國 立 臺 北 科 技 大 學

NATIONAL TAIPEI UNIVERSITY OF TECHNOLOGY

計 算 機 組 織 報 告

課程名稱: 計算機組織

單元名稱: Chapter.7 UART

班級: 電機二丙

姓名及學號: 黃莅翔 111310313
許桓瑜 111310324
林琮曜 111310340
楊祐全 111310330

中 華 民 國 113 年 6 月 4 日

7.1 介紹 Introduction

通用**非同步收發器（UART）**是一個**通過串行線傳送平行數據**的電路，其中包含了**發射器(Transmitter)和接收器(Receiver)**。

①發射器：本質上是個特殊的移位暫存器，以並行的方式加載數據，並按照特定速率逐位移出。
②接收器：逐位移入數據，然後重新組合數據。

UART 通常與 EIA（電子工業聯盟）RS-232 標準一起使用，該標準指定了兩個數據通信設備的功能和程序特性。由於 RS-232 中定義的電壓級別與 FPGA 1/0 不同，因此在序列埠和 FPGA 的 1/0 引腳之間需要一個電壓轉換器芯片。S3 板上有一個帶有標準九針連接器的 RS-232 端口且包含了必要的電壓轉換器芯片，將各種 RS-232 的控制信號配置為自動生成 PC 序列埠的確認信號。可以使用標準的直通串行電纜來連接 S3 板和 PC 的序列埠。

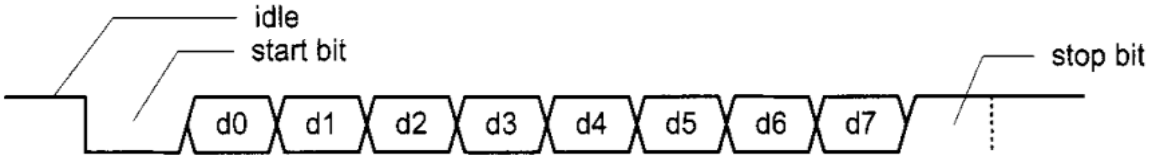


圖 7.1 位元傳送

在空閒時，串行線設為'1'。傳輸從起始位開始，**起始位(start bit)**設為'0'，**停止位(stop bit)**設為'1'，停止位可以是 1、1.5 或 2 位元，**數據位(data bit)**可以是 6、7 或 8 位元，**奇偶校驗位(可選) (optional parity bit)**用於錯誤檢測。當數據位中有奇數個 1 時奇數校驗被設置為'0'，當數據位中有偶數個 1 時偶數校驗被設置為'0'。使用 8 個數據位、無奇偶校驗和 1 個停止位的傳輸如圖 7.1 所示，先被傳輸的是最低有效位(LSB)。串行線不傳遞時鐘信息。在傳輸開始之前，發送器和接收器必須事先達成一組參數協議，包括波特率（即每秒鐘的位數）、數據位和停止位的數量，以及奇偶校驗位的使用情況。

7.2 UART 接收器子系統 Receiving Subsystem

由於從傳送的信號中沒有傳遞時脈訊號，接收器只能使用預定參數來檢索數據位。我們使用一個**過取樣(oversampling)**方案來估計傳輸位的中點，然後在相對應的時間點檢索它們。

7.2.1 過取樣程序

最常用的取樣率是波特率的 16 倍，意思是每個串行位元被取樣 16 次。假設通信使用 N 個數據位和 M 個停止位。過取樣方案的工作方式如下:

1. 等待收到的信號變為 '0'，即開始位開始，然後開始取樣計數器。
2. 當計數器數到 7 時，收到的信號達到開始位的中點後，將計數器清零並重新啟動。
3. 當計數器數到 15 時，收到的信號向前進一位，並到達第一個數據位的中點。檢索其值，將其移入暫存器中後，重新啟動計數器。
4. 重複步驟 3，執行 N-1 次，以檢索其餘的 data bits。
5. 如果使用了可選的奇偶校驗位，則重複步驟 3 一次以獲取奇偶校驗位。
6. 重複步驟 3，M 次，以獲取停止位。

過取樣方案基本上就是執行時鐘信號的功能，**利用取樣時刻來估計每個位元的中點**。雖然接收器對開始位的準確開始時間沒有信息，但估計最多可能偏差 1/16。隨後的數據位檢索也最多可能偏離中點。由於過取樣，波特率只能是系統時鐘率的一小部分，因此執行此方案時不適用於高數據率。

UART 接收子系統的概念圖如圖 7.2 所示。它由三個主要組件組成：

1. UART 接收器：通過過取樣獲取數據字的電路

2. 波特率發生器：生成取樣時刻的電路

3. 接口電路：在 UART 接收器和使用 UART 的系統之間提供緩沖和狀態的電路

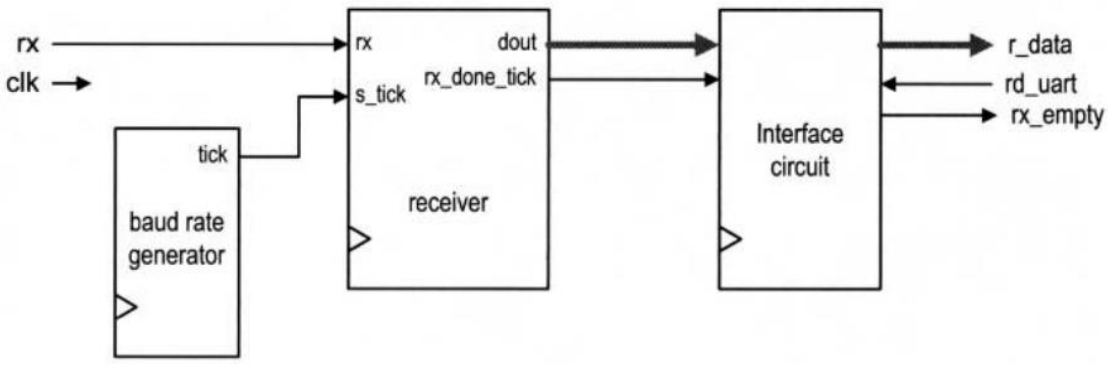
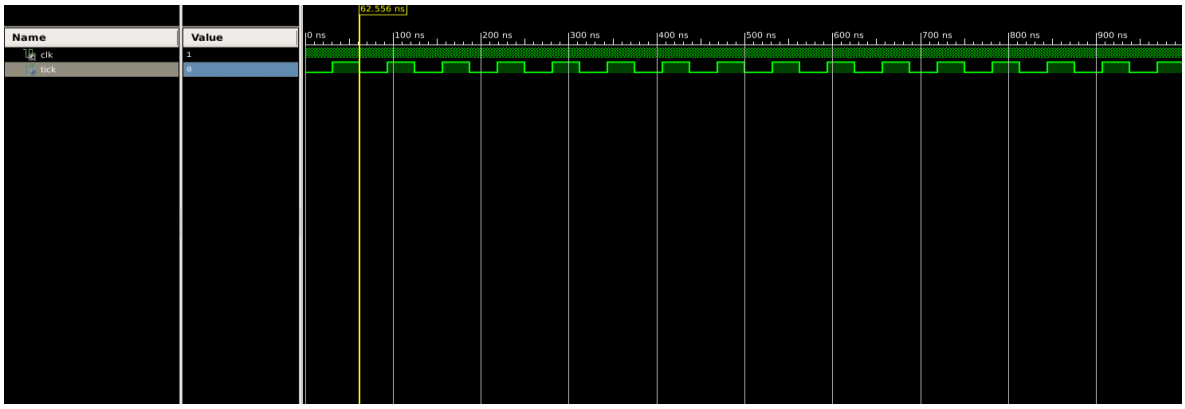


圖 7.2 UART 子系統概念圖

7.2.2 波特率產生器

波特率產生器生成一個取樣信號，其頻率恰好是 UART 指定的波特率的 16 倍。為了避免創建新的時脈且違反同步設計原則，取樣信號應該作為 enable ticks，而不是作為 clk 提供給 UART 接收器。

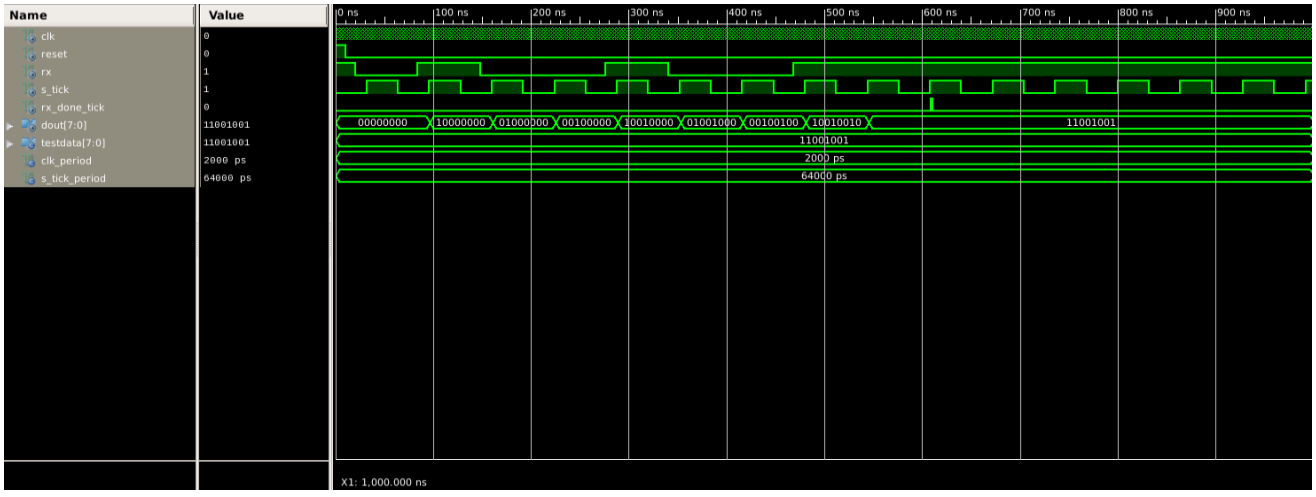
對於 19,200 波特率，取樣率必須是每秒 307,200 個（即 19,200*16）。由於系統時鐘率為 50 MHz，波特率發生器需要一個 mod-163 計數器（即$\frac{50*10^6}{307200}$），其中一個 one-clock-cycle tick 在每 163 個時脈周期中宣告一次。



程式 7.2.2 波特率產生器

7.2.3 UART 接收器

在理解了過取樣程序之後，可以相應地推導出 ASMD 圖如圖 7.3 所示。D-BIT 表示數據位的位元數，SB-TICK 表示停止位所需的時鐘周期數。該圖包括三個主要狀態，即**起始、數據和停止**狀態。s-tick 信號是來自波特率發生器的可使用時刻，一個位元間隔中有 16 個時脈周期，除非 s-tick 信號被宣告，否則 FSMD 都會保持在相同的狀態。有兩個計數器，由 s 和 n 暫存器表示，s 暫存器持續跟蹤 sample tick 的數量，在起始狀態中計到 7，在數據狀態中計到 15，在停止狀態中計到 SB-TICK。n 暫存器追踪在數據狀態中接收到的數據位數量。檢索的位被移入並重新組裝在 b 暫存器中。還包括一個狀態信號 rx_done_tick，在接收過程完成後宣告一個時鐘周期。



程式 7.2.3 UART 接收器 testbench

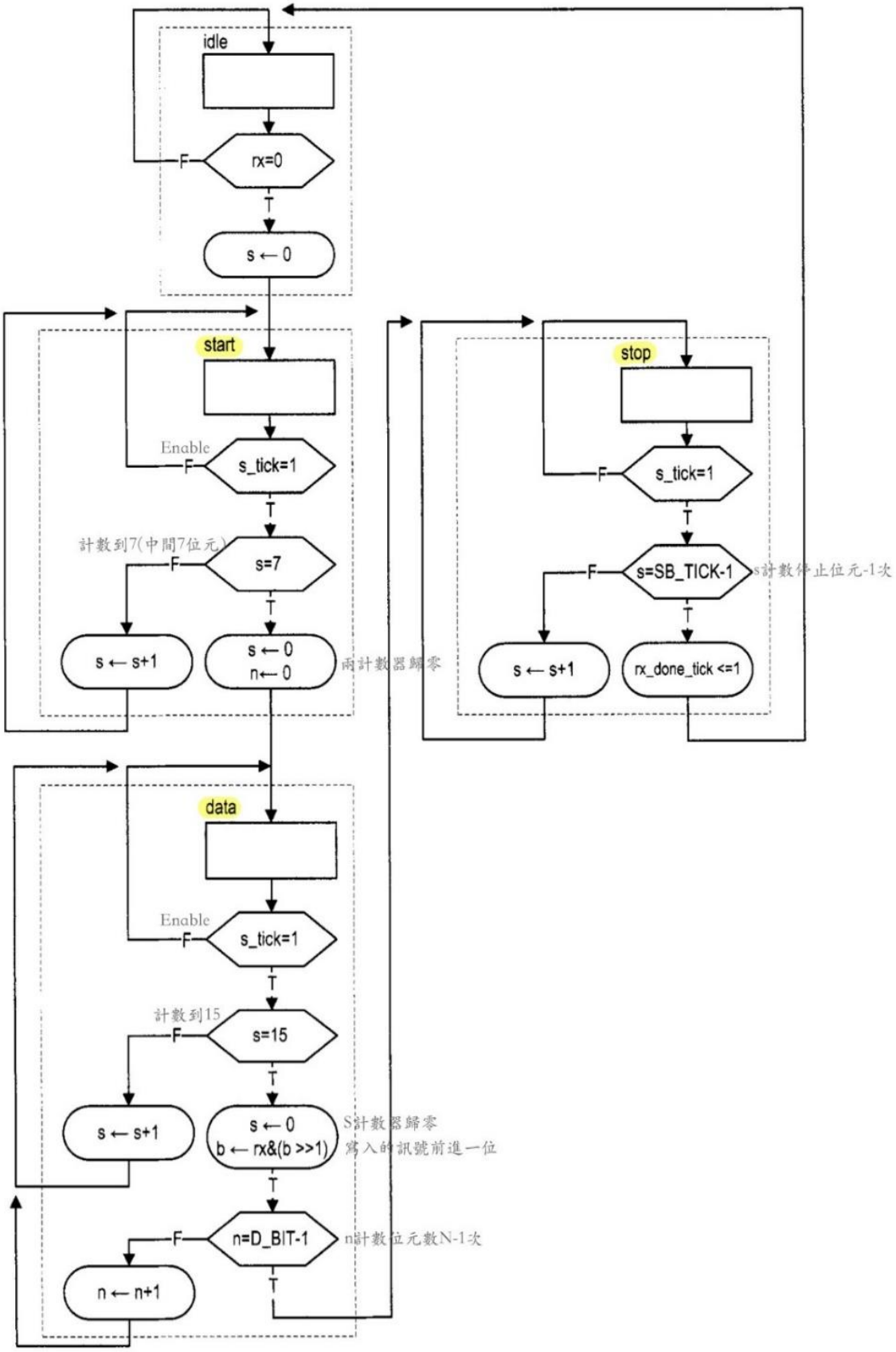


圖 7.3 UART 接收器流程

7.2.4 介面電路

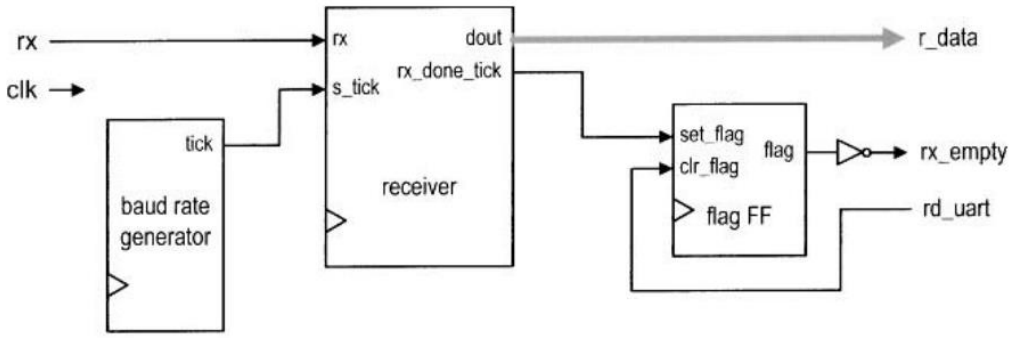
在一個大型系統中，UART 通常是用於串列數據傳輸的外部電路。主系統定期檢查其狀態並處理接收到的字。接收器的介面電路有兩個功能

①提供一個機制來表示 new word 是否可用，並防止接收到的字被多次檢索。

②在接收器和主系統之間提供緩衝空間。有三種常用的方案：

- ●一個旗標 FF
- ●一個旗標 FF 和一個 one-word 緩衝器
- ●一個 FIFO 緩衝器

UART 接收器在接收到數據字後一個時鐘周期後宣告 rx_ready_tick 信號。第一個方案使用一個觸發器 FF 來跟踪新數據字是否可用。觸發器 FF 有兩個輸入信號。一個是 set-flag，它將標誌位 FF 設置為 '1'，另一個是 clr_flag，它將旗號 FF 清零為 '0'。rx_ready_tick 信號連接到 set-flag 信號，當一個新的數據字到達時設置標誌。主系統檢查標誌 FF 的輸出以查看是否有新的數據字可用。它在檢索字後的一個時鐘周期內宣告 clr_flag 信號，如圖 7.4(a)所示。為了與其他方案保持一致，將標誌 FF 的輸出反轉以生成最終的 rx_empty 信號，這個信號表示沒有新字可用。在這個方案中，主系統直接從 UART 接收器的移位暫存器中檢索數據字，並**不提供任何額外的緩衝空間**。如果遠程系統在啟動了一個新的傳輸（即標誌 FF 仍然被宣告）**則舊數據字將被覆蓋**，這種錯誤稱為數據溢出。



7.4 (a) 旗標 FF

為了提供一些緩衝，可以添加一個 one-word 緩衝器，如圖 7.4(b)所示。當 rx_ready_tick 信號被宣告時，接收到的字被載到緩衝器中，並且設置旗標 FF。只要主系統在新字到達之前檢索字，就不會發生數據溢位。

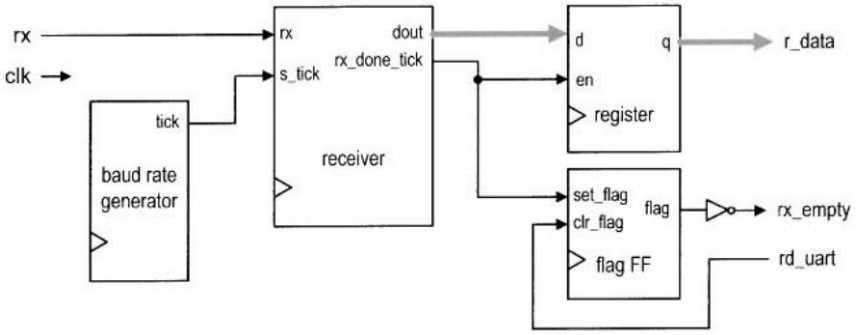


圖 7.4(b) 一個旗標 FF 和一個 one-word 緩衝器

第三種方案使用了 FIFO 緩衝器，可提供更多的緩衝空間，且進一步降低了數據溢位的機會。我們可以調整 FIFO 中所需的 word，以滿足主系統的處理需求如圖 7.4(c)所示。

rx_ready_tick 信號連接到 FIFO 的 wr 信號。當接收到新的數據字時，wr 信號在一個時鐘周期內被宣告，並將相應的數據寫入 FIFO 中。主系統從 FIFO 的讀取端獲取數據，在檢索到一個字後，在一個時脈周期內宣告 FIFO 的 rd 信號以刪除相應的項目。FIFO 的 empty 信號用來指示是否有任何接收到的數據字可用。當接收到新的 data 且 FIFO 已滿時，將發生數據溢位錯誤。

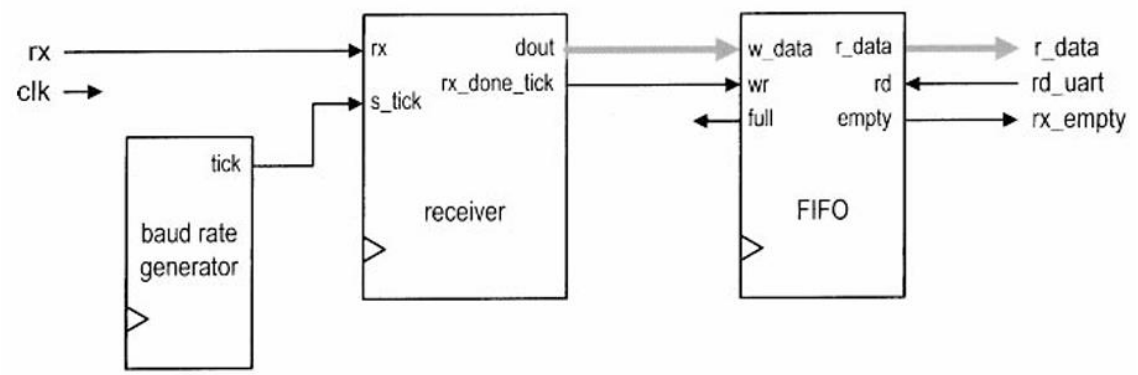
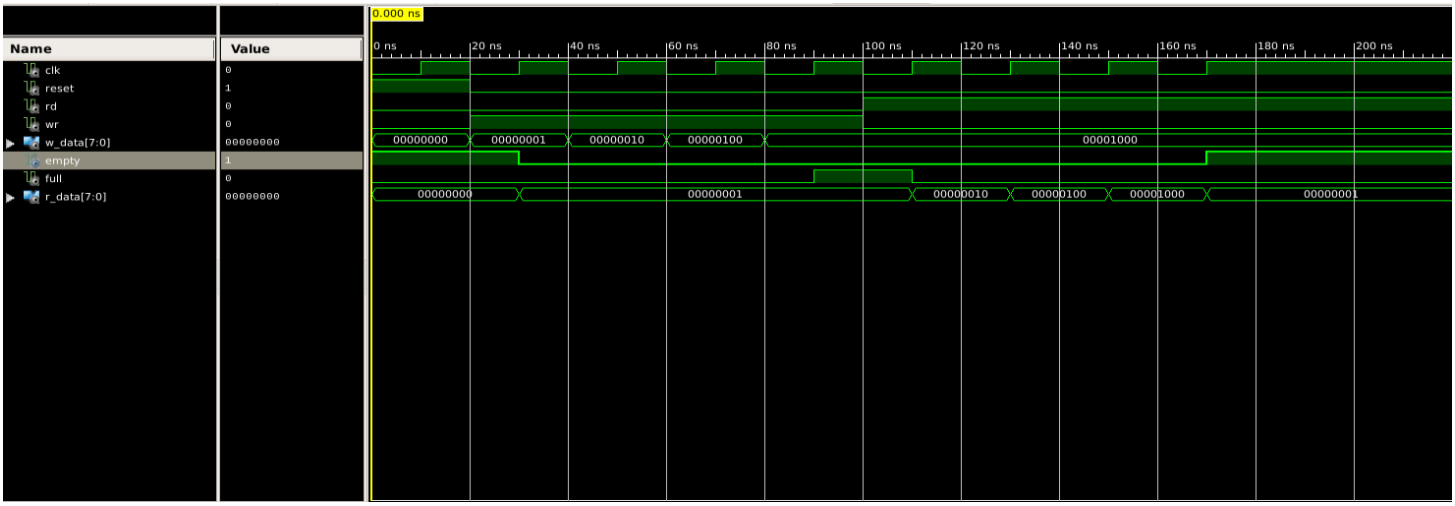


圖 7.4(c) 添加一個 FIFO 緩衝器



程式 7.2.4 FIFO 緩衝器 testbench

7.3 UART 發送器器子系統 Transmitting Subsystem

UART 發送子系統的組織與接收子系統類似，由 UART 發送器、鮑率產生器和接口電路組成。接口電路與接收子系統的類似，只是主系統設置標誌 FF 或寫入 FIFO 緩衝器，而 UART 發送器清除標誌 FF 或讀取 FIFO 緩衝器。

UART 發送器本質上是一個位移暫存器，以特定速率位移出數據位，該速率可以由鮑率產生器生成的 one_clock_cycle 啟用信號控制。由於不涉及過取樣，啟用信號的頻率比 UART 接收器慢 16 倍，UART 發送器通常與 UART 接收器共享鮑率產生器，並使用內部計數器追蹤啟用信號的數量，而不是引入新的計數器，每 16 個啟用信號，一位被移出。

UART 發送器的 ASMD 圖與 UART 接收器的類似。在 tx_start 信號宣告後，FSMD 加載 data word，然後逐漸進行起始、數據和停止狀態，以移出相應的位。它通過在一個時鐘周期內宣告 tx_done_tick 信號來表示完成。一位元緩衝器 tx_reg 用於濾除任何潛在的故障。

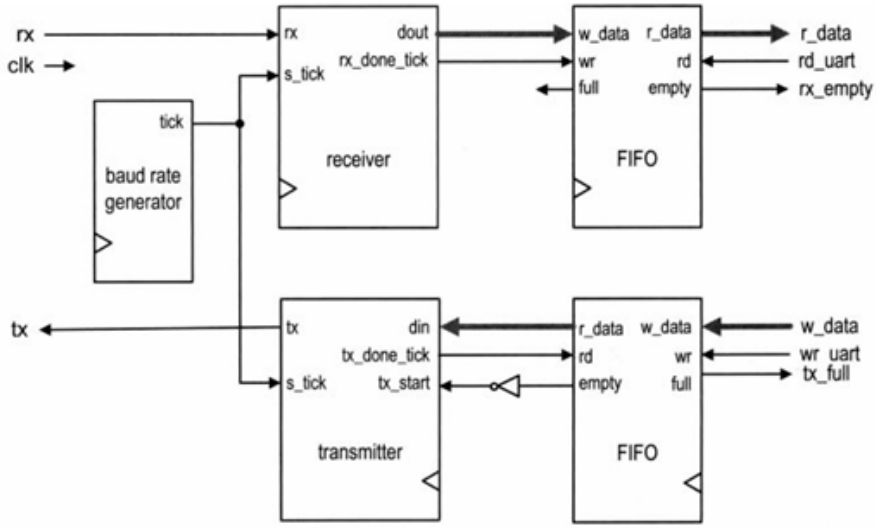
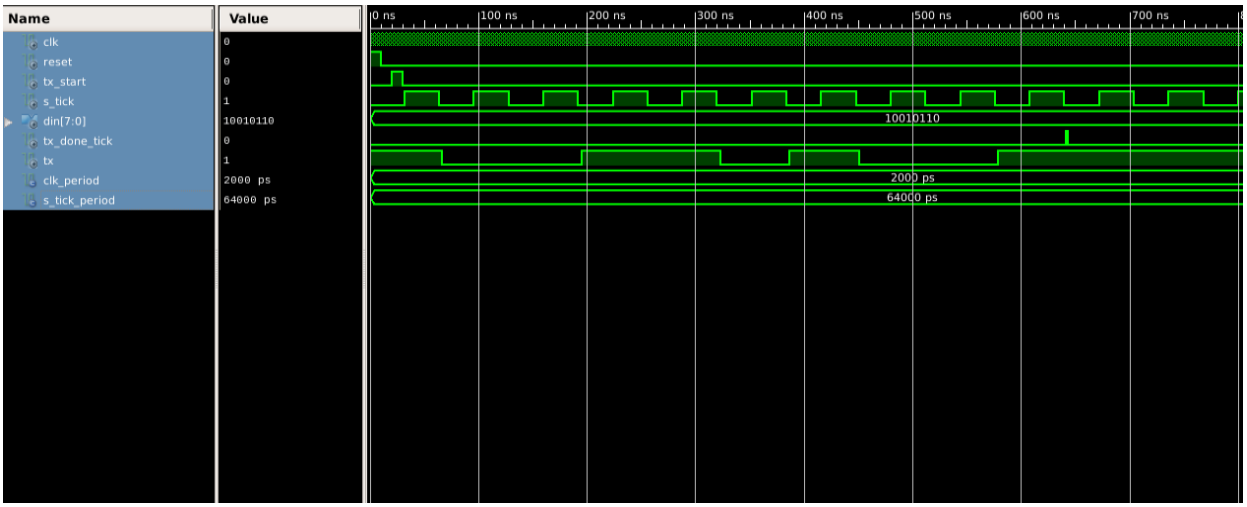


圖 7.5 UART 完整方塊圖



程式 7.3 UART 發送器 testbench

7.4 UART 驗證配置 verification configuration

驗證電路

我們使用一個迴授電路來驗證 UART 的操作。在電路中，S3 板的串口連接到 PC 的序列埠。當我們從 PC 發送一個字時，接收到的 data word 被存儲在 UART 接收器的四個字的 FIFO 緩衝器中。當通過 r-data 端口檢索時，data word 加 1，然後發送回發射器（通過 w_data 端口）。防彈跳的按鈕開關在按下時產生一個 one-clock-cycle 信號，並且連接到 rd_uart 和 wr_uart 信號。當產生時鐘周期時，它從接收器的 FIFO 中移除一個字，並將增加的字寫入發射器的 FIFO 進行傳輸。UART 的 r-data 端口連接到 S3 板的八個 LED，其 tx_full 和 rx_empty 信號連接到七段顯示器最右邊的兩個水平橫槓。

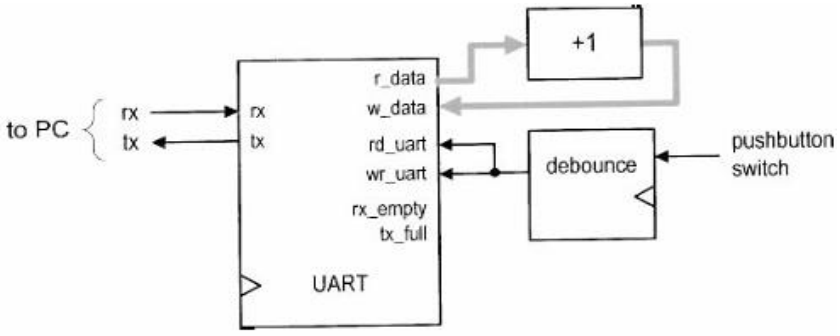
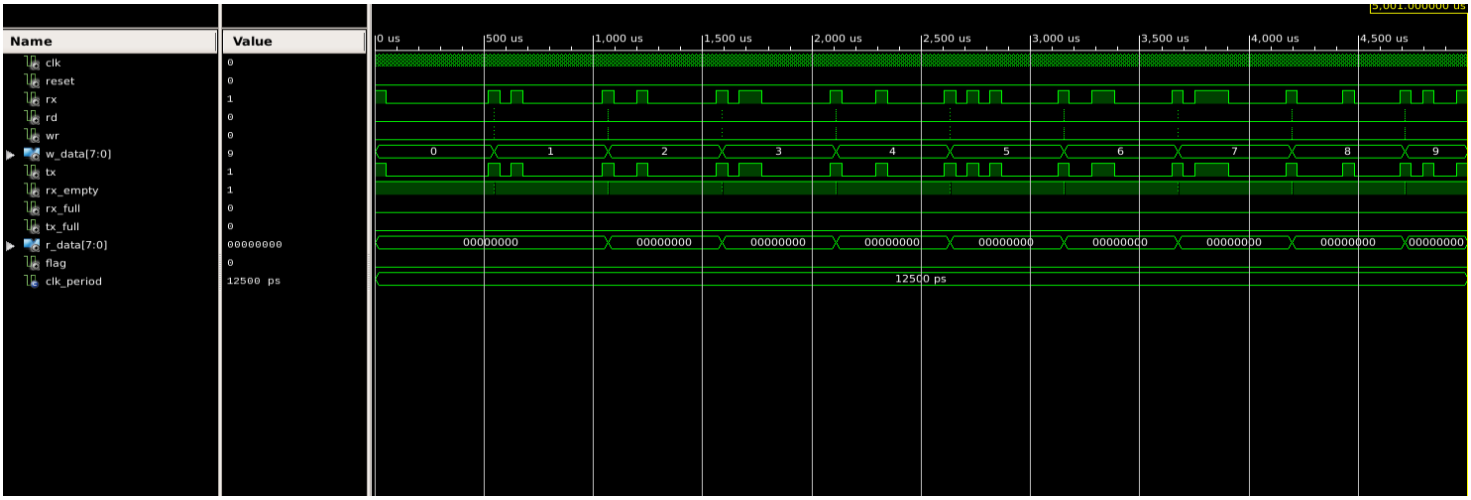


圖 7.6 UART 驗證電路方塊圖



程式 7.4.1 UART 驗證 testbench

HyperTerminal of Windows

在個人電腦端，Windows 的 HyperTerminal 程序可以用作虛擬終端，與 S3 板進行交互。為了與我們定制的 UART 兼容，它必須配置為 19,200 鮑率，8 個 data bit，1 個 stop bit，並且不使用奇偶校驗。

完成 HyperTerminal 程序的設置，與 S3 板進行傳輸。我們可以輸入一些按鍵並觀察 S3 板上的 LED。接收到的 data bit 存儲在 FIFO 緩衝器中，且只顯示第一個接收到的 data bit，在按下按鈕後，第一個 data bit 將從 FIFO 中刪除，增加的字將回到個人電腦的序列埠並顯示在 HyperTerminal window，而 FIFO 緩衝器的 full 和 empty 狀態可以透過連續接收和傳輸超過四個數據字來進行測試。

ASCII 碼

在 HyperTerminal 中，字符以 ASCII 碼發送，是由 7 位元組成，包含了 128 個代碼字(常規字母、數字、標點符號和控制字符) ，由於 ASCII 碼僅使用 7 位元，但 data bit 通常由 8 位元組成。PC 使用 extended set 最高位是 1，characters 是特殊的圖形符號。characters 和其代碼字以十六進制格式，不可列印字會用括號括起來顯示，當接收到多個不可列印字時，可能會引入特殊操作：

- (nul)：空字元，即全部為零的模式
- (bel)：生成響鈴聲
- (bs)：退格
- (ht)：水平制表符
- (nl)：換行
- (vt)：垂直制表符
- (np)：新頁
- (cr)：回車
- (esc)：逃逸
- (sp)：空格
- (del)：刪除

由於我們許多實驗項目中使用 PC 的序列埠與 S3 板進行傳輸，以下觀察有助於我們操作和處理 ASCII 碼：

- 當代碼字中的第一個十六進制數字為 0x0 或 0x1 時，對應的字符是控制字符。
- 當代碼字中的第一個十六進制數字為 0x2 或 0x3 時，對應的字符是數字或標點符號。
- 當代碼字中的第一個十六進制數字為 0x4 或 0x5 時，對應的字符通常是大寫字母。
- 當代碼字中的第一個十六進制數字為 0x6 或 0x7 時，對應的字符通常是小寫字母。
- 如果代碼字中的第一個十六進制數字為 0x3，則較低的十六進制數字表示相應的十進制數字。
- 大寫和小寫字母在一位上不同，可以通過添加或減去 0x20 或反轉第六位來互相轉換。

表 7.1 ASCII code

Code	Char	Code	Char	Code	Char	Code	Char
00	(nul)	20	(sp)	40	@	60	‘
01	(soh)	21	!	41	A	61	a
02	(stx)	22	"	42	B	62	b
03	(etx)	23	#	43	C	63	c
04	(eot)	24	$	44	D	64	d
05	(enq)	25	%	45	E	65	e
06	(ack)	26	&	46	F	66	f
07	(bel)	27	’	47	G	67	g
08	(bs)	28	(	48	H	68	h
09	(ht)	29	)	49	I	69	i
0a	(nl)	2a	*	4a	J	6a	j
0b	(vt)	2b	+	4b	K	6b	k
0c	(np)	2c	,	4c	L	6c	l
0d	(cr)	2d	-	4d	M	6d	m
0e	(so)	2e	.	4e	N	6e	n
0f	(si)	2f	/	4f	O	6f	o
10	(dle)	30	0	50	P	70	p
11	(dc1)	31	1	51	Q	71	q
12	(dc2)	32	2	52	R	72	r
13	(dc3)	33	3	53	S	73	s
14	(dc4)	34	4	54	T	74	t
15	(nak)	35	5	55	U	75	u
16	(syn)	36	6	56	V	76	v
17	(etb)	37	7	57	W	77	w
18	(can)	38	8	58	X	78	x
19	(em)	39	9	59	Y	79	y
1a	(sub)	3a	:	5a	Z	7a	z
1b	(esc)	3b	;	5b	[	7b	{
1c	(fs)	3c	<	5c	\	7c	¦
1d	(gs)	3d	=	5d	]	7d	}
1e	(rs)	3e	>	5e	^	7e	~
1f	(us)	3f	?	5f	_	7f	(del)

7.5 UART 客製化 customizing

在前面的章節中討論的 UART 是針對特定配置進行定制的，代碼可以輕鬆修改以適應其他所需功能

- 鮑率：鮑率由鮑率產生器的頻率控制，可通過修改 mod-m 計數器的 M 通用型（在代碼中表示為 DVSR 常量）來改變頻率。
- data bit 位元數：可以通過修改 n-reg 暫存器的上限（在代碼中指定為 DBIT 常量）來改變數據位數。
- 奇偶校驗位：可以在 ASMD 圖的數據狀態和停止狀態之間引入一個新狀態來包含奇偶校驗位。
- 停止位數：可以通過修改 ASMD 圖中停止狀態的 s-reg 暫存器的上限來改變停止位數，使用 SB-TICK 常量，可以是 1、1.5 或 2 個停止位。
- 錯誤檢查：在 UART 接收子系統中可以檢測到三種類型的錯誤

  -奇偶校驗錯誤。如果包含了奇偶校驗位，接收器可以檢查接收到的奇偶校驗位的正確性。

  -幀率錯誤：接收器可以檢查在停止狀態接收到的值如果該值不是 '1'，則發生幀率錯誤。

  -緩衝區溢出錯誤：當主系統未能及時檢索接收到的字時，就會發生這種情況。UART 接收器可以在接收字準備存儲時（即當生成 rx_done_tick 信號時）檢查緩衝區的 flag_reg 信號或 FIFO 的 full 信號，如果 flag-reg 或 full 信號仍然被宣告斷言，則發生數據溢出。

7.6 參考文獻說明 BIBLIOGRAPHIC NOTES

雖然 RS-232 standard 非常古老，但它仍提供了一種簡單可靠的低速通信鏈接，用於連接兩個設備，維基百科網站上有一篇關於此主題的很好的概述文章以及一些有用的鏈接。Jan Axelson 所著的《Serial Port Complete》提供了有關如何將硬件設備連接到 PC 的序列埠的信息。

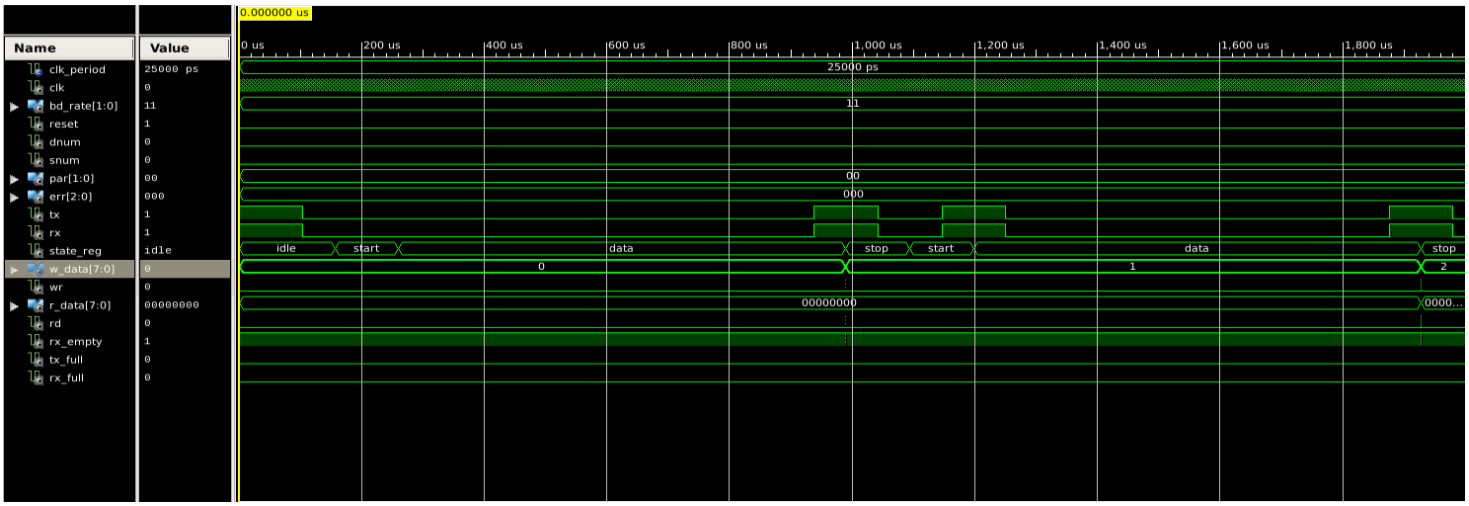
7.7 實驗建議 SUGGESTED EXPERIMENTS

7.7.1 全功能 UART

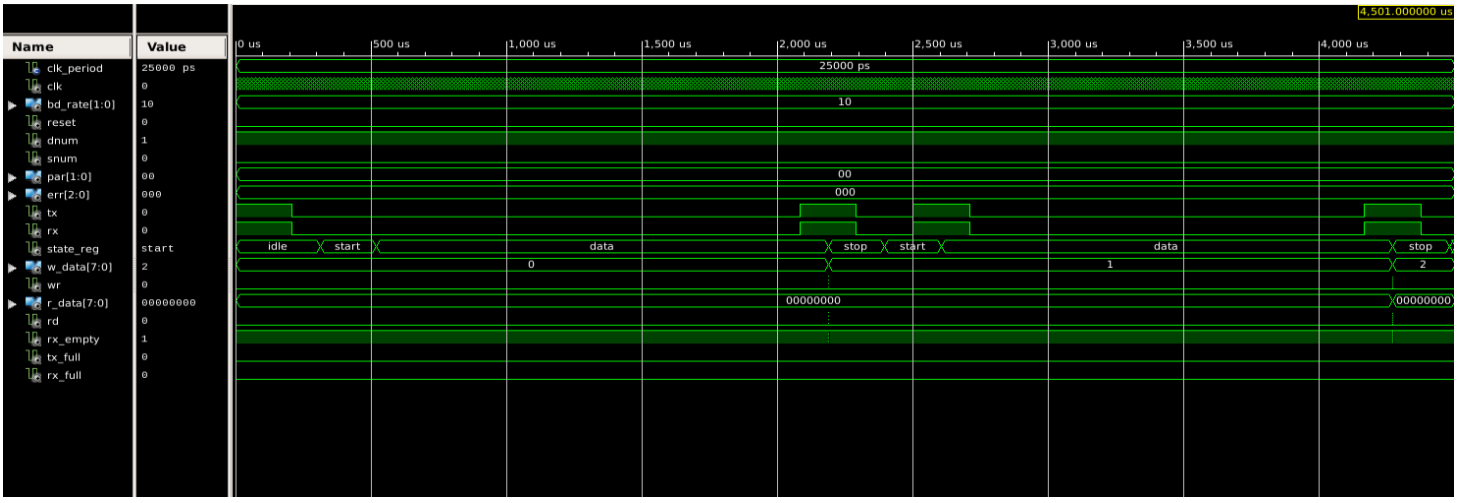
另一種定制UART的替代方案是將所有功能都包含在設計中，並根據需求配置UART。一個全功能 UART 使用額外的輸入信號來指定鮑率、奇偶校驗位類型以及 data bits 和停止位的數量，還包括一個錯誤信號。除了列表 7.4 中的 uart-top 設計的 I/O 信號，還需要以下信號：

- bd-rate：2 位輸入信號，指定鮑率，可以是 1200、2400、4800 或 9600 波特
- dnum：1 位元的輸入信號，指定 data bit 位元數，可以是 7 或 8 位元
- snum：1 位元的輸入信號，指定停止位的位元數，可以是 1 或 2 位元
- par：2 位元輸入信號，指定所需的奇偶校驗方案，可選擇無校驗、偶或奇校驗
- err：3 位元的輸出信號，表示存在奇偶校驗錯誤、幀率錯誤和數據溢出錯誤

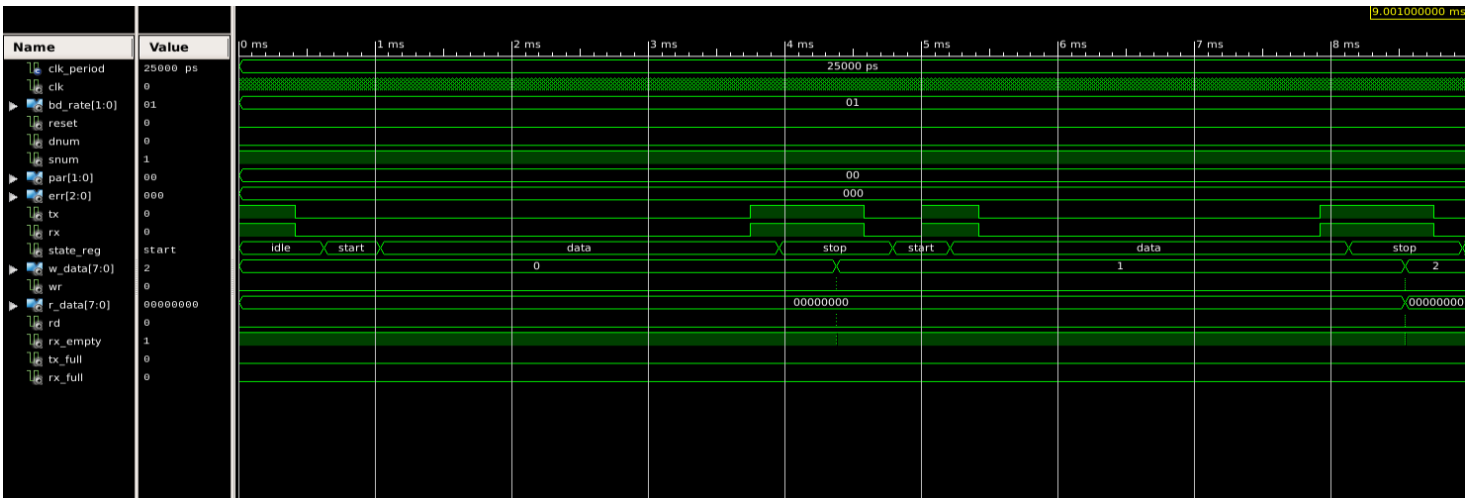
●設計結果



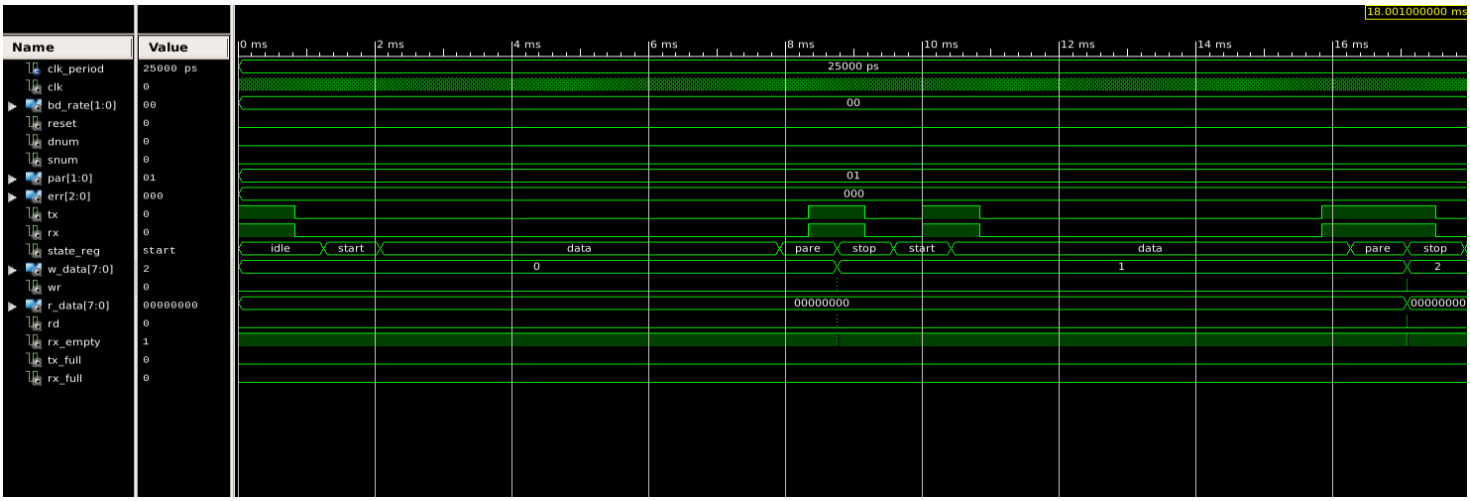
程式 7.7.1(a) (b_rate = 11 , dnum = 0 ,snum =0 ,par = 00)



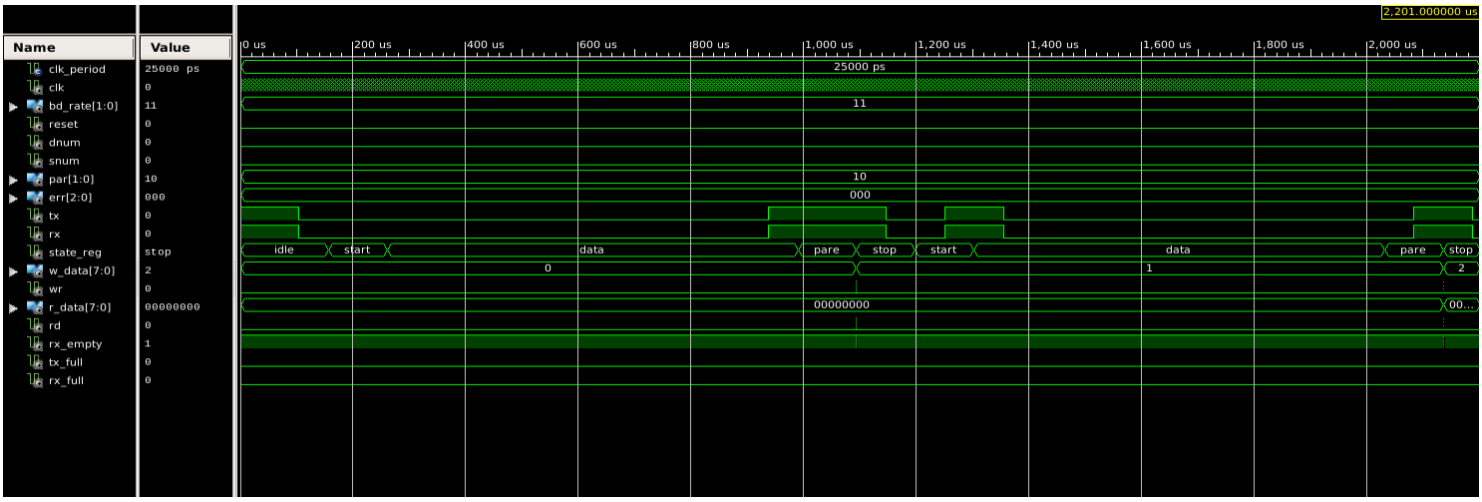
程式 7.7.1(b) (b_rate = 10 , dnum = 1 ,snum =0 ,par = 00)



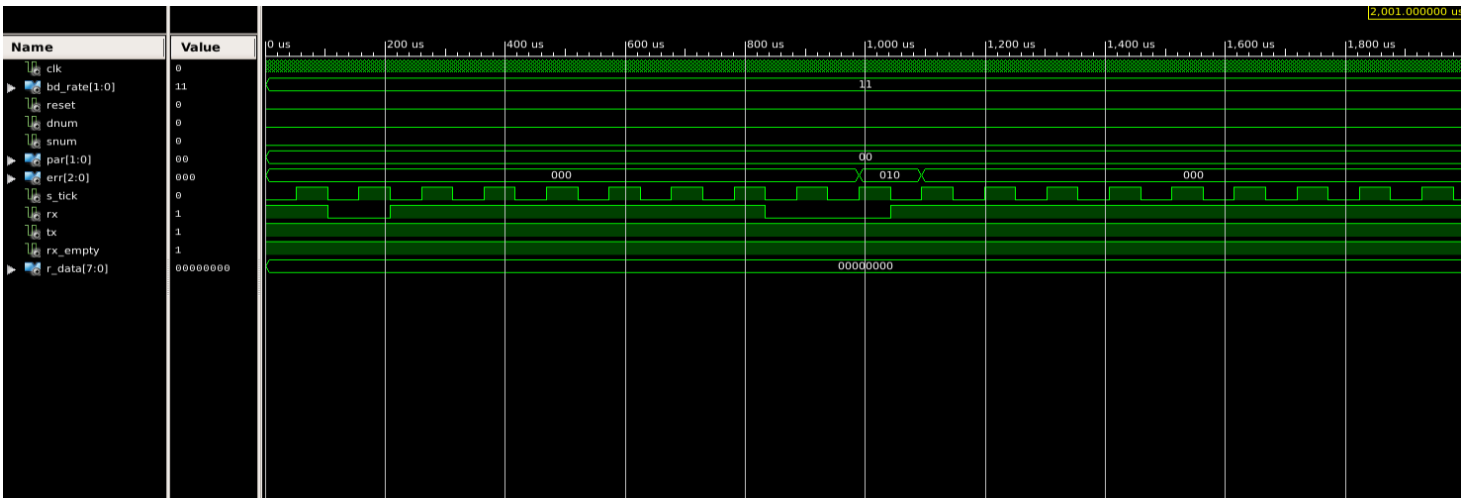
程式 7.7.1(c) (b_rate = 01 , dnum = 0 ,snum =1 ,par = 00)



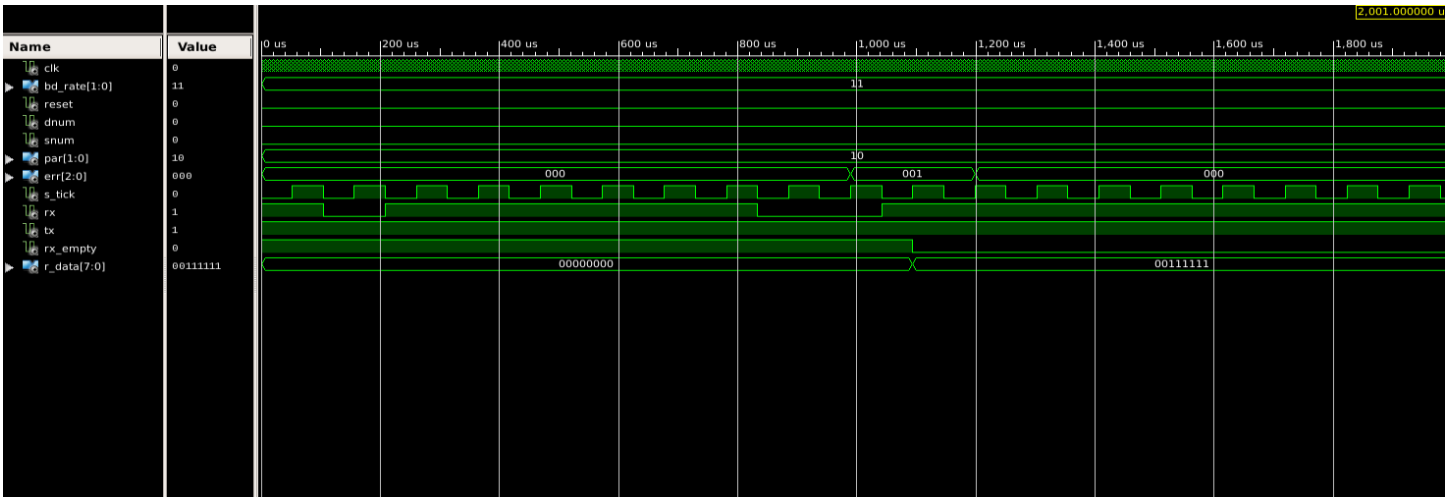
程式 7.7.1(d) (b_rate = 00, dnum = 0 ,snum =0 ,par = 01)



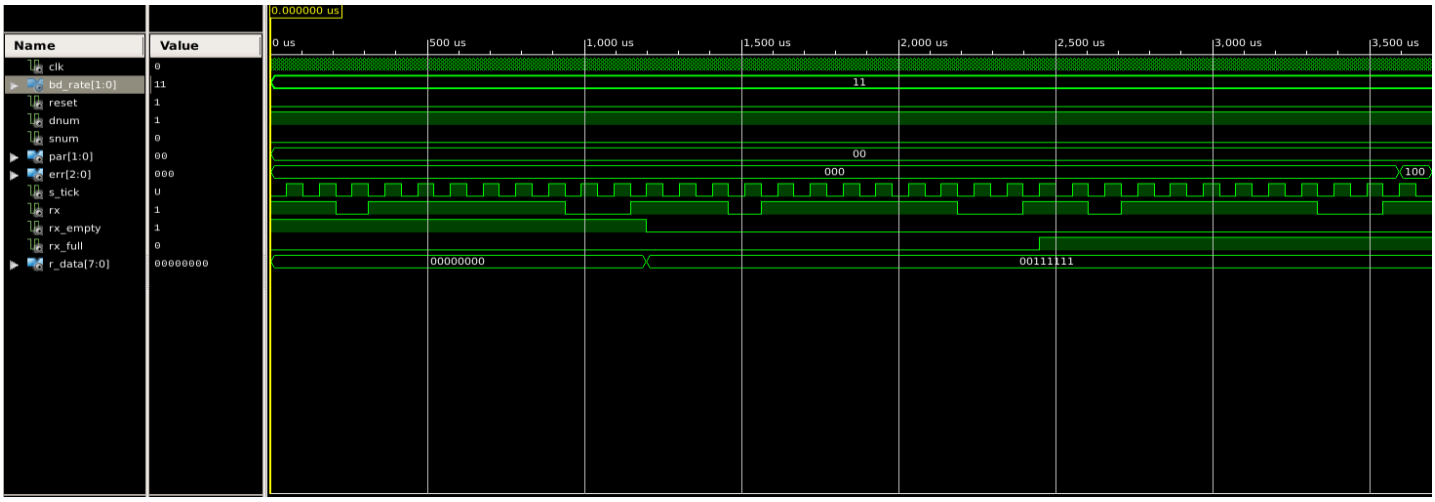
程式 7.7.1(e) (b_rate = 11, dnum = 0 ,snum =0 ,par = 10)



程式 7.7.1(f) 幀率錯誤



程式 7.7.1(g)奇偶校驗錯誤



程式 7.7.1(h)數據溢出錯誤

7.7.2 自動鮑率檢測

串接中最常用的 data bit 是八位，當使用常規 ASCII 碼進行通信時，僅使用七個最低有效位，最高有效位為'0'。如果 UART 配置為 8 個數據位、1 個停止位和無校驗位，則接收到的字將是形式為 0-dddd-ddd0-1，其中 d 是數據位，可以是'0'或'1'。假設第一個字和後續傳輸之間有足夠的時間，我們可通過測量第一個'0'和最後一個'0'之間的時間間隔來確定波特率，推導出一個帶有自動鮑率檢測電路的 UART。傳輸系統先發送一個 ASCII 碼進行速率檢測，並恢復正常操作，接收子系統使用第一個字來確定鮑率，並使用該鮑率作為剩餘傳輸的鮑率產稱器。

假設 UART 配置為 8 個 data bits、1 個停止位和無校驗位，鮑率可以是 4800、9600 或 19200 波特。修訂後的 UART 接收器應有兩個操作模式，最初處於「檢測模式」並等待第一個字。接收到該字並確定鮑率後接收器進入「正常模式」，UART 以常規方式運行，推導步驟如下：

1.繪製自動波特率檢測電路的 ASMD 圖。

2.根據 ASMD 圖推導 VHDL 代碼，使用 S3 板上的三個 LED 指示輸入信號的鮑率。

3.修改 UART 包括三種不同的波特率：4800、9600 和 19200，可以透過使用暫存器作為鮑率生成器的除數並根據所需鮑率來達成。

4.創建 top-level FSMD 追蹤、控制和協調波特率檢測電路和常規 UART 接收器的操作，使用 S3 板上的按鈕開關強制 UART 進入檢測模式。

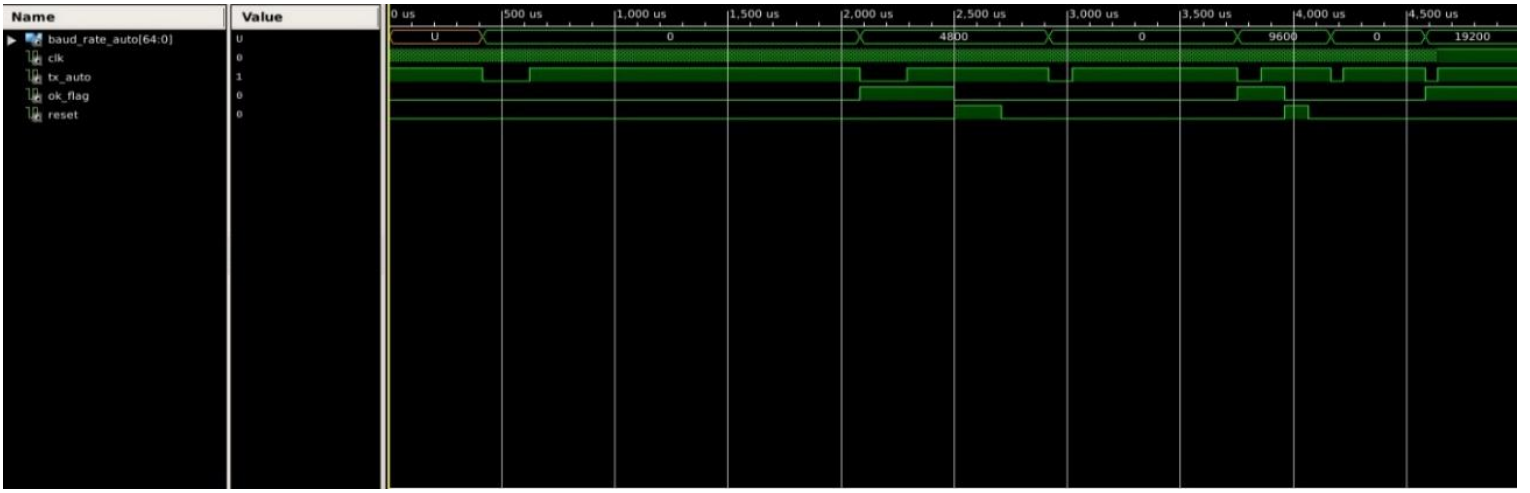
5.修改 top-level UART 代碼和驗證電路，綜合驗證電路。

6.在 HyperTerminal 中創建不同配置，並驗證 UART 的操作。

●設計結果

加入 I/O 信號:

- BAUD_RATE_auto：64 位輸出信號，輸出檢測完成的鮑率，可以是 4800/9600/19200
- reset：1 位元的輸入信號，切換至檢測模式。
- ok_flag：1 位元的輸出信號，表示檢測結束，切回正常模式。



程式 7.7.2 鮑率自動檢測(4800,9200,19200)

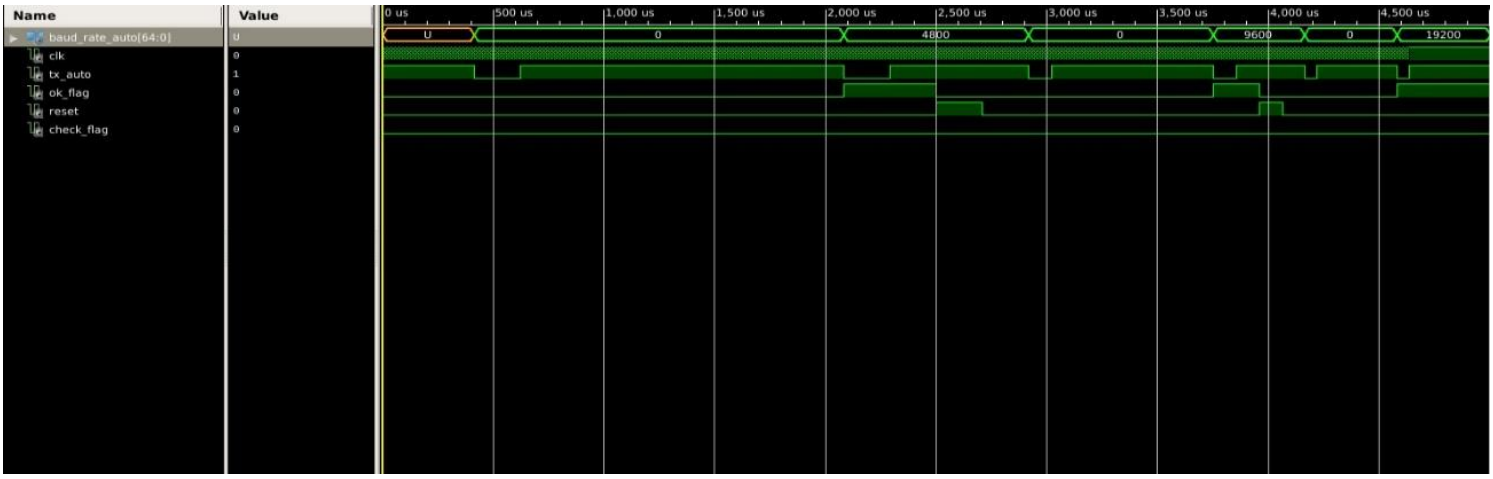
7.7.3 自動鮑率和校驗檢測電路

除了鮑率，我們還假設需要自動確定校驗方案，可以是無校驗、偶或奇校驗。擴展之前的自動波特率檢測電路以檢測校驗配置並重複實驗 7.7.2，以下為推導步驟：擴展 ASMD 圖：修改先前的 ASMD 圖以包括校驗檢測功能。在檢測模式中，除波特率外，還要檢測校驗類型。

●設計結果

加入 I/O 信號:

- chick_flag：1 位元的輸出信號表示有無校驗位

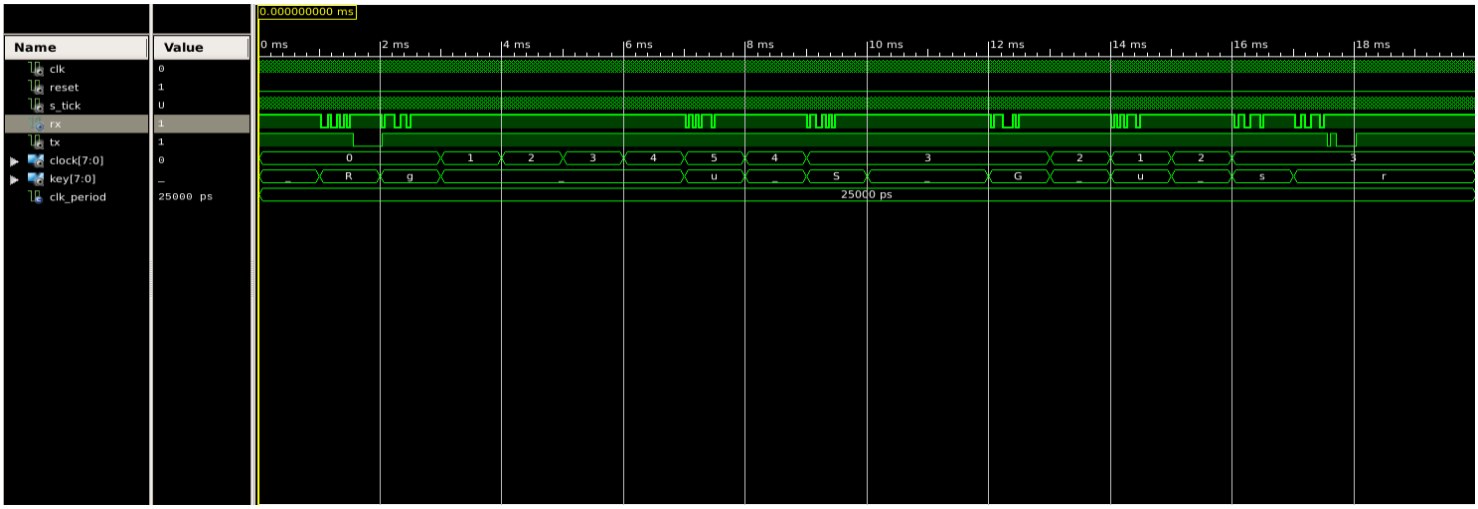


程式 7.7.3 鮑率自動檢測以及校驗檢測(4800 無校驗,9200 有校驗,19200)

7.7.4 UART 控制的碼表

在實驗 4.7.6 中的增強碼表的操作由 S3 板上的三個開關控制。透過 UART，我們可以使用 PC 的 HyperTerminal 發送命令並從秒表檢索時間：

- 當接收到 ASCII 碼 c 或 C（表示「清除」）時，碼表中止當前計數，清零並設置計數方向為“向上”。
- 當接收到 ASCII 碼 g 或 G（表示「開始」）時，碼表開始計數。
- 當接收到 ASCII 碼 p 或 P（表示「暫停」）時，計數暫停。
- 當接收到 ASCII 碼 u 或 U（表示「上下」）時，碼表反轉計數方向。
- 當接收到 ASCII 碼 r 或 R（表示「接收」）時，碼表將當前時間傳輸到 PC，時間應顯示為“DD.DD”，其中 D 是十進制數字。
- 所有其他代碼將被忽略。



程式 7.7.4 UART 控制碼表之模擬圖