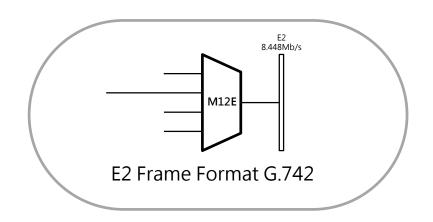


## 傳輸系統電路設計與模擬 - E12 Mapper IC 設計



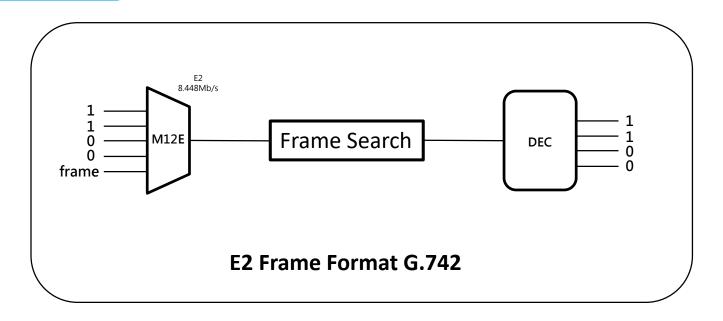
教授: 郭昭宗

電子三乙 110510216 蔡承宏 學生:

電子三甲 110510134 王維碩

2019/5/14

## 功能概要



Frame length 212x4 = 848 bits Bit rate = 8.448 MHz Bit interleaved by 4 E1

## 程式介紹 - 初值設定

```
module T1(clk,reset,data,count,sel,in,a,frame,inframe,Rdata,DEdata,DECdata);
input clk,reset;
//TX傳送端
                                //data: 引入輸入in用 sel: 選擇線用來控制輸入in值1100
output [3:0] data, sel;
output [9:0] count;
                                //count: 計數用,每個訊框長度848bits,要數 0~847
output [1:0]in;
                                //in: 用sel控制的1100一個一個的不斷輸入至data
reg [3:0] data,sel;
                                //data: [3:0] 為0~3 sel: [3:0] 為資料4個一組
reg [9:0] count;
                                //count: [9:0] 0~847 最少需要 2<sup>10</sup> = 1024 bits
reg [1:0]in;
                                //in: 輸入只有1和0只需[1:0]即可
//TX傳送端
//RX接收端
output [9:0] a;
                                //a: 解碼用
                                //frame: 訊框開頭用 inframe: 偵測3個訊框開始解碼
output frame, inframe;
reg [9:0] a;
reg frame, inframe;
output [3:0]Rdata,DEdata,DECdata; //Rdata:接收到的訊號 Dedata: 濾掉訊框部分 DECdata:濾掉Cxx、Sxx 部分
reg [3:0]Rdata, DEdata, DECdata;
//RX接收端
```

## 程式介紹 - 傳送端

```
//TX
always@(posedge clk)
begin
if (reset)
  begin
  count=0; sel=0;
  end
 else if (count = = 847)
  count = 0;
 else
  begin
   count=count+1;
   data[3] = data[2];
   data[2] = data[1];
   data[1] = data[0];
   data[0] = in;
  end
end
```

- 這裡利用正緣 clk 作為觸發
- 將選擇線還有計數器初值設為 0
- 讓計數器數 0~847 共 848 bits
- data[]用來一個一個的導入 in 的值

```
always@(posedge clk)
begin
 if (sel = 3)
 sel=0;
 else
 sel=sel+1;
end
always@(sel)
begin
 if (sel = 0)
 in=1;
 else if (sel = = 1)
 in=1;
 else if (sel = = 2)
 in=0;
 else if (sel = 3)
 in=0;
end
```

- sel 用來控制 in ,輸入有4組,所以數到 3
- sel 為 0 時,將 in 設為 1
- sel 為 1 時 · 將 in 設為 1
- sel 為 2 時,將 in 設為 0
- sel 為 3 時,將 in 設為 0
- 即可完成輸入data為 1100 的目的

```
always@(count)
begin
 if (count \leq 3)
 data = 4'b1111;
 else if (count >=4 && count <=7)
 data = 4'b0100;
 else if (count >=8 && count <=11)
 data = 4'b0000;
 else if (count > = 212 && count < = 215)
 data = 4'b1010:
 else if (count > = 424 && count < = 427)
 data = 4'b1010:
 else if (count >=636 && count <=639)
 data = 4'b1010;
 else if (count >=640 && count <=643)
 data = 4'b0000;
end
//TX
```

- 使用 count 控制 Frame開頭用來辨識
- Frame 開頭要求為 111101000000
- 分別對 0~11 bits 共12 bits 做每4個一組 的組合填上該填的數字 1010
- 212~215·424~427·636~639 為Cxx
- 640~643 為 Sxx 0000

### 程式介紹 - 接收端

```
//RX
always@(clk or data)
begin
if (reset)
 begin
  a=0; frame=0;
end
 else if (count < 13 && data = = 4'b1111 || count < 13 && data = = 4'b0100 || count < 13 && data = = 4'b0000)
frame=1:
 else
frame=0;
end
always@(frame or clk)
begin
 if(a = 50)
a=a;
else if (frame==1)
a=a+1:
else
a=a;
end
```

- Frame在這裡代表訊框開頭
- 偵測到訊框開頭則跳起來, Frame = 1
- a 用來控制in frame 3個Frame跳起來進 行解碼動作

```
always@(a)
begin
if (reset)
inframe=0;
else if (a==50)
inframe=1;
else
inframe=0;
end
always@(data or clk or inframe)
begin
if (reset)
Rdata=0;
 else if (inframe==1)
Rdata=data;
end
```

- 偵測frame跳3次,來啟動inframe
- Rdata 為接收到的所有資料
- inframe 為1時,開始進行接收

```
always@(Rdata)
begin
 if (reset)
 DEdata=0;
 else if (data==4'b1111 || data==4'b0100 || data==4'b0000 || data==4'b1010)
 DEdata=4'bxxxx;
else
DEdata=Rdata;
End
always@(DEdata)
begin
if(reset)
 begin w=0; x=0; y=0; z=0; end
 else if(inframe==1 && count==12)
 begin
  w = 1; x = 1; y = 0; z = 0;
 end
end
```

● DEdata 為將接收到的Rdata進行去除訊框 開頭以及Cxx、Sxx動作,回復為完整Data

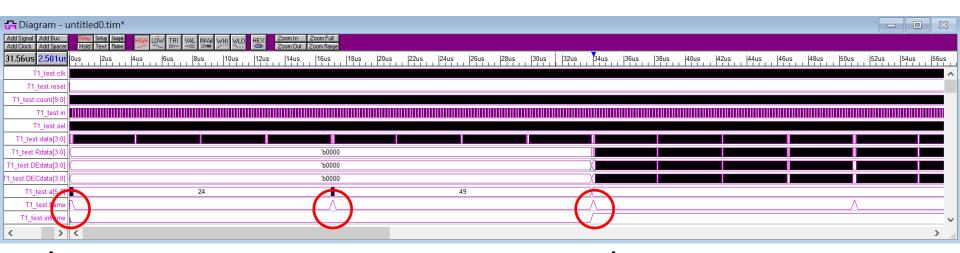
● 輸出線分別輸出data 序進資料轉為列出

### 測試檔

```
module T1 test;
reg clk,reset;
wire [9:0] count;
wire [1:0]in;
wire [3:0] sel,data,Rdata,DEdata,DECdata;
wire [9:0] a;
wire frame, inframe;
T1 d1(clk,reset,data,count,sel,in,a,frame,inframe,Rdata,DEdata,DECdata);
initial
  begin
   clk = 1'b1;
   reset = 1'b1;
   #10 reset = 1'b0;
 end
 always #10 \text{ clk} = \sim \text{clk};
initial
 #100000 $finish;
endmodule
```

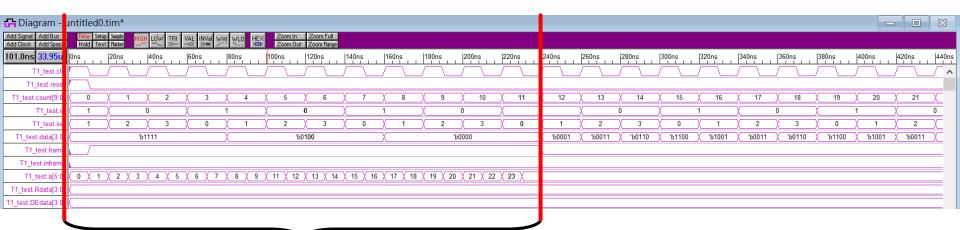
本次執行的檔案已以往的測試檔方式太龐大跑不了且過於不便,所以利用選擇線方式進行傳輸,主要程式皆在主體檔

# 完整視圖



找尋Frame,直到找到第3個Frame為止,開始解碼

## 細部視圖



Frame前面部分,共12bits,用來辨別848 bits開頭從哪開始,使用了848 bits的0~11Bits。

訊框內容以每4個為一組 1111 0100 0000

4.434us <mark>29.62</mark> us	4.08us	4.10us	4.12us	4.14us	4.16us	4.18us	4.20us	4.22us	.24us	4.26us	4.28us	4.30us	4.32us	4.34us	4.36us	4.38us	4.40us	4.42us	4.44us	4.46us	4.48us	4.50us	4.52us
T1_test.clk	T1_test.clk												$\overline{}$						$\overline{}$	$\overline{}$			<u>_</u> ^
T1_test.reset	t																						
T1_test.count[9:0]	204	205	206	207	208	209	210	(211	212	213	214	215	216	( 217	218	( 219	220	221	222	223	224	225	$\mathbf{x}$
T1_test.in	1	χ	0	Χ	1	χ	0	χ1		Х	0	X	1		0	Х	1	Χ	0	Х	1	Χ	0
T1_test.se	1	2	3	( 0	( 1	2	3	( 0	1	2	( 3	( 0	( 1	2	) 3	( 0	χ 1	2	3	χ ο	)( 1	2	$\mathbf{x}$
T1_test.data[3:0]	'b1001	( 'b00 <b>1</b> 1	'b0110	/ 'Ь1100	'ь1001	( 'b0011	/ 'b0110	( 'b1100			b1010		( 'b0101	( 'b1011	/ 'b0110	( 'b1100	/ 'b1001	/ 'b0011	'ь0110	( 'b1100	/ 'b1001	/ 'b0011	$\mathbf{x}$

#### C<sub>11</sub> C<sub>21</sub> C<sub>31</sub> C<sub>41</sub> 占用 212~215 bit · 填入1010

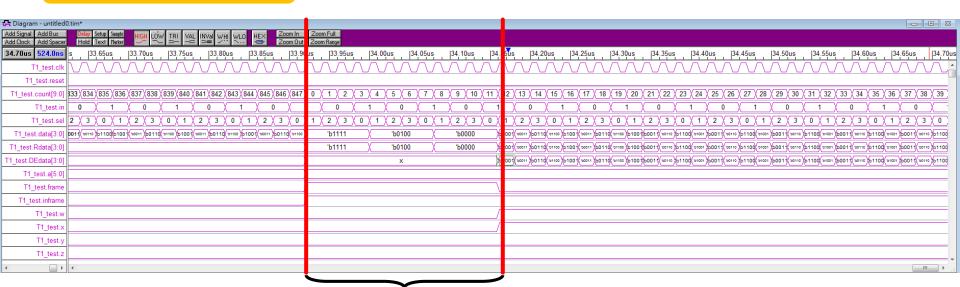
8.393us <mark>25.66us</mark>	.  8	3.32us	8.34us	8.36us	8.38us	8.40us	8.42us	8.44us	8.46us	8.48us	8.50us	8.52us	8.54us	.56us	8.58us	8.60us	8.62us	8.64us	8.66us	8.68us	8.70us	8.72us	8.74us  8
T1_test.clk									$\overline{}$		$\overline{}$				$\overline{}$								
T1_test.reset																							
T1_test.count[9:0]	5 <u>)</u> (	416	417	418	419	420	421	422	423	X 424	425	426	427	428	429	430	431	432	433	434	435	436	437
T1_test.in	1		( (	)	(	1	( 0	1	X	1	X	0		1	χ	0	Χ	1	χ	0	(1		
T1_test.sel	X	1	2	3	0	( 1	2	3	( 0	χ 1	2	3	( 0	1	2	3	0	1	2	3	( 0	1	2
T1_test.data[3:0]	00 )(	'ь1001	'b0011	'b0110	'b1100	'b1001	(b0011)	'ь0110	'b1100	χ		Ь1010		'ь0101	/b1011	'b0110	'b1100	'b1001	'ь0011	'b0110	'b1100	'ь1001	'b0011
									•	•				•									

#### C<sub>12</sub> C<sub>22</sub> C<sub>32</sub> C<sub>42</sub> 占用 424~427 bit · 填入1010

12.84us <mark>21.22us</mark>	8us	12.60us	12.62us	12.64us	12.66us	12.68us	12.70us	12.72us	12.74us	12.76us	12.78us	12.80us	12.82us	12.84us	12.86us	12.88us	12.90us	12.92us	12.94us	12.96us	12.98us	13.00us	13.02us
T1_test.clk	,k																						
T1_test.reset	st.reset																						
T1_test.count[9:0]	629	630	631	632	633	634	635	636	637	638	639	(640	641	642	643	644	(645	646	647	648	649	(650	65
T1_test.in		0	Χ	1 )		)		1	X	0	Х	1	Х	0	XX		X	0	Χ	1	χ	0	
T1_test.sel	2	3	( 0	( 1 )	2	3	( 0	1	2	) 3	( 0	χ 1	2	) 3	)( 0	1	2	( 3	χ ο	χ 1	2	) 3	O
T1_test.data[3:0]	0011	'ь0110	'b1100	(b1001)	'ь0011	'b0110	'ь1100	'b1010				χ	'b	0000		'b0001	/b0011	Ъ0110	/b1100	(b1001	) 'b0011	( 'b0110	<u>'b11</u>
																•							

C<sub>13</sub> C<sub>23</sub> C<sub>33</sub> C<sub>43</sub> 占用 636~639 bit,填入1010 同步用位元 S<sub>1</sub> S<sub>2</sub> S<sub>3</sub> S<sub>4</sub> 占用 640~643 bit,填入0000

# 接收解碼視圖



找到第3個Frame時,開始接收解碼

Rdata: 接收訊號(未經捨去訊框開頭辨識字元)

DEdata:接收訊號(已經捨去訊框開頭辨識字元、Cxx、Sxx)



#### DECdata: 接收訊號(已經捨去訊框開頭變識字元) Cxx 、Sxx

