Course number: CSC.T341

コンピュータ論理設計
Computer Logic Design

3. ハードウェア記述言語：順序回路
Hardware Description Language: Sequential Circuit

吉瀬 謙二　情報工学系
Kenji Kise, Department of Computer Science
kise _at_ c.titech.ac.jp　www.arch.cs.titech.ac.jp/lecture/CLD/
W621 講義室　月 10:45-12:15, 木 9:00-12:15

Inside code041.v

- シミュレーションのためのクロックの記述例を示す.
- forever文は，続くブロックの処理を無限に繰り返す．この例では，reg型の信号r_clkを開始時に0に初期化し，#50の後にr_clkの値の反転を繰り返す.
- Vivado Waveformウィンドウで波形を確認する．クロック周波数10MHz（クロック周期100ns）のクロックが生成されていることがわかる．Waveformウィンドウの操作は次スライドを参照.

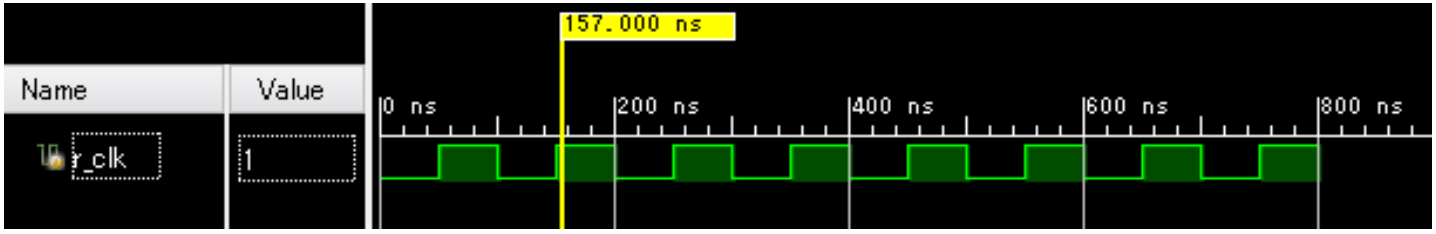
code041.v

```
module m_top ();
  reg r_clk=0;
  initial forever #50 r_clk = ~r_clk;

  always@(*) #1 $write("%3d %d¥n", $time, r_clk);
  initial #800 $finish;
endmodule
```

Simulation output

```
 51 1
101 0
151 1
201 0
251 1
301 0
351 1
401 0
451 1
501 0
551 1
601 0
651 1
701 0
751 1
```

Waveform window of Vivado



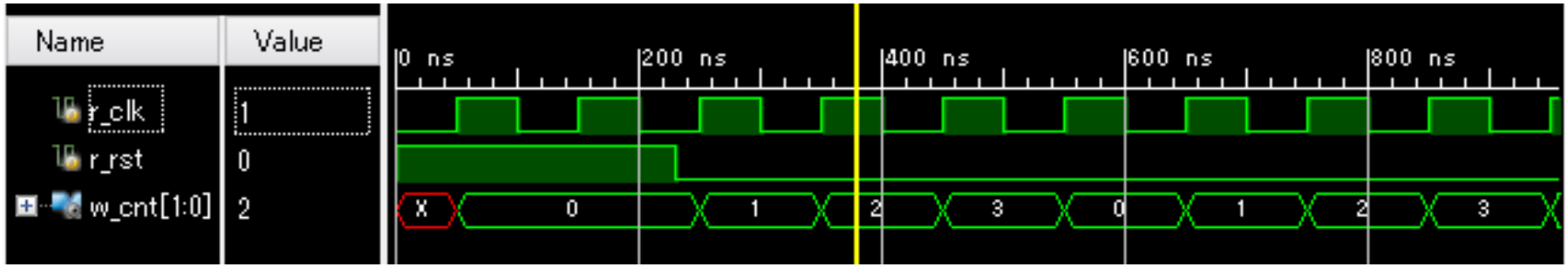
Inside code042.v

- 同期リセット付き2ビットカウンタの記述例を示す.
- クロック信号 w_clkの立ち上がりの時に，リセット信号 r_rst が1の時にはカウンタの値はゼロで初期化され，そうでなければ1インクリメントされる.
- 2ビットカウンタなので，最大値3の次は0となる点に注意.

code042.v

```
module m_top ();
  reg r_clk=0;
  initial forever #50 r_clk = ~r_clk;
  reg r_rst=1;
  initial #230 r_rst=0;
  wire [1:0] w_cnt;
  m_main m_main0 (r_clk, r_rst, w_cnt);
endmodule

module m_main (w_clk, w_rst, w_cnt);
  input  wire w_clk, w_rst;
  output wire [1:0] w_cnt;

  reg [1:0] r_cnt;
  always@(posedge w_clk) begin
    if (w_rst) r_cnt <= 0;
    else r_cnt <= r_cnt + 1;
  end
  assign w_cnt = r_cnt;
endmodule
```

Waveform window



Inside main03.xdc

- main.xdcをmain03.xdcの内容となるように入力する.
- The Nexys4 DDR board includes a single 100 MHz crystal oscillator connected to pin E3.
- 入力ピン clk が, 10.00ns (100MHz)のクロックであることを指定する.
  - すなわち,「100MHzで動作する回路を合成する」という制約を追加する.

main03.xdc

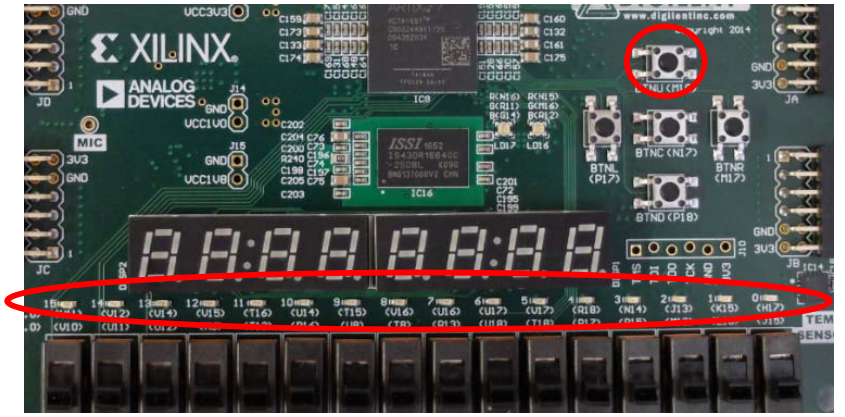
```
set_property -dict { PACKAGE_PIN M18 IOSTANDARD LVCMOS33} [get_ports { w_rst }];
set_property -dict { PACKAGE_PIN E3  IOSTANDARD LVCMOS33} [get_ports { w_clk }];
create_clock -add -name sys_clk -period 10.00 -waveform {0 5} [get_ports {w_clk}];

set_property -dict { PACKAGE_PIN H17 IOSTANDARD LVCMOS33} [get_ports { w_led[0] }];
set_property -dict { PACKAGE_PIN K15 IOSTANDARD LVCMOS33} [get_ports { w_led[1] }];
set_property -dict { PACKAGE_PIN J13 IOSTANDARD LVCMOS33} [get_ports { w_led[2] }];
set_property -dict { PACKAGE_PIN N14 IOSTANDARD LVCMOS33} [get_ports { w_led[3] }];
set_property -dict { PACKAGE_PIN R18 IOSTANDARD LVCMOS33} [get_ports { w_led[4] }];
set_property -dict { PACKAGE_PIN V17 IOSTANDARD LVCMOS33} [get_ports { w_led[5] }];
set_property -dict { PACKAGE_PIN U17 IOSTANDARD LVCMOS33} [get_ports { w_led[6] }];
set_property -dict { PACKAGE_PIN U16 IOSTANDARD LVCMOS33} [get_ports { w_led[7] }];
set_property -dict { PACKAGE_PIN V16 IOSTANDARD LVCMOS33} [get_ports { w_led[8] }];
set_property -dict { PACKAGE_PIN T15 IOSTANDARD LVCMOS33} [get_ports { w_led[9] }];
set_property -dict { PACKAGE_PIN U14 IOSTANDARD LVCMOS33} [get_ports { w_led[10] }];
set_property -dict { PACKAGE_PIN T16 IOSTANDARD LVCMOS33} [get_ports { w_led[11] }];
set_property -dict { PACKAGE_PIN V15 IOSTANDARD LVCMOS33} [get_ports { w_led[12] }];
set_property -dict { PACKAGE_PIN V14 IOSTANDARD LVCMOS33} [get_ports { w_led[13] }];
set_property -dict { PACKAGE_PIN V12 IOSTANDARD LVCMOS33} [get_ports { w_led[14] }];
set_property -dict { PACKAGE_PIN V11 IOSTANDARD LVCMOS33} [get_ports { w_led[15] }];
```

Inside code043.v（ここまでが演習(2)のコード）

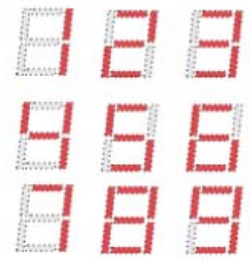
- 32ビットカウンタを用いて16個のLEDを点滅させる回路の例を示す.
- main.xdcをmain03.xdcの内容となるように入力する.
- main.vをcode043.vとなるように入力する.
- FPGAのコンフィギュレーションができたら，担当のTAを呼んで確認してもらう（Check Point 2）.
  - BTNUによってカウンタの値が0に初期化され，LEDが消えることを確認する.

code043.v

```
module m_main (w_clk, w_rst, w_led);
  input  wire w_clk;
  input  wire w_rst;
  output wire [15:0] w_led;

  reg [31:0] r_cnt;
  always@(posedge w_clk) begin
    if (w_rst) r_cnt <= 0;
    else r_cnt <= r_cnt + 1;
  end
  assign w_led = r_cnt[31:16];
endmodule
```



Inside code051.v（ここからが演習(3)のコード）

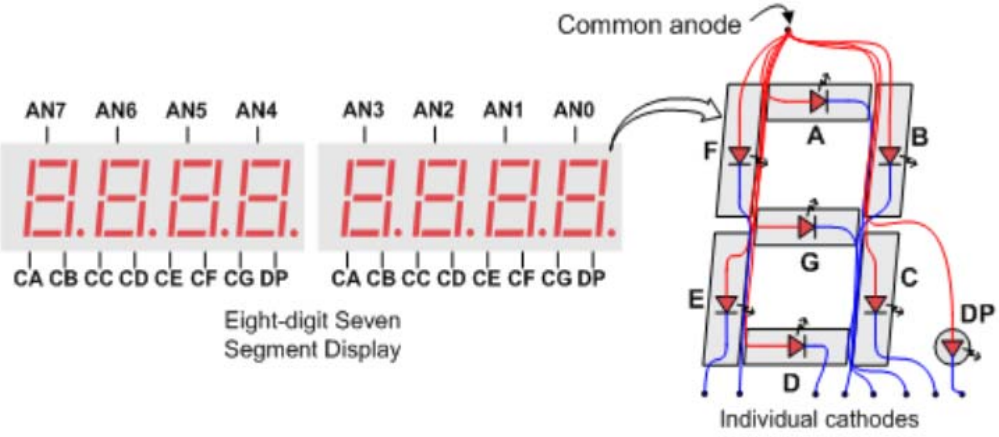


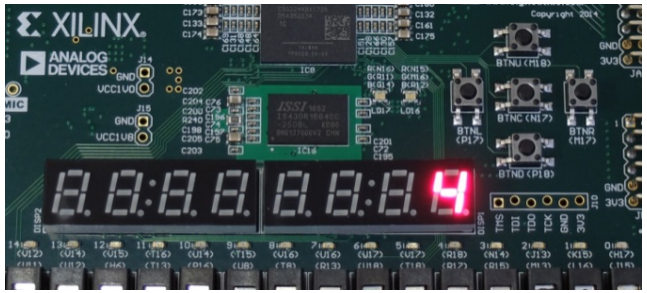
- Seven-segment LED を点灯させる回路の例を示す.
- main.xdcをmain04.xdcの内容となるように編集する.
- main.vをcode051.vとなるように入力，合成し，FPGAをコンフィギュレーション.
  - AN0をゼロとすることで，右端のseven-segment LEDを選択する.
  - AからGに対応するセグメントをゼロにすることで点灯させる. ビット列を反転している点に注意.
  - FPGAボード右端のseven-segment LEDが 4 を表示することを確認する.

code051.v

```
module m_main (w_clk, r_sg, r_an);
  input  wire w_clk;
  output reg [6:0] r_sg;  // cathode segments
  output reg [7:0] r_an;  // common anode

  initial r_an <= 8'b11111110; // select AN0
  initial r_sg <= ~7'b0110011; // ‘4’
endmodule
```

code018.vを参照.





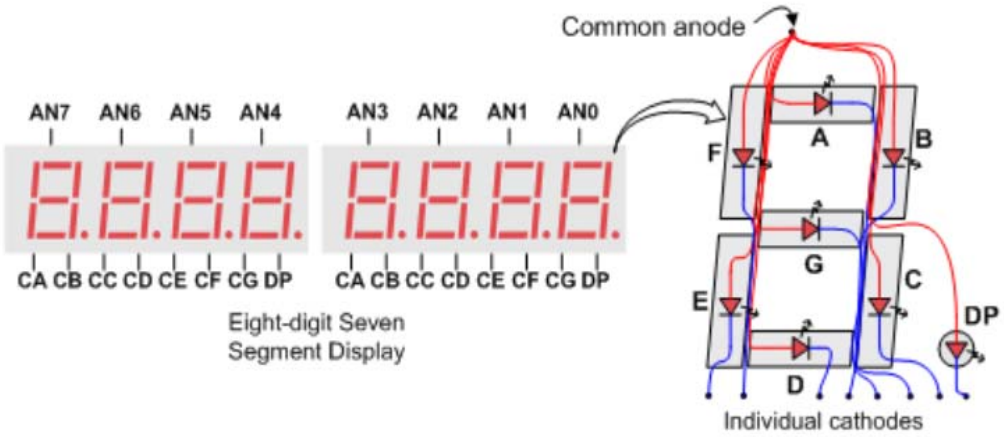
Inside code052.v

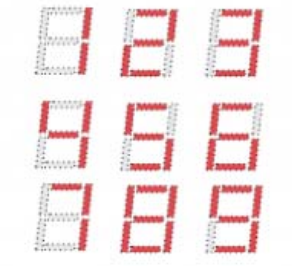
- Seven-segment LED を点灯させる回路の例を示す.
- main.vをcode052.vとなるように入力，合成し，FPGAをコンフィギュレーション.
  - AN1をゼロとすることで，右から2番目のseven-segment LEDを選択する.
  - AからGに対応するセグメントをゼロにすることで点灯させる. ビット列を反転している点に注意
  - FPGAボードの右から2番目のseven-segment LEDが 3 を表示することを確認する.

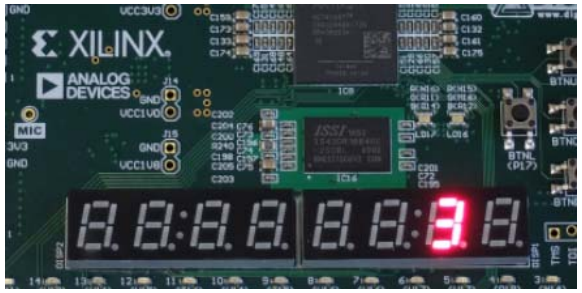
code052.v

```
module m_main (w_clk, r_sg, r_an);
  input  wire w_clk;
  output reg [6:0] r_sg;  // cathode segments
  output reg [7:0] r_an;  // common anode

  initial r_an <= 8'b11111101; // select AN1
  initial r_sg <= ~7'b1111001; // ‘3’
endmodule
```

code018.vを参照.







Inside code053.v

- Seven-segment LED を点灯させる回路の例を示す.
  - 右から2番目のseven-segment LEDに 3 を，右端に 4 を表示するにはどうすればよいか？
- 一定の間隔で（2サイクル毎に），右端と右から2番目とを切り替える.
  - main.vをcode053.vとなるように入力して，シミュレーションする.

code053.v

```
module m_top ();
  reg r_clk=0; initial forever #50 r_clk = ~r_clk;
  wire [6:0] w_sg;
  wire [7:0] w_an;
  m_main m_main0 (r_clk, w_sg, w_an);
  always@(*) $write("%3d %b %b¥n", $time, w_an, w_sg);
endmodule

module m_main (w_clk, r_sg, r_an);
  input  wire w_clk;
  output reg [6:0] r_sg;  // cathode segments
  output reg [7:0] r_an;  // common anode

  reg [31:0] r_cnt=0;
  reg        r_digit=0;
  always@(posedge w_clk) begin
    r_cnt <= (r_cnt>=(2-1)) ? 0 : r_cnt + 1;
    if(r_cnt==0) begin
      r_digit <= r_digit+ 1;
      if      (r_digit==0) begin r_an <= 8'b11111110; r_sg <= ~7'b0110011; end
      else if (r_digit==1) begin r_an <= 8'b11111101; r_sg <= ~7'b1111001; end
    end
  end
endmodule
```

```
 50 11111110 1001100
250 11111101 0000110
450 11111110 1001100
650 11111101 0000110
850 11111110 1001100
```

Inside code054.v

- Seven-segment LED を点灯させる回路の例を示す.
- main.vをcode054.vとなるように修正，合成し，FPGAをコンフィギュレーション.
  - 右から2番目のseven-segment LEDに 3 を，右端に 4 を表示するにはどうすればよいか？
- 一定の間隔で（200,000サイクル，2msec 毎に），右端と右から2番目とを切り替える.

code054.v

```
module m_main (w_clk, r_sg, r_an);
  input  wire w_clk;
  output reg [6:0] r_sg;  // cathode segments
  output reg [7:0] r_an;  // common anode

  reg [31:0] r_cnt=0;
  reg        r_digit=0;
  always@(posedge w_clk) begin
    r_cnt <= (r_cnt>=(200000-1)) ? 0 : r_cnt + 1;
    if(r_cnt==0) begin
      r_digit <= r_digit+ 1;
      if      (r_digit==0) begin r_an <= 8'b11111110; r_sg <= ~7'b0110011; end
      else if (r_digit==1) begin r_an <= 8'b11111101; r_sg <= ~7'b1111001; end
    end
  end
endmodule
```

Inside code055.v

- 1秒間隔で2つの seven-segment LED に点灯させる数字を交換する回路の例を示す.
  - 1秒間隔で，34, 43, 34, 43 と表示を変更する回路.
- main.vをcode055.vとなるように入力，合成し，FPGAをコンフィギュレーション.
- 一定の間隔（100,000,000サイクル毎）で，表示するセグメントのパターンを変更する.

code055.v

```
module m_main (w_clk, r_sg, r_an);
  input  wire w_clk;
  output reg [6:0] r_sg;  // cathode segments
  output reg [7:0] r_an;  // common anode

  reg [31:0] r_tcnt=0;
  always@(posedge w_clk) r_tcnt <= (r_tcnt>=(100000000-1)) ? 0 : r_tcnt + 1;
  reg r_num=0;
  always@(posedge w_clk) if(r_tcnt==0) r_num <= r_num + 1;

  reg [31:0] r_cnt=0;
  reg        r_digit=0;
  always@(posedge w_clk) begin
    r_cnt <= (r_cnt>=(200000-1)) ? 0 : r_cnt + 1;
    if(r_cnt==0) begin
      r_digit <= r_digit+ 1;
      if      (r_digit==0) begin r_an <= 8'b11111110; r_sg <= (r_num==0) ? ~7'b0110011 : ~7'b1111001; end
      else if (r_digit==1) begin r_an <= 8'b11111101; r_sg <= (r_num==0) ? ~7'b1111001 : ~7'b0110011; end
    end
  end
endmodule
```

Inside code061.v

- シフトレジスタの例を示す．クロック等に同期して1ビットずつシフトしていくレジスタをシフトレジスタと呼ぶ．
- main.v をcode061.vの内容となるように入力し，シミュレーションをおこなう．
- w_data として指定された 8'b11001111 を毎サイクル1ビット右にシフトし，最下位ビットを出力．

code061.v

```
module m_top ();
  reg r_clk=0; initial forever #20 r_clk = ~r_clk;
  reg r_we=0;  initial begin #210 r_we = 1; #20 r_we = 0; end
  wire w_dout;
  always@(posedge r_clk) begin #1
    $write("%3d %d %b %d¥n", $time, r_we, m_main0.r_shiftreg, w_dout);
  end
  initial #1000 $finish();
  m_main m_main0 (r_clk, 8'b11001111, r_we, w_dout);
endmodule

module m_main (w_clk, w_data, w_we, w_dout);
  input wire w_clk;
  input wire [7:0] w_data;
  input wire w_we;
  output wire w_dout;

  reg [7:0] r_shiftreg = 0;
  always@(posedge w_clk) begin
    if (w_we) r_shiftreg <= w_data;
    else      r_shiftreg <= {1'b0, r_shiftreg[7:1]};
  end
  assign w_dout = r_shiftreg[0];
endmodule
```

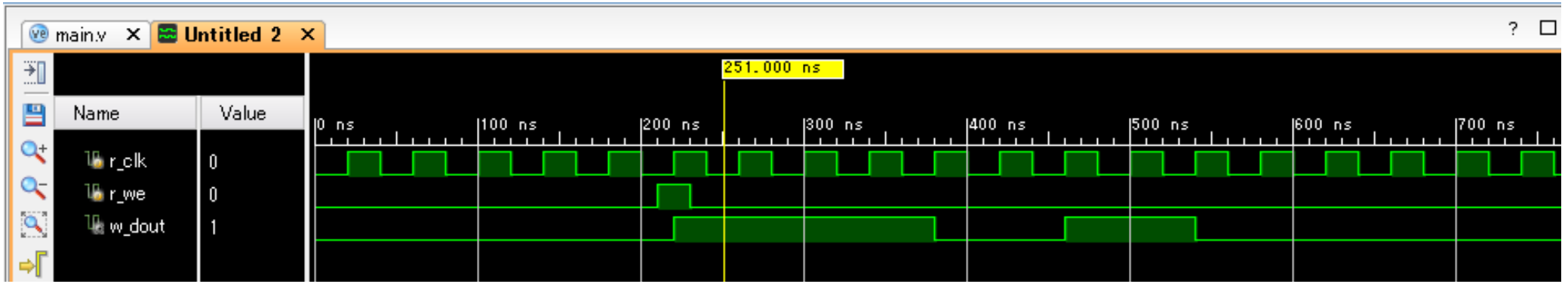
Simulation output

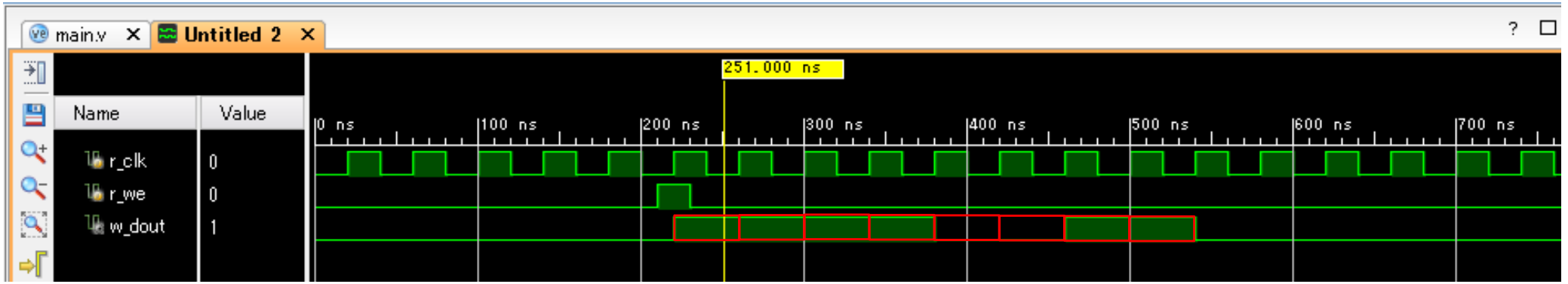
```
 21 0 00000000 0
 61 0 00000000 0
101 0 00000000 0
141 0 00000000 0
181 0 00000000 0
221 1 11001111 1
261 0 01100111 1
301 0 00110011 1
341 0 00011001 1
381 0 00001100 0
421 0 00000110 0
461 0 00000011 1
501 0 00000001 1
541 0 00000000 0
581 0 00000000 0
621 0 00000000 0
661 0 00000000 0
701 0 00000000 0
741 0 00000000 0
781 0 00000000 0
821 0 00000000 0
861 0 00000000 0
901 0 00000000 0
941 0 00000000 0
981 0 00000000 0
```

Inside code061.v

- この回路は Parallel Input / Serial Output (PISO) のシフトレジスタとも呼ばれる.
- シフトレジスタの最上位ビットには1'b0が格納されていくので，8ビットのデータ8'b11001111が出力された後は0が出力され続ける.

Waveform window

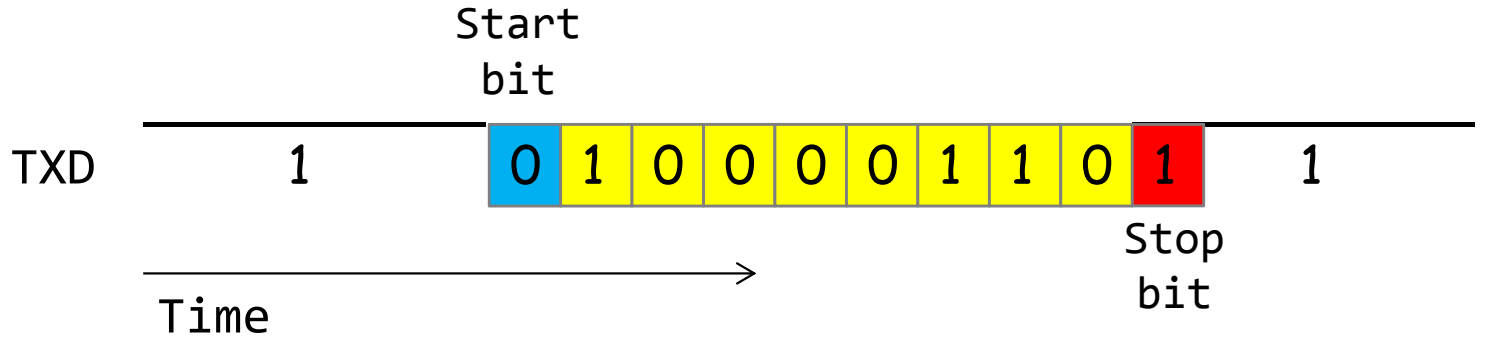




出力データを赤色で強調

UART (Universal Asynchronous Receiver/Transmitter)

- 調歩同期方式によるシリアル信号をパラレル信号に変換したり，その逆方向の変換をおこなう集積回路をUARTと呼ぶ．8ビット（1バイト）単位でデータを送信・受信する．
- UARTを用いることで，FPGAとコンピュータの間でのお手軽なデータ通信が可能．
- 例えば，'a' という文字を送信する場合，'a' は 8'h61, 8'b01100001（次スライドのASCII Tableを参照）なので，下図のタイミングで送信線TXDを制御する．
  - データが送信されるまで送信線TXD を1とする．
  - まず，青色で示した0（これをスタートビットと呼ぶ）を送信することで，データ送信の開始を明示．
  - 次に，黄色で示した様に送信したいデータ 8'b01100001 の最下位ビットから順番に送信する．
  - 最後に，赤色で示した1（これをストップビットと呼ぶ）を送信する．
- 1ビットを送受信するための時間間隔は送信側と受信側で同じレートを用いる．これをボー・レート (baud) と呼ぶ．
  - 9,600, 14,400 baud 等が一般に用いられる．この演習では主に 1,000,000 baudを用いる．



ASCII (American Standard Code for Information Interchange) Table

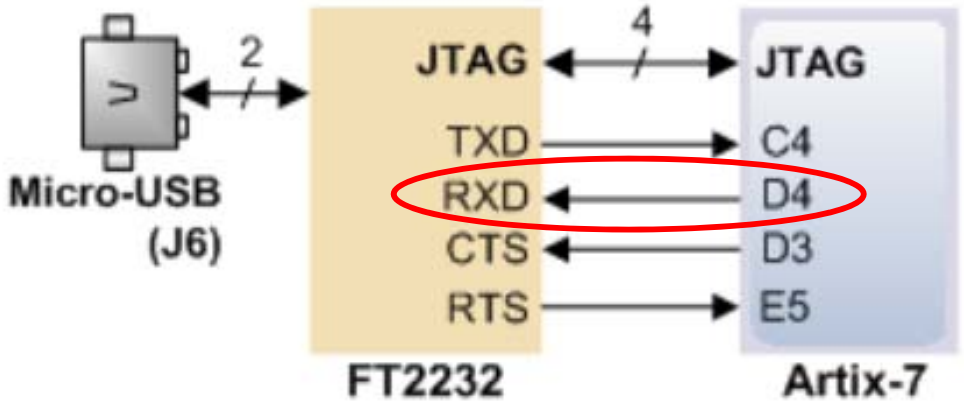
Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	[space]	64	40	100	@	96	60	140	`
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	"	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	'	71	47	107	G	103	67	147	g
8	8	10		40	28	50	(	72	48	110	H	104	68	150	h
9	9	11		41	29	51	)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	-	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	.	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	70	8	88	58	130	X	120	78	170	x
25	19	31		57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	73	;	91	5B	133	[	123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	|
29	1D	35		61	3D	75	=	93	5D	135	]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37		63	3F	77	?	95	5F	137	_	127	7F	177	

Inside main05.xdc

- main.xdcをmain05.xdcの内容となるように入力する.
- ピン w_txd は, UARTのFPGAからのデータ送信のために用いる.
  - コンピュータからはUARTのデータ受信 (RXD) となる.

main05.xdc

```
set_property -dict { PACKAGE_PIN E3   IOSTANDARD LVCMOS33} [get_ports { w_clk }];
create_clock -add -name sys_clk -period 10.00 -waveform {0 5} [get_ports {w_clk}];

set_property -dict { PACKAGE_PIN D4   IOSTANDARD LVCMOS33} [get_ports { w_txd }];
```



Inside code062.v

- シリアル通信によるFPGAからの送信回路の例を示す.
- シリアル通信のボー・レートを 25Mbaudとする(速すぎてあまり実用的ではない).
  - すなわち, クロックが100MHzなので, 4サイクルで1ビットを送信する.
- main.vをcode062.vの内容となるように入力し, 3000nsec までシミュレーションする.

code062.v

```
module m_top ();
  reg r_clk=0; initial forever #50 r_clk = ~r_clk;
  reg r_we=0;
  wire w_txd;
  initial begin #130 r_we = 1; #100 r_we = 0; end
  m_UartTx m_UartTx0(r_clk, 8'h61, r_we, w_txd);
  always@(posedge r_clk) $write("%4d %b %b %b¥n", $time, r_we, w_txd, m_UartTx0.r_data);
endmodule

module m_UartTx (w_clk, w_data, w_we, r_txd);
  input wire       w_clk, w_we;
  input wire [7:0] w_data;;
  output reg       r_txd;
  initial r_txd = 1;
  reg [8:0] r_data =~0;
  reg [7:0] r_wait = 0;
  always@(posedge w_clk) begin
    if (w_we) begin
      r_data <= {w_data, 1'b0}; // add start bit
      r_wait <= 0;
    end else if (r_wait >= (4-1)) begin
      r_txd  <= r_data[0];
      r_data <= {1'b1, r_data[8:1]};
      r_wait <= 0;
    end else begin
      r_wait <= r_wait + 1;
    end
  end
endmodule
```

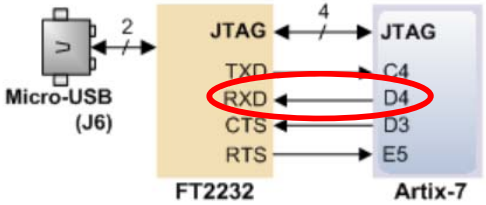
```
  50 0 1 111111111
 150 1 1 111111111
 250 0 1 011000010
 350 0 1 011000010
 450 0 1 011000010
 550 0 1 011000010
 650 0 0 101100001
 750 0 0 101100001
 850 0 0 101100001
 950 0 0 101100001
1050 0 1 110110000
1150 0 1 110110000
1250 0 1 110110000
1350 0 1 110110000
1450 0 0 111011000
1550 0 0 111011000
1650 0 0 111011000
1750 0 0 111011000
1850 0 0 111101100
1950 0 0 111101100
2050 0 0 111101100
2150 0 0 111101100
2250 0 0 111110110
2350 0 0 111110110
2450 0 0 111110110
2550 0 0 111110110
2650 0 0 111111011
2750 0 0 111111011
2850 0 0 111111011
2950 0 0 111111011
```

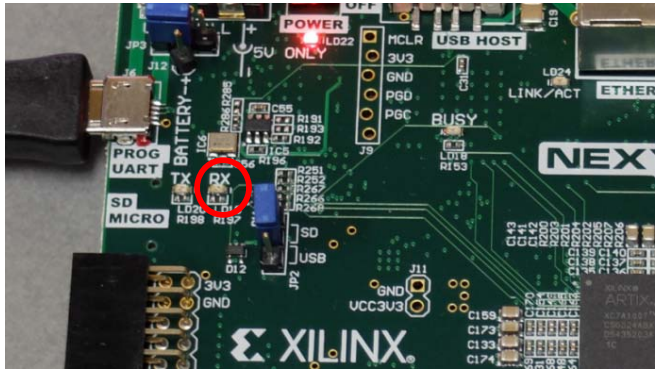
Inside code063.v

- シリアル通信によるFPGAからの送信回路の例を示す.
- シリアル通信のボー・レートを 1,000,000 baud (1Mbaud) とする.
  - すなわち，クロックが100MHzなので，100サイクルで1ビットを送信する.
- main.vをcode063.vの内容となるように入力し，合成，コンフィギュレーションする.
  - FPGAボードのRXと書かれたLEDが一定間隔で点滅する.

code063.v

```
module m_main(w_clk, w_txd);
  input  wire w_clk;
  output wire w_txd;
  reg r_we;
  reg [31:0] r_cnt = 1;
  always@(posedge w_clk) r_cnt <= (r_cnt>50000000) ? 0 : r_cnt + 1;
  always@(posedge w_clk) r_we <= (r_cnt==0);
  m_UartTx m_UartTx0(w_clk, 8'h61, r_we, w_txd);
endmodule
```



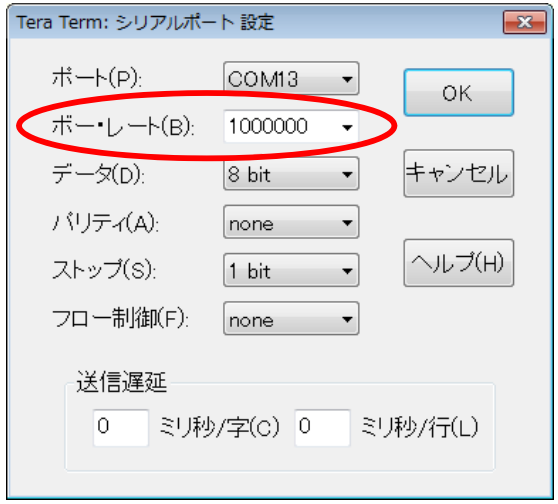


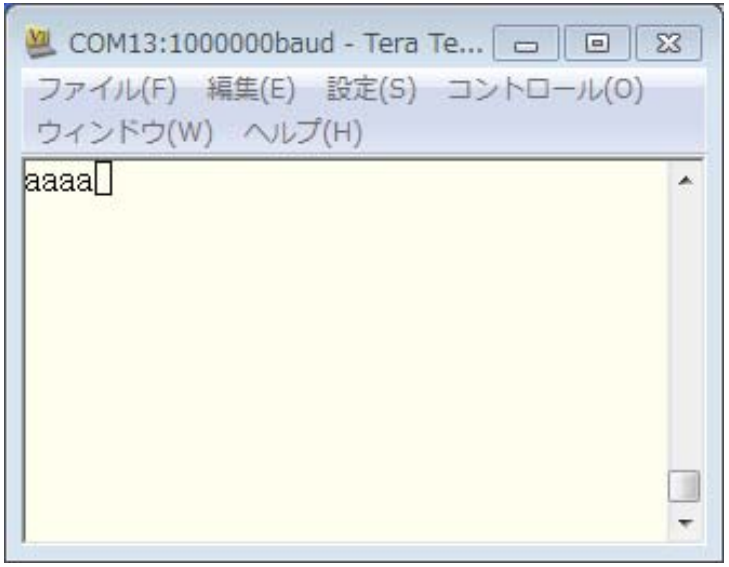
code063.v

```
module m_UartTx (w_clk, w_data, w_we, r_txd);
  input wire       w_clk, w_we;
  input wire [7:0] w_data;;
  output reg       r_txd;
  initial r_txd = 1;
  reg [8:0] r_data =~0;
  reg [7:0] r_wait = 0;
  always@(posedge w_clk) begin
    if (w_we) begin
      r_data <= {w_data, 1'b0}; // add start bit
      r_wait <= 0;
    end else if (r_wait >= (100-1)) begin
      r_txd  <= r_data[0];
      r_data <= {1'b1, r_data[8:1]};
      r_wait <= 0;
    end else begin
      r_wait <= r_wait + 1;
    end
  end
endmodule
```

Inside code063.v

- main.vをcode063.vの内容となるように入力し，合成，コンフィギュレーションする．
- Tera Term を起動する．
  - シリアルポートを選択．複数のポートが選択できる場合には最も大きい番号を選択．
  - 設定，シリアルポートを選択し，ボー・レートに 1000000 を入力
  - Tera Term に一定の間隔で a の文字が表示される．



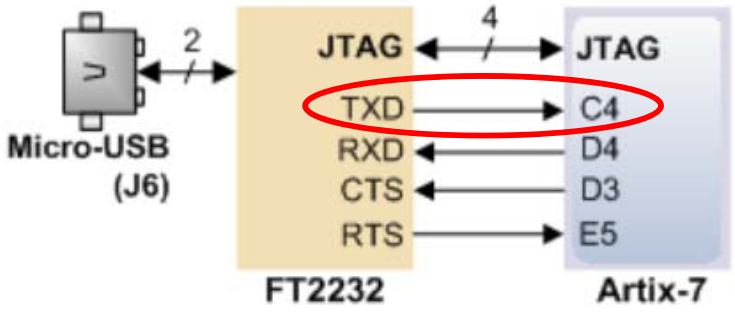


Inside main06.xdc

- main.xdcをmain06.xdcの内容となるように入力する.
- ピン w_rxd は, FPGAのデータ受信. コンピュータからはデータ送信 (TXD) となる.

main06.xdc

```
set_property -dict { PACKAGE_PIN E3  IOSTANDARD LVCMOS33} [get_ports { w_clk }];
create_clock -add -name sys_clk -period 10.00 -waveform {0 5} [get_ports {w_clk}];

set_property -dict { PACKAGE_PIN C4  IOSTANDARD LVCMOS33} [get_ports { w_rxd }];

set_property -dict { PACKAGE_PIN H17 IOSTANDARD LVCMOS33} [get_ports { w_led[0] }];
set_property -dict { PACKAGE_PIN K15 IOSTANDARD LVCMOS33} [get_ports { w_led[1] }];
set_property -dict { PACKAGE_PIN J13 IOSTANDARD LVCMOS33} [get_ports { w_led[2] }];
set_property -dict { PACKAGE_PIN N14 IOSTANDARD LVCMOS33} [get_ports { w_led[3] }];
set_property -dict { PACKAGE_PIN R18 IOSTANDARD LVCMOS33} [get_ports { w_led[4] }];
set_property -dict { PACKAGE_PIN V17 IOSTANDARD LVCMOS33} [get_ports { w_led[5] }];
set_property -dict { PACKAGE_PIN U17 IOSTANDARD LVCMOS33} [get_ports { w_led[6] }];
set_property -dict { PACKAGE_PIN U16 IOSTANDARD LVCMOS33} [get_ports { w_led[7] }];
```



Inside code064.v

- シリアル通信を用いたFPGA側の受信回路の例を示す.
- シリアル通信のボーレートを25M baudとする. クロックが100MHzなので 4サイクルで1ビットを受信.
- main.vをcode064.vの内容となるように入力し, 5000nsec をシミュレーションする.
  - code063.v の m_UartTx も必要.

code064.v

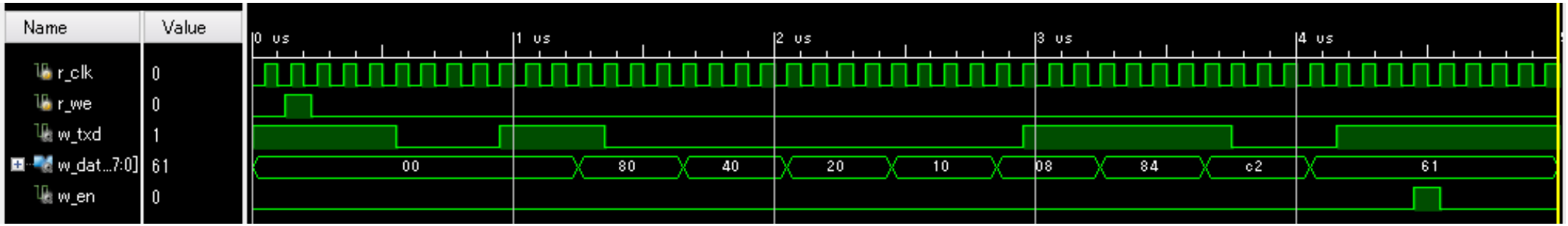
```
module m_top ();
  reg r_clk=0; initial forever #50 r_clk = ~r_clk;
  reg r_we=0;
  wire w_txd;
  initial begin #130 r_we = 1; #100 r_we = 0; end
  m_UartTx m_UartTx0(r_clk, 8'h61, r_we, w_txd);
  wire [7:0] w_data;
  wire       w_en;
  m_UartRx m_UartRx0(r_clk, w_txd, w_data, w_en);

  always@(posedge r_clk) begin
    $write("%4d %b %b", $time, r_we, w_txd);
    $write(" %b -> ", m_UartTx0.r_data);
    $write("%b %b¥n", w_data, w_en);
  end
endmodule
```

code064.v

```
module m_UartRx(w_clk, w_rxd, r_data, r_en);
  input  wire      w_clk, w_rxd;
  output reg [7:0] r_data;
  output reg       r_en;
  initial begin r_data = 0; r_en = 0; end
  reg [5:0] r_detect = 0;
  always @(posedge w_clk) r_detect <= (w_rxd) ? 0 : r_detect + 1;
  reg       r_recv = 0;
  reg [6:0] r_wait = 0;
  reg [3:0] r_cnt  = 0;
  always@(posedge w_clk) begin
    if (r_recv==0) begin
        if(r_detect>1) begin r_recv <= 1; r_data<= 0; end
        r_cnt  <= 8;
        r_wait <= 0;
        r_en   <= 0;
    end else begin
      if (r_wait >= (4-1)) begin
        r_cnt  <= r_cnt - 1;
        r_data <= (r_cnt==0) ? r_data : {w_rxd, r_data[7:1]};
        r_wait <= 0;
        if (r_cnt == 0) begin r_en <= 1; r_recv <= 0; end
      end else begin
        r_wait <= r_wait + 1;
      end
    end
  end
endmodule
```

Inside code064.v

Waveform window




```
  50 0 1 111111111 -> 00000000 0
 150 1 1 111111111 -> 00000000 0
 250 0 1 011000010 -> 00000000 0
 350 0 1 011000010 -> 00000000 0
 450 0 1 011000010 -> 00000000 0
 550 0 1 011000010 -> 00000000 0
 650 0 0 101100001 -> 00000000 0
 750 0 0 101100001 -> 00000000 0
 850 0 0 101100001 -> 00000000 0
 950 0 0 101100001 -> 00000000 0
1050 0 1 110110000 -> 00000000 0
1150 0 1 110110000 -> 00000000 0
1250 0 1 110110000 -> 00000000 0
1350 0 1 110110000 -> 10000000 0
1450 0 0 111011000 -> 10000000 0
1550 0 0 111011000 -> 10000000 0
1650 0 0 111011000 -> 10000000 0
1750 0 0 111011000 -> 01000000 0
1850 0 0 111101100 -> 01000000 0
1950 0 0 111101100 -> 01000000 0
2050 0 0 111101100 -> 01000000 0
2150 0 0 111101100 -> 00100000 0
2250 0 0 111110110 -> 00100000 0
2350 0 0 111110110 -> 00100000 0
2450 0 0 111110110 -> 00100000 0
```

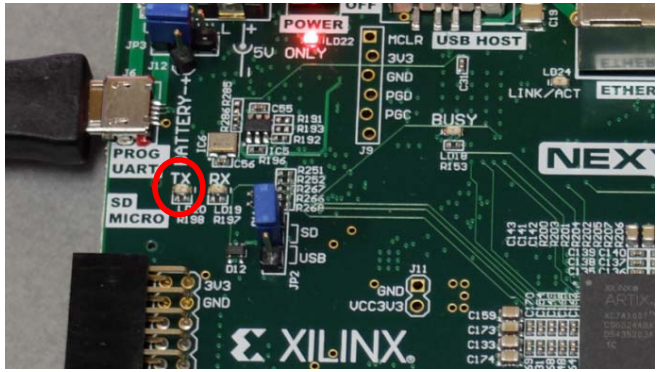
```
2550 0 0 111110110 -> 00010000 0
2650 0 0 111111011 -> 00010000 0
2750 0 0 111111011 -> 00010000 0
2850 0 0 111111011 -> 00010000 0
2950 0 0 111111011 -> 00001000 0
3050 0 1 111111101 -> 00001000 0
3150 0 1 111111101 -> 00001000 0
3250 0 1 111111101 -> 00001000 0
3350 0 1 111111101 -> 10000100 0
3450 0 1 111111110 -> 10000100 0
3550 0 1 111111110 -> 10000100 0
3650 0 1 111111110 -> 10000100 0
3750 0 1 111111110 -> 11000010 0
3850 0 0 111111111 -> 11000010 0
3950 0 0 111111111 -> 11000010 0
4050 0 0 111111111 -> 11000010 0
4150 0 0 111111111 -> 01100001 0
4250 0 1 111111111 -> 01100001 0
4350 0 1 111111111 -> 01100001 0
4450 0 1 111111111 -> 01100001 0
4550 0 1 111111111 -> 01100001 1
4650 0 1 111111111 -> 01100001 0
4750 0 1 111111111 -> 01100001 0
4850 0 1 111111111 -> 01100001 0
4950 0 1 111111111 -> 01100001 0
```

Inside code065.v

- シリアル通信を用いたFPGA側の受信回路の例を示す.
- シリアル通信のボーレートを 1,000,000 baud (1Mbaud) とする.
  - すなわち, クロックが100MHzなので, 100サイクルで1ビットを受信する.
- main.vをcode065.vの内容となるように入力し, 合成, コンフィギュレーションする.
- Tera Term からの入力で, LEDの点滅を確認.

code065.v

```
module m_main(w_clk, w_rxd, w_led);
  input  wire w_clk;
  input  wire w_rxd;
  output wire [7:0] w_led;

  wire w_en;
  m_UartRx m_UartRx0 (w_clk, w_rxd, w_led, w_en);
endmodule
```



code065.v

```
module m_UartRx(w_clk, w_rxd, r_data, r_en);
  input  wire      w_clk, w_rxd;
  output reg [7:0] r_data;
  output reg       r_en;
  initial begin r_data = 0; r_en = 0; end
  reg [5:0] r_detect = 0;
  always @(posedge w_clk) r_detect <= (w_rxd) ? 0 : r_detect + 1;
  reg       r_recv = 0;
  reg [6:0] r_wait = 0;
  reg [3:0] r_cnt  = 0;
  always@(posedge w_clk) begin
    if (r_recv==0) begin
        if(r_detect>10) begin r_recv <= 1; r_data<= 0; end
        r_cnt  <= 8;
        r_wait <= 0;
        r_en   <= 0;
    end else begin
      if (r_wait >= (100-1)) begin
        r_cnt  <= r_cnt - 1;
        r_data <= (r_cnt==0) ? r_data : {w_rxd, r_data[7:1]};
        r_wait <= 0;
        if (r_cnt == 0) begin r_en <= 1; r_recv <= 0; end
      end else begin
        r_wait <= r_wait + 1;
      end
    end
  end
endmodule
```

References

- Computer Logic Design support page
  - http://www.arch.cs.titech.ac.jp/lecture/CLD/
- 情報工学系計算機室
  - http://www.csc.titech.ac.jp/
- Xilinx Vivado Design Suite
  - https://japan.xilinx.com/products/design-tools/vivado.html
- Digilent Nexys 4 DDR Atrix-7 FPGA
  - https://reference.digilentinc.com/reference/programmable-logic/nexys-4-ddr/start
  - https://store.digilentinc.com/nexys-4-ddr-artix-7-fpga-trainer-board-recommended-for-ece-curriculum/
- Verilog HDL
  - https://ja.wikipedia.org/wiki/Verilog
- Frix (Feasible and Reconfigurable IBM PC Compatible SoC)
  - http://www.arch.cs.titech.ac.jp/a/Frix/