JOURNAL OF SCIENCE AND APPLIED ENGINEERING (JSAE), October 2018, Vol 1(2), 68-74
E-ISSN: 2621-3753
P-ISSN: 2621-3745

Designing VHDL to Simulate the Error Correction of Hamming Code

A. Mahmudi [1], S. Achmad [2]
[1, 2] Department of Informatics Engineering, Institut Teknologi Nasional Malang, Malang, Indonesia
Email: amahmudi@hotmail.com

Abstract

The role of error detection and error correction for the data bit by the receiver is very important because the sender does not need to repeat the transmissions [1]. Thus, the speed and reliability in transmitting data information can be maintained. This study aims to implement simulation the Forward Error Correction (FEC) method in verifying and correcting data errors received by using simulation. To support FEC method, study utilizes visual basic software so that it can be used as one of the quasi-experimental modules in the data communication laboratory. The Forward Error Correction (FEC) method is a method that can correct data errors in the receiver. This method uses simulated Hamming codes on the computer so that the detection and correction process can be clearly demonstrated on the monitor screen. This simulation can be used as a quasi-experimental module in a data communication laboratory. The simulation results show that the Hamming code (17, 12) codec has been running as expected.

***Keywords :** Simulation, Forward Error Correction (FEC), quasi-experimental module..*
Paper type Research paper

INTRODUCTION (*HEADING 1*)

The success in delivering information from the sender to the receiver is the key important outcome in determining the reliability of a communication system [2]. Reliability of a data communication system is not only measured by the data transfer rates in the bits per second, namely bit rate, but also the success of the data sent. The success of the data communication is determined by the clear, understandable, and correct information to the recipient [3], [4]. In delivering data information by voice and data, the transmitter can be transmitted by wires such as coaxial and fibre optic, by the terrestrials, and by satellites. For example, fig 1 illustrates the simple communication block diagram.

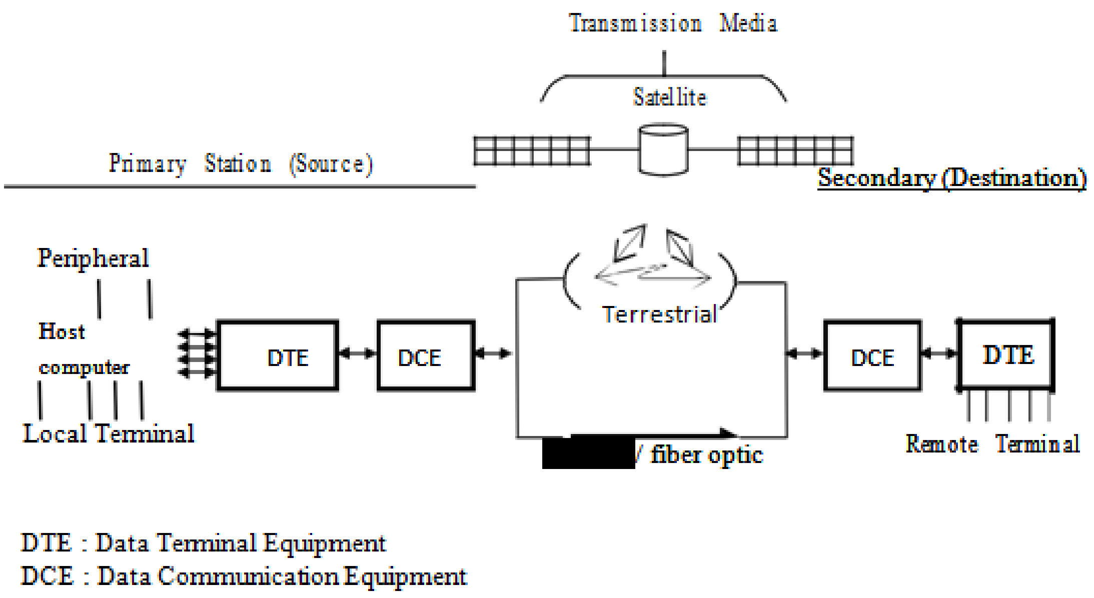


Fig 1. The simple block diagram of communication network

In processing the communication data, there is a possibility of error received by the recipient. Therefore, the sender must resend data until it is corrected and fitted. It causes the overall time in delivering data delayed. There are two causes of errors in receiving data. Firstly, the error is caused by synchronising data or the clock by the sender or the recipient. Secondly, there is interferes of the canals which are closed to the transmission media.

Data sent by the source to the recipient as the destination will be synchronised and modulated. Then, it will be sent to transmission media. The recipient should detect, check, and correct the data errors to deliver data quickly. Also, it is not requisite to re-transfer the error data received by the recipient. One of the methods to detect and correct the error data is known as the Forward Error Correction (FEC) method [3]. The researchers utilise this method because this method has high accuracy. To date, this method is employed the most as the correction tool in communication data.

FEC can detect and correct the inaccuracy data send by the source [1], [3], [5]. Then, error data can be re-sent by fixing the data, thus data sent will be the same as the previous data sent by the source. The FEC method detects and corrects data errors in the recipient without asking the sender to send again data. In this method, the arrangement of the bits of data sent (string data) is added into the Hamming code. The formula of the number of bits of data sent is:

$$2^n \geq m + n + 1 \qquad (1)$$

Where:

n = the bits number of the Hamming codes
m = the bits number of character data

By using the aforementioned formula, the bits number of Hamming codes can be determined. This Hamming code is inserted between the bits of character data that will be sent by the sender. Character data (string data) that has been inserted in the Hamming code will create a data called data stream. The location of the insertion of each bit of the Hamming code must be agreed between the sender and the recipient. Therefore, the recipient knows the exact location of each bit of the Hamming code on the data stream received. After the data stream is created, then an exclusive OR between the positions of binary and bit ‘1’). The first bit ‘1’of the data stream consists of one character. After determining the first bit ‘1’, binary is located at the position of the second bit ‘1’ [6], [7]. It is then continued until reaching the end of result which is the exclusive OR of the bits of the Hamming code. The bits of the Hamming code derived is inserted into the data stream. Then, it is sent to the receiver.

METHOD

The design of data error correction by simulation using the FEC method. In this case, the researchers utilize visual basic software to support the design. There are the following stages to process this simulation design.

1. Entering binary data as the input string data. In this case, the data should be made at least 4 bits and a maximum of 20 bits. The assumption is that no character data used is smaller than 4 bits and more than 20 bits. By inputting data whether it is smaller than 4 bits or greater than 20 bits, the data can be rejected.
2. Calculating the length of the string data entered.
3. Inserting the value of n that express the bit length for the Hamming code. In this step, the process should meet the rule of $2n \geq m + n + 1$. There are two ways to determine the value of n in the simulation. First, entering the value of any n. The calculation process will be displayed. Then, the result will appear whether the value of n meets the requirements of $2n \geq m + n + 1$ or not. This process will be recounted if the value does not meet the requirement. The user should enter another n value, then, recount again until meeting the requirement. Second, the direct calculation analyses n value that meet the requirement.
4. After the value of n is obtained, the length of the data stream can be determined and displayed by m + n.
5. Determine the position of every bit in the Hamming code.
6. Calculate the Hamming code with the calculation method as described above, this calculation process is displayed on the monitor screen. The results of the Hamming code obtained are displayed, as well as the bits arrangement in the data stream that has inserted the Hamming code displayed.
7. Determine whether there is a bit error in the data stream during the transmission. If there is no bit error then the data stream is received by the recipient, this data stream is displayed on the monitor screen.
8. Check the error on the data stream received. Add the data stream with Exclusive OR. The result is ‘0’, if there is no error. It means that the bit of ‘0’ implies no errors bit occur in the data stream received.
9. If the bit error in the data stream received occurs, then the process should return to step 7. The researcher should determine the location of the wrong bit of position. The data stream bit arrangement is displayed on the monitor screen, then do steps 10 and 11.

10. Check the error on the data stream received by the recipient by making an addition with Exclusive OR. If an error occurs, the calculation results will produce bits that are not equal to zero. It means that a bit error occurs in the

data stream received. The final value received is expressed as a decimal, which means that the decimal value indicates the location of the wrong bit position in the data stream.

11. Correct these bit errors by copying the bit value to the bit position indicated by the decimal value. The final result after correction is displayed on the monitor screen in the form of a bit arrangement in the data stream.

The error of the data that can be corrected by the FEC method is the error of one bit of data in the received data stream. However, this error does not belong to the Hamming code bits.

DISCUSSION

The process undertaken by the recipient is reading the bits of the Hamming code on the data stream received according to the location of the bit of Hamming code. It should be agreed on by the sender. The Hamming code received in Exclusive OR is placed in the position of the first bit '1' (binary from the position of the bit position '1') to the received data stream. The result on Exclusive OR replaced to the position of the second bit '1'. It is continued until reaching the final result that indicates whether the bits obtained wrong or not. If the end result produces a zero decimal, it is concluded that there is no data error. Meanwhile, data error confirmed means the end result does not produce a zero decimal and the decimal states the location of the wrong bit position in the data stream. To correct the bit error, the researchers complement the wrong bit. For example, the bit data string sent is 12 bits, i.e. 1001-1001-1001. In detecting the error, the researchers utilise some steps as follow.

1. Determining the number of bits of the Hamming codes by using the formulation (1):

Where:
m= 12 (the numbers of bit)
n = the number bits of the Hamming code

If	n	=	2	,	then	2^2	$\geq 12+2+1$	, so	$4 \geq 15$	wrong
If	n	=	3	,	then	2^3	$\geq 12+3+1$	, so	$8 \geq 16$	wrong
If	n	=	4	,	then	2^4	$\geq 12+4+1$	, so	$16 \geq 17$	wrong
If	n	=	5	,	then	2^5	$\geq 12+5+1$	, so	$32 \geq 18$	correct

The bit specified is 5 (n=5). So, the number of bit of stream data which is the data sent is the number of data added by the number of the Hamming code [3]. For example, the number of bit of stream data is 12 and the number of bit of the Hamming code is 17. Then, 12 bit added by 5 is 17 bit (12 bit + 5 bit = 17 bit).

2. Determining the position of the bit of the Hamming code (H_1, H_2, H_3, H_4, H_5) in the stream data, such as:

The bits position	17 16 15 14 -	13	121110-	9	8	7	6	-	5	4	3	2	1
	1001-	1	001-	1	0	0	1	-	H_5	H_4	H_3	H_2	H_1

3. Adding the position of the bits with Exclusive OR which equals to bit '1'.

The bits position	Binary
6	0 0 1 1 0
9	0 1 0 0 1
ExOR	0 1 1 1 1
10	0 1 0 1 0
ExOR	0 0 1 0 1
13	0 1 1 0 1
ExOR	0 1 0 0 0
14	0 1 1 1 0
ExOR	0 0 1 1 0
17	1 0 0 0 1
ExOR	1 0 1 1 1

Therefore, the bits of the Hamming code are $H_5 = 0$, $H_4= 0$, $H_3= 1$, $H_2= 1$, and $H_1= 1$. Also, the stream data sent by the sender are 1001-1001-1001-00111.

The bits position	17	16	15	14-13	12	11	10 - 9	8	7	6 - 5	4	3	2	1
	1	0	0	1 - 1	0	0	1 - 1	0	0	1 - 1	0	1	1	1

4. At the recipient, the data stream will be added to the Exclusive OR through the position of bits '1'

The bits position	Binary	
Hamming code	1 0 1 1 1	
6	0 0 1 1 0	
ExOR	1 0 0 0 1	
9	0 1 0 0 1	
ExOR	1 1 0 0 0	
10	0 1 0 1 0	
ExOR	1 0 0 1 0	
13	0 1 1 0 1	
ExOR	1 1 1 1 1	
14	0 1 1 1 0	
ExOR	1 0 0 0 1	
17	1 0 0 0 1	
ExOR	0 0 0 0 0	Decimal 0

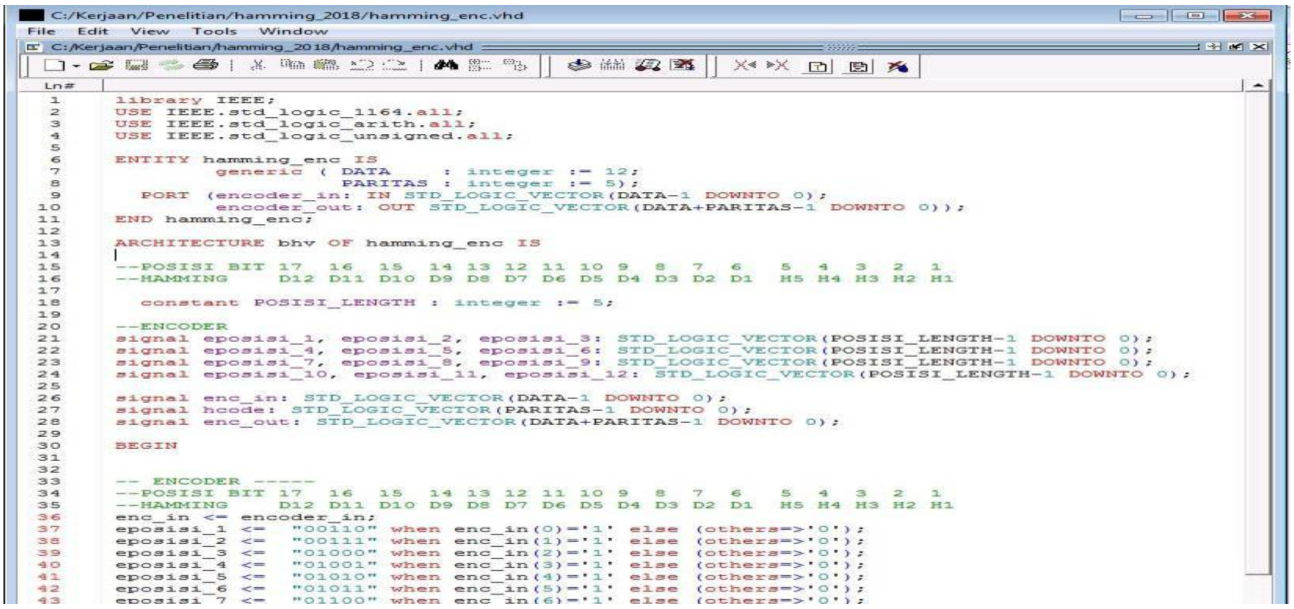
The final result is decimal 0. It means that there are no errors in the data received by the recipient. For example, there is a data error that occurs during transmission, namely the change of the 6th bit in the data stream. Then, the bit '1' shift to be the bit '0' by following the process below.

The bits position	17	16	15	14 - 13	12	11	10-9876-5	4	3	2	1
	1	0	0	1 - 1	0	0	1-1000-1	0	1	1	1

The bits position	Binary	
Hamming code	1 0 1 1 1	
9	0 1 0 0 1	
ExOR	1 1 1 1 0	
10	0 1 0 1 0	
ExOR	1 0 1 0 0	
13	0 1 1 0 1	
ExOR	1 1 0 0 1	
14	0 1 1 1 0	
ExOR	1 0 1 1 1	
17	1 0 0 0 1	Decimal 6
ExOR	0 0 1 1 0	

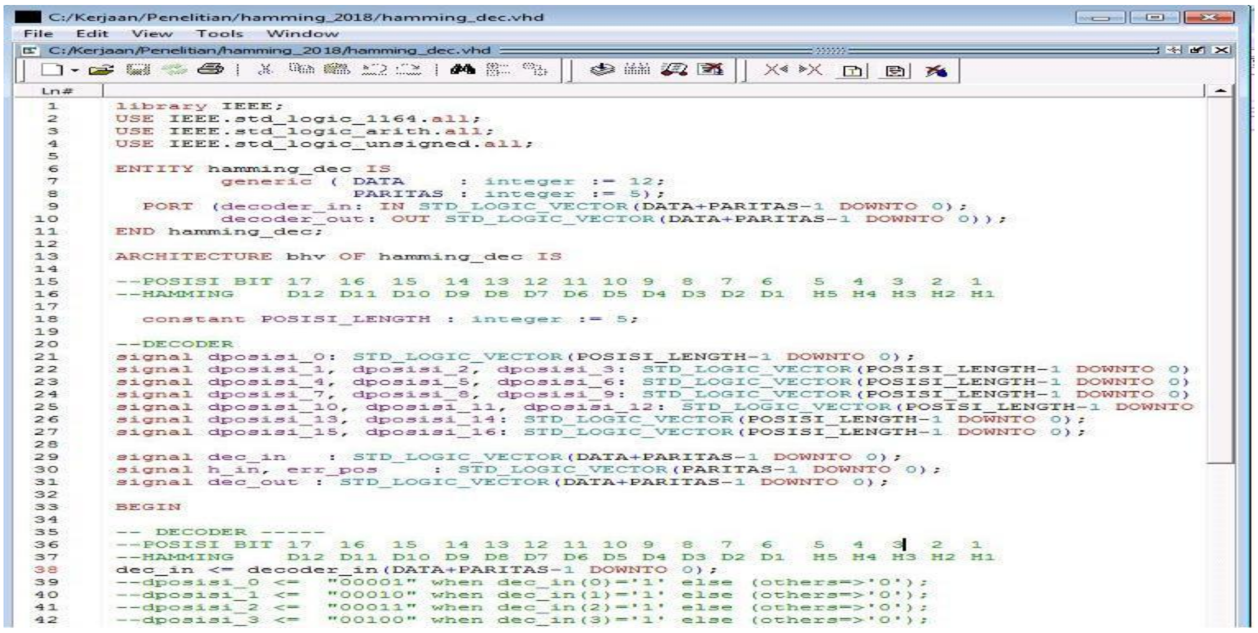
The final result is decimal 6. It means that the error data received by the recipient is in the bit position of 6. Therefore, in the stream data, the correction was conducted by complementing the bit of 6 from the bit ‘0’ to be the bit of ‘1’. So, the bit of data received is the same as the data sent.

The result of VDHL design is illustrated in the following fig 2. The programming uses VDHL programming language. Also, the script display from VHDL Hamming encoder depicts in fig 2. In fig 3. The script of VHDL Hamming decoder is shown.


C:/Kerjaan/Penelitian/hamming_2018/hamming_enc.vhd
File Edit View Tools Window
Ln#

```vhdl
library IEEE;
USE IEEE.std_logic_1164.all;
USE IEEE.std_logic_arith.all;
USE IEEE.std_logic_unsigned.all;

ENTITY hamming_enc IS
        generic ( DATA     : integer := 12;
                  PARITAS  : integer := 5);
  PORT (encoder_in: IN STD_LOGIC_VECTOR(DATA-1 DOWNTO 0);
        encoder_out: OUT STD_LOGIC_VECTOR(DATA+PARITAS-1 DOWNTO 0));
END hamming_enc;

ARCHITECTURE bhv OF hamming_enc IS

--POSISI BIT 17  16  15  14 13 12 11 10 9  8  7  6   5  4  3  2  1
--HAMMING      D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 D1  H5 H4 H3 H2 H1

  constant POSISI_LENGTH : integer := 5;

--ENCODER
signal eposisi_1, eposisi_2, eposisi_3: STD_LOGIC_VECTOR(POSISI_LENGTH-1 DOWNTO 0);
signal eposisi_4, eposisi_5, eposisi_6: STD_LOGIC_VECTOR(POSISI_LENGTH-1 DOWNTO 0);
signal eposisi_7, eposisi_8, eposisi_9: STD_LOGIC_VECTOR(POSISI_LENGTH-1 DOWNTO 0);
signal eposisi_10, eposisi_11, eposisi_12: STD_LOGIC_VECTOR(POSISI_LENGTH-1 DOWNTO 0);

signal enc_in: STD_LOGIC_VECTOR(DATA-1 DOWNTO 0);
signal hcode: STD_LOGIC_VECTOR(PARITAS-1 DOWNTO 0);
signal enc_out: STD_LOGIC_VECTOR(DATA+PARITAS-1 DOWNTO 0);

BEGIN


-- ENCODER -----
--POSISI BIT 17  16  15  14 13 12 11 10 9  8  7  6   5  4  3  2  1
--HAMMING      D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 D1  H5 H4 H3 H2 H1
enc_in <= encoder_in;
eposisi_1 <=  "00110" when enc_in(0)='1' else (others=>'0');
eposisi_2 <=  "00111" when enc_in(1)='1' else (others=>'0');
eposisi_3 <=  "01000" when enc_in(2)='1' else (others=>'0');
eposisi_4 <=  "01001" when enc_in(3)='1' else (others=>'0');
eposisi_5 <=  "01010" when enc_in(4)='1' else (others=>'0');
eposisi_6 <=  "01011" when enc_in(5)='1' else (others=>'0');
eposisi_7 <=  "01100" when enc_in(6)='1' else (others=>'0');
```

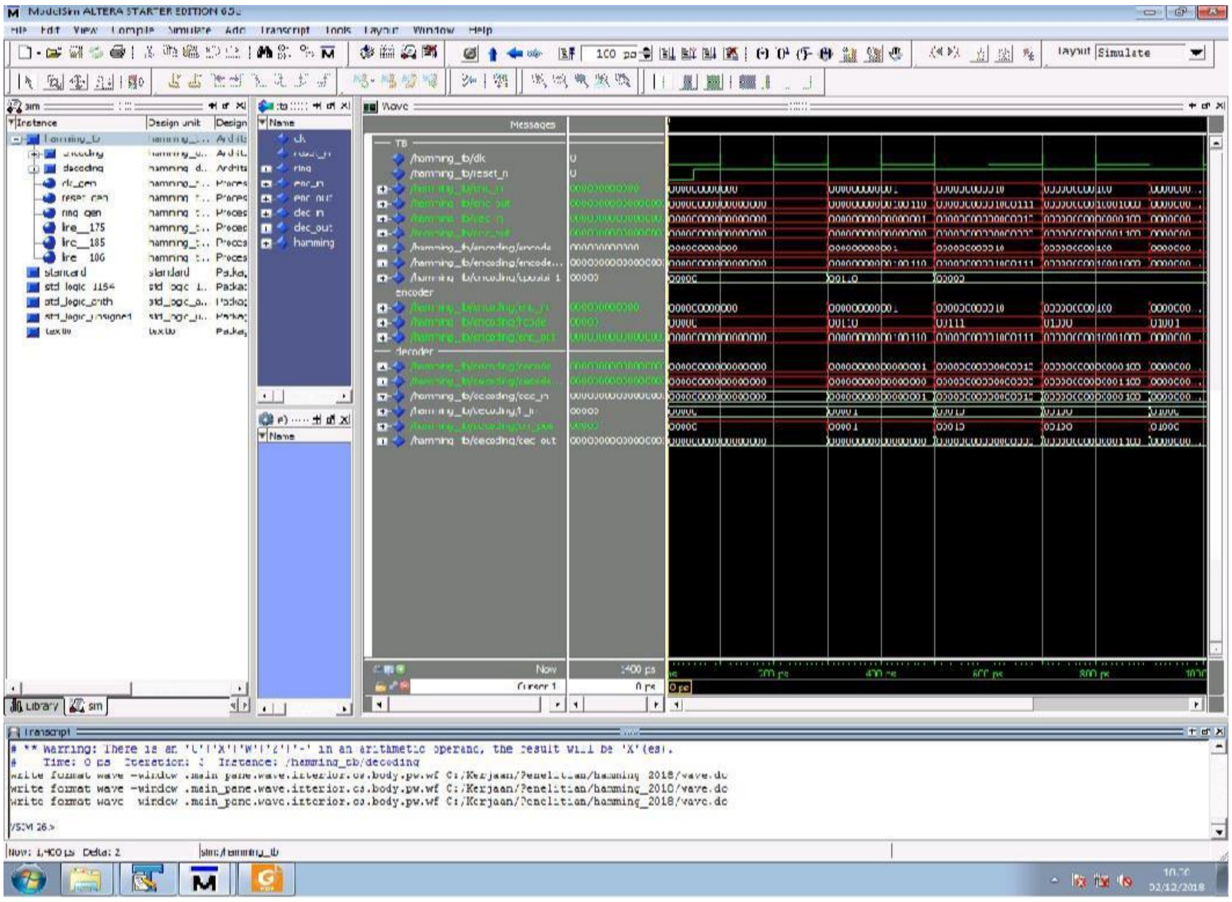
Fig 2. The design of VHDL Hamming encoder


C:/Kerjaan/Penelitian/hamming_2018/hamming_dec.vhd
File Edit View Tools Window
Ln#

```vhdl
library IEEE;
USE IEEE.std_logic_1164.all;
USE IEEE.std_logic_arith.all;
USE IEEE.std_logic_unsigned.all;

ENTITY hamming_dec IS
        generic ( DATA     : integer := 12;
                  PARITAS  : integer := 5);
  PORT (decoder_in: IN STD_LOGIC_VECTOR(DATA+PARITAS-1 DOWNTO 0);
        decoder_out: OUT STD_LOGIC_VECTOR(DATA+PARITAS-1 DOWNTO 0));
END hamming_dec;

ARCHITECTURE bhv OF hamming_dec IS

--POSISI BIT 17  16  15  14 13 12 11 10 9  8  7  6   5  4  3  2  1
--HAMMING      D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 D1  H5 H4 H3 H2 H1

  constant POSISI_LENGTH : integer := 5;

--DECODER
signal dposisi_0: STD_LOGIC_VECTOR(POSISI_LENGTH-1 DOWNTO 0);
signal dposisi_1, dposisi_2, dposisi_3: STD_LOGIC_VECTOR(POSISI_LENGTH-1 DOWNTO 0)
signal dposisi_4, dposisi_5, dposisi_6: STD_LOGIC_VECTOR(POSISI_LENGTH-1 DOWNTO 0)
signal dposisi_7, dposisi_8, dposisi_9: STD_LOGIC_VECTOR(POSISI_LENGTH-1 DOWNTO 0)
signal dposisi_10, dposisi_11, dposisi_12: STD_LOGIC_VECTOR(POSISI_LENGTH-1 DOWNTO
signal dposisi_13, dposisi_14: STD_LOGIC_VECTOR(POSISI_LENGTH-1 DOWNTO 0);
signal dposisi_15, dposisi_16: STD_LOGIC_VECTOR(POSISI_LENGTH-1 DOWNTO 0);

signal dec_in    : STD_LOGIC_VECTOR(DATA+PARITAS-1 DOWNTO 0);
signal h_in, err_pos     : STD_LOGIC_VECTOR(PARITAS-1 DOWNTO 0);
signal dec_out : STD_LOGIC_VECTOR(DATA+PARITAS-1 DOWNTO 0);

BEGIN

-- DECODER -----
--POSISI BIT 17  16  15  14 13 12 11 10 9  8  7  6   5  4  3  2  1
--HAMMING      D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 D1  H5 H4 H3 H2 H1
dec_in <= decoder_in(DATA+PARITAS-1 DOWNTO 0);
--dposisi_0 <=  "00001" when dec_in(0)='1' else (others=>'0');
--dposisi_1 <=  "00010" when dec_in(1)='1' else (others=>'0');
--dposisi_2 <=  "00011" when dec_in(2)='1' else (others=>'0');
--dposisi_3 <=  "00100" when dec_in(3)='1' else (others=>'0');
```

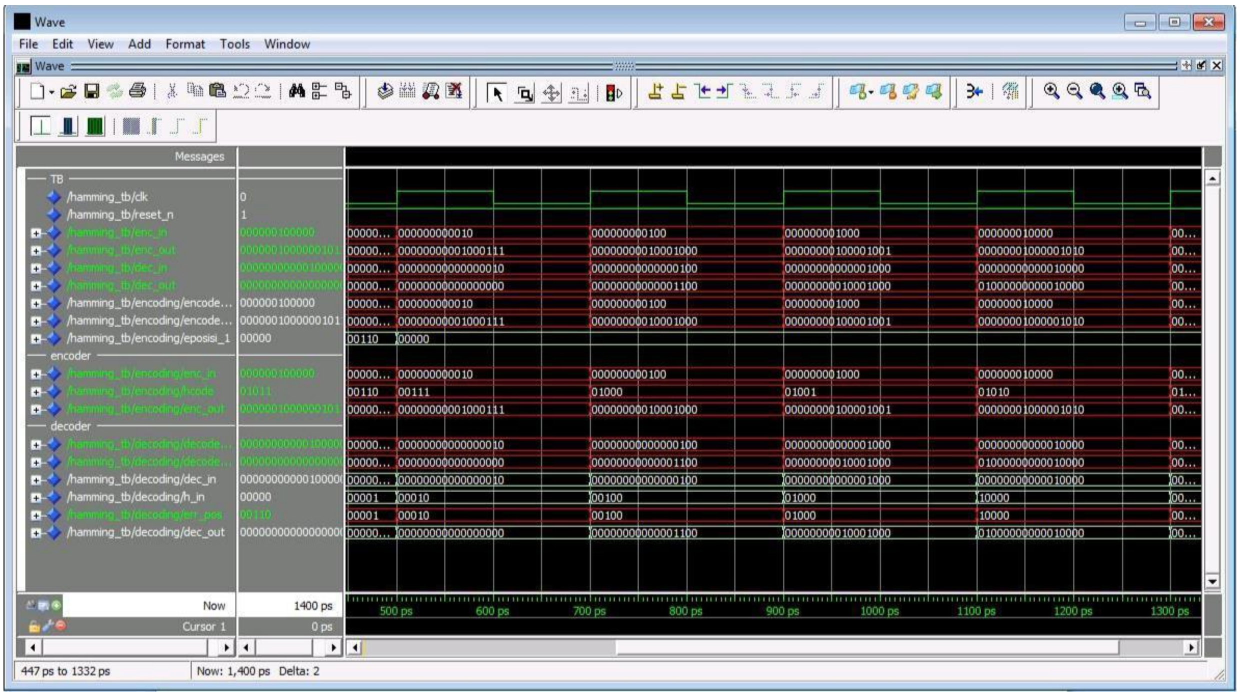
Fig 3. The design of VHDL Hamming decoder

The code Hamming design is simulated by Sim Altera 6.5e Model as shown in fig 4. The simulation result is shown in fig 5. The result shows that VHDL design for the Hamming code is associated with the expectation.



```
# ** Warning: There is an 'U'|'X'|'W'|'Z'|'-' in an arithmetic operand, the result will be 'X'(es).
#    Time: 0 ps  Iteration: 3  Instance: /hamming_tb/decoding
write format wave -window .main_pane.wave.interior.cs.body.pw.wf C:/Kerjaan/Penelitian/hamming_2018/wave.do
write format wave -window .main_pane.wave.interior.cs.body.pw.wf C:/Kerjaan/Penelitian/hamming_2010/wave.do
write format wave -window .main_pane.wave.interior.cs.body.pw.wf C:/Kerjaan/Penelitian/hamming_2018/wave.do

VSIM 26>
```

Fig 4. The display of ModelSim Altera 6.5e



```
TB
/hamming_tb/clk                       0
/hamming_tb/reset_n                   1
/hamming_tb/enc_in                    000000100000
/hamming_tb/enc_out                   00000010000000101
/hamming_tb/dec_in                    00000000000100000
/hamming_tb/dec_out                   00000000000000000
/hamming_tb/encoding/encode...        000000100000
/hamming_tb/encoding/encode...        00000010000000101
/hamming_tb/encoding/eposisi_1        00000
encoder
/hamming_tb/encoding/enc_in           000000100000
/hamming_tb/encoding/hcode            01011
/hamming_tb/encoding/enc_out          00000010000000101
decoder
/hamming_tb/decoding/decode...        00000000000100000
/hamming_tb/decoding/decode...        00000000000000000
/hamming_tb/decoding/dec_in           00000000000100000
/hamming_tb/decoding/h_in             00000
/hamming_tb/decoding/err_pos          00110
/hamming_tb/decoding/dec_out          00000000000000000
Now        1400 ps
Cursor 1   0 ps
447 ps to 1332 ps     Now: 1,400 ps  Delta: 2
```

Fig 5. The result of the Hamming code simulation

CONCLUSION

This research finds that the VHDL language programming was successfully applied to design the Hamming code (17, 12) encoder decoder. Then, the result from ModelSim Altera 6.5e showed that the Hamming code (17, 12) is running properly and meet the expectation. In addition, it can correct the data up to one error.

ACKNOWLEDGMENT

-

REFERENCES

[1] A. Fauzi and R. Rahim, "Bit Error Detection and Correction with Hamming Code Algorithm," *International Journal of Scientific Research in Science, Engineering and Technology (IJSRSET)*, vol. 3, no. 1, pp. 76–81, 2017.

[2] W. Tomasi, *Electronic communications systems: fundamentals through advanced*. Prentice Hall PTR, 1987.

[3] A. Masoomi and R. Hamzehiyan, "A New Approach for Detecting and Correcting Errors in the Satellite Communications Based on Ha mming Error Correcting Code," *International Journal of Computer Theory and Engineering*, vol. 5, no. 2, pp. 227–231, 2013.

[4] W. Stallings, "Data and computer communications." 2017.

[5] A. S. Ahmad Alfi Albar Lubis, Poltak Sihombing, "Perancangan Error Detection System And Error Correction System Menggunakan Metode Hamming Code Pada Pengiriman Data Text," *Jurnal Online Program Studi S1 Ilmu Komputer Fakultas Ilmu Komputer dan Teknologi Informasi*, vol. 1, 2012.

[6] R. L. Tokheim, *Digital electronics*. Glencoe, 1994.

[7] J. P. Hayes, *Introduction to digital logic design*. Addison-Wesley Longman Publishing Co., Inc., 1993.