Verilog-HDL ゼミ 第 4 週・練習問題「順序回路と assign 文」

問 1.

第 3 週のゼミ課題・問 1 で作成したモジュール counter の出力について、奇数個のビットが 1 のときには 1 を出力し、偶数個のビットが 1 のとき(4'b0000 を含む)には 0 を出力するモジュール counter_trans を作成してください。なお、作成したモジュールのシミュレーションを行う際は、テストベンチ counter_trans_tb.v を用いて行ってみてください。

```
module counter_trans ( CLK, RESETL, OUT ); // 4bit counter
input  CLK, RESETL;
output OUT;
wire [3:0] tmp;      // counter の出力に接続

// counter モジュールを呼び出す
…;
// 奇数個のビットが 1 の時には 1, 偶数個のビットが 1 のときには 0 を出力
assign OUT = …;

endmodule
```

プログラムリスト 1. counter_trans.v

```
`timescale 1ps / 1ps

module counter_trans_tb;
reg    CLK, RESETL;
wire   OUT;

parameter STEP = 1000;

// counter_trans を呼び出す
counter_trans counter_trans ( CLK, RESETL, OUT );

// クロック生成
always begin
       CLK = ~CLK; #(STEP/2); // 半ステップ毎に CLK を反転してクロック生成
end

// テスト入力
initial begin
       $dumpfile("counter_trans_tb.vcd");
       $dumpvars(0, counter_trans_tb);
       $monitor( $stime, " CLK=%b RESETL=%b Q=%b OUT=%b", CLK,
RESETL, counter_trans.counter.Q, OUT);

                      CLK = 0; RESETL = 1;
       #STEP          RESETL = 0;
       #STEP          RESETL = 1;
       // 20 回カウントアップして終了(20 クロック分進める)
       …              $finish;
end

endmodule
```

プログラムリスト 2. counter_trans_tb.v

以下にコマンド実行例を示します。今回は前回の問 2 の場合と同様に、出力ファイルと期待値ファイルとの比較によってモジュールが正しく作成できたかを確認しています。動作確認については、テストベンチにおいて 4 ビットカウンタモジュール counter のカウント値と counter_trans の出力を同時に出力しているため、テストベンチの実行結果を目視することでも行うことができます。また、赤文字で示したファイルの指定に注意してください。

コマンド例) > iverilog -o counter_trans_tb -s counter_trans_tb counter_tb.v counter_trans.v counter.v

```
vvp counter_trans_tb > counter_trans_tb_out.txt
fc counter_trans_tb_out.txt counter_trans_tb_true_iv.txt
gtkwave counter_trans_tb.vcd
```

or

```
cver counter_trans_tb.v counter_trans.v counter.v > counter_trans_tb_out.txt
fc counter_trans_tb_out.txt counter_trans_tb_true_cv.txt
gtkwave counter_trans_tb.vcd
```

※ ファイル比較コマンドの fc は、Linux 系の OS では diff となります。

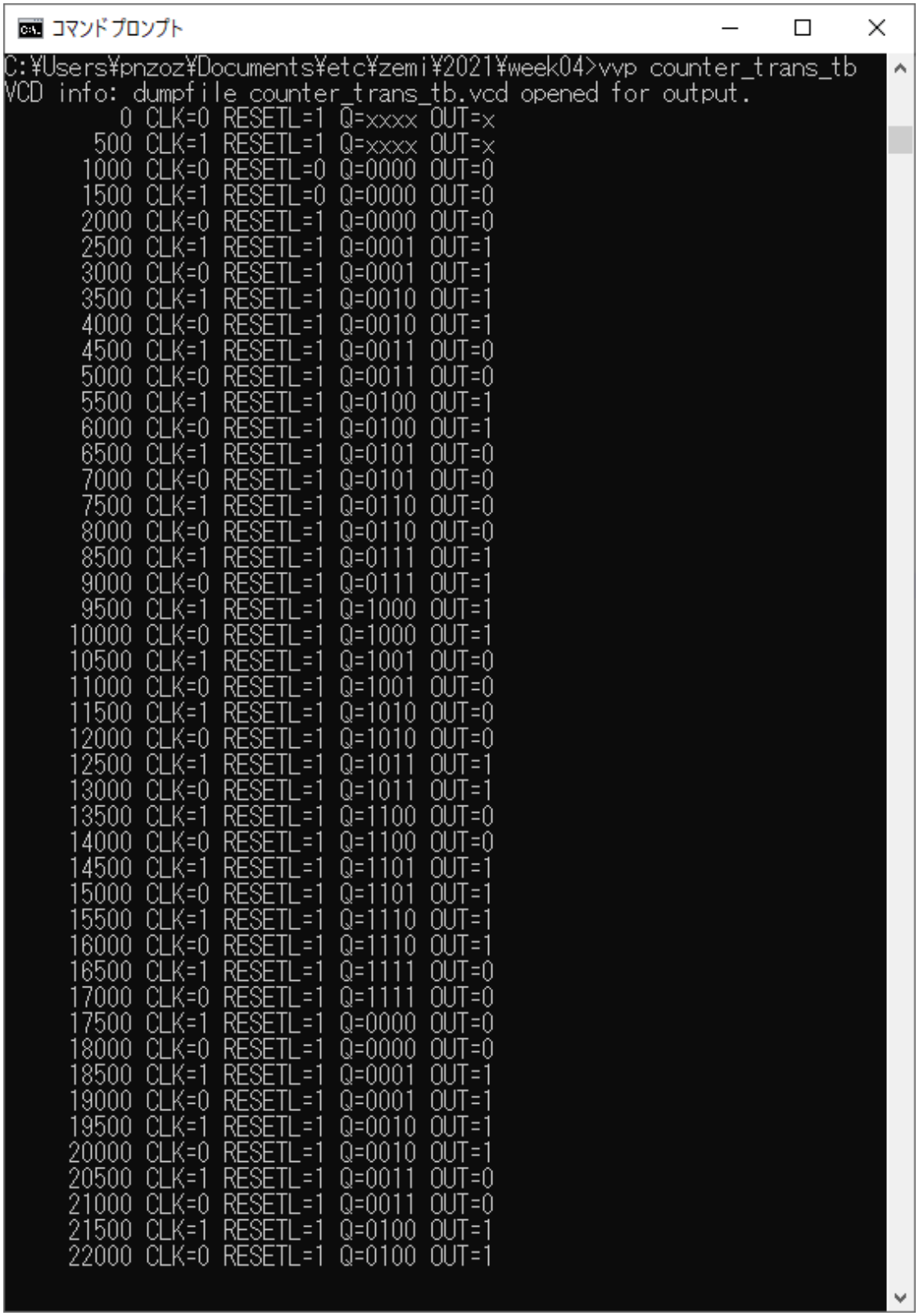

図 1. 出力結果の例(vvp)

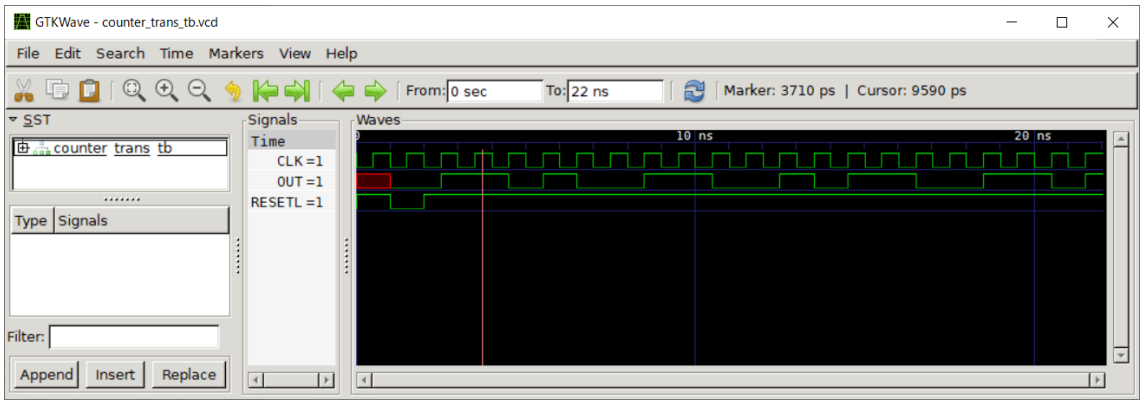

図 2. 波形表示例 (gtkwave)

次に、グレイコードカウンタについて考えてみます。グレイコードカウンタは、表 1 に示すように常に出力の 1 ビットのみが変化するカウンタのことをいいます。グレイコードカウンタは図 3 のブロック図のようになり、4 ビットのフリップフロップと、フリップフロップに入力するグレイコード生成の組み合わせ回路から構成されます。

表 1. グレイコードカウンタの真理値表

現在値	次の値
0000	0001
0001	0011
0011	0010
0010	0110
0110	0111
0111	0101
0101	0100
0100	1100
1100	1101
1101	1111
1111	1110
1110	1010
1010	1011
1011	1001
1001	1000
1000	0000

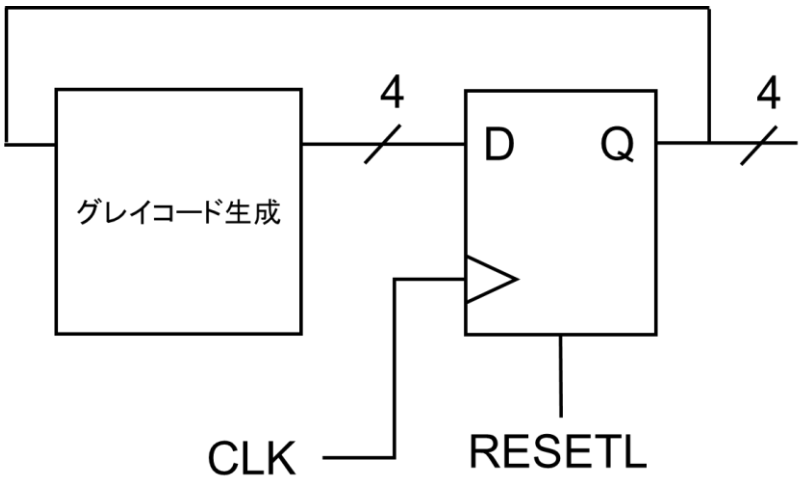

図 3. グレイコードカウンタのブロック図

回路としては、プログラムリスト 3 のように記述されます。グレイコードカウンタのグレイコード生成部は function 文で記述した部分となります。function 文においては、表 1 を参考に、現在のカウンタ値を入力として次のカウンタ値を出力するように記述してみてください。最後の always 文はフリップフロップです。リセット信号の立下りによって非同期にカウンタがリセットされます。また、リセット信号が 0 でなければクロック信号 CLK の立ち上がり毎にカウントアップされます。レジスタ Q にカウントアップ値を代入する際は、function 文で定義した gray を呼び出します。gray は次のカウンタ値を出力するため、呼び出しの際には現在のカウンタ値 Q を入力として与えることに注意してください。

問 2.

プログラムリスト 3 に示すグレイコードカウンタ gray_counter を完成させ、シミュレーションによりグレイコードカウンタが正しく動作することを確認してください。シミュレーションに使用するテストベンチについては、プログラムリスト 4 を使用してください。

```
module gray_counter ( CLK, RESETL, Q );
input CLK, RESETL;
output [3:0] Q;
reg [3:0] Q;

// グレイコード生成部
function [3:0] gray;
input [3:0] gray_in;
	case ( gray_in )
		// 現在のカウンタ値によって次のカウンタ値を代入
		4'b0000: gray = 4'b0001;
		4'b0001: gray = 4'b0011;
		// 以下より表 1 の残りのカウンタ値の場合について記述
		…;
		default: gray = 4'bxxxx;
	endcase
endfunction

always @( posedge CLK or negedge RESETL ) begin
	if ( RESETL == 1'b0 )
		Q <= 4'h0;
	else
		// gray の出力を代入
		Q <= …;
end

endmodule
```

プログラムリスト 3. gray_counter.v

```
`timescale 1ps / 1ps

module gray_counter_tb;
reg    CLK, RESETL;
wire   [3:0]  Q;

parameter STEP = 1000;

// グレイコードカウンタを呼び出す
gray_counter gray_counter( CLK, RESETL, Q );

// クロック生成
always begin
       CLK = ~CLK; #(STEP/2); // 半ステップ毎に CLK を反転してクロック生成
end

// テスト入力
initial begin
       $dumpfile("gray_counter_tb.vcd");
       $dumpvars(0, gray_counter_tb);
       $monitor( $stime, " CLK=%b RESETL=%b Q=%b", CLK, RESETL, Q);

                     CLK = 0; RESETL = 1;
       #STEP         RESETL = 0;
       #STEP         RESETL = 1;
       // 20 回カウントアップして終了(20 クロック分進める)
       …             $finish;
end

endmodule
```

プログラムリスト 4. gray_counter_tb.v

コマンド例) > iverilog -o gray_counter_tb -s gray_counter_tb gray_counter_tb.v gray_counter.v

vvp gray_counter_tb > gray_counter_tb_out.txt

fc gray_counter_tb_out.txt gray_counter_tb_true_iv.txt

or

cver gray_counter_tb.v gray_counter.v > gray_counter_tb_out.txt

fc gray_counter_tb_out.txt gray_counter_tb_true_cv.txt

※ ファイル比較コマンドの fc は、Linux 系の OS では diff となります。

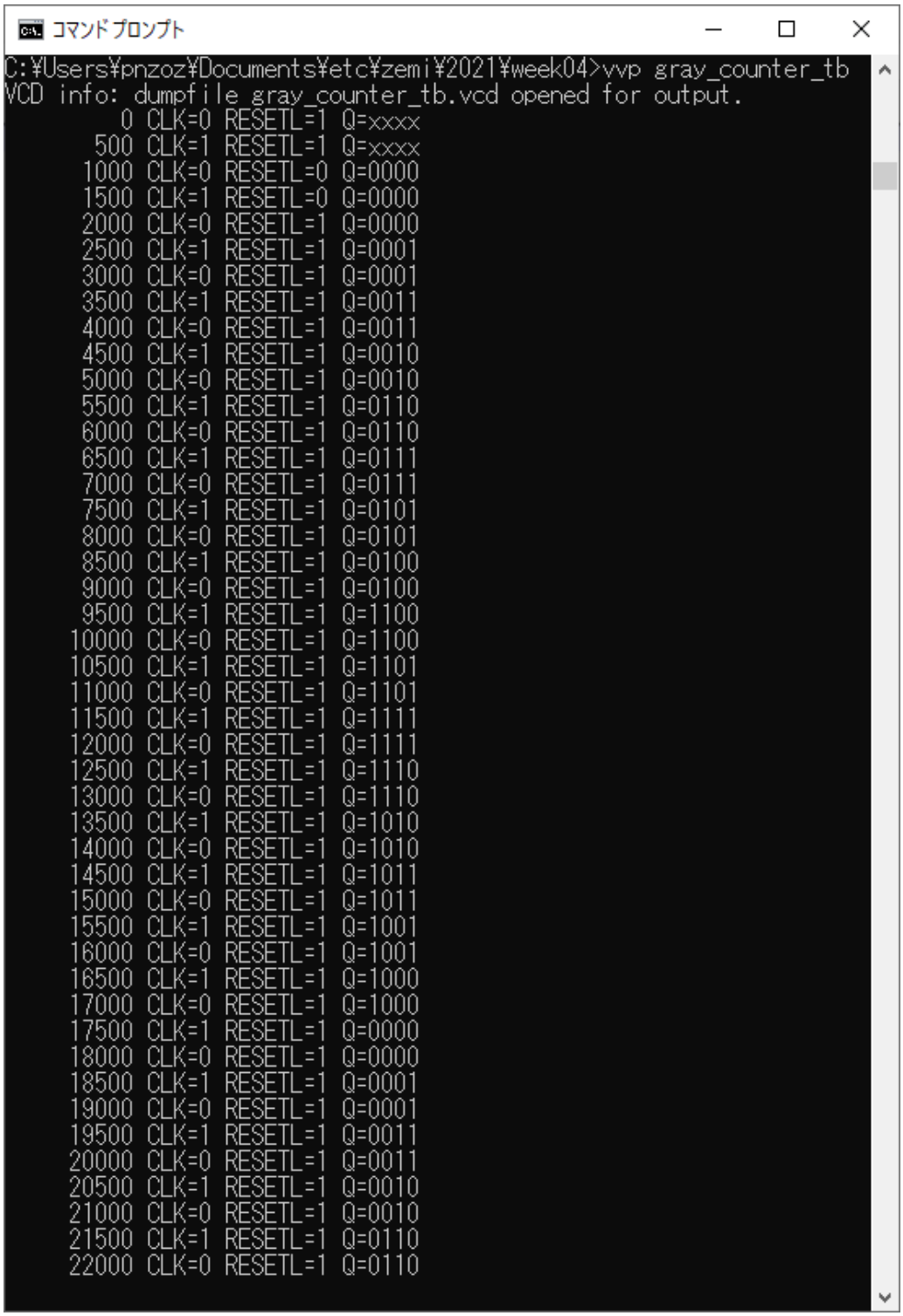

図 4. 出力結果の例(vvp)

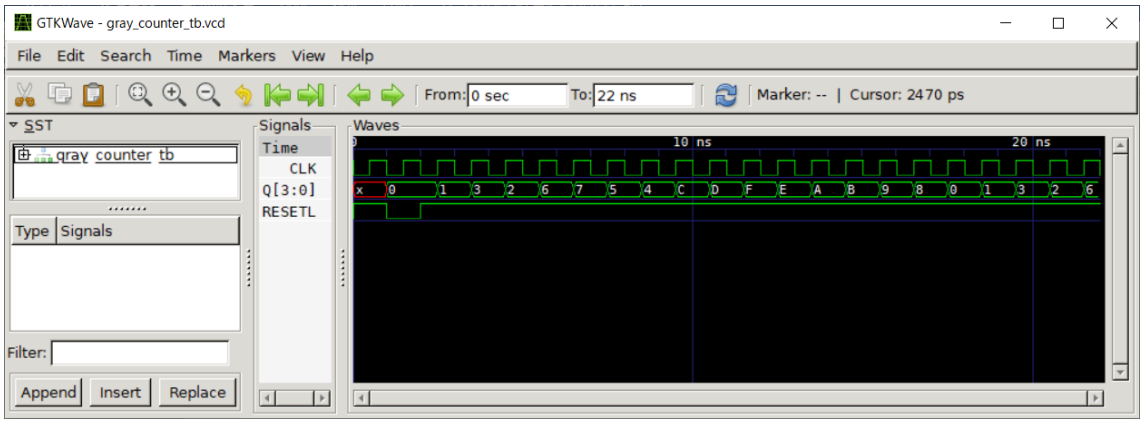

図 5. 波形表示例 (gtkwave)

参考文献

小林 優,「入門 Verilog HDL 記述」, CQ 出版社, 1996 年(初版)