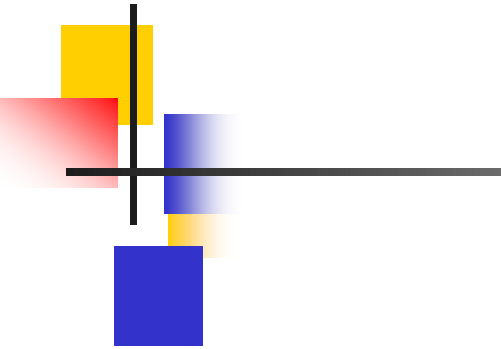


ECE641 Project

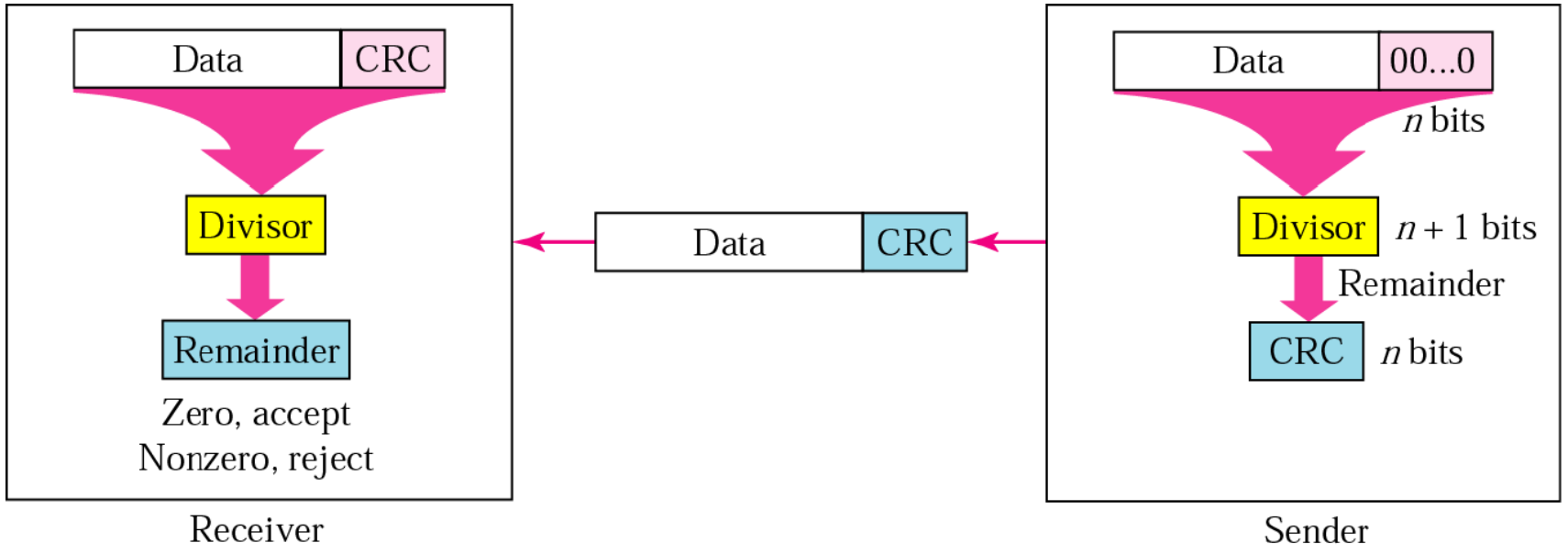
Cyclic Redundancy Check (CRC) on FPGA

Yanjia Sun

Introduction

- **Cyclic redundancy check (CRC) codes**
  - CRC is an error-detecting code designed to detect accidental changes to raw computer data, and is commonly used in digital networks and storage devices such as hard disk drives.
    - Invented by *W. Wesley Peterson*, and published in 1961
    - A type of linear block codes & a systematic *error detecting code*
  - A group of *error control bits* (which is the remainder of a polynomial division of a message polynomial by a generator polynomial) is appended to the end of the message block

CRC Generator and Checker



CRC Codes Format

- **Binary (N, k) CRC codes**
  - k message or data bits are encoded into N code bits by appending to the message bits a sequence of $n=N-k$ bits.

✓ ***Message bits:*** $\mathbf{m} = [m_{k-1} \quad m_{k-2} \quad \dots \quad m_1 \quad m_0]$

$$\mathbf{m}(X) = m_{k-1}X^{k-1} + m_{k-2}X^{k-2} + \dots + m_1X + m_0 \qquad \text{degree (k - 1)}$$

✓ ***Appended bits:*** $\mathbf{R} = [r_{n-1} \quad r_{n-2} \quad \dots \quad r_1 \quad r_0]$

$$\mathbf{R}(X) = r_{n-1}X^{n-1} + r_{n-2}X^{n-2} + \dots + r_1X + r_0 \qquad \text{degree (n - 1)}$$

✓ ***CRC code bits:***

$$\mathbf{C} = [c_{N-1} \quad c_{N-2} \quad \dots \quad c_1 \quad c_0] = [m_{k-1} \quad m_{k-2} \quad \dots \quad m_1 \quad m_0 \quad r_{n-1} \quad r_{n-2} \quad \dots \quad r_1 \quad r_0]$$

$$\mathbf{C}(X) = c_{N-1}X^{N-1} + c_{N-2}X^{N-2} + \dots + c_1X + c_0 \qquad \text{degree (N - 1)}$$

$$= X^n\mathbf{m}(X) + \mathbf{R}(X)$$

CRC Codes Format (Cont'd)

Example:

($k=10$, $N=13$, $n=N-k=3$) CRC code

$$\mathbf{m} = [1010100101]$$

$$\mathbf{m}(X) = X^9 + X^7 + X^5 + X^2 + 1$$

$$\mathbf{R} = [111]$$

$$\mathbf{R}(X) = X^2 + X + 1$$

$$\mathbf{C} = [1010100101111]$$

$X^n m(X)$ is the polynomial corresponding to the message bit sequence to which a number n of 0's is appended. [1010100101000]

$$\mathbf{C}(X) = X^n \mathbf{m}(X) + \mathbf{R}(X)$$

$$= X^3(X^9 + X^7 + X^5 + X^2 + 1) + X^2 + X + 1$$

$$= X^{12} + X^{10} + X^8 + X^5 + X^3 + X^2 + X + 1$$

Obtaining R(x)

- How to obtain the polynomial R(X) (the appended bits)
  - CRC codes are designated by a *generator polynomial* g(X) with degree of *n*
  - Divide $X^n m(X)$ by g(X) (modulo-2 division) and obtain the remainder, which is R(X)

$$\mathbf{g} = \begin{bmatrix} g_n & g_{n-1} & \cdots & g_1 & g_0 \end{bmatrix}$$

$$\mathbf{g}(X) = g_n X^n + g_{n-1} X^{n-1} + \ldots + g_1 X + g_0 \qquad \text{degree (n)}$$

- $X^n m(X) = p(X)g(X)+R(X)$
  - *The remainder R(X) is always a polynomial of maximum order (n-1).*

Obtaining R(x) (Cont'd)

Example:

The polynomial R(X) (the appended bits)

Message $[11100110]$ 8 bits

$\mathbf{m}(X) = X^7 + X^6 + X^5 + X^2 + X$

Given N-k=n=4, generator polynomial $\mathbf{g}(X) = X^4 + X^3 + 1 \rightarrow [11001]$

$$\frac{X^n \mathbf{m}(X)}{\mathbf{g}(X)} = \frac{X^{11} + X^{10} + X^9 + X^6 + X^5}{X^4 + X^3 + 1}$$

$X^n m(X)$ is the polynomial corresponding to the message bit sequence to which a number n of 0's is appended. [111001100000]

The remainder $\mathbf{R}(X) = X^2 + X$,

therefore the appended bits are $[0110]$ (since n=4).

The CRC code bits are [111001100110]

Error Detection

- The polynomial for the received code word T(X) is divided by the generator polynomial g(X).

- **Upon the reception without error**
  - T(X)=C(X)=Xnm(X)+R(X)
  - The remainder of T(X)/g(X) =R(X)+R(X)=*the all-zero row*
  - (modulo-2 addition).
  - ***Example:***

$\mathbf{g}(X) = X^4 + X^3 + 1$

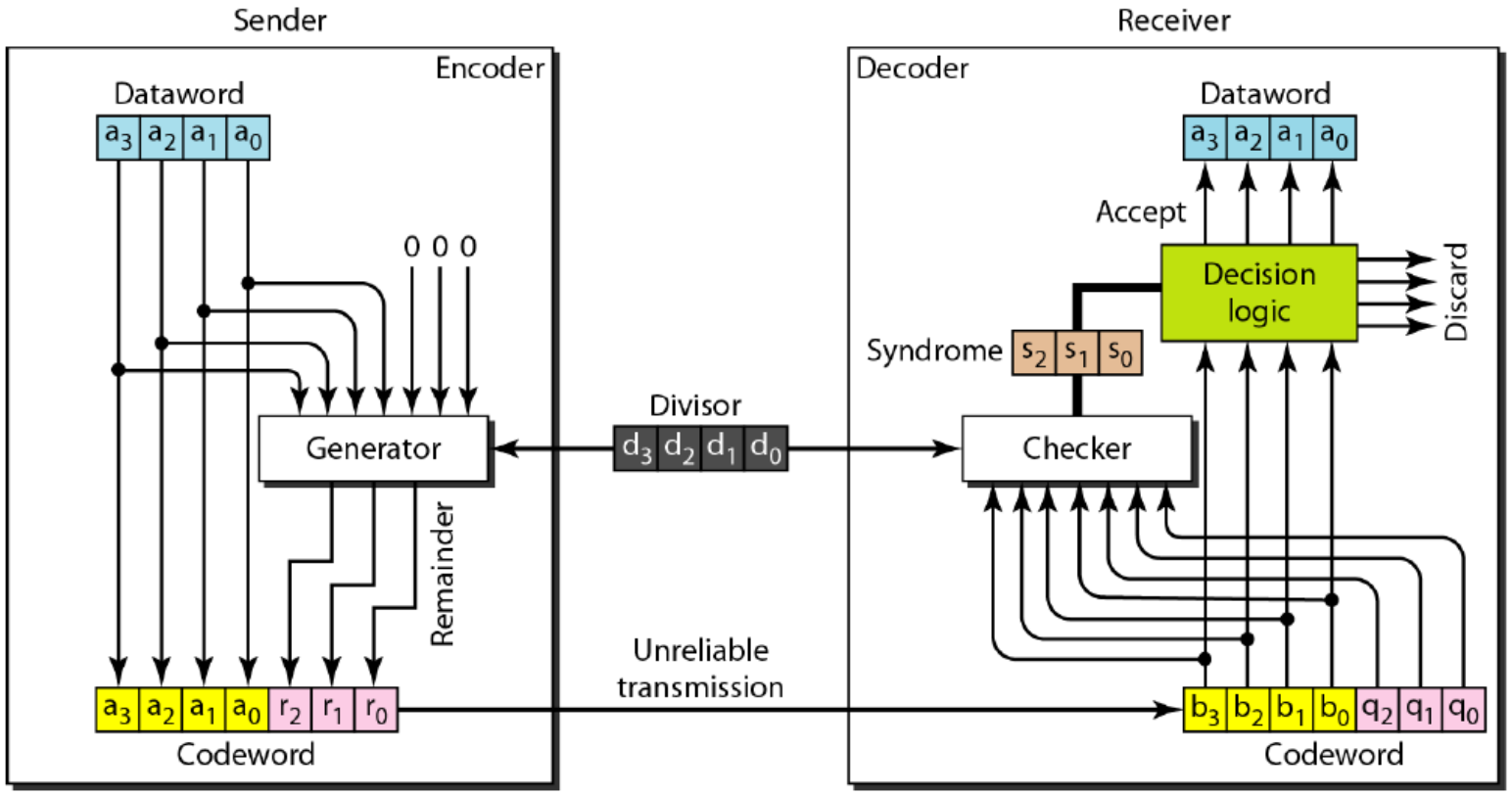
The transmitted CRC code bits are $[111001100110]$

$\mathbf{T(X)} = \mathbf{C}(X) = X^{11} + X^{10} + X^9 + X^6 + X^5 + X^2 + X$

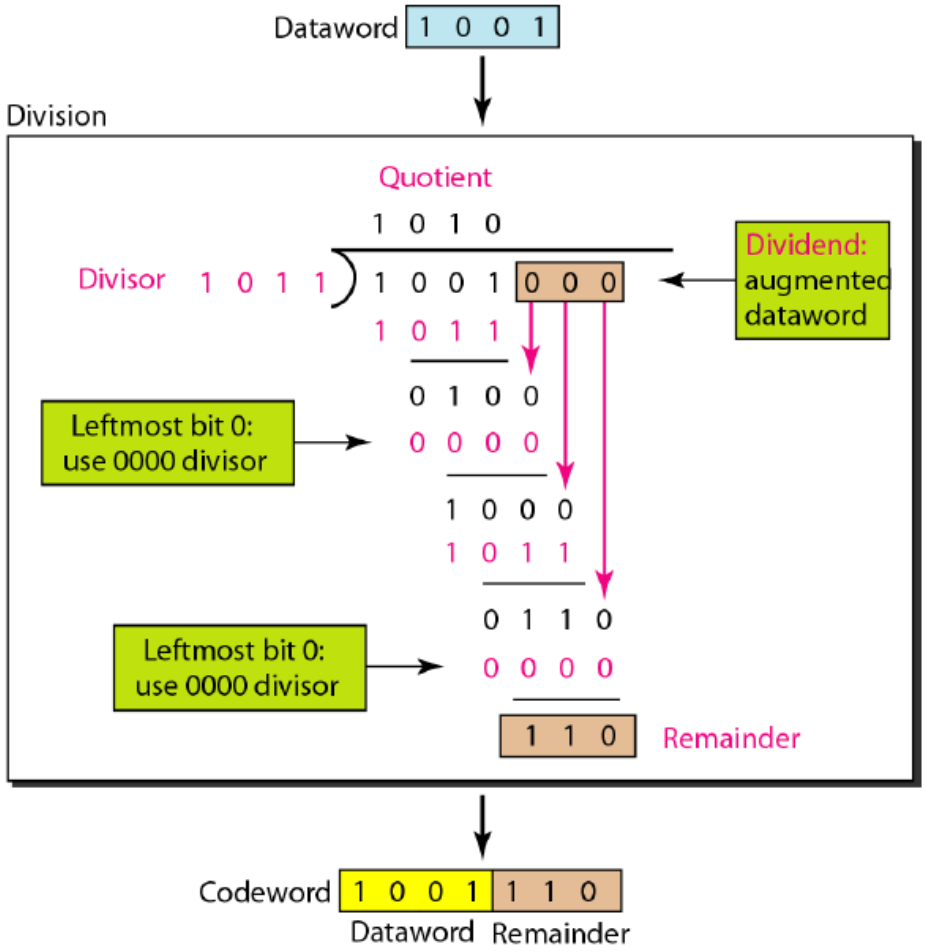
$= (X^7 + X^5 + X^4 + X^2 + X)\mathbf{g}(X)$

The remainder of $[\mathbf{C}(X)/\mathbf{g}(X)] = \mathbf{0} \longrightarrow [0000]$

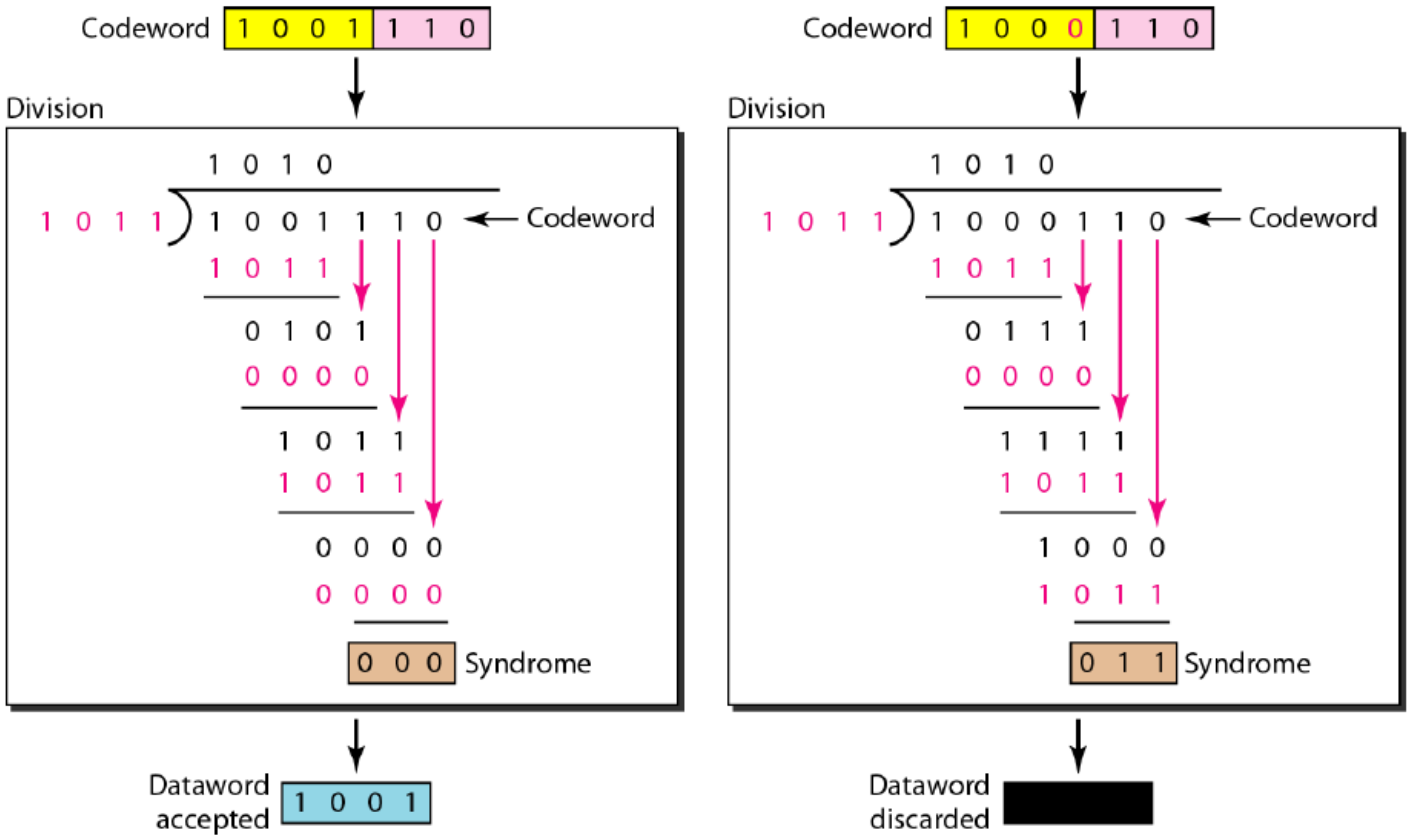
Encoder and Decoder Diagram



Sender Section

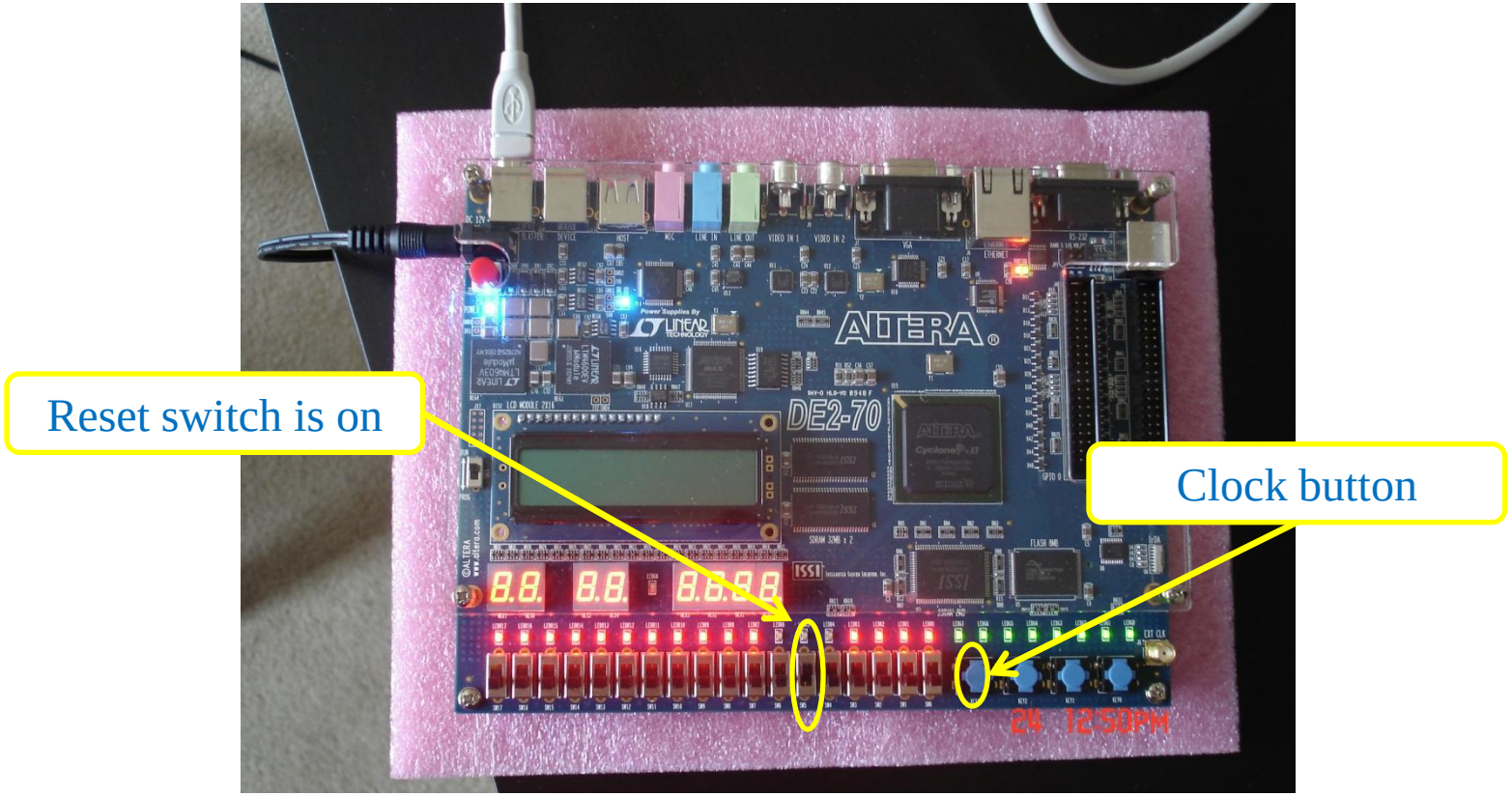


Receiver Section



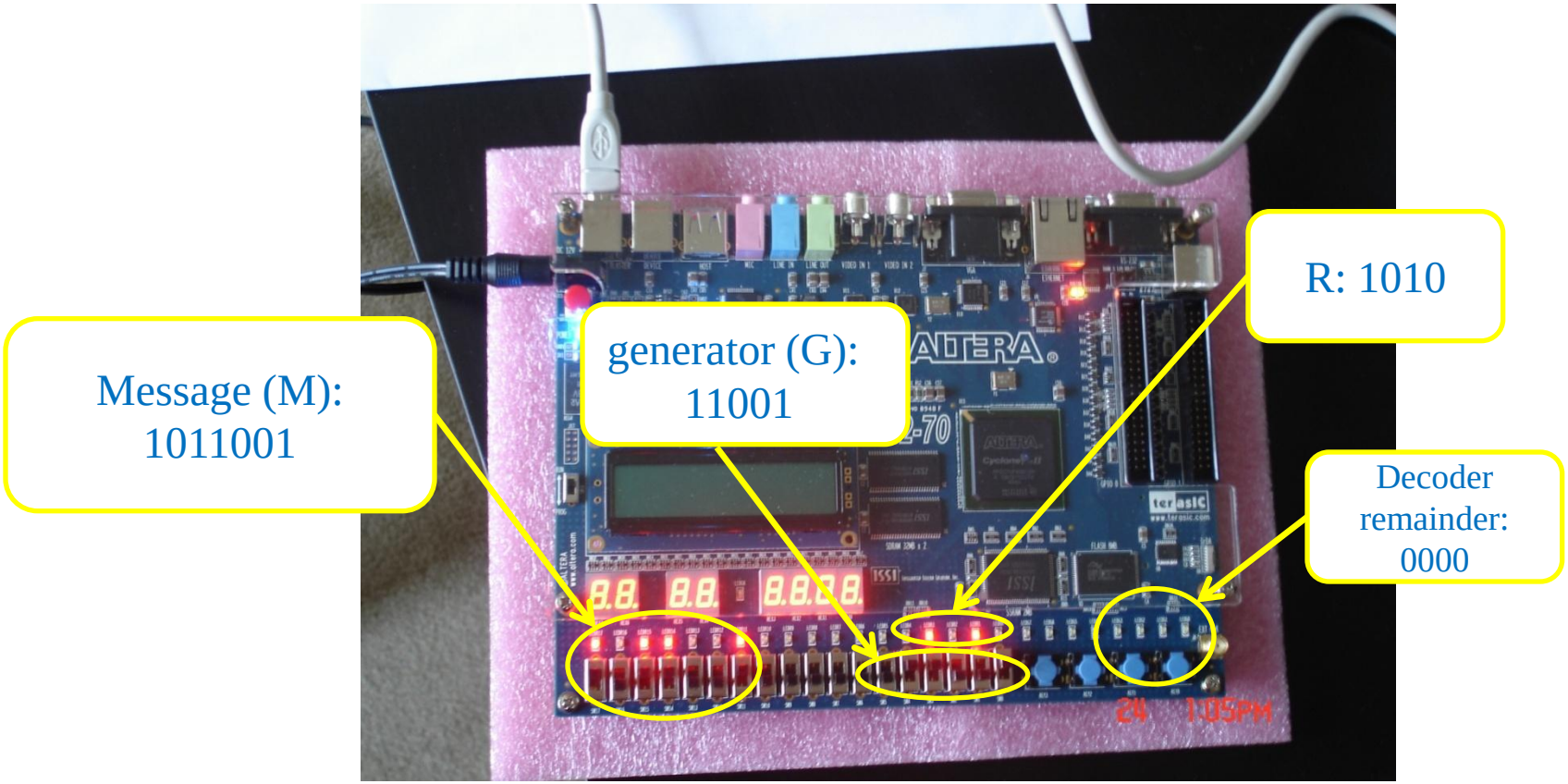
Run on FPGA

- **Initialization**
  - All lights are on when reset switch (SW5) is on.
  - When reset is off and the rising-edge clock starts the program with specific input switches.



Run on FPGA

- **Result:**
  - Input switch: message and generator.
  - Output green lights: all off represents deocding with no errors.





Thank you ！