accommodate service for high-speed and real-time communications (such as video and voice)

4.3 Functional spec of IO21000 board

From now on, four major functions of the IO 21000 board have roughly been introduced before. In this section, we continuously go further inside about discussion of four function of the IO 21000 board. Here is the sequential relation of four functions in Fig. 4.3. In addition, much software information about our Formosa II which I get in the MEOS software is also given in each section. Algorithm of each function is also introduced.

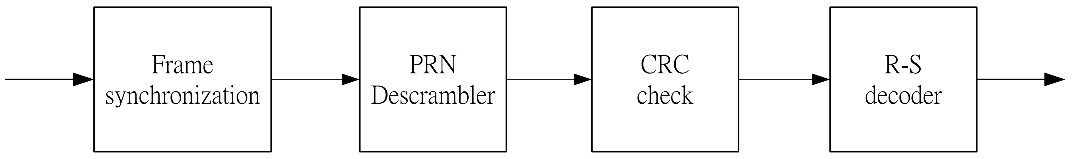


Fig. 4.3 IO 21000 board functional block diagram [6]

4.3.1 Frame Synchronization

To gain the original signal of each sensor which multiplexed the measurements signal in the frame from Formosa II satellite to the ground station, frame synchronization is necessarily for the baseband receiver to locate the exact frame start point and then can extract information for each sensor for upper layer application. We also describe the software information of particular interest in our interface analysis and algorithm of frame synchronization later.

4.3.1.1 Software information

Software information of frame synchronization which related to our interface analysis gained from MEOS capture system is shown as follows:

✓ **Software information:**

- Frame size 16 to 65536 bytes ➔ 1279 Bytes
- Sync pattern size: 8 to 64 bits ➔ 4 Bytes
- Synchronization strategy: Search-check-lock
  - Check to lock threshold : 0 to 15 frames
  - Lock to check threshold : 0 to 15 frames
- Sync pattern tolerance:
  - Bit errors: 0 to 31 bits errors
  - Bit slips: 0 to 7 bits slips

4.3.1.2 Algorithm

The algorithm of frame synchronization utilizes the pattern to recognize if the frame is correctly identified. Once the input frame is received, the first four bytes will compare to synchronization pattern and see if the received bits match synchronization pattern. In IO 21000 board, the synchronization pattern is often a pre-determined according to Consultive Committee for Space Data System (CCSDS) standard [10]. It is shown as follows

0x1acffc1d

Fig. 4.4 Synchronization pattern in CCSDS standard [10]

The core spirit of algorithm of the IO 21000 board is the search-check-lock strategy. The state diagram is shown in Fig. 4.5. Three states are search, check and lock play an important role in frame synchronization. When we received the first frame, we examines the sync pattern to see if any disagreement with known sync pattern. Then, if no error, the state of frame synchronization forwards to check state and if there are errors, the state remains the same situation. Next, following second and three or more frames run the same examination. If the error frame numbers are larger than some pre-determined value, the state forwards to lock state. The action of the Lock-to-Check-to-Search state transition is similar to the above mentioned Search-Check-Lock action procedure.

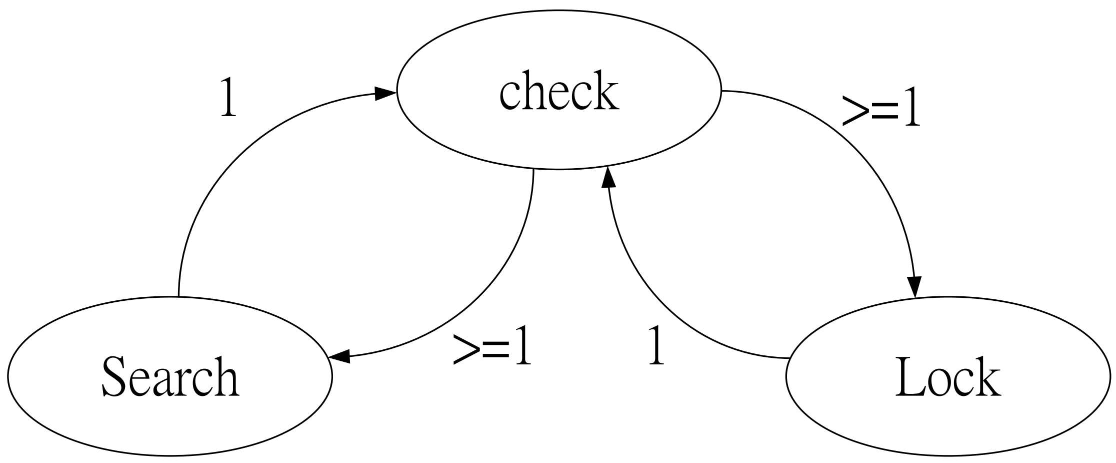


Fig. 4.5 Search-check-lock synchronization strategy state diagram [7]

4.3.2 Pseudo Random Number (PRN) descrambler

To help the synchronization of the receiver, we often need a block called scrambler which make exclusive-or addition with information bit at transmitter side. Hence, it makes sense that we need a descrambler to recover the encode bits at receiver side. Here are the soft information in MEOS capture system and algorithm of PRN descrambler. Discussion about interface analysis of the IO 21000 board can go more fluent after knowing these information.

4.3.2.1 Software information

Software information of PRN descrambler which related to our interface analysis gained from MEOS capture system is shown as follows:

✓ **Software information:**

- frame start offset: programmable from 0 to 255 bytes ➔ 4 bytes
- frame end offset: programmable from 0 to 65535 bytes ➔ 1278 bytes

4.3.2.2 Algorithm

In IO 21000 board, the generator polynomial of descrambler is chosen according to Consultive Committee for Space Data System (CCSDS) standard [10]. It is shown as follows:

$$\mathbf{g(x) = x^8 + x^7 + x^5 + x^3 + 1}$$

where g(x) means that there are nine registers in generator of PRN. The implementation of PRN generator only need register and exclusive-or addition unit. Every input bit performs exclusive-or addition with each output bit of PRN generator. Then, the descrambled bit is generated.

4.3.3 CRC checker

For detecting if the received frame is error, CRC checker is used in this system. After we know the remainder of each frame, we can judge if there is an error in such frame. Then, the receiver can decide to do retransmit or just pass the error message to upper layer which have the decision right to reserve or discard the frame. It is worthwhile to note that our MEOS capture system close the CRC checker function. That is to say, no error detection mechanism is in our board. For convenience of analysis, soft information of MEOS capture system and algorithm of CRC checker are given in next two sections.

4.3.3.1 Software information

Software information of CRC checker which related to our interface analysis gained from MEOS capture system is shown as follows:

✓ Software information:

- ◆ frame start offset: programmable from 0 to 65535 bytes
- ◆ frame end offset: programmable from 0 to 65535 bytes

4.3.3.2 Algorithm

In IO 21000 board, the generator polynomial of CRC checker is chosen according to Consultive Committee for Space Data System (CCSDS) standard [10]. It is shown as follows:

$$\mathbf{g(x) = x^{16} + x^{12} + x^{5} + 1}$$

A 16-bit cyclic redundancy checksum is calculated by g(x) in finite field GF(2) operation. The remainder of CRC which is calculated by g(x) always appends to frame tail. Implementation of CRC checker to calculate the remainder can also use the combination of shift register and exclusive-or addition. It is widely used in many communication systems.

4.3.4 Reed-Solomon Decoder

Reed-Solomon code is the main point of IO 21000 board and it is also our major focus on this project. We have introduced the basic concept of Reed-Solomon code in section 3.3.3.7 which encoder and decoder are also included. Hence, we don’t described them again in this section. Reference to section 3.3.3.7 or chapter 5 is the best choice.

4.3.4.1 Software information

Software information of Reed-Solomon decoder which related to our interface analysis gained from MEOS capture system is shown as follows:

✓ Software information:

- Two mode R-S (255,223) and R-S (10,6) of R-S decoder ➔ R-S (255,223)
- Codeword length: from 33 to 255 bytes. ➔ 255 Bytes
- Interleaving factor: from 1 to 16. ➔ 5
- frame start offset: programmable from 0 to 255 bytes ➔ 4 bytes →1278 Bytes
- coding block start offset: programmable from 0 to 15 bytes ➔ 4 bytes

4.3.4.2 Algorithm

Because introduction of algorithm of R-S encoder and decoder have been introduced before. The further discussion can trace back to the section 3.3.3.7 or go to the chapter 5.

4.4 Hardware components of IO21000 board

In this section, we start to discuss about further detail introduction of our hardware elements of the IO 21000 board. Basic function and block diagram of these components that we can find in Fig. 4.2 are described sequentially.

4.4.1 Dual high speed PECL comparator

There are two Dual high speed PECL chips on this board due to the two pair of data and clock input of the board. Each pair has one positive and negative connector for data and clock transmission. The main function of Dual high speed PECL comparator is like a one-bit decision device. When the input is higher than zero voltage, it makes a high logic level decision and the output is positive high. If the input is negative than zero, the low logic level decision is made and output is negative high. Note that the input of PECL comparator is analog amplitude signal and output is digital signal. The manufacture company of this chip is Analog device and the type is ADCMP 561. In the following, we summarize some important features of Dual high speed PECL comparator. Other description which the reader is interested in can go through [20].

- Features:
  - Differential PECL compatible outputs
  - 700-ps propagation delay input to output
  - 75-ps propagation delay dispersion
  - Input common-mode range: -2.0V to 3.0V
  - Robust input protection
  - Differential latch control
  - Internal latch pull-up resistors
  - Power supply rejection greater than 85 Db
  - 700-ps minimum pulse width
  - 1.5 GHz equivalent input rise time bandwidth
  - Typical output rise/fall time of 500-ps

- ESD protection > 4kV HBM > 200V MM
- Programmable hysteresls

The functional block diagram is in the following Fig. 4.6:

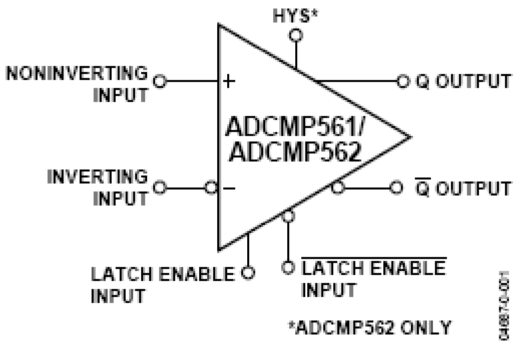


Fig. 4.6 Functional block diagram of Dual high speed PECL comparator [20]

4.4.2 Austin LynxTM DC-DC converter

DC to DC converter is a common device which provides DC voltage transformation and also provides the protection from the fluctuation form other interference in digital circuit. The manufacture company of this chip is Austin. The type of this chip is SIP (Non-isolated power modules) AXH010A0X3Z – CC109104923.

In the following, we summarize some important features of Dual high speed PECL comparator. Other description which the reader is interested in can go through [21].

- Features:

■ Non-isolated

■ DOSA industry standard package & pin out (SIP)

■ Wide input voltage range

■ Up to 10A output current

■ Adjustable output voltage via external resistors

■ Remote On/Off Control

■ Remote sense

■ Low output noise and ripple

■ Under-voltage lockout, over-current and over-temperature protection

The following Fig. 4.7 is the physical figure of Austin LynxTM DC-DC converter.

There are ten pins in this DC-DC converter.

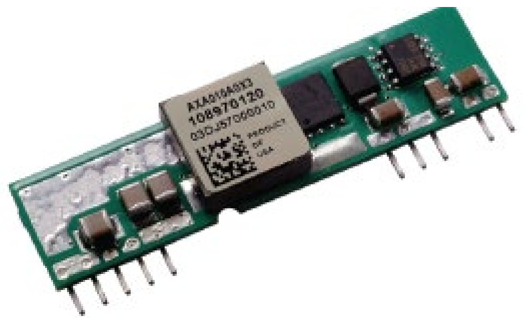


Fig. 4.7 Austin Lynx DC-DC converter [21]

4.4.3 TDK LAMBA DC-DC converter

DC to DC converter is a common device which provides DC voltage transformation and also provides the protection from the fluctuation form other interference in digital circuit. The manufacture company of this chip is different from former section. It is another company called TDK LAMBA which produces this kind of DC-DC converter. The type of this chip is PL5C-05C and IC number is AXH010A0X3Z – CC109104923. Next, we summarize some important features of TDK LAMBA DC-DC converter. Other description which the reader is interested in can go through [22].

- Characteristics
  - High Efficiency up to 94%
  - Reduces Input Current Draw
  - Wide Output Voltage Adjustment Range
  - Stock One Part for all Voltages
  - SMT or Through Hole Packages
  - Multiple Mounting Methods
  - Industry Standard Pin Out
  - Second Sourcing

The following Fig 4.8 is the TDK-LAMBA DC-DC converter appearance.

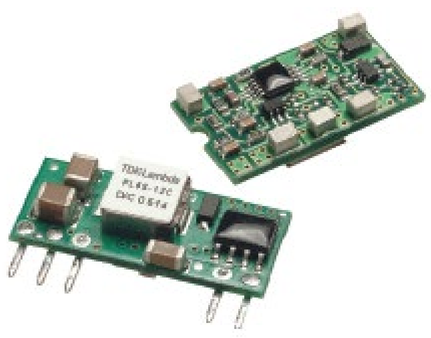


Fig. 4.8 TDK-LAMBA DC-DC converter [22]

4.4.4 Intel IO Processor

Intel IO processor is the critical role of this board that it provides the regulation and scheduling of input and output digital signal. Instruction set of Intel IO processor is ARM instruction set. The design company of this processor is Intel and the type of IO processor is FW80321M600. In the following, we summarize some important features of Intel IO processor. Other description in which the reader is interested can go through [11].

- Product features:
  - Core features:
    - ARM V5T instruction set and V5e DSP extension
    - 400 MHz and 600 MHz
  - PCI Bus interface
    - 64-bit/133MHz operation in PCI-X mode.

- Memory controller
- Address translation unit
- Dynamic memory access DMA controller
- Application accelerator unit
- IIC Bus interface
- SSP serial port
- Peripheral performance monitoring unit
- Timers
- 554-Ball, Plastic Ball Grid Array (PBGA)
- Eight general purpose I/O pins

4.4.5 Xilinx Virtex-4 FPGA processor

Xilinx FPGA processor is a general purpose processor that we can burn our code on it and determine what kind of behavior the processor should be. Combining advanced silicon modular block architecture with a wide variety of flexible features, the Virtex-4 family from Xilinx greatly enhances programmable logic design capabilities, making it a powerful alternative to ASIC technology. The manufacture company of this chip is Xilinx and the type of this chip is XC4VLX25-FF668-10C/I. In the IO 21000 board, three FPGA processors are embedded for specific purpose. Summary of Xilinx Virtex-4 FPGA processor is in the later. Other description in which the reader is interested can go through [16].

- Features:
  - Virtex-4 LX : High performance logic application solution.
  - Xesium clock technology
  - XtremeDSP slice
  - Smart RAM memory Hierarchy
  - SelectIO technology
  - Flexible logic resource
  - Secure chip AES Bitstream encryption
  - 90-nm copper CMOS process
  - 1.2 V core voltage
  - Flip-chip packaging including Pb-free package choices

4.4.6 Xilinx CoolRunner-II CPLD

Xilinx CPLD is another processor of this board that it provides the main device for designation of combinational logic circuit. The Coolrunner II device is designed for both high performance and low power application. This lends power savings to high-end communication equipment and high speed to battery operated devices. The manufacture company of this chip is Xilinx and the type of this chip is XC2C256 Coolrunner-II. In the following, we summarize some important features of Xilinx Coolrunner-II CPLD. Other description which the reader is interested in can go through [13].

- Features:
  - Optimized for 1.8V systems
  - Industry's best 0.18 micron CMOS CPLD
  - Available in multiple package options
  - Advance system features

4.4.6.1 Comparison of FPGA and CPLD

Programmable logic device (PLD) is a digital integrated circuit whose logic function can be designed by users themselves. PLD is classified as three categories by architecture and logic cell density. That is simple programmable logic device (SPLD), Complex programmable logic device, and field programmable gate array. SPLD is a small scale PLD while CPLD and FPGA are the large scale PLD. With use of SRAM technique, PLD needs an EPROM or Flash to store the programming message. Interconnection of logic cell inside FPGA is segmented interconnected and the interconnection of CPLD is continuous interconnected that other characteristics are predictable delay and faster global interconnect. Generally speaking, the speed of CPLD is faster than FPGA. However, FPGA is more flexible than CPLD in programming. FPGA is suitable for more flip-flops design architecture and CPLD is suitable for finite flip-flops and more product term design architecture.

4.4.7 Xilinx programmable ROM

Xilinx programmable ROM is another critical role of this board that many instructions are saved in PROM which is used for processor arithmetic operation. The manufacture company of this PROM is Xilinx and the type of this PROM is XVF32VOG48. In the following, we summarize some important features of Xilinx Programmable ROM. Other description which the reader is interested in can go through [12].

Features:

- In-system programmable PROMs for configuration of Xilinx FPGAs

- Low-power advanced CMOS NOR FLASH Process
- Endurance of 20000 Programs/Erase cycles
- Operation over full Idustrial Temperature range (-45’c to +85’c)
- IEEE standard 1149.1/1532 Boundary-scan (JTAG) Support for programming, prototyping, and testing.
- JTAG Command initiation of standard FPGA Configuration.
- Cascadable for strong longer or multiple bitstreams
- I/O pins compatible with voltage levels ranging from 1.5V to 3.3V
- Design support using the Xilinx Alliance ISE and foundation ISE series software packages.
- 1.8V supply voltage and serial or parallel FPGA configuration interface (up to 33MHz)
- Built-in data de-compressor compatible with Xilinx advanced compression technology

4.4.8 Numonyx strata Flash embedded memory

Numonyx strata flash is another critical role of this board that many instructions are saved in flash which is dedicated for Intel processor to perform arithmetic operation. The manufacture company of this flash is Numonyx and the type of this flash is strata flash embedded memory. In the following, we summarize some important features of Numonyx strata flash embedded. Other description which the reader is interested in can go through [23].

- Features
  - High performance
    - 85 ns initial access and 52 MHz with zero wait states, 17ns clock-to-data output synchronous-burst read mode
    - 1.8V buffered programming at 7 us/Byte (Typ)
    - Architecture
      - Multi-level cell technology: Highest Density at lowest cost
      - Asymmetrically-blocked architecture
      - Standby current: 20 uA for 64 Mbit
      - Four 32-Kbyte parameter blocks: top or bottom configuration
    - Voltage and Power
      - Vcc: 1.7-2.0 V
      - 4-word synchronous read current: 13 mA at 40 MHz
    - Quality and reliability
      - Operating temperature: -40'C to +85 'C
      - Minimum 100,000 erase cycles per block

- ETOX(TM) VIII process technology

4.4.9 IDT FIFO buffer

IDT FIFO buffer is another important point of this board that it provides the extra space for registering the calculated signaling or others which are waiting to be called signal. FIFO buffer is an abbreviation of fist-in-first-out that means the first input signals are first output when system calls for them. Similarly, the last input signals to buffer are the last served signal to the buffer output. The manufacture company of this chip is IDT and the type of this chip is 72V36104, 72T18125. In the following, we summarize some important features of IDT FIFO buffer. Other description which the reader is interested in can go through [14]-[15].

- Features:
  - Clock frequencies up to 100 MHz (6.5 ns access time)
  - Two independent clocked FIFOs buffering data in opposite directions
  - Programmable Almost-Empty and Almost-Full flags; each has five default offsets (8,16,64,256, and 1024)
  - Serial or parallel programming of partial flags
  - Retransmit capability

- Big- or Little-Endian format for word and byte bus sizes
- Master reset clears data and configures FIFO, Partial reset clear data but remains configuration setting
- Mailbox bypass registers for each FIFO
- Auto power down minimizes power dissipation
- Industrial temperature range -40’c to + 85‘c is available

4.5 Interface analysis of IO 21000 board

After introducing so many components of IO 21000 board, we think we should continue to go to the hardware interface analysis. In this section, the most possible hardware connection interface of this board. The inference of hardware interface connection refers to coordination of MEOS Capture User manual, data sheet of all components, observation of software system in NSPO, and measurements of hardware board in NSPO, etc. There are two directions in our interface analysis. One is software interface and the other is hardware interface analysis. Software interface analysis involves how the software instruction to control hardware circuit operation. It is so difficult that if we don’t know the original design pattern and try to recover how the software work on hardware cost a lot of time. For example, in MEOS capture system, we can close the IO 21000 hardware board function by software control. To know how the signal flow from your computer to the circuit board and how the protocol is performing require more effort to overcome many challenges. Hardware board interface still have some difficulty. One of main important problem is routing of clock line and data line. Due to modern PCB board implementation, there are always above two layers board (usually 4~12) combination that each layer can design its own circuit line. Hence, to unveil the layout routing mystery of IO 21000 board are time-consuming work.

In order to efficiently analyze the interface signal, we first focus on the hardware interface of the IO 21000 board. By reference of MEOS capture system and hardware components data sheets, we derive the most possible hardware connection way in this section. Each possible connection interface has its own representation number for convenience. Next, we go to further discussion.

4.5.1 Most possible hardware connection of IO21000 board

From the previous section, we know that all we could do first is the hardware interface analysis. By integrating software information observing in NSPO and hardware data sheet, we try to divide the main hardware interface into 11 portions for which each has a number stands. In Fig. 4.9, we call the eleven important connection lines between each block because we still not verify the connection by any method. Although some difficulty we meet during measurements, we still do our best to guess the interface connection as correctly as possible.

We classify the interfaces into two major categories. One is the so-called outward-board interface and the other is inward-board interface. The outward-board interfaces are exactly two main interfaces that are

- 1. Differential ECL interface
- 2. PCI-X interface

Which are remarked as blue line in Fig. 4.9 and we have the discussion in section 4.6. The inward-board interface are 3 to 11 total nine possible interfaces which are used black line to represent them. Nine interfaces are:

- 3. Dual high speed PECL comparator ⇔ DC to DC converter
- 4. DC to DC converter ⇔ Intel IO processor
- 5. PCI-X interface ⇔ Intel IO processor
- 6. Intel IO processor ⇔ IDT FIFO buffer
- 7. Intel IO processor ⇔ Strata flash memory
- 8. Xilinx FPGA processor ⇔ IDT FIFO buffer
- 9. Xilinx FPGA processor ⇔ Xilinx programmable ROM

- 10. IDT FIFO buffer ⇔ Xilinx Coolrunner CPLD
- 11. Xilinx programmable ROM ⇔ Xilinx Coolrunner CPLD

And we will defer the detailed interface 3 to 11 analyses to section 4.7.

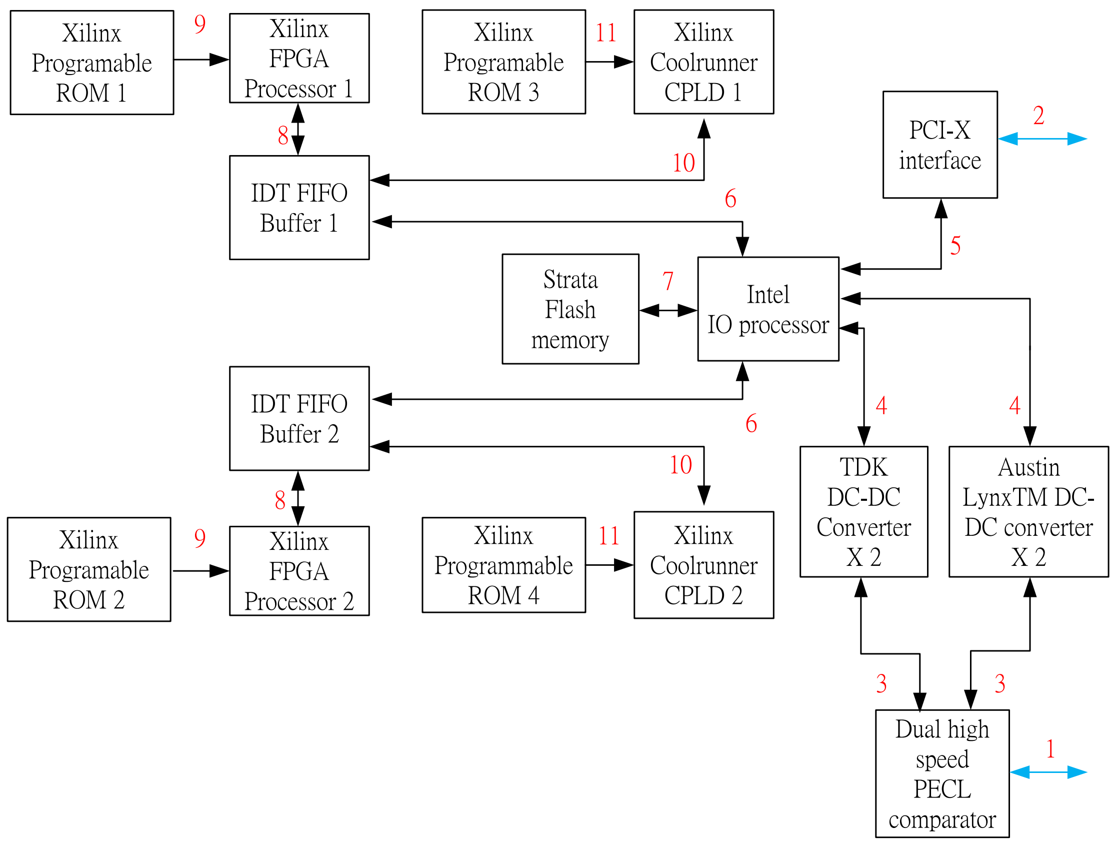


Fig. 4.9 Most possible hardware connection diagram of IO 21000 board

4.6 Outward-board hardware interface I/O frame structure

In order to exactly analyze all interfaces 1 to 11 in IO 21000 board, inter-board 1-2 interfaces and inter-board 3-11 interfaces connection problems are both required to be solved. The first important thing we should do is try to get the input and output data format that another method to replace all IO 21000 board may be feasible. In other words, we can consider the IO 21000 board as a black box and if we know what the input and output is, we have large probability to reproduce entire board. Due to four known functions of IO 21000 board, it is more easy to reproduce the same functions on another board by ourselves own capability that the implementation way may be different from this board than definitely reproduce the board with the same implementation way. However, it is not our original purpose to replace the board by regarding this board as a block box. Hence, inner-board interfaces still go ahead until the effort we can make as best as possible.

1. Differential ECL interface:

By observing the software information and operation manual in MEOS capture system, we deduce that two input and output frame formats are most possible input and output to IO 21000 board. In Fig. 4.10, we see that the interface 1 with two input frame formats. Those are Virtual channel data unit (VCDU) frame format for Kongsberg IO 21000 board and DRD frame format for Formosa II which is two most possible frame formats of input of IO 21000 board. We know each frame sizes and synchronization length but we don’t know what the content is in the data field and VCDU header or DRD header and its data field.

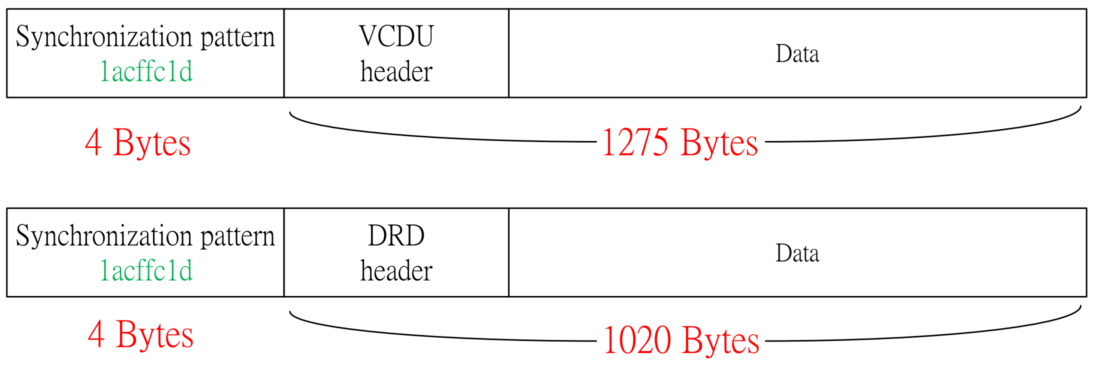


Fig. 4.10 Frame structure of differential ECL interface input of IO 21000 board

2. PCI-X interface [17],[18]:

Interface 2 which is PCI-X interface also has two output frame formats shown in Fig. 4.10. The same as we discuss before, DRD and VCDU format both have their own frame format which are a little bit different from interface 1 format. In Fig 4.11, we could see that two different things from before are that PCI-X header and Quality signal added to the original frame. PCI-X interface is a general standard used in industrial and need a lot of time to be familiar with, but it is not hard to reproduce, just time-consuming. Hence, in this analysis, we assume the PCI-X header is known [17],[18] and we focus more on quality signal field.

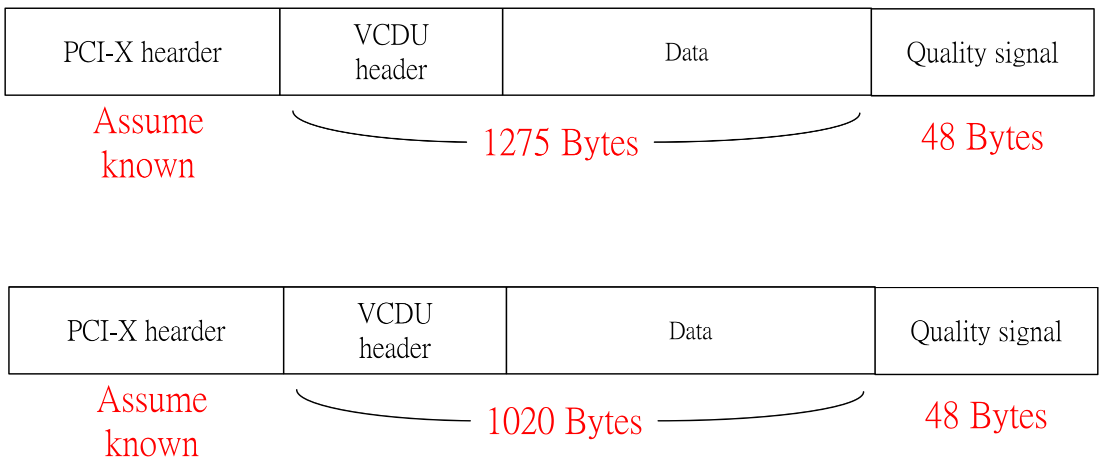


Fig. 4.11 Frame format of PCI-X output of IO 21000 board

Quality status field which is appended to the tail of two entire frames consist of three parts: time tag information, frame synchronization status, Reed-Solomon code status. The sequence of the quality signal we can see from Fig 4.12.

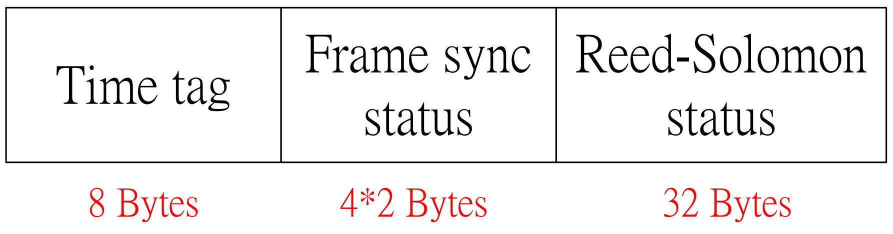


Fig. 4.12 The sequence of quality signal [7]

Time tagging field of quality signal records the time information with seconds, minutes, hours, and date. Eight bytes needs to represent what time the frame is received. The top fields meaning of table which are byte, bit, bit-definition are the byte sequence, bit sequence, and what the bits stand for. All sequence from zero to seven means from LSB to MSB. The exact bit position and what each bit means is in following Table 4.1.

- Time tagging format:

Byte	Bit	Bit definition
0	7-0	Day number (15-8)
1	7-0	Day number (7-0)
2	7-3	All zeros
2	2-0	Mili-seconds of Day (26-24)
3	7-0	Mili-seconds of Day (23-16)
4	7-0	Mili-seconds of Day (15-8)
5	7-0	Mili-seconds of Day (7-0)
6	7-2	All zeros
6	1-0	Micro-seconds of Mili-seconds (9-8)
7	7-0	Micro-seconds of Mili-seconds (7-0)

Table 4.1 Bytes information of time tagging [7]

Frame synchronization consists of much information about frame status such as frame length, sync errors, slip frames, and etc. Note that two repetitive frame sync status are in that field in this board by observing the output frame of IO 21000 board in term of software [24]. Hence, 4*2 bytes are used to represent all frame sync status information. The exact bit position and what each bit means is in following Table 4.2.

- Frame synchronization status format

Byte	Bit	Bit definition
0	7-6	Sync state
0	5	Inverted Frame
0	4	CRC errors
0	3	Best match frame
0	2	Flush frame
0	1	Slip frame
0	0	Slip detection
1	7-5	Slip bits
1	4-0	Sync errors
2-3	15-0	Frame length

Table 4.2 Byte information of frame sync status [7]

Reed-Solomon status are the last field of quality signal that consists of codeword error numbers, bit error numbers, codeword error location and bit error location and etc. 32 bytes are need for record of Reed-Solomon status signals. The exact bit position and what each bit means is in following Table 4.3.

- Reed-Solomon status:

	Byte							
Bytes	0		1		2		3	
0-3	Static annotation #1				Static annotation #2			
4-7	Static annotation #3				Static annotation #4			
8-11	Frame Cunter							
12-15	Reserved		Frame Qality		Rejection and errors			
16-19	Corrected codeword numbers		Uncorrectable codeword numbers		Errored bitsmap			
20-23	Uncorrectable bitmap				Bits error numbers			
24-27	CW 16 Error #	CW15 Error #	CW14 Error #	CW13 Error #	CW12 Error #	CW11 Error #	CW10 Error #	CW9 Error #
28-31	CW 8 Error #	CW7 Error #	CW6 Error #	CW5 Error #	CW4 Error #	CW3 Error #	CW15 Error #	CW1 Error #

Table 4.3 Byte information of Reed-Solomon status [7]

- CW 16 error # means error numbers of codeword 16

According to MEOS Capture operation manual, we find that the IO 21000 board has some relation with CCSDS AOS standard. Hence, we make a hypothesis that AOS transfer frame may be the input and output frame format of IO 21000 board. We revisit CCSDS AOS standard and find out the AOS transfer frame at Fig 4.13.

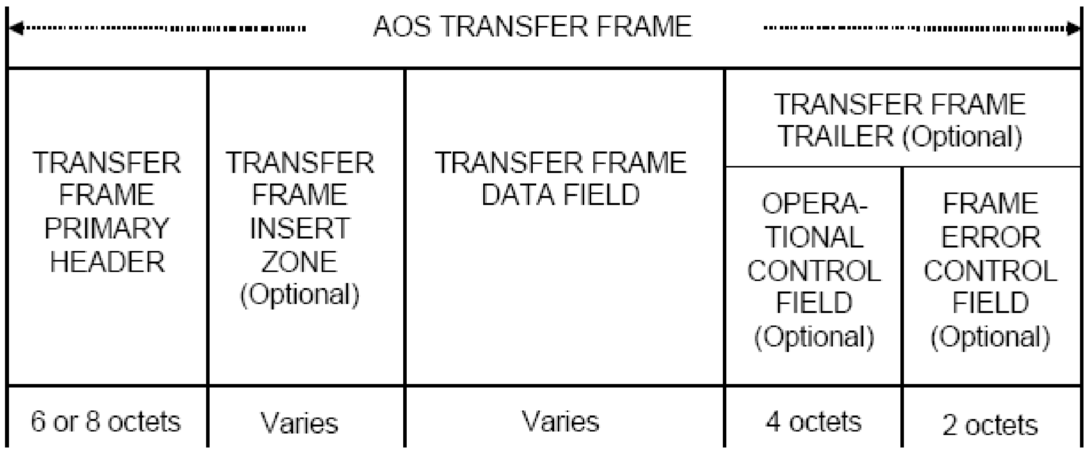

AOS TRANSFER FRAME
TRANSFER FRAME PRIMARY HEADER
TRANSFER FRAME INSERT ZONE (Optional)
TRANSFER FRAME DATA FIELD
TRANSFER FRAME TRAILER (Optional)
OPERA-TIONAL CONTROL FIELD (Optional)
FRAME ERROR CONTROL FIELD (Optional)
6 or 8 octets
Varies
Varies
4 octets
2 octets

Fig. 4.13 AOS transfer frame [25]

At AOS transfer frame, we consider the transfer frame as our indicator to identify if the AOS transfer frame is our desired frame. Fig 4.14 is primary header of AOS transfer frame.

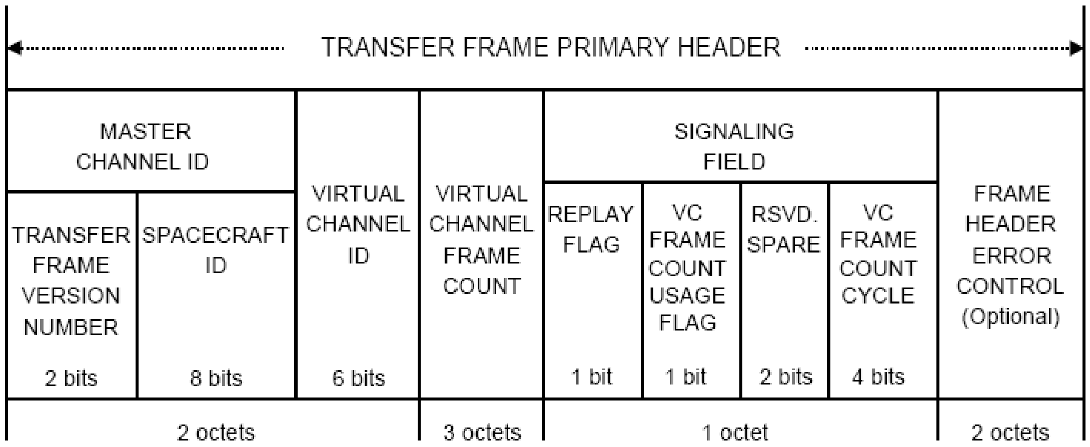


Fig. 4.14 Transfer frame primary header of IO 21000 board [25]

By examining the only information of output frame format of IO 21000 board [24], we can clearly find that there are some field with constant bit from frame to frame which are:

- Transfer Frame Version Number (TFVN) : 2 bits
- Spacecraft Identifier (SCID) : 8 bits
- Virtual Channel Identifier (VCID): 6 bits
- Signaling field: 1 byte
    - Reply flag: 1 bit
    - VC frame count usage flag: 1 bit
    - RSVD spare: 2 bits
    - VC frame count cycle: 4 bits

These four columns are invariant and only virtual channel frame count column (3 bytes) varies frames to frames. Frame header error control is optional with 2 bytes, this column needs to be confirmed if it exists. Once the optional field is confirmed, and the remained things to prove that transfer frame primary header is one portion of our IO 21000 board frame header are to verify the TFVN, VCID, SCID, signaling field and virtual channel frame counts. The former four fields can be verified by searching if any other information about Formosa II which we have never found due to secretary or other reasons. The last filed could be verified by observing some frames information in software [24] and see if the total frame is the same as the field which records virtual channel frame counts.

After confirming above uncertain information in primary frame header, we can say that the AOS transfer frame has large probability to be our desired IO 21000 board frame format. But if we want to claim that the whole frame structure of IO 21000 board is unveiled, we should accomplish some other checks such as the boundary of transfer frame data field, optional transfer frame insert zones, and optional transfer frame trailer including operational control field (4 bytes) and frame error control field (2 bytes). If we all confirm these things, we can definitely say that we get through and know the entire frame structure. Although we don’t know what the hardware implementation is used in the IO 21000 board, we still can claim that we have capability to reproduce a board to replace IO 21000 board by our-own implementation skill if we exactly know the input and output spec and frame structure. Verification method of the proposed frame in this section describes in section 4.6.1.

4.6.1 Verification method of outward -board interface

After we proposed our frame format for outward-board interfaces which are differential ECL interface and PCI-X interface, we require a verification method to identify whether our proposed frame structure is true or not. In this section, the work we should do is to give a method to confirm if our proposed frame format is correct or not. Although the method have not been sure if it is feasible, it can provide the next research worker of this project for a direction at verification.

The main concept of verification method is to use some important pattern as a landmark to locate each frame position when we record the bits stream which enters into the input of board. This pattern in differential interface is synchronization pattern which is a 32-bit pattern. In PCI-X interface, the critical pattern is quality status signal which is a 48-byte pattern.

The procedure of this verification method is divided into two classes which are Differential ECL interface and PCI-X interface. We described them as follows:

- Differential ECL interface:
    - ✓ 1. We record all input signals by the logic analyzer.
    - ✓ 2. We utilize the synchronization pattern and quality status signal as a landmark to distinguish from frame to frame.
    - ✓ 3. We compare this hardware frame to our proposed frame and identify the extra hardware information and the uncertain column of frame.

However, after a 7-day measurement we have done described in section 4.8, we find that we can not get any signal by direct measuring differential ECL interface. In order to solve this problem, we consider it carefully and find out the reason of this problem is that there should be some protection mechanism on IO 21000 board and the logic analyzer is suitable for digital circuit and not for analog input signal.

- ✓ 4. For suggestion method to solve this problem, the probe of logic analyzer should contact with pins of dual high speed PECL comparator, instead of SMA connector of the IO 21000 board.
- ✓ 5. Once we can get the measured hardware signal from dual high speed PECL interface output, we also identify each frame contents and find out the uncertain columns of frame.
- ✓ 6. If other these unknown columns are unveiled, we can say that the differential ECL interface input frame structure of IO 21000 board is established.

- ● PCI-X interface:

The other class is the verification of PCI-X interface. We need not only the logic analyzer but also a PCI-X analyzer to carry out this interface measurement. The function of PCI-X analyzer is to help us avoid studying the complex PCI-X protocol for PCI-X interface frame format or header information. Without considering the troubling PCI-X protocol, we can focus on the verification of proposed frame format.

- ✓ 1. The procedure of PCI-X protocol is almost in common with differential ECL interface. Assume the total signal can be recorded, then we use the quality signal as our index to identify frame format from PCI-X output.
- ✓ 2. In addition, we can not only analyze if the measured frame is the same as our proposed frame structure but also can examine if any other extra signals or feedback signals exist.
- ✓ 3. After checking out the exactly hardware interface frame structure, we get the uncertain information part of frame. The remained things we should do are to identify all un-determined columns of measured frame structure.
- ✓ 4. The same as differential ECL interface, to certify all un-determined columns of measured frame, we can say that the output frame structure of IO 21000 board is established.

By determination of differential ECL and PCI-X interface frame format, we can replace the IO 21000 board by making another same-function board due to awareness of IO 21000 board function.

4.7 Inward-board hardware interface

After finishing the outward-board hardware interface, we do our best to go further discussion for our complex inward-board interface analysis in this section. We make many efforts on every inward-board hardware interface to find the possible connected pins. Besides, some interfaces doesn't have exact pin definition and location such as FPGA processor and CPLD whose pins are all configurable, and we should do more extra efforts on guessing what functions of these pins are.

In order to certify what the way of all inward-board interfaces are inter-connected, we first identify the possible inter-connected pins of interface 3 to interface 11 from related datasheets and by engineering common sense. Then, these possible connection pins will be reduced, increase or maintain after verify these possible pins by short-circuit characteristic of multi-meter. After all pins connection diagram on this board are completed, we just can go to further analysis for software interface analysis. In the following discussion, we put more emphasis on physical hardware and all related information is listed as follows, and the method how to guess and verify the connection pins function as correctly as possible is described in section 4.7.1.

3. Dual high speed PECL comparator [20] ⇔ DC-DC converter [21-22]

The pin information diagram of dual high speed PECL comparator is in Fig. 4.15:

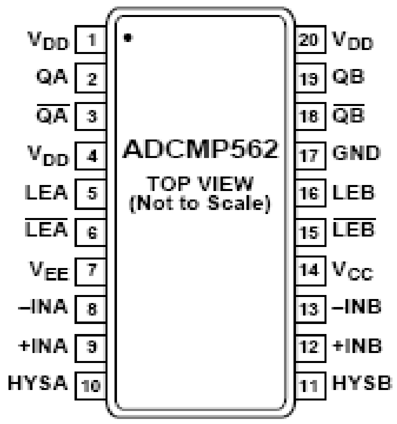


Fig. 4.15 Dual high speed PECL comparator pins location [20]

To combine the following Table 4.4 and upper pins location diagram, we can know exactly about pins function and location.

- Pin information of Dual high speed PECL comparator

Pin	Signal name	Function
1	VDD	Logic supply Terminal

2	QA	Channel A output
3	QA’	Complementary Channel A output
4	VDD	Logic supply Terminal
5	LEA	Channel A Latch enable
6	LEA’	Complementary Channel A latch enable
7	VEE	Negative supply Terminal
8	-INA	Inverting input of differential stage for channel A
9	+INA	Non- inverting input of differential stage for channel A
10	HYSA	Programmable Hysteresis input
11	HYSB	Programmable Hysteresis input
12	+INB	Non- inverting input of differential stage for channel B
13	-INB	Inverting input of differential stage for channel B
14	VCC	Positive supply Terminal
15	LEB’	Complementary Channel B latch enable
16	LEB	Channel B Latch enable
17	GND	Analog ground
18	QB’	Complementary Channel B output
19	QB	Channel B output
20	VDD	Logic supply Terminal

Table 4.4 Pin information of Dual high speed PECL comparator [20]

There are two DC-DC converters in this board. Hence, we should introduce their pins function and location individually. The first kind DC-DC converter is Austin LynxTM DC-DC converter. The pins location diagram and information is shown in Fig. 4.16and Table 4.5.

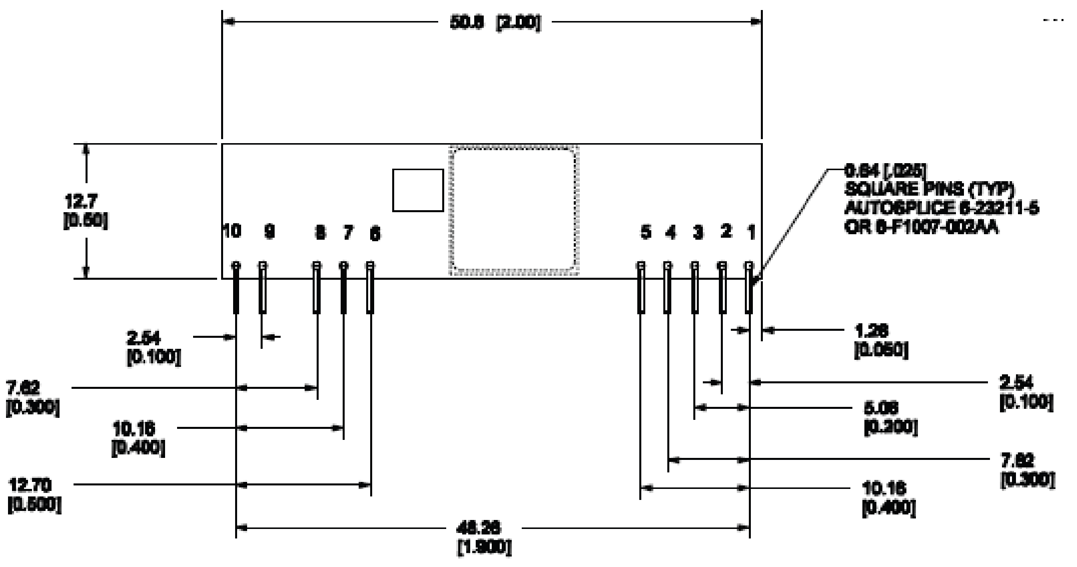


Fig. 4.16 Pin location of Austin Lynx DC-DC converter [21]

- Interface pins of Austin LynxTM DC-DC converter

Pin	Function
1	Vo
2	Vo
3	Vo,sense
4	Vo
5	GND
6	GND

7	Vin
8	Vin
9	TRIM
10	ON/OFF

Table 4.5 Pin information of Austin LynxTM DC-DC converter [21]

The second kind DC-DC converter is TDK LAMBA DC-DC converter and its pins location and diagram is shown in Fig 4.17 and Table 4.6.

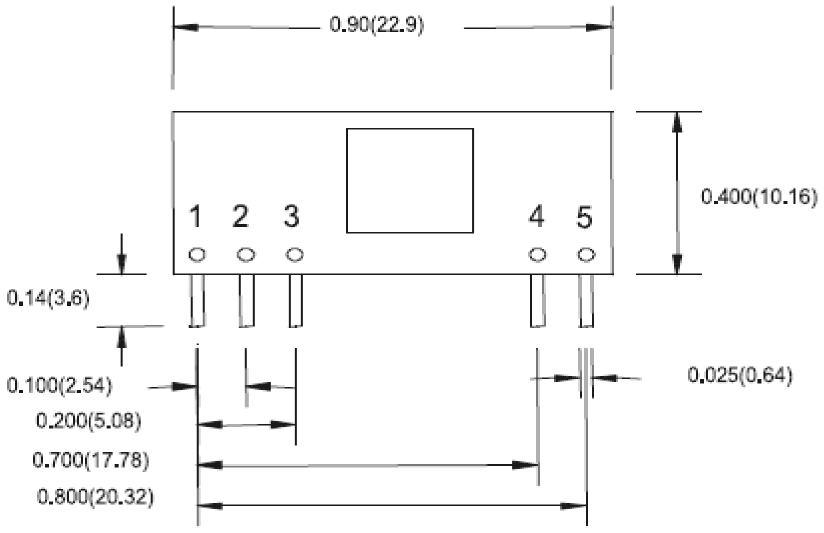


Fig 4.17 Pin location of TDK-LAMBA DC-DC converter [22]

- Interface pins of TDK-LAMBA DC-DC converter

Pin	Function
1	+ Output
2	Trim
3	Common
4	+V input
5	ON/OFF

Table 4.6 Pins of TDK-LAMBA DC-DC converter [22]

Once we know the all pins information of DC-DC converter and dual high speed PECL comparator, we can guess the most possible connected candidate pins by experience and system functional diagram. Two possible way of hardware interface connection between DC-DC converter and dual high speed PECL comparator are shown in the following.

- Possible hardware interface signal of 3 with Austin LynxTM DC-DC converter

  ✓ PECL comparator [20]

2	QA	Channel A output
3	QA'	Complementary Channel A output
18	QB'	Complementary Channel B output
19	QB	Channel B output

  ✓ Austin LynxTM DC-DC converter [21]

7	Vin
8	Vin
10	ON/OFF

- Possible hardware interface signal of 3 with TDK-LAMBA DC-DC converter

  ✓ PECL comparator [20]

2	QA	Channel A output
3	QA'	Complementary Channel A output
18	QB'	Complementary Channel B output

19	QB	Channel B output

✓ TDK-LAMBA DC-DC converter [22]

4	+V input
5	ON/OFF

4. DC-DC converter [21],[22] ⇔ Intel IO processor [11]

Interface 4 is the connection between DC-DC converter and Intel IO processor. All hardware information about DC-DC converter is discussed before. Now, we pay more attention on Intel IO processor pins. Many possible connection pins including Synchronous serial port interface and GPIO interface but not all pins are shown in following tables.

● Possible hardware interface signal of Intel IO processor [11],[21],[22]:

✓ Synchronous serial port interface of IO processor [11]

Signal name	Count	Description
SSCKO	1	SERIAL PORT CLOCK OUTPUT
SFRM	1	SERIAL FRAME
TXD	1	TRANSMITT DATA
RXD	1	RECEIVE DATA
SSCKI	1	SERIAL PORT CLOCK IN

✓ GPIO interface [11]

Signal Name	Count	Description

GPIO[3:0]	4	GENERAL PURPOSE INPUT-OUTPUT
GPIO[4]/SDA 1	1	GENERAL PURPOSE INPUT-OUTPUT / I2C DATA
GPIO[5]/SCL 1	1	GENERAL PURPOSE INPUT-OUTPUT / I2C CLOCK
GPIO[6]/SDA 0	1	GENERAL PURPOSE INPUT-OUTPUT / I2C DATA
GPIO[7]/SCL 0	1	GENERAL PURPOSE INPUT-OUTPUT / I2C CLOCK

✓ Possible Interface pins of Austin LynxTM DC-DC converter [21]

Pin	Function
1	Vo
2	Vo
3	Vo,sense
4	Vo
9	TRIM
10	ON/OFF

✓ Interface pins of TDK-LAMBA DC-DC converter [22]

Pin	Function
1	+ Output
2	Trim
3	Common

5. PCI-X interface [17],[18] ⇔ Intel IO processor [11]

Interface 5 involves the connection between PCI-X interface and Intel IO processor. As mentioned above, Intel IO processor possible pins are discussed in the previous section and the work we have done in this section is to list all possible interface of PCI-X pins information including address and data path, interrupt, error signal, arbitration, and system signal by the datasheet [17],[18].

- PCI-X hardware interface signal [17],[18]:

Signal Name	Count	Description
Address and data path		
P_AD[31:0]	32	PCI ADDRESS/DATA
P_AD[63:32]	32	PCI DATA
P_PAR	1	PCI BUS PARITY
P_PAR_64	1	PCI BUS UPPER WORD PARITY
P_C/BE[3:0]	4	PCI BUS COMMAND and BYTE ENABLES
P_C/BE[7:4]	4	PCI BYTE ENABLES
P_REQ64	1	PCI BUS REQUEST 64-BIT TRANSFER
P_ACK64	1	PCI BUS REQUEST 64-BIT TRANSFER
P_FRAME	1	PCI BUS CYCLE FRAME
P_IRDY	1	PCI BUS INITIATOR READY
P_TRDY	1	PCI BUS TARGET READY
P_STOP	1	PCI BUS STOP
P_IDSEL	1	PCI BUS INITIALIZATION DEVICE SELECT

P_DEVSEL	1	PCI BUS DEVICE SELECT
P_M66EN	1	PCI BUS 66 MHz ENABLE
	Interrupt	
P_INT[A:D]	4	PCI BUS INTERRUPT
	Error signals	
P_PERR	1	PCI BUS PARITY ERROR
P_SERR	1	PCI BUS STSTEM ERROR
	Arbitration	
P_REQ	1	PCI BUS REQUEST
P_GNT	1	PCI BUS GRANT
	System Signals	
P_CLK	1	PCI BUS INPUT CLOCK
P_RST	1	RESET

6. Intel IO processor [11] ⇔ IDT FIFO buffer [14],[15]

The interface 6 talks about Intel IO processor and IDT FIFO buffer. In this interface, Intel IO processor pins about DDR SDRAM interface pins and all IDT FIFO buffer pins are listed in following. Our main point is to get what the relation between this two hardware components. Many related pins information and function are shown as follows.

- Possible hardware interface signal of 6:

  ✓ DDR SDRAM interface signals of IO processor [11]

Signal name	Count	Description
RCVENI	1	RECEIVE ENABLE IN
RCVEN0	1	RECEIVE ENABLE OUT
M_CK[2:0]	3	MEMORY CLOCKS (POSITIVE)
M_CK[2:0]#	3	MEMORY CLOCKS (NEGATIVE)
M_RST	1	MEMORY RESET
SA[12:0]	13	MEMORY ADDRESS BUS
SBA[1:0]	2	SDRAM BANK ADDRESS
SRAS	1	SDRAM ROW ADDRESS STROBE
SCAS	1	SDRAM COLUMN ADDRESS STROBE
SWE	1	SDRAM CLOCK ENABLE
SCE[1:0]	2	SDRAM CHIP SELECT
SCKE[1:0]	2	SDRAM CLOCK ENABLE
DQ[63:0]	64	SDRAM DATA BUS
SCB[7:0]	8	SDRAM ECC CHECK BITS
DQS[8:0]	9	SDRAM DATA STROBES
SDQM[8:0]	9	SDRAM DATA MASK
Vref	1	SDRAM VOLTAGE REFERENCE

- ✓ IDT FIFO buffer pins information [14],[15]

Signal Name	I/O	Description
A0-A35	I/O	PORT A Data
AEA	O	PORT A ALMOST EMPTY FLAG
AEB	O	PORT B ALMOST EMPTY FLAG
AFA	O	PORT A ALMOST FULL FLAG
AFB	O	PORT B ALMOST FULL FLAG
B0-B35	I/O	PORT A DATA
BE/FWFT	I	BIG-ENDIAN / FIRST WORD FALL THROUGH SELECT
BM	I	BUS-MATCH-SELECT (PORT B)
CLKA	I	PORT A CLOCK
CLKB	I	PORT B CLOCK
CSA	I	PORT A CHIP SELECT
CSB	I	PORT B CHIP SELECT
EFA/ORA	O	PORT A EMPTY / OUTPUT READY FLAG
EFB/ORB	O	PORT B EMPTY / OUTPUT READY FLAG
ENA	I	PORT A ENABLE
ENB	I	PORT B ENABLE
FFA/IRA	O	PORT A FULL/INPUT READY FLAG
FFB/IRB	AO	PORT B FULL/INPUT READY FLAG
FS0/SD	I	FLAG OFFSET SELECT 0 / SERIAL DATA
FS1/SEN	I	FLAG OFFSET SELECT 1 / SERIAL ENABLE
FS2	I	FLAG OFFSET SELECT2

MB	I	PORT A MAILBOX SELECT
MBB	I	PORT B MAILBOX SELECT
MBF1	O	MAIL 1 REGISTER FLAG
MBF2	O	MAIL 2 REGISTER FLAG
MRS1	I	FIFO 1 MASTER RESET
MRS2	I	FIFO 2 MASTER RESET
PRS1/RT1	I	PARALLEL RESET / RETRANSMIT FIFO1
PRS2/RT2	I	PARALLEL RESET / RETRANSMIT FIFO2
RTM	I	RETRANSMIT MODE
SIZE	I	BUS SIZE SELECT
W/RA	I	PORT-A WRITE / READ SELECT
W/RB	I	PORT-B WRITE / READ SELECT

Pin location diagram of IDT FIFO buffer is shown in Fig. 4.18.

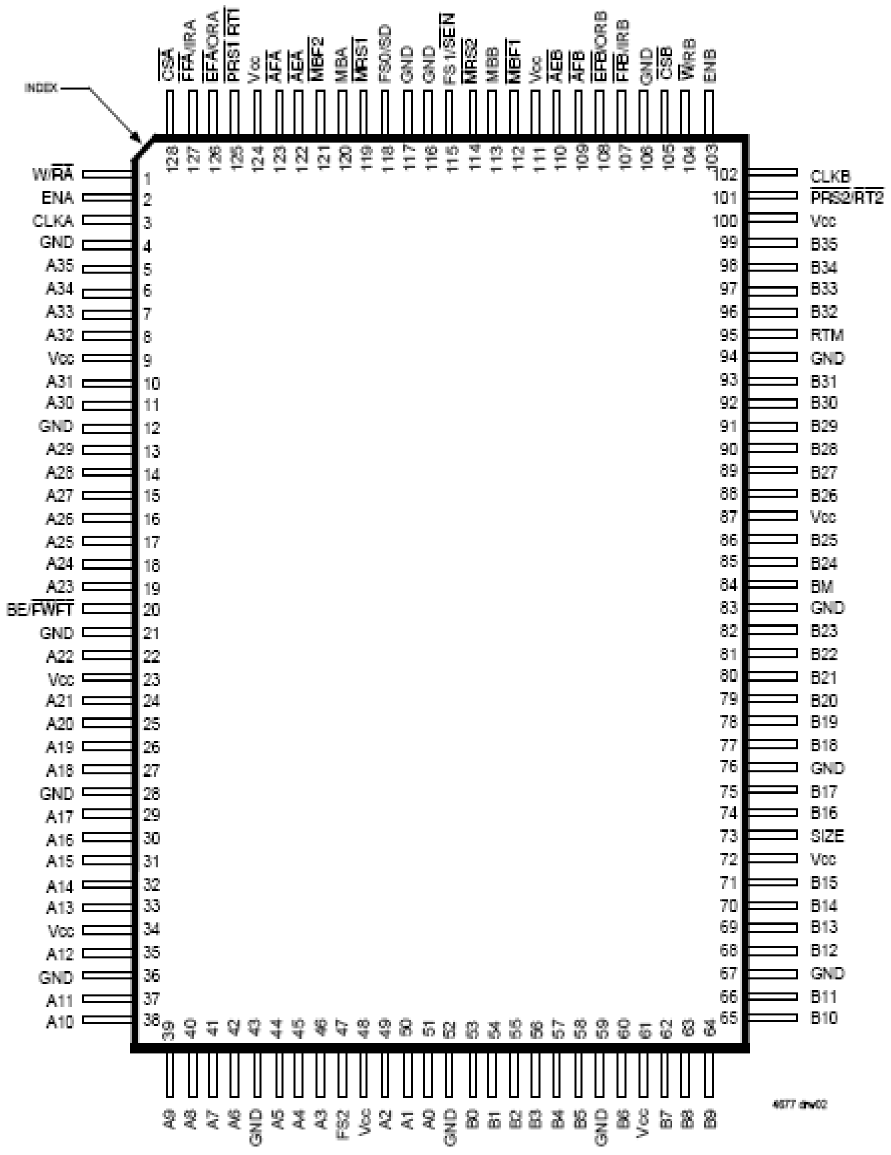


Fig. 4.18 Pins location of IDT FIFO buffer [14],[15]

7. Intel IO processor [11] ⇔ Strata flash memory [23]

The connection of Intel IO processor and strata flash memory are main point in interface 7. The same as before, only most possible pins information will be listed for convenience to verify them. Intel IO processor pins which includes peripheral bus interface and strata flash memory are in following tables and pin location diagram of strata flash are shown in Fig 4.19.

- Possible hardware interface signal of 6:
  - ✓ Peripheral Bus Interface signals of IO processor [11]

Signal name	Count	Description
AD[31:0]	32	ADDRESS / DATA BUS
BE[3:0]	4	BYTE ENABLES
ALE	1	ADDRESS LATCH EBABLE
ADS	1	ADDRESS STROBE
PB_CLK	1	PERIPHERAL BUS CLOCK
W/R	1	WRITE / READ
FEW	1	FLASH WRITE ENABLE
DEN	1	DATA ENABLE
BLAST	1	BURST LAST
RDYRCV	1	READY / RECOVER
HOLD	1	HOLD
HOLDA	1	HOLD ACKNOWLEDGE
PB_RST	1	PERIPHERAL BUS REST
WIDTH[1:0]	2	WIDTH

Configuration pins

PCE[5] / PBI 100MHz	1	PERIPHERAL CHIP ENABLES / PERIPHERAL BUS 100 MHz ENABLE
PCE[4] / PBI 66MHz	1	PERIPHERAL CHIP

		ENABLES / PERIPHERAL BUS 66 MHz ENABLE
PCE[3] / P_BOOT16 MHz	1	PERIPHERAL CHIP ENABLES / PERIPHERAL BUS BOOT WIDTH 16 ENABLE
PCE[2] / 32BITPCI	1	PERIPHERAL CHIP ENABLES / 32 BIT PCI
PCE[1] / RETRY	1	PERIPHERAL CHIP ENABLES / RETRY
PCE[0] / RST_MODE	1	PERIPHERAL CHIP ENABLES / RESET MODE

- ✓ Strata flash Interface signals [23]

Signal name	Count	Description
A[MAX:1]	unknown	ADDRESS INPUTS
DQ[15:0]	16	DATA INPUT/OUTPUT
ADV	1	ADDRESS VALID
CE	1	FLASH CHIP ENABLE
CLK	1	CLOCK
OE	1	OUTPUT ENABLE
RST	1	Reset
WAIT	1	WAIT
WE	1	WRITE ENABLE
RFU	1	RESERVED FOR FUTURE USE
DU	1	DO NOT USE
NC	1	NO CONNECT

The pin location of Strata flash memory is shown in Fig. 4.19.

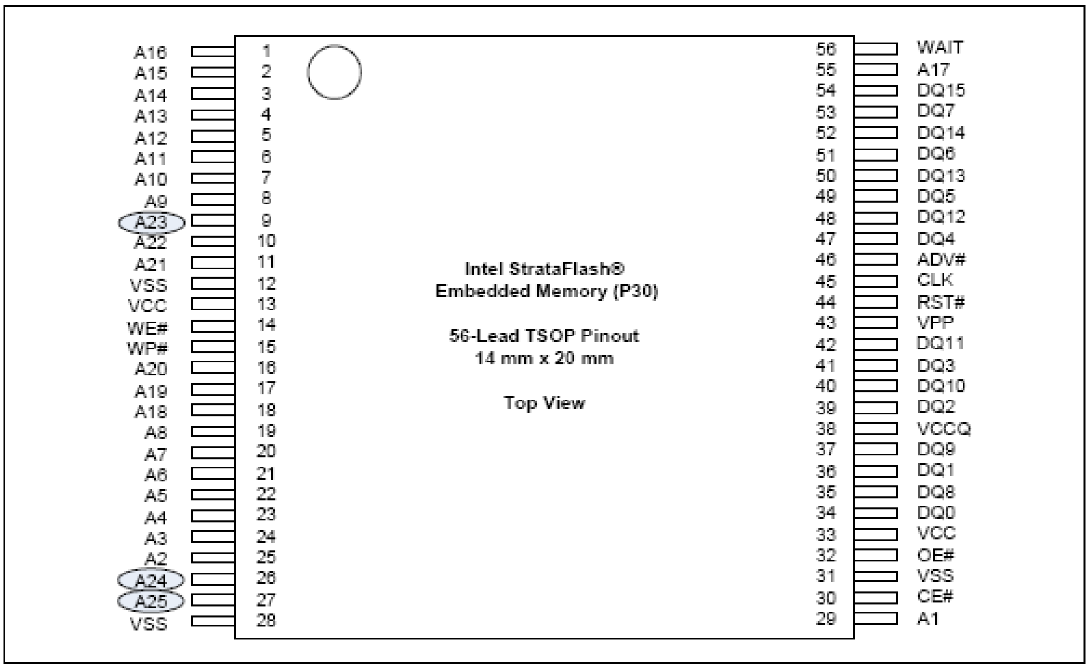


Fig. 4.19 Pins location of Strata Flash memory [23]

8. Xilinx FPGA processor [16] ⇔ IDT FIFO buffer [14],[15]

In interface 8, Xilinx FPGA processor and IDT FIFO buffer are our concerned things. The purpose of connections between Xilinx FPGA processor and IDT FIFO buffer is that Xilinx FPGA processor needs some space for computed signal which may communicate among hardware board interfaces to complete an algorithm computation. FIFO buffer is a good choice for registering these signals. Xilinx FPGA processor has many configurable pins and we can not know exactly what function of each pin works. A method is proposed in section 4.6.1 for guessing every pin function as exactly as possible. Total pins information IDT FIFO buffer has shown before, so we don’t list again.

9. Xilinx FPGA processor [16] ⇔ Xilinx Programmable ROM [12]

Two components of interface 9 are about Xilinx FPGA processor and Xilinx programmable ROM. The reason we require programmable ROM are that FPGA processor need instructions to perform computation and these instructions are needed a space to be saved. Programmable ROM is often a best choice because the instruction does not disappear when system is shutdown. In addition, we can read out and write in the instructions form programmable ROM if requires. So, it is reusable and many instructions will be put in programmable ROM.

Due to configurable Xilinx FPGA processor pins, we only know how many pins on FPGA processor and don’t know how to work on each pins without software control information. In section 4.7.1, we have a method to test all pins and guess them function if measurements are possible in feature. Hence, we only listed the possible connection pins in the following table.

- Possible hardware interface signal of 9:

✓ Programmable ROM Interface signals [12]

Signal name	Location	Description
D0	28	DATA
D1	29	DATA
D2	32	DATA
D3	33	DATA
D4	43	DATA
D5	44	DATA
D6	47	DATA
D7	48	DATA
CLK	12	CONFIGURATION CLOCK
OE/RESET	11	OUTPUT ENALE/RESET
CE	13	CHIP ENABLE INPUT
CF	6	CONFIGURATION PULSE
CEO	10	CHIP ENABLE OUTPUT
EN_EXT_SEL	25	ENABLE EXTERNAL SELECTION INPUT
REV_SEL[0:1]	26,27	REVISION SELECT[1:0] INPUTS
BUSY	5	BUSY INPUT
CLKOUT	9	CONFIGURATION CLOCK OUPUT
TMS	21	JTAG MODE SELECT INPUT
TCK	20	JTAG CLOCK INPUT
TDI	19	JTAG SERIAL DATA INPUT
TDO	22	JTAG SERIAL DATA

		OUTPUT

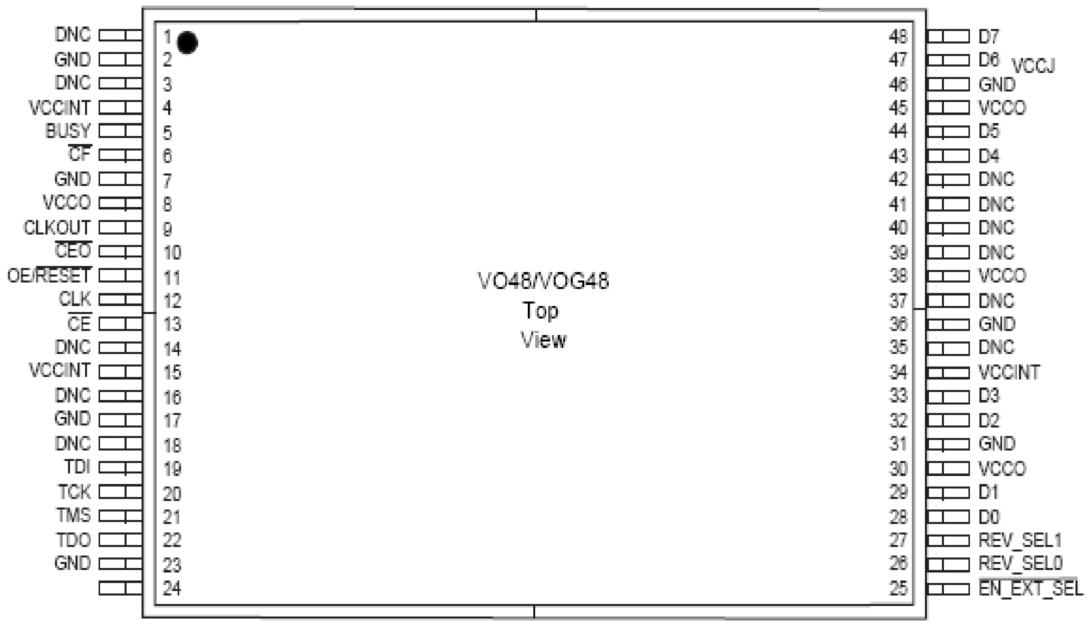


Fig 4.20 Pins location of Xilinx Programmable ROM [12]

10. Xilinx CPLD [13] ⇔ IDT FIFO buffer [14],[15]

We talk about CPLD and IDT FIFO buffer in interface 10. We can see CPLD as another kind of FPGA but CPLD is more suitable for combinational logic design. Hence, as we know, CPLD also has configurable pins which defined by the manufacture. It is difficult to guess any information on this interface but we still try to do something. Hence, we give the same method for measurement of CPLD as FPGA and hope to find any information as clearly as possible. IDT FIFO buffer pins are described before, and we don't list them again.

11. Xilinx CPLD [13] ⇔ Xilinx Programmable ROM [12]

It is the last possible inward-board interface of IO 21000 board. The relation between Xilinx CPLD and Xilinx programmable ROM are the same as Xilinx FPGA and Xilinx programmable ROM that FPGA and CPLD always requires to load instructions from programmable ROM for any algorithm when they need to perform. In interface 11, Xilinx programmable ROM is mentioned above where we won't list again and Xilinx CPLD pins location and function still need the verification way which we will describe in section 4.7.1.

4.7.1 Verification method of inward-board interface

In this section, we propose our verification method for inward-board interface of IO 21000 board. The logic analyzer, multi-meter and PCI-X analyzer are indispensible devices to this verification method. Verification method of interface 3 to 11 can also be classified into two categories. One is verification of FPGA and CPLD which have unknown definite pins function and location among interface 8 to 11 and the other are those which have the definite pin information among interface 3 to 7. The procedures are listed as follows:

- For interface 8 to 11 which involve FPGA and CPLD:
  - ✓ 1. Indentify each pin function and location of FPGA and CPLD
    - ➢ We first connect all pins in FPGA or CPLD with probe of logic analyzer. Assume the number of probes and logic analyzers are large enough.
    - ➢ Then, we let the MEOS system run and observe the signal waveform on the screen of logic analyzers.
    - ➢ We can certify that the pins whose waveforms are almost positive and negative are VDD and GND.
    - ➢ By change the system operation mode (such as change of frame length), we can observe which waveforms of pins changes on the screen of logic analyzer and these changed pins are configuration pins.
    - ➢ By receiving different input data, we can observe which pins changes and we consider these pins as input/output pins.
  - ✓ 2. We use the short-circuit characteristics of multi-meter to identify all connection way among interface 8 to 11.
  - ✓ 3. For instruction set which saved in programmable ROM:
    - ➢ We record all pins waveform of programmable ROM.

    - ➢ Close some function of IO 21000 board and record the pins waveform of programmable ROM when system is running.
    - ➢ Compared these waveforms to original waveforms and extract the different part to analyze meaningful machine languages.
    - ➢ Do the above three steps iteratively with another test way and hope to extract all instructions of Programmable ROM instruction set.
- ✓ 4. The same step 3 is for FIFO buffer to FPGA and CPLD.

- ● For interface 3 to 7 which have definite pin information:
  - ✓ 1. We use the short-circuit characteristics of multi-meter to identify all connection way among interface 3 to 7.
  - ✓ 2. For instruction set which saved in strata flash memory:
    - ➢ We record all pins waveform of strata flash memory.
    - ➢ Close some function of IO 21000 board and record the pins waveform of strata flash memory when system is running.
    - ➢ Compared these waveforms to original waveforms and extract the different part to analyze meaningful machine languages.
    - ➢ Do the above three steps iteratively with another test way and hope to extract all instructions of strata flash memory instruction set.

- ✓ 3. The same step is for other interfaces.

We describe our proposed verification method above. Although we can not promise that all inward-board interfaces can be analyzed by this method, this method still give the next project researcher a good guide to reference.

4.8 Appendix: Measurements work in NSPO

We go to NSPO to check and carry out our proposal of interface signal measurement of the IO 21000 board on Sep 9-11, 14-16 in 2009. Our major work on this week focus on outward-board interface measurement and verification. Under no consideration to decompose the overall system, all we can do is to perform measurement on differential ECL interface. Hence, we first go to buy the T-type SMA connectors which are silver in Fig. 4.21 and make these connectors replace the original connectors as shown in Fig. 4.21.

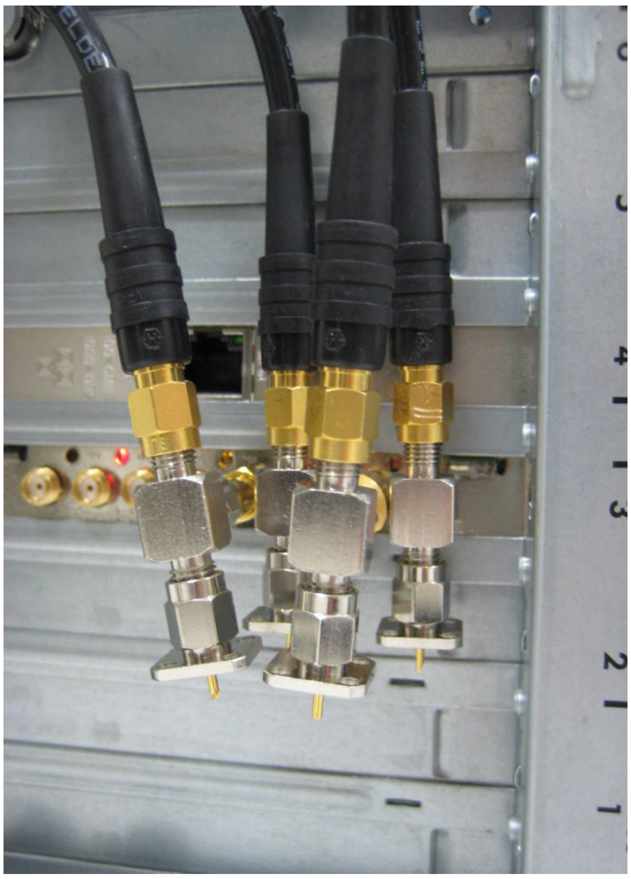


Fig. 4.21 T-type SMA connectors (silver)

Four black lines are the input signal of IO 21000 board. Two are differential data and clock signals which one line represents positive signal and the other stand for negative signal. After we finish setting up the environment, we spent one day on getting familiar with logic analyzer whose brand is Tektronix TLA 614 Logic analyzer whose picture is in Fig 4.22.

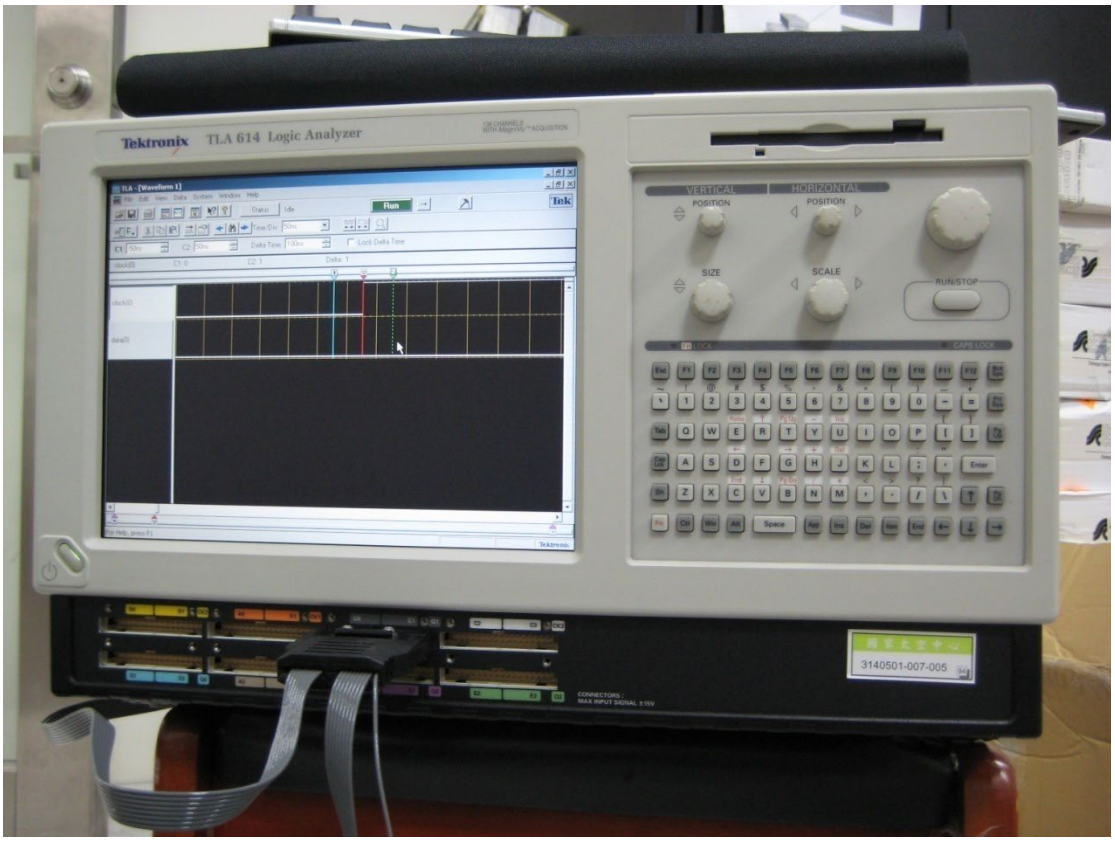


Fig. 4.22 Tektronix TLA 614 Logic analyzer

Then, we put the probes of logic analyzer on T-type SMA connector pins as shown in Fig. 4.23. Different from Fig. 4.23 is that four lines are all with probes in practical measurements. After finishing the setting, we can go to further step. Before going to further measurement, we roughly describe the background of our measurement.

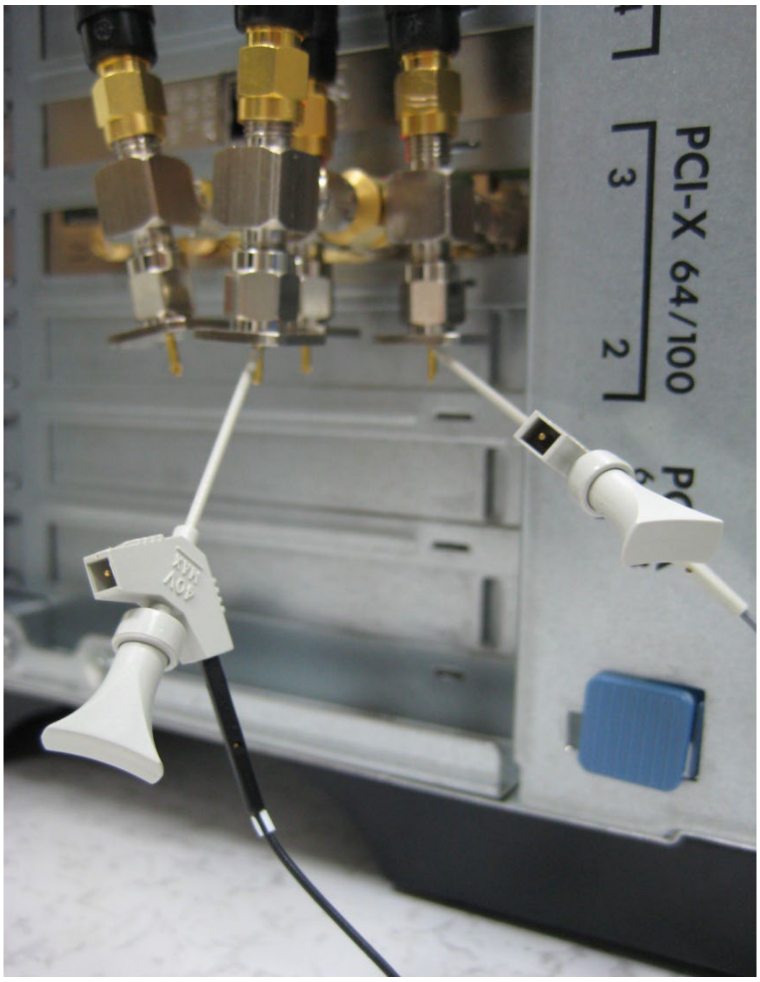


Fig. 4.23 Probes on differential ECL

For first two day, we only have ten minutes during AM 9:40 ~ AM 9:50 to perform measurements everyday. Due to the characteristics of LEO satellite, Formosa II satellite has higher elevation angles to transmit information at this time interval in the sky. Ten minutes measurements are too less to perform efficient measurement for us. Hence, we try to find some other method to overcome this obstacle. Fortunately, two days later, we find the other two satellites which can provide test data for our measurements. These two satellites are Aqua and Terra that they are meteorological satellite and also LEO satellites and each give us ten minutes for measurement. That is total 30 minutes per day we can perform measurements. But, it is unfortunate that we have not measured any information from differential ECL input on screen shot of the logic analyzer during that week. We consider what happened to this problem once more and infer that there must be some protection mechanism on DC-DC converter of IO 21000 board. Our suggestion to get the measured hardware signal is to measure the pins at dual high speed PECL comparator. Due to not enough time for measurement, we are not able to take our suggestion method to measurement in time. However, we think that the suggest method can be a future work for next project research worker. The more detailed for suggestion measurement is introduced in section 4.6.1.

4.9 summary

In this chapter, we describe the interface analysis and verification method of IO 21000 board which consist of inward-board and outward-board interface and more discussion about verification and interface analysis are also include. We first introduce the IO 21000 board and what function of IO 21000 board works. Then, the hardware components of IO 21000 board are described. After we are familiar with the hardware components, we perform some analysis on inward-board and outward-board interface of IO 21000 board. Verification methods are also proposed for certification of our proposed interface structure. Although we can not do the whole reverse engineering of IO 21000 board, we also proposed another method to replace the whole board if we can know the outward-board interface of IO 21000 board. It is good reference guide for the next project research worker.

5 Reed-Solomon code implementation [26]-[41]

In this chapter, we start go to discuss the R-S code hardware implementation. First, we introduce the basic introduction and skill of how R-S code is implemented in FPGA hardware board. Then, the implementation method and evaluation is also described. Finally, the simulation and hardware implementation result of Reed-Solomon decoder is shown and discussed.

5.1 Introduction of Reed-Solomon code implementation

We have simulated Matlab or C programming of CCSDS standard R-S code (255,223) with error correcting capability t=16 and convolutional code (2,1,7) with constraint length 7. The hardware description has programmed in Verilog. The hardware implementation of R-S decoder is illustrated in Fig 5.1 as follows.

Each SRAM in this design is required 64×32 bits to store receiving decoding bit from previous stage, but TriMatrix memory feature for Stratix II set the memory size and we choose 128×32. For buffering data, RSIN pin are 16bits per clock cycle. When writing each address, we write the MSB 16 bits for first clock and then LSB

16bit for next clock. Timing chart and finite state machine are illustrated at Fig. 5.2 and Fig. 5.3.

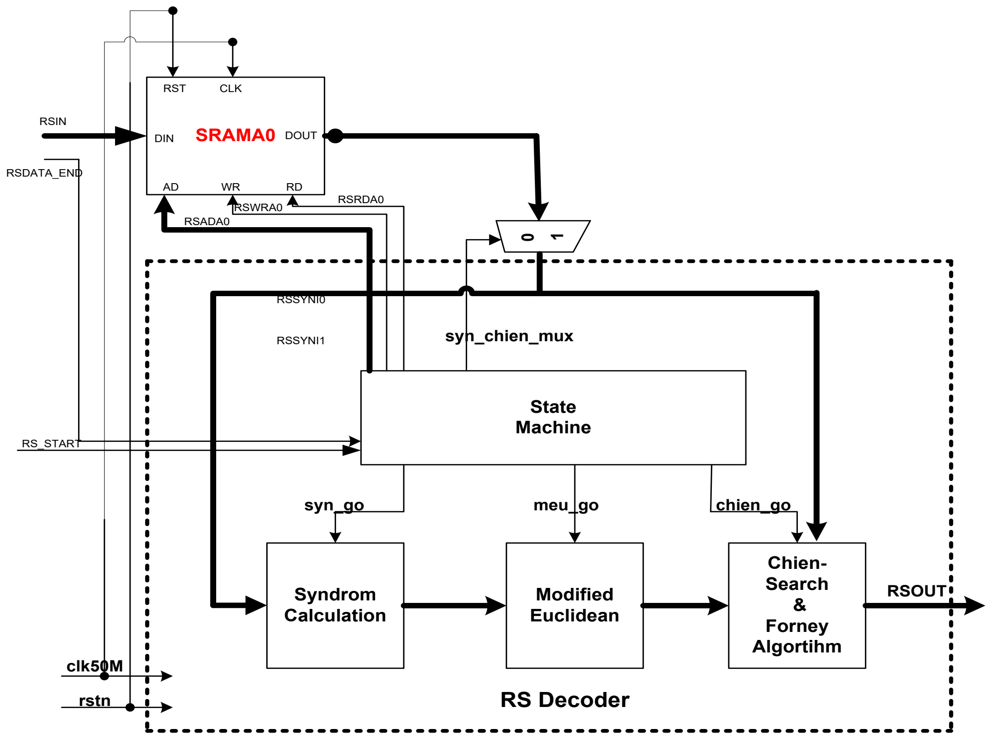


Fig. 5.1 Block diagram of R-S decoder scheme

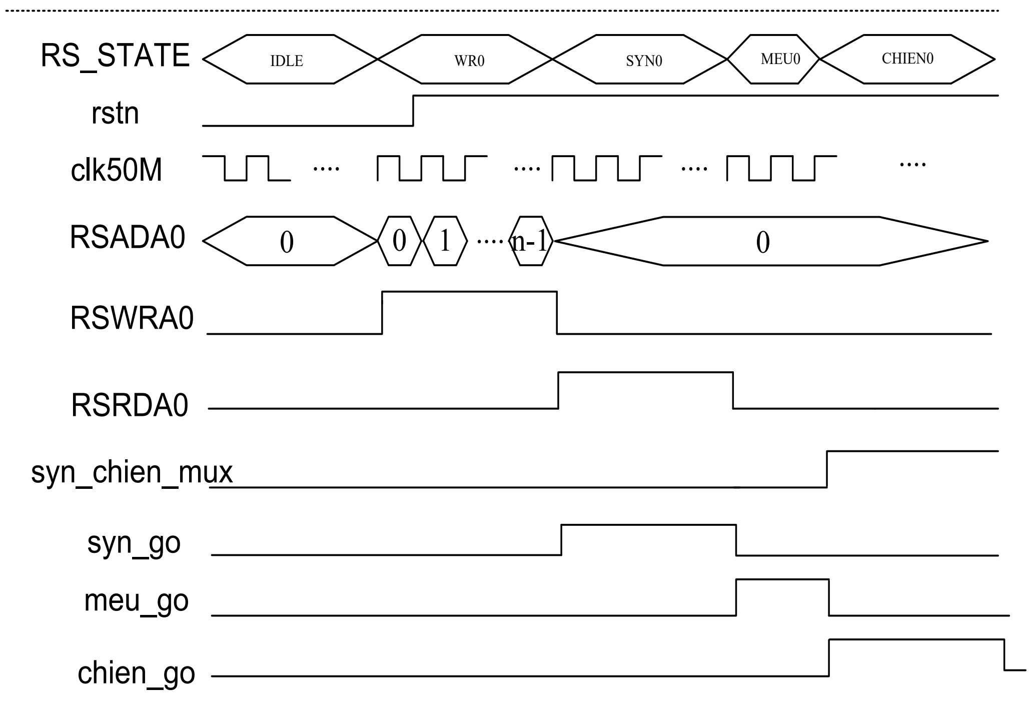


Fig. 5.2 Timing chart of R-S decoder scheme

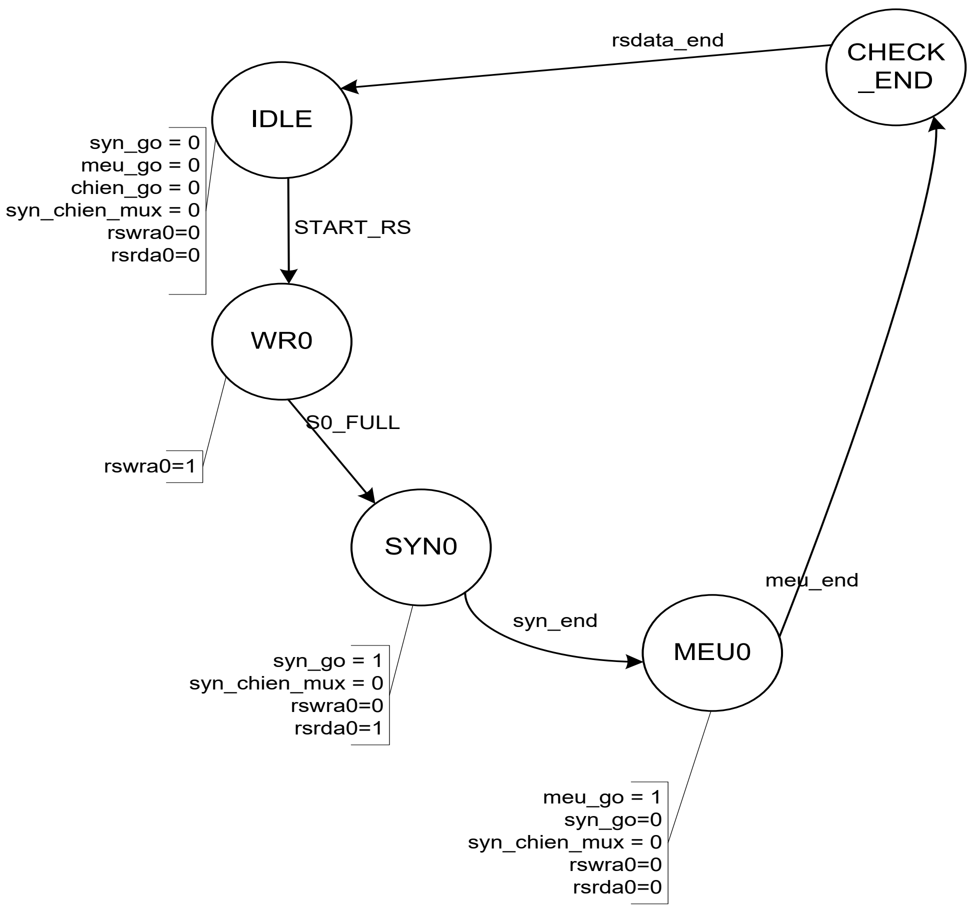


Fig. 5.3 Finite state machine of R-S decoder schem

The parallel structure for syndrome block is illustrated at Fig. 5.4. We set 16 bits for each clock input and process $\left\lfloor \frac{255}{16} \right\rfloor$ clock cycles to obtain syndrome of a block R-S codeword. The parallel syndrome generator and Chien search unit is illustrated and depict to process 16 bits per clock cycle, only $\left\lfloor \frac{255}{16} \right\rfloor$ clock cycle are sufficient.

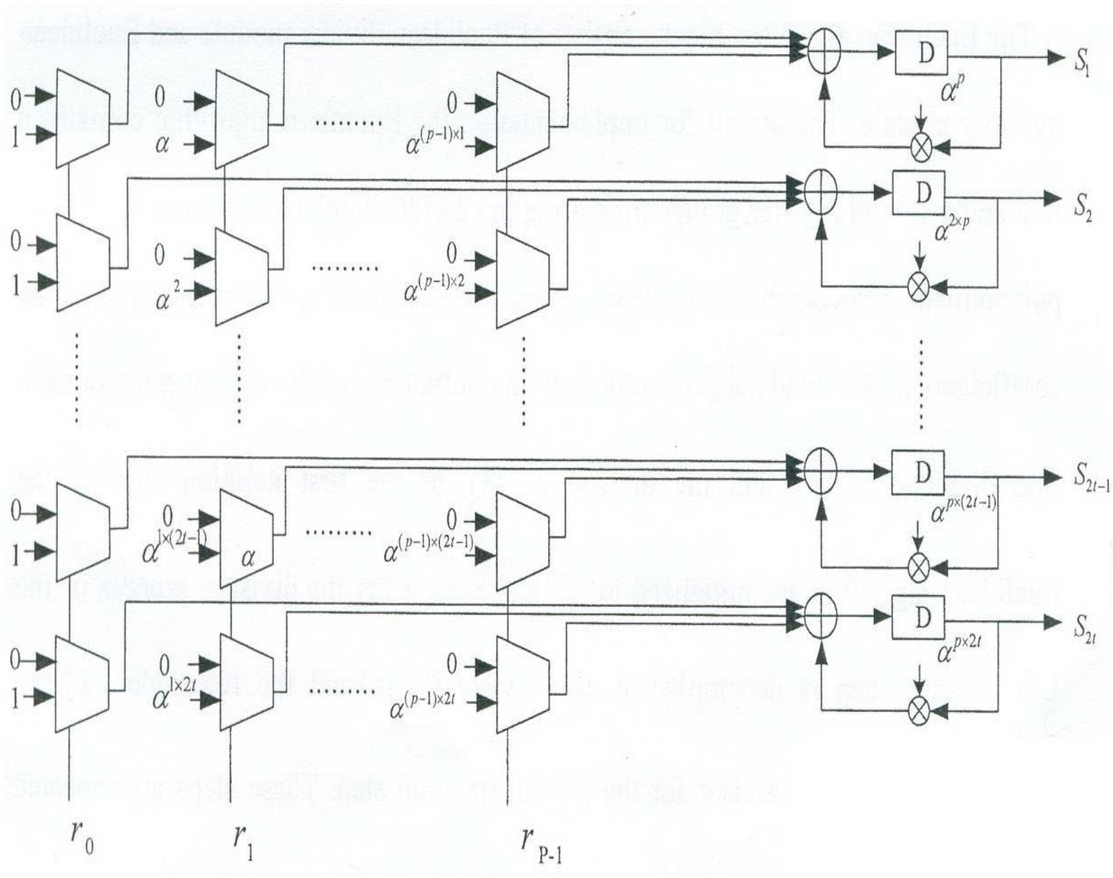


Fig. 5.4 Parallel syndrome generator with parallel factor p=16

Most complicated block of R-S decoder is modified Euclidean algorithm block. We require 2t iterations to compute the key equation, so the data that processing from cell1 to cell4 require to feedback from the end of cell4 and feed into mux selector until 8th feedback is meet. We would follow the above hardware architecture to implement R-S decoder. We compute the ideal overall throughput for this design as follows: Input buffering data require 16 cycles, syndrome unit require 16 cycle, key equation solving unit require 2t*3=96 cycle and the last stage is Chien search and Forney algorithm require 32 cycle. The overall clock cycles are 160 cycles if we operate at 50 MHz.

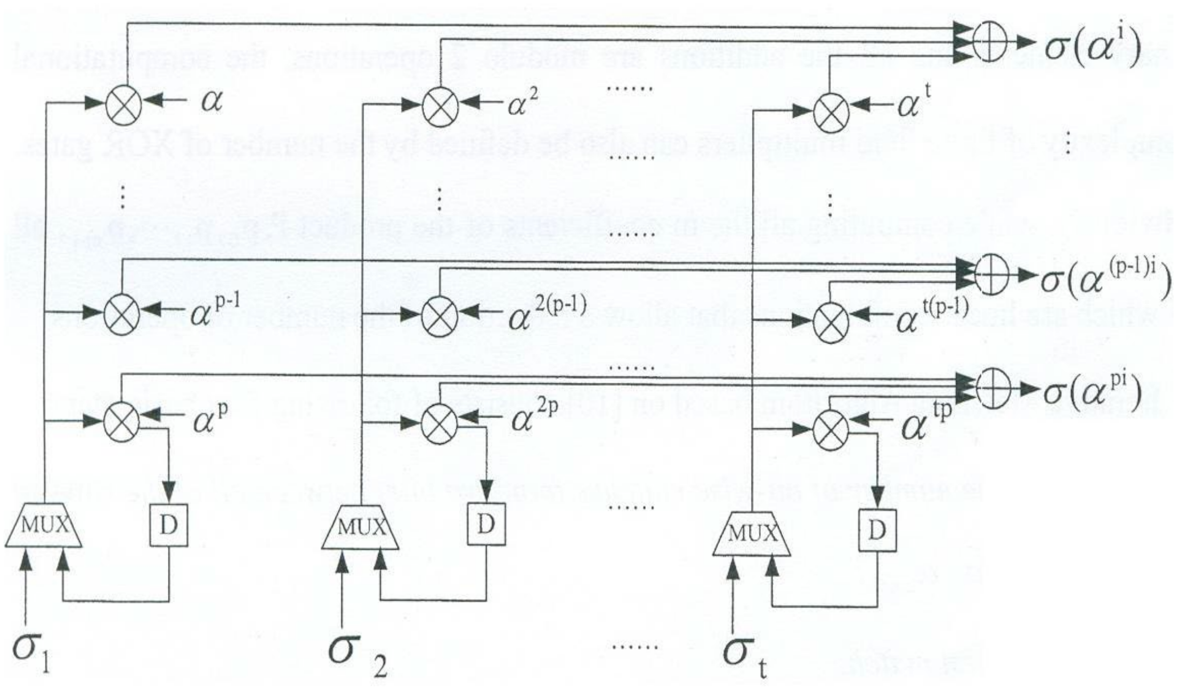


Fig. 5.5 Parallel Chien search with parallel factor p=16

The throughput is ideally 223*8bit/160cycle*50*10^8=557.5Mbps. Although we probably suffer from any kinds of retiming solution to meet the clock frequency, the latency would be more than this ideal calculation. This design may seems to meet our target 150Mbps

The I/O pins are summarized in Table 5.1 as follows:

Signal	**I/O**	**Description**
CLK50Mhz	I	Input clock 50 MHz from Clock Gen
rstn	I	Reset (*active low*) for clock generation
RS_START	I	R-S decoder starting
RSDATA_END	I	No data to R-S decoder, So, R-S decoder ending
RSSYNI0[N-1:0]	I	Input date from SRAMA0
RSADA0[M-1:0]	O	Address for SRAMA0
RSWRA0	O	Write enable for SRAMA0
RDA0	O	Read enable for SRAMA0
ERRADD1[7:0]	O	Error address for 1st Error
ERRADD2[7:0]	O	Error address for 2st Error
ERRADD3[7:0]	O	Error address for 3st Error
ERRADD4[7:0]	O	Error address for 4st Error
ERRADD5[7:0]	O	Error address for 5st Error
ERRADD6[7:0]	O	Error address for 6st Error
ERRADD7[7:0]	O	Error address for 7st Error

ERRADD8[7:0]	O	Error address for 8st Error
ERRADD9[7:0]	O	Error address for 9st Error
ERRADD10[7:0]	O	Error address for 10st Error
ERRADD11[7:0]	O	Error address for 11st Error
ERRADD12[7:0]	O	Error address for 12st Error
ERRADD13[7:0]	O	Error address for 13st Error
ERRADD14[7:0]	O	Error address for 14st Error
ERRADD15[7:0]	O	Error address for 15st Error
ERRADD16[7:0]	O	Error address for 16st Error
SERR[3:0]	O	Corrected error amount

Table 5.1 I/O pins of R-S decoder summary

5.2 Hardware design of Reed-Solomon code

➢ **Syndrome generator:**

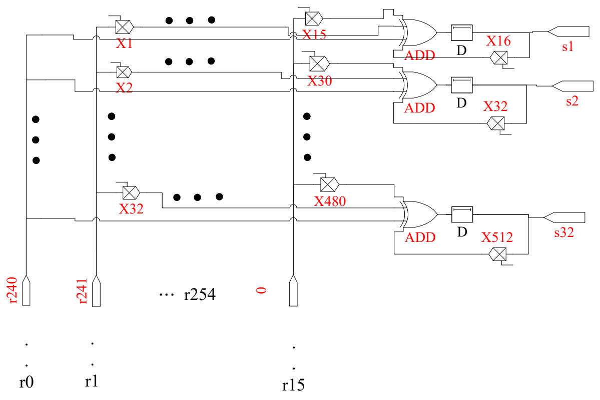


Fig 5.6

Fig 5.6 shows the design of syndrome generator. The polynomial

$S(X)=S_1+S_2X+\ldots+S_{2t}X^{2t-1}$ can be rearranged as

$(\ldots((r_{240}+r_{241}\alpha+\ldots r_{254}\alpha^{14})*\alpha^{16}+r_{224}+r_{225}\alpha+\ldots r_{239}\alpha^{15})\ldots.+r_{15}\alpha^{15}$. Hence, a

16-parallel syndrome generator can be used.

➢ **ME algorithm**

Fig 5.7 shows the design of ME algorithm. We modified the ME algorithm to facilitate the hardware implementation. The initial conditions of the algorithm are

$R_0(x)=x^{2t},\ Q_0(x)=S(x),\ L_0(x)=0,\ U_0(x)=1.$

In i-th iteration

$$R_i(x) = b_{i-1}R_{i-1}(x) - x[a_{i-1}Q_{i-1}(x)]$$

$$Q_i(x) = Q_{i-1}(x)$$

$$L_i(x) = [b_{i-1}L_{i-1}(x) - x[a_{i-1}U_{i-1}(x)]$$

$$U_i(x) = U_{i-1}(x)$$

where $\deg(R_{i-1}(x)) > \deg(Q_{i-1}(x))$. If $\deg(R_{i-1}(x)) < \deg(Q_{i-1}(x))$, the exchanger

exchanges

$R_{i-1}(x) \leftrightarrow Q_{i-1}(x)$, $L_{i-1}(x) \leftrightarrow U_{i-1}(x)$, and $\deg(R_{i-1}(x)) \leftrightarrow \deg(Q_{i-1}(x))$, the signal ”sw” is

used

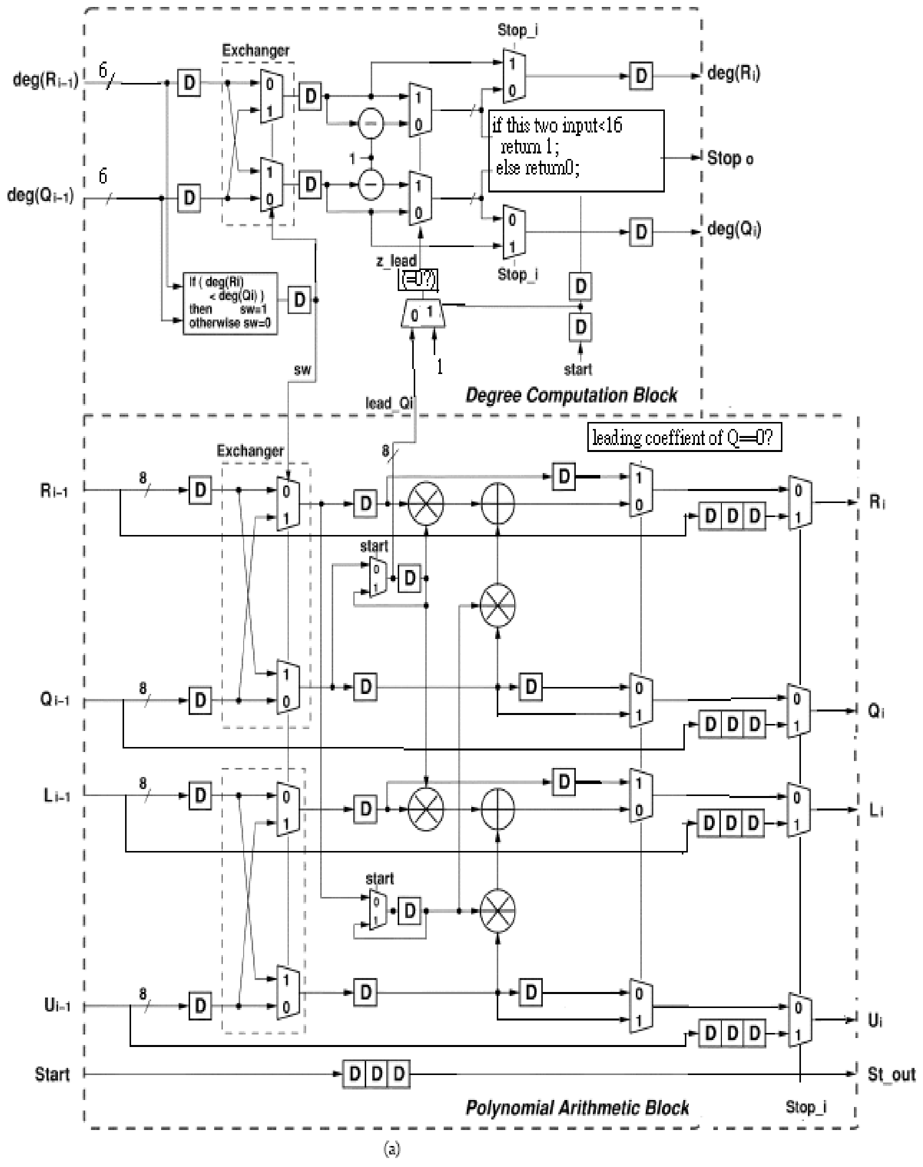


Fig 5.7

to indicate whether it exchanges or not. The signal ”stop” is used to indicate whether the computation is to be stopped or not. Signal “lead_Qi” has to be generated in order to check whether the leading coefficient of $Q_i(x)$ is equal to zero. The signal ”start” is equal to zero when leading coefficient Qi(x) comes. Fig 5.8 show the input of ME cell

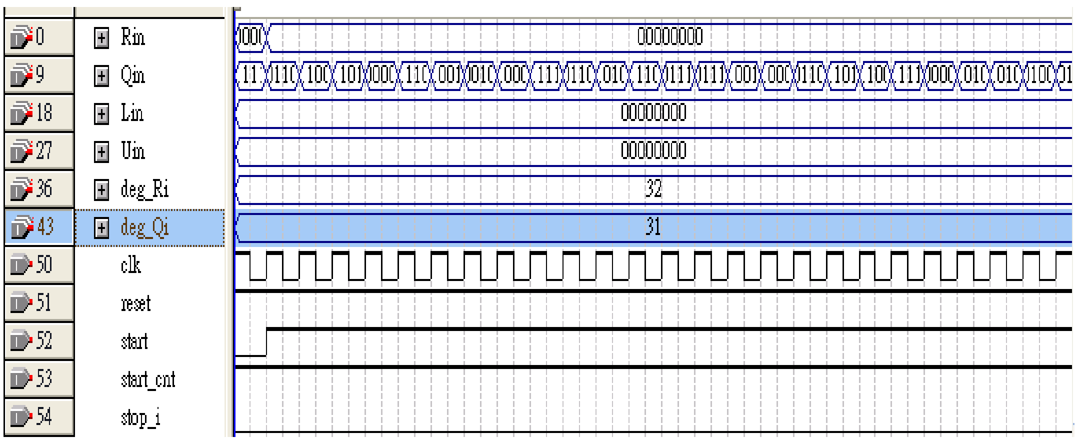


Fig 5.8

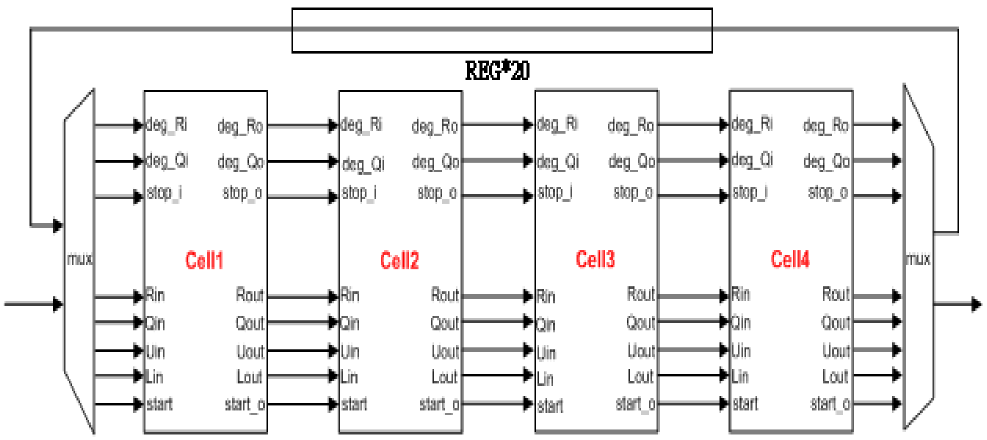


Fig 5.9 Modified Euclidean algorithm block

Using the above ME algorithm, we can decrease Q(x) or R(x) by one degree. In worst case, we need 32 cells to decrease 1 by 6 degree of R(x) (8 cycle when using 4 ME cell).

The design of 4-cell ME algorithm is shown in Fig 5.9.

Fig. 5.10 shows the output of ME cell. As we see Fig 5.10, block 1 shifts Q(x) and U(x) component to become block 2. This situation occurs when deg(R(x))-deg(Q(x))>2. Block 2 is identical to block 3, because stop condition is matched.

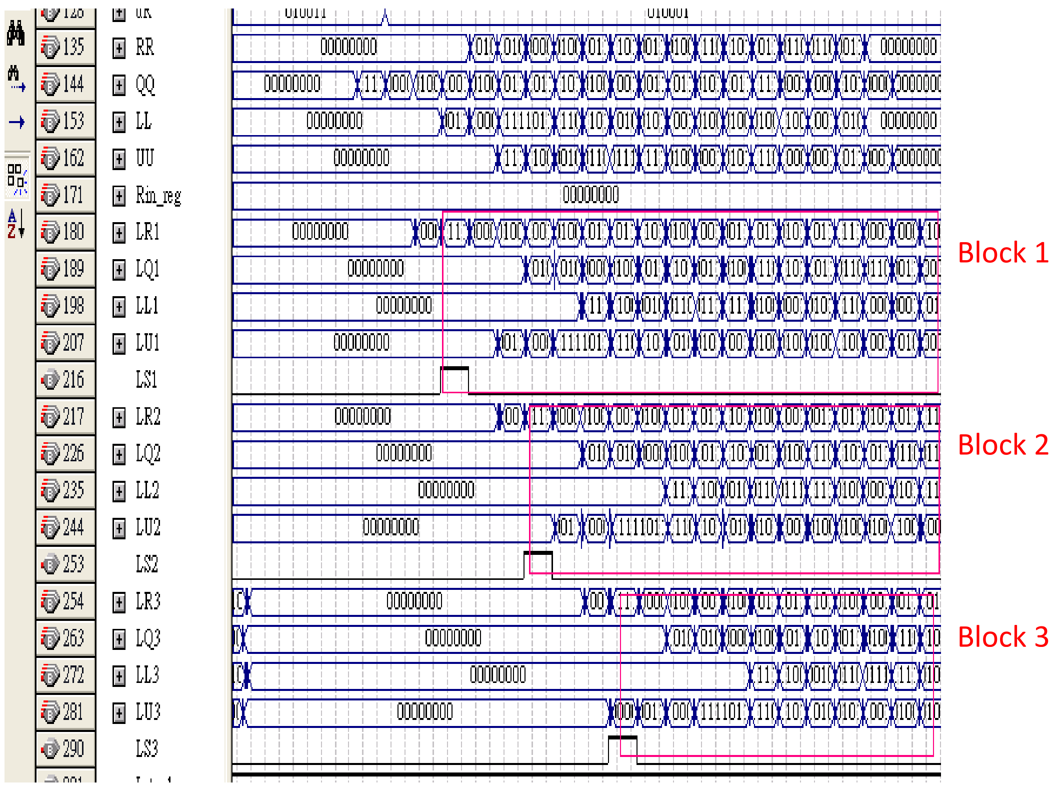


Fig. 5.10

➢ **Chien Search:**

Fig. 5.11 shows the design of Chien search. We use 16 parallelism to compute $\sigma(\alpha^x)$ for $1 \leqq x \leqq n$ and see whether it is zero or not. The same circuit can be used to calculate $\omega(\alpha^x)$ and $\sigma'(\alpha^x)$. The multipliers muliply fixed number α^i here,so we can reduce unit in Fig. 5.11. Therefore, the logic gate in this circuit will be reduce.

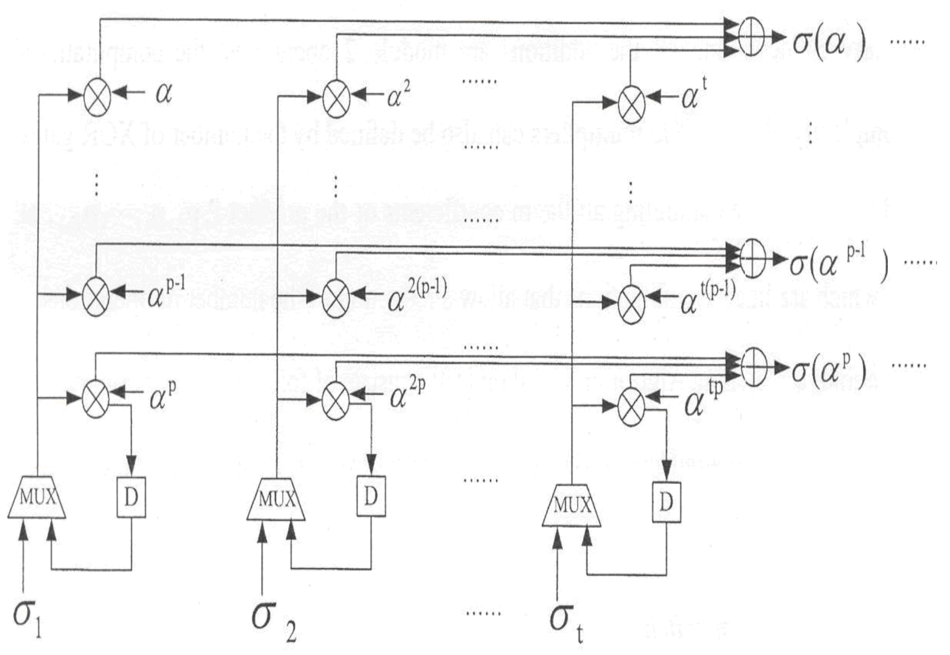


Fig. 5.11

Let $f(X)=A_xX^n+A_{n-1}X^{n-1}+\ldots A_0$, then $f'(X)=nA_xX^{n-1}+(n-1)A_{n-1}X^{n-2}+\ldots A_1$. Since it is calculate in binary, the even component of $f'(X)$ is equal to 0. Therefore, $\sigma'(\alpha^x)$ can be derived by decreasing one degree of input, and input the odd component only.

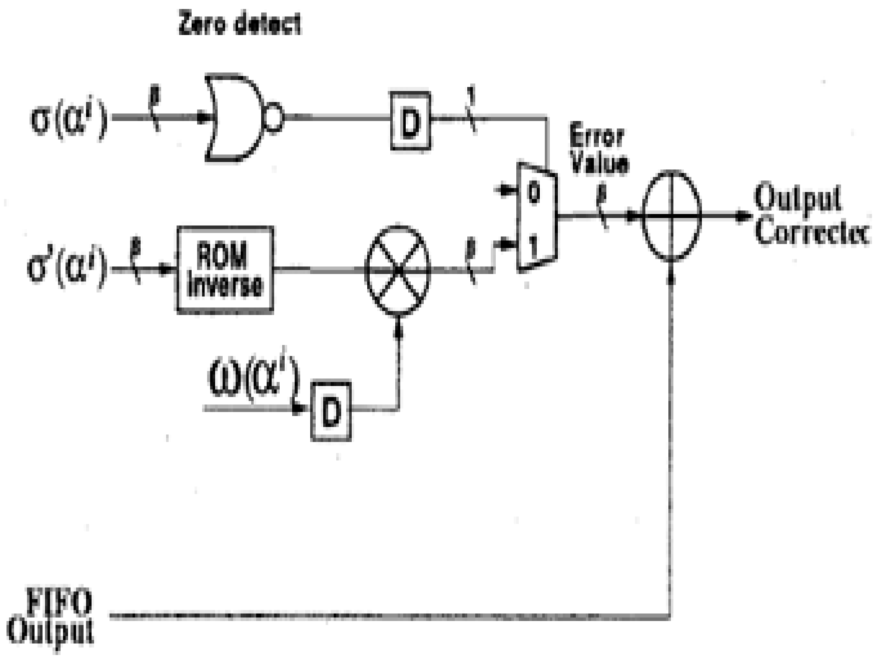


Fig. 5.12

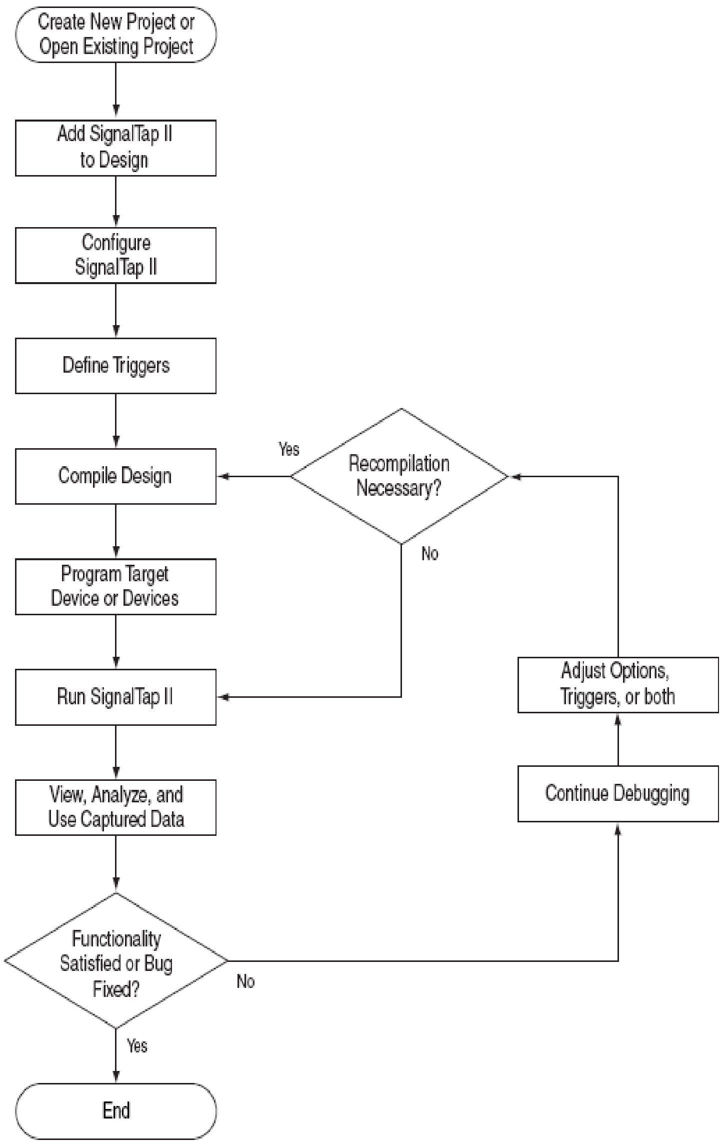
Fig. 5.12 is the circuit of correcting part, we use a ROM inverse block to generate $\frac{1}{\sigma'(\alpha)}$, and resulting σ($\alpha^x$) decide whether the error magnitude $\frac{\omega(\alpha^x)}{\sigma'(\alpha^x)}$ is added to the received polynomial. At last, we find the correct codeword.

5.3 Design debugging tasks using SignalTap II embedded logic analyzer

For the verification of the hardware design, a testing approach using SignalTap II has been applied to justify our RS decoder. First, the operation flow of SignalTap II was revealed. Second, a block diagram for verification was illustrated to clarify our work.

To help with the process of design debugging, Altera provides a solution for us to examine the behavior of internal signals, without using extra I/O pins, while the design is running at full speed on an FPGA device. To use the SignalTap II Embedded Logic Analyzer to debug this design, we performed a number of tasks to add, configure, and run the logic analyzer. Fig. 5.13 shows a typical flow of the tasks to debug this design.

For testing our design, a simple testing circuit consists of three components, which are RS data generator, Error injector and XOR comparator. These components are connected as Fig. 5.14. RS data passes into error injector and the randomly generated error apply on the input data sequence. For random error, 11 kinds of error patterns were tested using symbol error number from 1 to 11. Each kind of error pattern was random generated 4 samples to justify our design. For revealing the correctness of decoded data, XOR comparator matched the original data and the decoded data. Through SignalTapII captured the result of XOR comparator, Jtag passed the result data to computer. This is the testing block diagram for our work.



Create New Project or Open Existing Project
Add SignalTap II to Design
Configure SignalTap II
Define Triggers
Compile Design
Yes
Recompilation Necessary?
No
Program Target Device or Devices
Run SignalTap II
Adjust Options, Triggers, or both
View, Analyze, and Use Captured Data
Continue Debugging
Functionality Satisfied or Bug Fixed?
No
Yes
End

Fig. 5.13 SignalTap II debugging verification

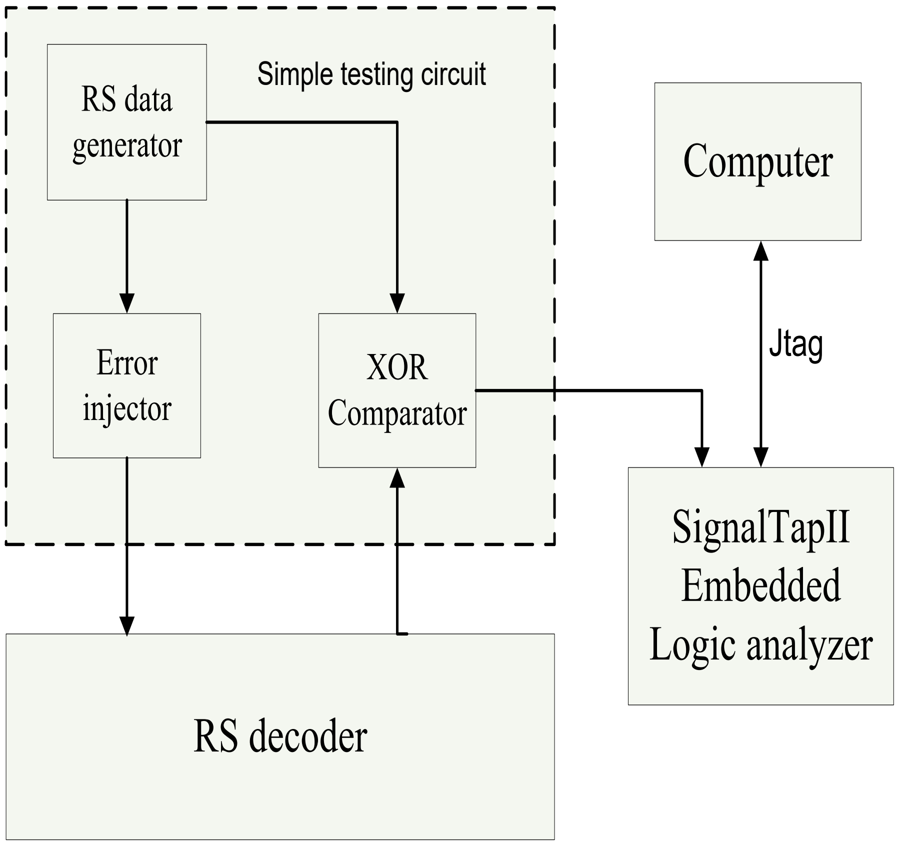


Fig. 5.14 Verification block diagram

5.4 Simulation result

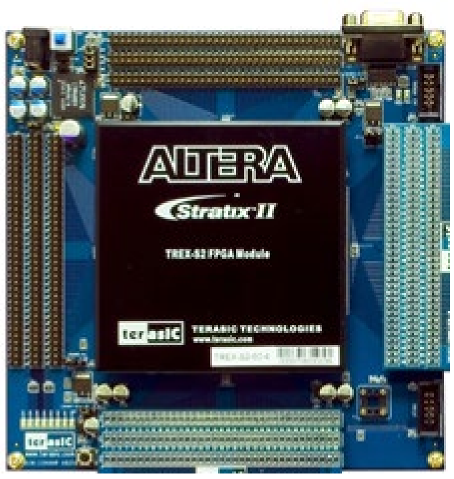
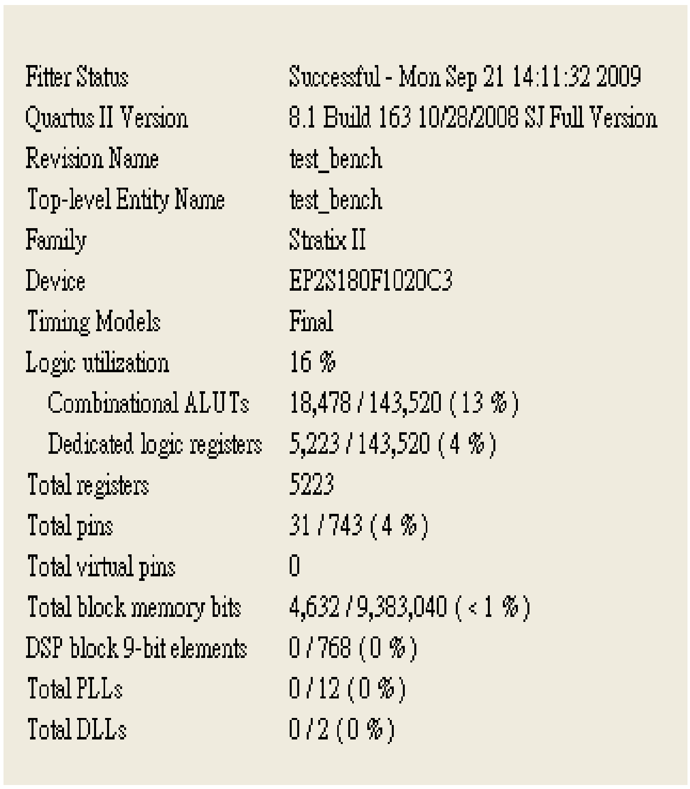
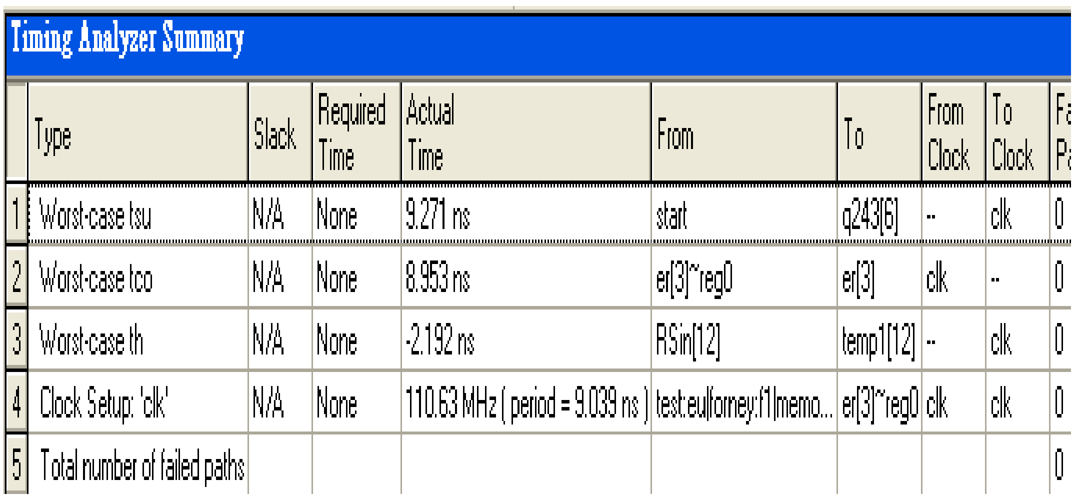


Fig. 5.15



Fitter Status	Successful - Mon Sep 21 14:11:32 2009
Quartus II Version	8.1 Build 163 10/28/2008 SJ Full Version
Revision Name	test_bench
Top-level Entity Name	test_bench
Family	Stratix II
Device	EP2S180F1020C3
Timing Models	Final
Logic utilization	16 %
Combinational ALUTs	18,478 / 143,520 (13 %)
Dedicated logic registers	5,223 / 143,520 (4 %)
Total registers	5223
Total pins	31 / 743 (4 %)
Total virtual pins	0
Total block memory bits	4,632 / 9,383,040 (< 1 %)
DSP block 9-bit elements	0 / 768 (0 %)
Total PLLs	0 / 12 (0 %)
Total DLLs	0 / 2 (0 %)

Fig. 5.16



Timing Analyzer Summary

	Type	Slack	Required Time	Actual Time	From	To	From Clock	To Clock	Fa Pa
1	Worst-case tsu	N/A	None	9.271 ns	start	q243[6]	--	clk	0
2	Worst-case tco	N/A	None	8.953 ns	er[3]~reg0	er[3]	clk	--	0
3	Worst-case th	N/A	None	-2.192 ns	RSin[12]	temp1[12]	--	clk	0
4	Clock Setup: 'clk'	N/A	None	110.63 MHz (period = 9.039 ns)	test:eu|forney:f1|memo...	er[3]~reg0	clk	clk	0
5	Total number of failed paths								0

Fig. 5.17

Fig. 5.15 is the FPGA board we used to test our program. Fig. 5.16 is the simulation report with Quartus II. Fig. 5.17 is the timing report, which show that the fastest possible clock frequency is 110.63MHz in our program.

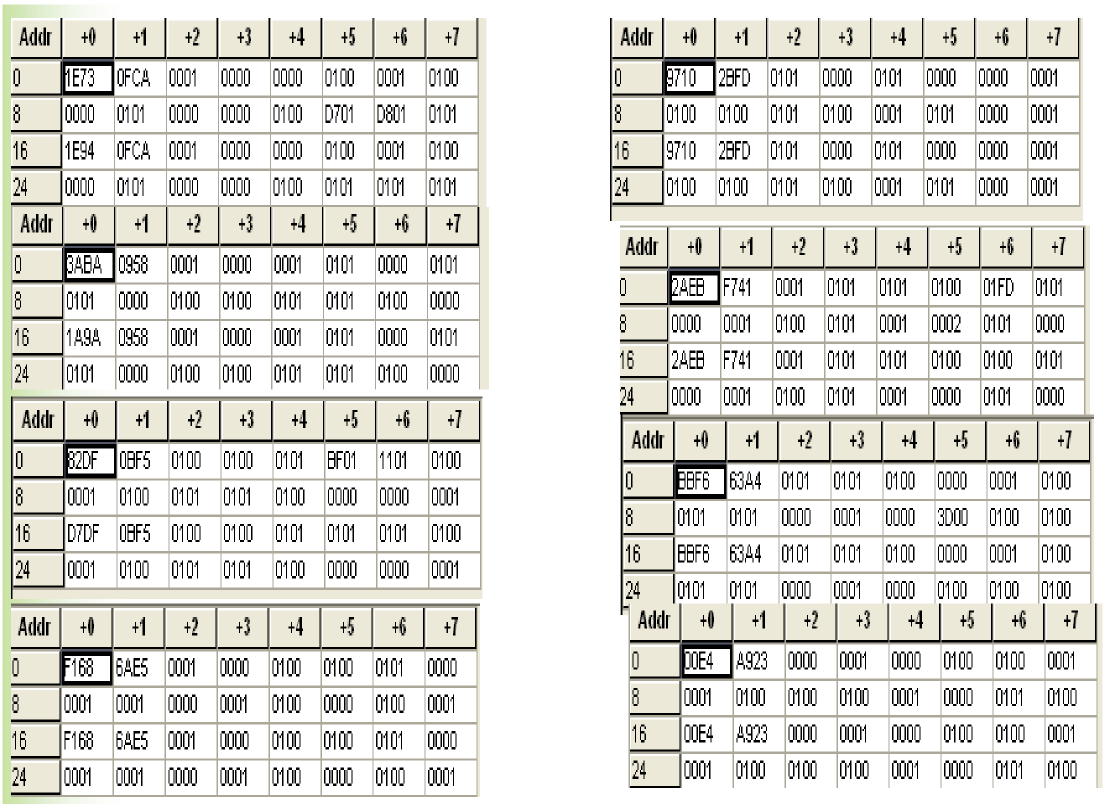
Clock:0~128:buff input memory

419~435:buff output memory

So the through is:

(best): (223*8bit/435clock)*110.63MHz= 453Mbps>150Mbps

(50Mhz): (223*8bit/435cycle)*50MHz=205Mbps >150Mbps



Addr	+0	+1	+2	+3	+4	+5	+6	+7
0	1E73	0FCA	0001	0000	0000	0100	0001	0100
8	0000	0101	0000	0000	0100	D701	D801	0101
16	1E94	0FCA	0001	0000	0000	0100	0001	0100
24	0000	0101	0000	0000	0100	0101	0101	0101

Addr	+0	+1	+2	+3	+4	+5	+6	+7
0	3ABA	0958	0001	0000	0001	0101	0000	0101
8	0101	0000	0100	0100	0101	0101	0100	0000
16	1A9A	0958	0001	0000	0001	0101	0000	0101
24	0101	0000	0100	0100	0101	0101	0100	0000

Addr	+0	+1	+2	+3	+4	+5	+6	+7
0	82DF	0BF5	0100	0100	0101	BF01	1101	0100
8	0001	0100	0101	0101	0100	0000	0000	0001
16	D7DF	0BF5	0100	0100	0101	0101	0101	0100
24	0001	0100	0101	0101	0100	0000	0000	0001

Addr	+0	+1	+2	+3	+4	+5	+6	+7
0	F168	6AE5	0001	0000	0100	0100	0101	0000
8	0001	0001	0000	0001	0100	0000	0100	0001
16	F168	6AE5	0001	0000	0100	0100	0101	0000
24	0001	0001	0000	0001	0100	0000	0100	0001

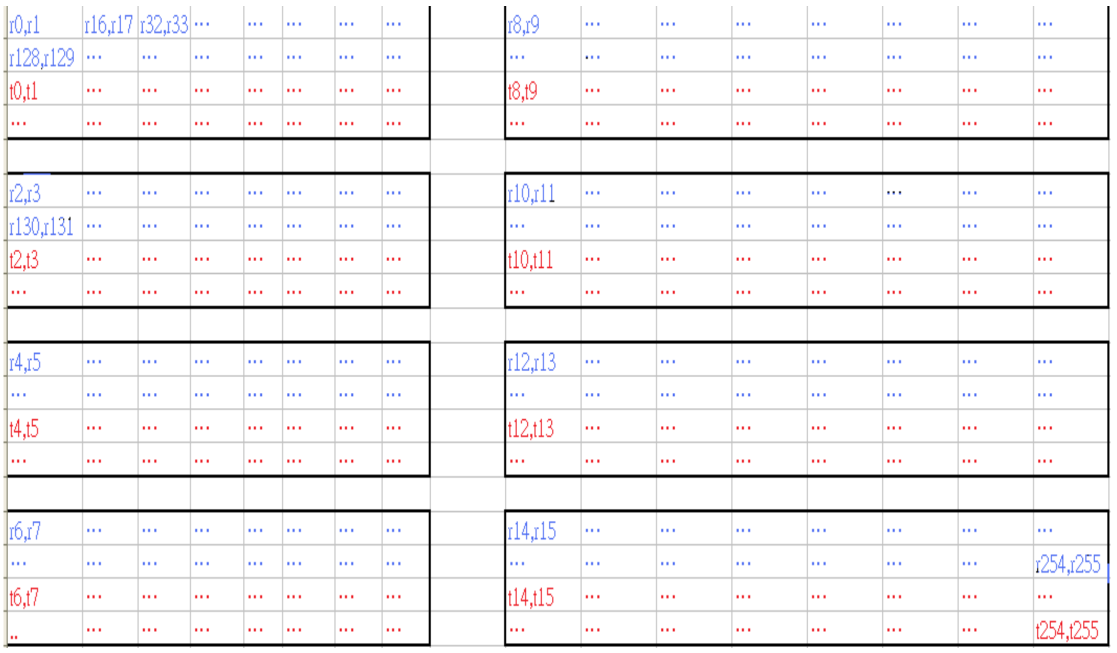
Addr	+0	+1	+2	+3	+4	+5	+6	+7
0	9710	2BFD	0101	0000	0101	0000	0000	0001
8	0100	0100	0101	0100	0001	0101	0000	0001
16	9710	2BFD	0101	0000	0101	0000	0000	0001
24	0100	0100	0101	0100	0001	0101	0000	0001

Addr	+0	+1	+2	+3	+4	+5	+6	+7
0	2AEB	F741	0001	0101	0101	0100	01FD	0101
8	0000	0001	0100	0101	0001	0002	0101	0000
16	2AEB	F741	0001	0101	0101	0100	0100	0101
24	0000	0001	0100	0101	0001	0000	0101	0000

Addr	+0	+1	+2	+3	+4	+5	+6	+7
0	BBF6	63A4	0101	0101	0100	0000	0001	0100
8	0101	0101	0000	0001	0000	3D00	0100	0100
16	BBF6	63A4	0101	0101	0100	0000	0001	0100
24	0101	0101	0000	0001	0000	0100	0100	0100

Addr	+0	+1	+2	+3	+4	+5	+6	+7
0	00E4	A923	0000	0001	0000	0100	0100	0001
8	0001	0100	0100	0100	0001	0000	0101	0100
16	00E4	A923	0000	0001	0000	0100	0100	0001
24	0001	0100	0100	0100	0001	0000	0101	0100

Fig 5.18



r0,r1	r16,r17	r32,r33	...	...	...	...	...
r128,r129	...	...	...	...	...	...	...
t0,t1	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
r2,r3	...	...	...	...	...	...	...
r130,r131	...	...	...	...	...	...	...
t2,t3	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
r4,r5	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
t4,t5	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
r6,r7	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
t6,t7	...	...	...	...	...	...	...
..	...	...	...	...	...	...	...

r8,r9	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
t8,t9	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
r10,r11	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
t10,t11	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
r12,r13	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
t12,t13	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
r14,r15	...	...	...	...	...	...	...
...	...	...	...	...	...	...	r254,r255
t14,t15	...	...	...	...	...	...	...
...	...	...	...	...	...	...	t254,t255

Fig. 5.19

Fig. 5.19 shows the contents of memories(in hexadecimal), and each block of memory contain two symbols. There are eight memories to meet the requirement of 16 parallelism (because we only can read one memory block in each memory). By using eight memories, the system can input or output 16 symbols in one clock, and the format of Fig 5.19 is showed in Fig. 5.18 (blue words mean received symbol, and red words mean corrected symbol). Each block has 32 bits, the MSB 16 bits for a symbol, and the LSB 16 bits for another symbol.

For example:

The first 16 received symbol:r0,r1,r2,r3…r15 is   1E,73,3A,BA,…,E4

The first 16 corrected symbol:t0,t1,t2,t3…t15 is   00,00,01,01,…,01

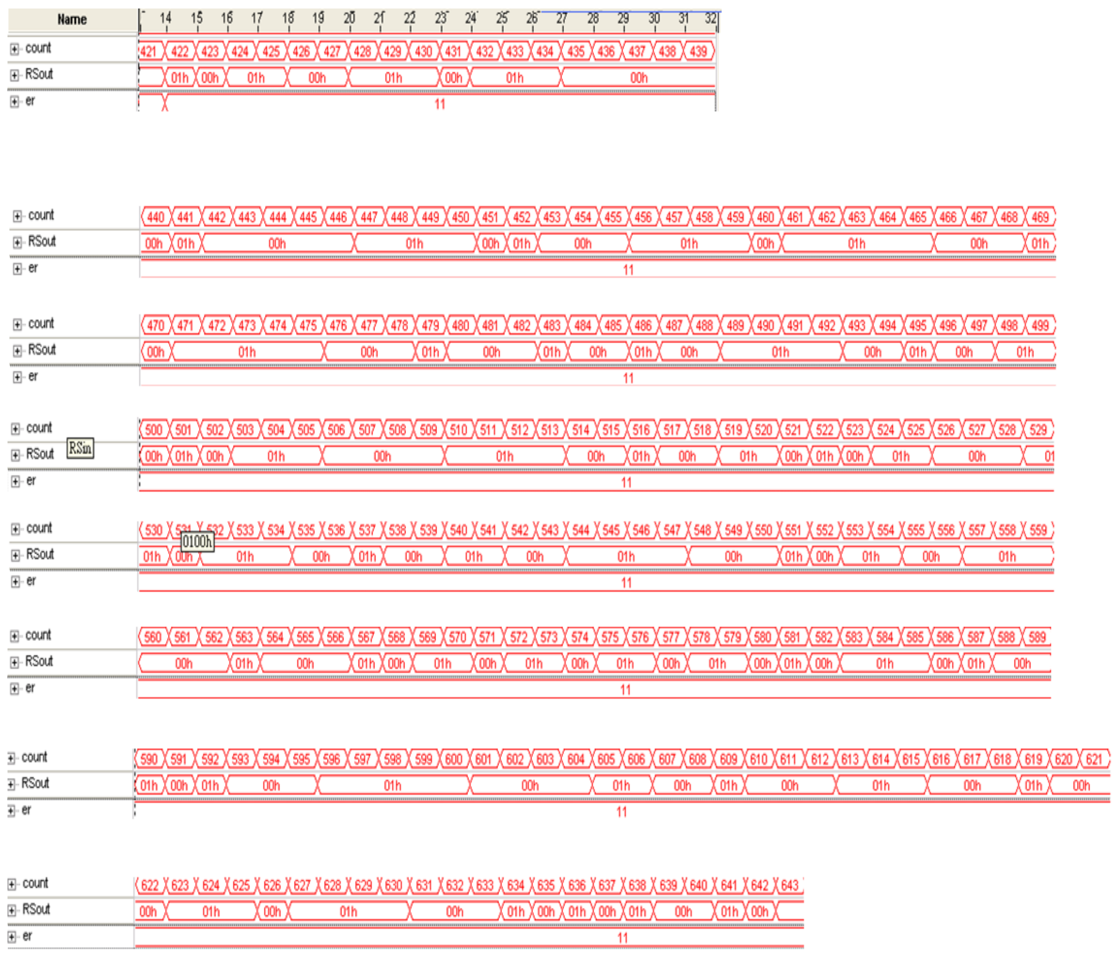


Fig. 5.20

The pin port indicate as follows.

count :how many clock we used

RSout:the output of our testing(corrected symbols without redundency)

er:error numbers we have found

Fig. 5.20 is the simulation result with SignalTap II, which show all of the captutured ouputs from FPGA board, it is equal to the data of Fig. 5.18 , so this program not only useful in software testing, but also useful in FPGA board.

6 Conclusion and future work

After introduction and discussion about the interface analysis and Reed-Solomon code implementation of the IO 21000 board, the evaluation of the IO 21000 board reproduction and future work suggestion are described in this chapter.

The interface analysis of IO 21000 board are classified as outward-board and inward-board interface. For outward-board interface, we propose the most possible frame structure and verification of this proposed structure. The future work for outward-board interface is that all the remained things is to make sure each un-identified column definitely by some other way which can be helped by NSPO or by measurement which is going to continue the unfinished tasks. After getting this unknown information, we can utilize our own technology to reproduce a brand-new IO 21000 board by our country. For inward-board interface, we provide all the information of inward-board hardware components from their datasheet which including pins information and each hardware function. The future work of inward-board interface is that we first should make sure the connection relation of all inward-board hardware components. That is to say, we should recover the original each hardware relation of connection when the IO 21000 board was made. Even if we fortunately obtain the whole-board interconnection of hardware components, we still a lot of obstacles to meet such as soft interface control (graphic user interface…) and so on. Therefore, by comparing and combining the two tasks which are inward- and outward-board interface analysis, we suggest that the further outward-board interface analysis is the best choice and a feasible way that the next project researchers can do in the future.

We have shown that the implementation of Reed-Solomon encoder and decoder are feasible for IO 21000 board reproduction in this project. In the future work, we think that we can do the implementation of the other IO 21000 board functions which are PRN-descrambler, CRC Checker and frame synchronization. Then, we combine Reed-Solomon decoder and the other three functions into an integrated function. Once we can unveil the secret of the outward board interface analysis, we consider that the day we reproduce a brand-new IO 21000 board by our-selves country is no so far.

Reference

[1] Elbert, Bruce R, Introduction to satellite communication, Artech House, Boston, 2008

[2] NSPO , http://www.nspo.org.tw/2008c/

[3] H, Myer, Digital communication receivers, 1998.

[4] B, Razavi, RF Microelectronics, Prentice Hall, 1998

[5] A. Loke and F. Ali, "Direct conversion radio for digital mobile phones—Design issues, status and trends," *IEEE Trans. Microwave Theory Tech.*, vol. 50, pp. 2422-2435, Nov. 2002.

[6] Kongsberg Spacetec AS, online source, http://www.spacetec.no/

[7] Kongsberg Spacetec AS, MEOS Capture Operational and Maintenance Manual, 2008

[8] J. G. Proakis, Digital communications, 4th edition, McGraw-Hill, 2001

[9] S. Lin and D. J. Costello, Jr., Error Control Coding: Fundamentals and Application*s*, Englewood Cliffs, NJ: Prentice-Hall, 1983.

[10] CCSDS 131.0-B-1, blue book, "TM SYNCHRONIZATION AND CHANNEL CODING ", September, 2003.

[11] Intel, "FW80321 IO Processor datasheet", January, 2005

[12] Xilinx, ”XCF32P VOG48 datasheet”, March 14, 2005

[13] Xilinx XC2C256 CoolRunner-II CPLD datasheet, March 8, 2007

[14] IDT,”72V36104 datasheet”, March, 2001

[15] IDT “72T18125 datasheet”, Feburary, 2009

[16] Xilinx, “Virtex-4 FPGA packaging and pin out specification datasheet”, September, 2008

[17] Xilinx, “Initiator/Target v6.5 for PCI-X User Guide”, 2007.

[18] Xilinx, “Virtex-4 ML455 PCI/PCI-X Development Kit *User Guide*”, 2005.

[19] Takahiro Yamada, “CCSDS Telemetry/Telecommand Standards Restructured as Communications Protocols”.

[20] Analog device, “ADCMP561/ADCMP562 Dual High Speed PECL Comparators”, 2004.

[21] Austin LynxTM, “SIP non-isolated Power Module, Progorammable: 3Vdc-5.5Vdc input; 0.75Vdc to 3,63Vdc Output; 10A output current”, lineage power, April 1, 2008.

[22] TDK-Lambda, “PL5 series 5A Non-isolated DC-DC Converters”, July 2009.

[23] Numonyx, “strata flash embedded memory”, Nov 2007.

[24] 100 frames extract from MEOS capture system in Appedix.

[25] CCSDS 732.0-B-2, blue book, “AOS space data link protocol recommend standard”, July 2006

[26] S. B. Wicker and V. K. Bhargava, *Reed–Solomon Codes and Their Applications*. Piscataway, NJ: IEEE Press, 1994.

[27] S. Lin, *Error Control Coding: Fundamentals and Applications*. Englewood Cliffs, NJ: Prentice-Hall, 1983.

[28] S. B. Wicker, *Error Control Systems for Digital Communication and Storage*. Englewood Cliffs, NJ: Prentice-Hall, 1995.

[29] G. Fettweis and H. Meyr, “High-speed parallel Viterbi decoding: Algorithm and VLSI-architecture,” *IEEE Commun. Mag.*, pp. 46–55, May 1991.

[30] T. B. Pei and C. Zukowski, “High-speed parallel CRC circuits in VLSI,” *IEEE Trans. Commun.*, vol. 40, pp. 653–657, Apr. 1992.

[31] T. K. Matsushima, T. Matsushima, and S. Hirasawa, “Parallel encoder and decoder architecture for cyclic codes,” *IEICE Trans. Fundamentals*, vol. E79-A, no. 9, pp. 1313–1323, Sept. 1996.

[32] ITU-T Recommendation G.975, “Forward Error Correction for Submarine Systems,” ITU, Geneva, Switzerland, Oct. 2000.

[33] R. T. Chien, “Cyclic decoding procedures for Bose–Chaudhuri–Hocquenghem codes,” *IEEE Trans. Inform. Theory*, vol. IT-10, pp. 357–363,1964.

[34] P. Tong, “A 40-MHz encoder-decoder chip generated by a Reed–Solomon code compiler,” in *Proc. EEE 1990 Custom Integrated Circuits Conf.*, May 1990, pp. 13.5.1–13.5.4.

[35] H. M. Shao, T. K. Truong, L. J. Deutsch, J. H. Yuen, and I. S. Reed, “A VLSI design of a pipeline Reed–Solomon decoder,” *IEEE Trans. Comput.*, vol. C-34, pp. 393–403, May 1985.

[36] H. M. Shao and I. S. Reed, “On the VLSI design of a pipeline Reed–Solomon decoder using systolic arrays,” *IEEE Trans. Comput.*, vol. 37, pp. 1273–1280, Oct. 1988.

[37] W. Wilhelm, “A new scalable VLSI architecture for Reed–Solomon decoders,” *IEEE J. Solid-State Circuits*, vol. 34, pp. 388–396, Mar. 1999.

[38] H. Lee, M. L. Yu, and L. Song, “VLSI design of Reed–Solomon decoder architectures,” *IEEE Int. Symp. Circuits Syst.*, vol. 5, pp. 705–708, May 2000.

[39] H. Lee, “A VLSI design of a high-speed Reed–Solomon decoder,” in *Proc. Int. ASIC/SOC Conf.*, Sept. 2001, pp. 316–320.

[40] S. R. Whitaker, J. A. Canaris, and K. B. Cameron, “Reed–Solomon VLSI Codec for advanced television,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 1, pp. 230–236, June 1991.

[41] S. Kwon and H. Shin, “An area-efficient VLSI architecture of a Reed–Solomon decoder/encoder for digital VCRs,” *IEEE Trans.Consumer Electron.*, vol. 43, pp. 1019–1027, Nov. 1997.

Appendix

Source code

1. **Fig 5.6: Syndrome generator**

- **syndrome**
- **syn_lib:** database of **syndrome**

2. **Fig. 5.7: Modify Euclidean cell**

- **euclidean_cell**
- **euclidean_lib:** database of **euclidean_cell** and **chien_search_p16_t16**
- **feedback_ckt:** a circuit used in **degree_computation**
- **degree_computation:** upper part of Figure

3. **Fig. 5.9: 4-cell Modify Euclidean**

- **euclidean_4cells**

4. **Fig. 5.11: Chien search**

- **chien_search_p16_t16**

5. Fig. 5.12: Forney algorithm

- **Forney**
- **memory_inverse:** rom inverse
- **multi:** reduced circuit of Fig. 3.10
- **omega_p16_t16:** same circuit of **chien_search_p16_t16**

6. Others

- **RAM_32_16:** RAM used for save input and output
- **top:** top of the design, which include all of the source codes above
- **test_bench:** test input circuit