コンピュータ科学実験レポート

坪井正太郎（101830245）

2020 年 10 月 22 日

はじめに

各実験で，シュミレータや論理合成のソフトウェアを使うために，以下の設定を行う。端末を終了した場合，再度 source コマンドを実行する。

ソースコード 1　設定の読み込み

```
$ ln -s /pub1/jikken/eda3/cadsetup.bash.altera ~/
$ source ~/cadsetup.bash.altera
```

各実験

1　実験 1

1.1　実験の目的，概要

本実験では，2 入力 1 出力のセレクタ回路を設計し，シュミレーションや論理合成を行って，FPGA ボードでの動作を確認する。これによって HDL による記述，シュミレータの使い方，回路を FPGA 上で動かすための方法を確認する。

入力は D0,D1(1bit データ)，S1(1bit セレクト信号)。出力は Y(1bit)。セレクタ信号が 0 ならば D0 のデータを，1 ならば D1 のデータを出力する。

ICE 計算機上の，ModelSim，Quartus，FPGA ボードは DE10-Lite を使用する。

1.2　実験方法

1.2.1　HDL での回路記述

以下のような回路記述を mux.v，テストベンチを test_mux.v として作成する。

ソースコード 2　mux21.v

```
module mux21 (S1, D0, D1, Y);
  input S1, D0, D1;
  output Y;
  assign Y = (~S1 & D0) | (S1 & D1);
endmodule
```

ソースコード 3　test_mux21.v

```
1  `timescale 1ns / 1ns
2
3  `include "mux21.v"
4
5  module test_mux21;
6    reg S1, D0, D1;
7    wire Y;
8    mux21 mux21a(S1, D0, D1, Y);
9
10   initial begin
11     S1=0; D0=0; D1=0;
12     #20 S1=0; D0=1; D1=0;
13     #20 S1=1; D0=1; D1=0;
14     #20 S1=0; D0=0; D1=0;
15     #80 $finish;
16   end
17 endmodule
```

1.2.2　機能レベルシュミレーション

作成したテストベンチをもとに，ModelSim で信号波形を出力する。入出力の値が仕様通りの真理値表と一致することを確認する。

1.2.3　コンパイル

以下の 2 ファイルを作成し，配置する。

ソースコード 4　mux21.qpf

```
1  # ------------------------------------------------------------------------- #
2  #
3  # Copyright (C) 1991-2013 Altera Corporation
4  # Your use of Altera Corporation's design tools, logic functions
5  # and other software and tools, and its AMPP partner logic
6  # functions, and any output files from any of the foregoing
7  # (including device programming or simulation files), and any
8  # associated documentation or information are expressly subject
9  # to the terms and conditions of the Altera Program License
10 # Subscription Agreement, Altera MegaCore Function License
11 # Agreement, or other applicable license agreement, including,
12 # without limitation, that your use is for the sole purpose of
13 # programming logic devices manufactured by Altera and sold by
14 # Altera or its authorized distributors. Please refer to the
15 # applicable agreement for further details.
16 #
17 # ------------------------------------------------------------------------- #
```

```
18 #
19 # Quartus II 64-Bit
20 # Version 13.0.1 Build 232 06/12/2013 Service Pack 1 SJ Web Edition
21 # Date created = 05:22:27 July 28, 2014
22 #
23 # ------------------------------------------------------------------------- #
24
25 QUARTUS_VERSION = "13.0"
26 DATE = "05:22:27 July 28, 2014"
27
28 # Revisions
29
30 PROJECT_REVISION = "mux21"
```

ソースコード 5　mux21.qsf

```
1  # Project-Wide Assignments
2  # ========================
3  set_global_assignment -name LAST_QUARTUS_VERSION "19.1.0 Lite Edition"
4
5  # Pin & Location Assignments
6  # ==========================
7  set_location_assignment PIN_C10 -to S1
8  set_location_assignment PIN_B8 -to D0
9  set_location_assignment PIN_A7 -to D1
10 set_location_assignment PIN_B11 -to Y
11
12 # Analysis & Synthesis Assignments
13 # =================================
14 set_global_assignment -name FAMILY "MAX 10"
15 set_global_assignment -name TOP_LEVEL_ENTITY mux21
16
17 # Fitter Assignments
18 # ==================
19 set_global_assignment -name DEVICE 10M50DAF484C6GES
20 set_global_assignment -name CYCLONEII_RESERVE_NCEO_AFTER_CONFIGURATION "USE AS REGULAR IO
       "
21 set_global_assignment -name RESERVE_ALL_UNUSED_PINS "AS INPUT TRI-STATED"
22
23 # ====================
24 set_global_assignment -name RESERVE_ALL_UNUSED_PINS_NO_OUTPUT_GND "AS INPUT TRI-STATED"
25
26 set_global_assignment -name SEED 1
27 set_global_assignment -name FIT_ONLY_ONE_ATTEMPT OFF
28 set_global_assignment -name OPTIMIZE_HOLD_TIMING "IO PATHS AND MINIMUM TPD PATHS"
29 set_global_assignment -name OPTIMIZE_MULTI_CORNER_TIMING OFF
30
```

```
set_global_assignment -name VERILOG_FILE mux21.v

set_global_assignment -name PARTITION_NETLIST_TYPE SOURCE -section_id Top
set_global_assignment -name PARTITION_FITTER_PRESERVATION_LEVEL PLACEMENT_AND_ROUTING -
    section_id Top
set_global_assignment -name PARTITION_COLOR 16764057 -section_id Top


set_global_assignment -name EDA_SIMULATION_TOOL "ModelSim-Altera (Verilog)"
set_global_assignment -name EDA_OUTPUT_DATA_FORMAT "VERILOG HDL" -section_id
    eda_simulation
set_global_assignment -name BOARD "MAX 10 DE10 - Lite"
set_instance_assignment -name PARTITION_HIERARCHY root_partition -to | -section_id Top
```

作成した回路記述を Quartus でコンパイルし，論理合成とレイアウトを行う。回路構成やロジックエレメント数，遅延時間について確認する。

1.2.4 FPGA ボードでの回路実現

計算機に FPGA ボードを接続し，dmesg コマンドで接続を確認する。以下のような設定ファイルを配置し，‘quartus-pgm mux21.cdf’ を実行して，接続した FPGA にダウンロードする。

ソースコード 6　mux21.cdf ダウンロード設定ファイル

```
/* Quartus II Version 6.0 Build 178 04/27/2006 SJ Web Edition */
JedecChain;
        FileRevision(JESD32A);
        DefaultMfr(6E);

        P ActionCode(Cfg)
                Device PartName(5CEBA4F23C7) Path("") File("mux21.sof") MfrSpec(OpMask
                    (1));

ChainEnd;

AlteraBegin;
        ChainType(JTAG);
AlteraEnd;
```

FPGA が仕様通りに動作するかを確認する。

1.3 実験結果

1.3.1 機能レベルシュミレーション

ModelSim で波形を作成した結果，以下のような波形になった。

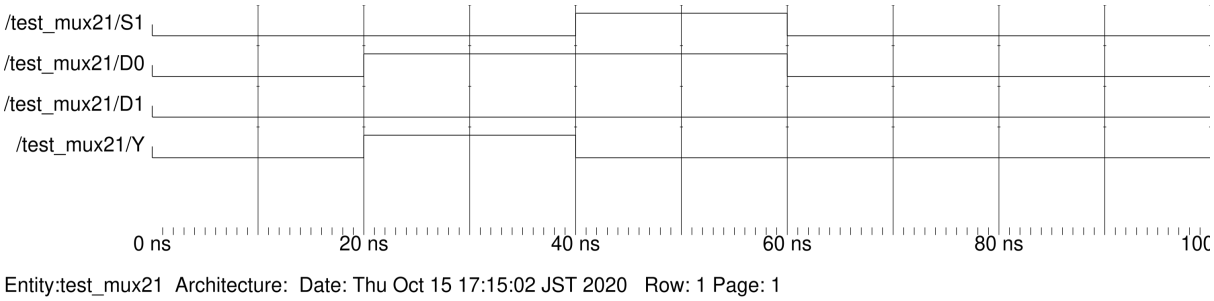

図 1　mux21 の波形

Y の出力値が期待どおりに 0 → 1 → 0 → 0 で推移し，S1 によって D0，D1 が切り替わっていることが確認できた。

1.3.2　論理合成

論理合成の結果，以下のような回路が作られた。

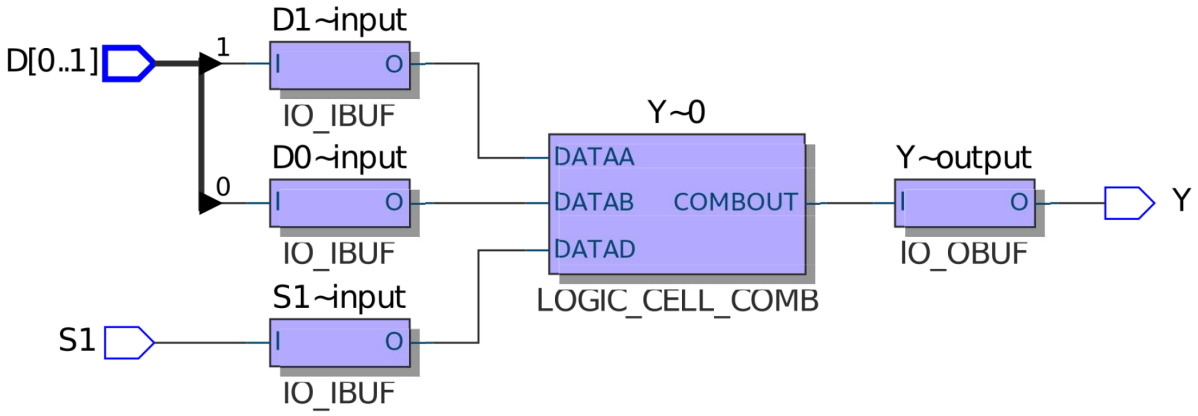

図 2　mux21 の回路

ロジックエレメント数は 2 だった。

回路全体の遅延時間は，以下のようになった。Y〜O の部分で遅延が大きくなった。

	Total	Incr	RF	Type	Fanout	Location	Element
1	7.071	7.071					data path
1	0.000	0.000			1	PIN_B8	D0
2	0.000	0.000	FF	IC	1	IOIBUF_X46_Y54_N29	D0~input|i
3	0.870	0.870	FF	CELL	1	IOIBUF_X46_Y54_N29	D0~input|o
4	3.968	3.098	FF	IC	1	LCCOMB_X50_Y53_N24	Y~0|datac
5	4.192	0.224	FF	CELL	1	LCCOMB_X50_Y53_N24	Y~0|combout
6	4.697	0.505	FF	IC	1	IOOBUF_X49_Y54_N9	Y~output|i
7	7.071	2.374	FF	CELL	1	IOOBUF_X49_Y54_N9	Y~output|o
8	7.071	0.000	FF	CELL	0	PIN_B11	Y

図 3　mux21 の遅延時間

1.3.3　FPGA ボードでの動作結果

スイッチによって S1 の値が変わり，それに対応して 2 つのボタンのうちどちらかの入力が LED で出力された。

1.4　考察

1.4.1　回路の HDL 記述

コード 2 では，input output で割り当てた入出力に，assign で単純な論理演算の結果を割り当てている。Y に $(\neg S1 \wedge D0)|(S1 \wedge D1)$ を割り当てることで，S1 の値によって D1 と D2 のどちらを出力するか選択することができている。(S1 が 0 の場合 D0 が，1 の場合 D1 が出力される)

コード 3 では，コード 2 から mux21 モジュールを読み込んで動作させながら，20ns 間隔で (S1=0, D0=0, D1=0) → (S1=0, D0=1, D1=0) → (S1=1, D0=1, D1=0) → (S1=0, D0=0, D1=0) と入力を遷移させている。

1.4.2　論理合成

回路の遅延は，処理が行われている Y〜O で大きくなっていた。実際，以下のように Y O にはゲートが配置され，他は入出力用のセルだった 。

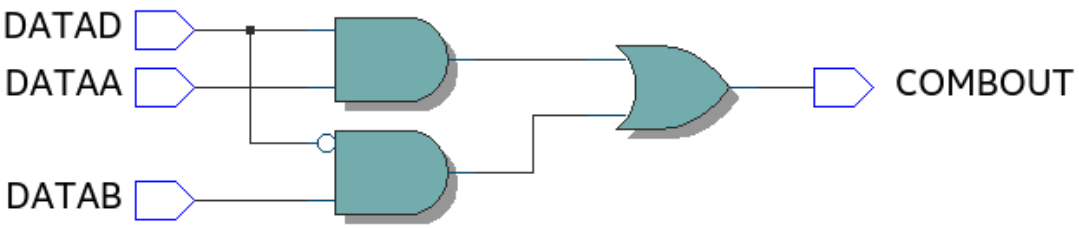

図 4　mux21 のゲート

また，この結果よりロジックエレメントの値はゲートの数に一致しないことも分かった。

この実験で，assign 句など，HDL 特有の逐次的な動作が理解できた。また，シュミレータの動作や，論理合成結果の参照が確認できた。

2　実験 2

2.1　実験の目的，概要

本実験では，16 ビット加算回路を組み合わせ回路で記述し，機能レベルシュミレーションと論理合成を実行する。これによって，出力値が仕様を満たしているかどうか，論理合成による回路構成を確認する。

入力は，x,y(16bit オペランド),y(1bit 桁上げ入力)。出力は，sum(16bit 演算結果),cout(1bit 桁上げ出力)。

ICE 計算機上の，ModelSim，Quartus を使用する。

組み合わせ回路の設計法と回路合成結果の確認を目的とする。

2.2　実験方法

2.2.1　回路の HDL 記述

以下のような回路記述を adder16.v，テストベンチを test_adder16.v として作成した。

ソースコード 7　adder16.v

```
1 module adder16 (x, y, cin, sum, cout);
2   input [15:0] x, y;
3   input cin;
4   output [15:0] sum;
5   output cout;
```

```
6
7   assign {cout,sum}=x+y+cin;
8 endmodule
```

ソースコード 8　test_adder16.v

```
1 ‘timescale 1ns / 1ns
2 ‘include "adder16.v"
3
4 module test;
5   reg [15:0] x, y;
6   reg cin;
7   wire [15:0] sum;
8   wire cout;
9
10  adder16 adder16a(x, y, cin, sum, cout);
11
12  always begin
13    #10 x = x + 100;
14  end
15
16  always begin
17    #5 y = y + 300;
18  end
19
20  initial begin
21    x = 0; y = 0; cin = 0;
22  end
23
24 endmodule
```

2.2.2　機能レベルシュミレーション

作成したテストベンチをもとに，ModelSim で信号波形を出力する。入出力の値が仕様通りの真理値表と一致することを確認する。

2.2.3　論理合成

以下の 2 ファイルを作成し，配置する。

ソースコード 9　adder16.qpf

```
1 # -------------------------------------------------------------------------- #
2 #
3 # Copyright (C) 1991-2013 Altera Corporation
4 # Your use of Altera Corporation's design tools, logic functions
5 # and other software and tools, and its AMPP partner logic
6 # functions, and any output files from any of the foregoing
```

```
# (including device programming or simulation files), and any
# associated documentation or information are expressly subject
# to the terms and conditions of the Altera Program License
# Subscription Agreement, Altera MegaCore Function License
# Agreement, or other applicable license agreement, including,
# without limitation, that your use is for the sole purpose of
# programming logic devices manufactured by Altera and sold by
# Altera or its authorized distributors. Please refer to the
# applicable agreement for further details.
#
# ---------------------------------------------------------------------------- #
#
# Quartus II 64-Bit
# Version 13.0.1 Build 232 06/12/2013 Service Pack 1 SJ Web Edition
# Date created = 05:22:27 July 28, 2014
#
# ---------------------------------------------------------------------------- #

QUARTUS_VERSION = "13.0"
DATE = "05:22:27 July 28, 2014"

# Revisions

PROJECT_REVISION = "adder16"
```

ソースコード 10　adder16.qsf

```
# Project-Wide Assignments
# ========================
set_global_assignment -name LAST_QUARTUS_VERSION "19.1.0 Lite Edition"

# Pin & Location Assignments
# ==========================
set_location_assignment PIN_C10 -to S1
set_location_assignment PIN_B8 -to D0
set_location_assignment PIN_A7 -to D1
set_location_assignment PIN_B11 -to Y

# Analysis & Synthesis Assignments
# ================================
set_global_assignment -name FAMILY "MAX 10"
set_global_assignment -name TOP_LEVEL_ENTITY adder16

# Fitter Assignments
# ==================
set_global_assignment -name DEVICE 10M50DAF484C6GES
```

```
20 set_global_assignment -name CYCLONEII_RESERVE_NCEO_AFTER_CONFIGURATION "USE AS REGULAR IO
       "
21 set_global_assignment -name RESERVE_ALL_UNUSED_PINS "AS INPUT TRI-STATED"
22
23 # ======================
24 set_global_assignment -name RESERVE_ALL_UNUSED_PINS_NO_OUTPUT_GND "AS INPUT TRI-STATED"
25
26 set_global_assignment -name SEED 1
27 set_global_assignment -name FIT_ONLY_ONE_ATTEMPT OFF
28 set_global_assignment -name OPTIMIZE_HOLD_TIMING "IO PATHS AND MINIMUM TPD PATHS"
29 set_global_assignment -name OPTIMIZE_MULTI_CORNER_TIMING OFF
30
31 set_global_assignment -name VERILOG_FILE adder16.v
32
33 set_global_assignment -name PARTITION_NETLIST_TYPE SOURCE -section_id Top
34 set_global_assignment -name PARTITION_FITTER_PRESERVATION_LEVEL PLACEMENT_AND_ROUTING -
       section_id Top
35 set_global_assignment -name PARTITION_COLOR 16764057 -section_id Top
36
37
38 set_global_assignment -name EDA_SIMULATION_TOOL "ModelSim-Altera (Verilog)"
39 set_global_assignment -name EDA_OUTPUT_DATA_FORMAT "VERILOG HDL" -section_id
       eda_simulation
40 set_global_assignment -name BOARD "MAX 10 DE10 - Lite"
41 set_instance_assignment -name PARTITION_HIERARCHY root_partition -to | -section_id Top
```

作成した回路記述を Quartus でコンパイルし，論理合成とレイアウトを行う。回路構成やロジックエレメント数，遅延時間について確認する。

2.3 実験結果

2.3.1 機能レベルシュミレーション

ModelSim での入出力波形は以下のようになった。(キャプチャーでは値が見えないので，赤数字で 10 進の値を追記した。)

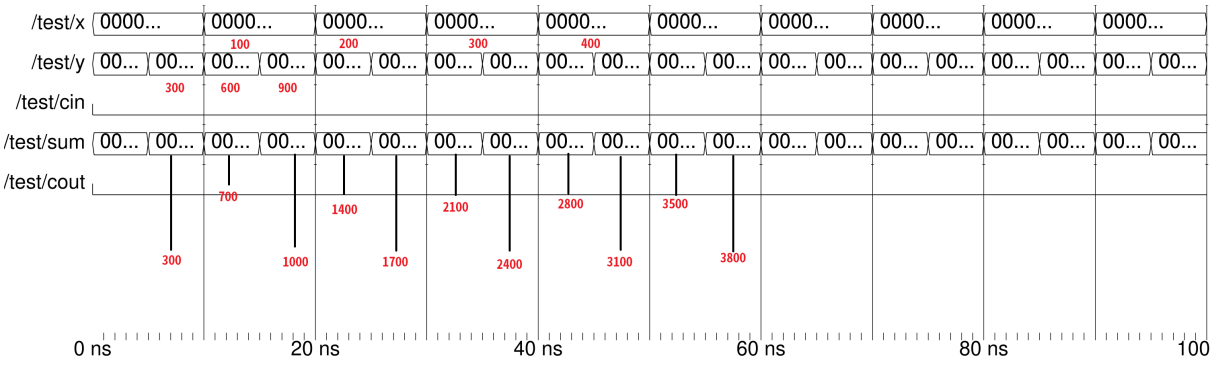

図 5　adder16 の波形

足し算の結果は正しく出力されていることが確認できた。キャプチャの範囲の外で，cout に桁上げが出力されていることも確認できた。

2.3.2　論理合成

論理合成の結果，以下のような回路が作られた。

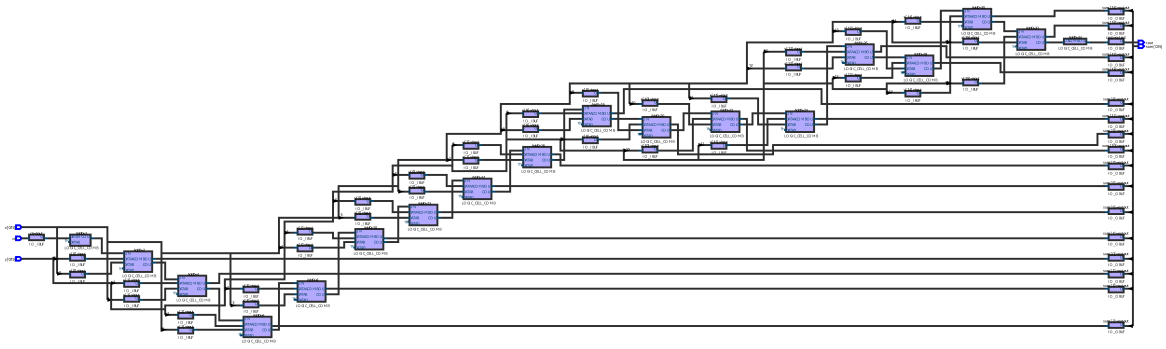

図 6　adder16 の回路

ロジックエレメント数は 19 だった。

回路全体の遅延時間は，以下のようになった。

	Total	Incr	RF	Type	Fanout	Location	Element
1	10.920	10.920					data path
1	0.000	0.000			1	PIN_A3	y[4]
2	0.000	0.000	FF	IC	1	IOIBUF_X26_Y39_N8	y[4]~input|i
3	0.900	0.900	FF	CELL	1	IOIBUF_X26_Y39_N8	y[4]~input|o
4	5.002	4.102	FF	IC	2	LCCOMB_X1_Y35_N24	Add0~10|datab
5	5.314	0.312	FR	CELL	1	LCCOMB_X1_Y35_N24	Add0~10|cout
6	5.314	0.000	RR	IC	2	LCCOMB_X1_Y35_N26	Add0~12|cin
7	5.361	0.047	RF	CELL	1	LCCOMB_X1_Y35_N26	Add0~12|cout
8	5.361	0.000	FF	IC	2	LCCOMB_X1_Y35_N28	Add0~14|cin
9	5.408	0.047	FR	CELL	1	LCCOMB_X1_Y35_N28	Add0~14|cout
10	5.408	0.000	RR	IC	2	LCCOMB_X1_Y35_N30	Add0~16|cin
11	5.455	0.047	RF	CELL	1	LCCOMB_X1_Y35_N30	Add0~16|cout
12	5.455	0.000	FF	IC	2	LCCOMB_X1_Y34_N0	Add0~18|cin
13	5.502	0.047	FR	CELL	1	LCCOMB_X1_Y34_N0	Add0~18|cout
14	5.502	0.000	RR	IC	2	LCCOMB_X1_Y34_N2	Add0~20|cin
15	5.549	0.047	RF	CELL	1	LCCOMB_X1_Y34_N2	Add0~20|cout
16	5.549	0.000	FF	IC	2	LCCOMB_X1_Y34_N4	Add0~22|cin
17	5.596	0.047	FR	CELL	1	LCCOMB_X1_Y34_N4	Add0~22|cout
18	5.596	0.000	RR	IC	2	LCCOMB_X1_Y34_N6	Add0~24|cin
19	5.643	0.047	RF	CELL	1	LCCOMB_X1_Y34_N6	Add0~24|cout
20	5.643	0.000	FF	IC	2	LCCOMB_X1_Y34_N8	Add0~26|cin
21	5.690	0.047	FR	CELL	1	LCCOMB_X1_Y34_N8	Add0~26|cout
22	5.690	0.000	RR	IC	2	LCCOMB_X1_Y34_N10	Add0~28|cin
23	5.737	0.047	RF	CELL	1	LCCOMB_X1_Y34_N10	Add0~28|cout
24	5.737	0.000	FF	IC	2	LCCOMB_X1_Y34_N12	Add0~30|cin
25	5.784	0.047	FR	CELL	1	LCCOMB_X1_Y34_N12	Add0~30|cout
26	5.784	0.000	RR	IC	2	LCCOMB_X1_Y34_N14	Add0~32|cin
27	6.279	0.495	RR	CELL	1	LCCOMB_X1_Y34_N14	Add0~32|combout
28	8.389	2.110	RR	IC	1	IOOBUF_X22_Y39_N16	sum[15]~output|i
29	10.920	2.531	RR	CELL	1	IOOBUF_X22_Y39_N16	sum[15]~output|o
30	10.920	0.000	RR	CELL	0	PIN_B2	sum[15]

図 7　adder16 の遅延時間

2.4　考察

2.4.1　回路の HDL 記述

コード 7 では，16bit の入出力を定義して，assign 句で出力に演算結果を割り当てている。17bit 連接に対して足し算の結果を割り当てているため，結果が 16bit に収まらない場合，cout の値は 1 になる。

コード 8 ではテストベンチとして，入力をすべてゼロに初期化し，10ns ごとに x を 100 づつ加算し，5ns ごとに y を 300 づつ加算している。

2.4.2 論理合成

合成結果より，一桁ごとに桁上りを含めた 3 入力の全加算器で加算している。単純な加算機として実装されており，当然桁上りの先読みも行われていないため，そのような記述をすることで，遅延速度が小さくなるのではないかと考える。

遅延時間の表 7 より，最長の遅延時間が発生するパスは，最初の全加算器ではなく，(半加算器も含めて)6つめの全加算器への入力から始まっている。これは，この段階までは配線長と inputIO による遅延よりも，加算機による遅延のほうが少ないということではないかと予想する。

3 実験 3

3.1 実験の目的，概要

本実験では，16 ビット加算回路を順序回路で記述し，機能レベルシュミレーションと論理合成を実行する。これによって，出力値が仕様を満たしているかどうか，論理合成による回路構成を確認する。

入力は，clk(1bit クロック),reset(1bit リセット信号),x,y(16bit オペランド),y(1bit 桁上げ入力)。出力は，sum(16bit 演算結果),cout(1bit 桁上げ出力)。ただし，入力値の演算結果は次のクロックの立ち上がりで出力に反映させる。

ICE 計算機上の，ModelSim，Quartus を使用する。

順序回路の設計法と回路合成結果の確認を行い，実験 2 の結果と比較することで，等価な回路が設計によって，どのように異なるかを考察することを目的とする。

3.2 実験方法

3.2.1 回路の HDL 記述

以下のような回路記述を adder16s.v，テストベンチを test_adder16s.v として作成する。

ソースコード 11　adder16s.v

```
1  module adder16s (
2    clk,reset,x,y,cin,sum,cout
3  );
4  input [15:0] x,y;
5  input clk, reset, cin;
6  output [15:0] sum;
7  output cout;
8
9  reg [15:0] r0, r1;
10
11 assign {cout, sum}=r0+r1+cin;
12
13 always @(posedge clk or negedge reset) begin
14   if(reset==1'b0) begin
15     r0<=0;r1<=0;
```

```
16   end
17   else begin
18     r0<=x;r1<=y;
19   end
20 end
21
22 endmodule //adder16s
```

ソースコード 12　test_adder16s.v

```
1
2 `timescale 1ns / 1ns
3
4 `include "adder16s.v"
5
6 module test ;
7         reg reset, clk, cin;
8         reg [15:0] x, y;
9
10        wire [15:0] sum;
11        wire cout;
12
13        adder16s adder16s(clk, reset, x, y, cin, sum, cout);
14
15        always begin
16                #5 clk = ~clk;
17        end
18
19        always begin
20                #8 x = x + 100;
21                        y = y + 200;
22        end
23
24        initial begin
25
26           reset = 1; clk = 0; x = 0; y = 0; cin = 0 ;
27           #20 reset = 0;
28           #20 reset = 1;
29        end
30
31 endmodule
```

3.2.2　機能レベルシュミレーション

作成したテストベンチをもとに，ModelSim で信号波形を出力する。入出力の値が仕様通りの真理値表と一致することを確認する。

3.2.3 論理合成

以下の 2 ファイルを作成し，配置する。

ソースコード 13　adder16s.qpf

```
# -------------------------------------------------------------------------- #
#
# Copyright (C) 1991-2013 Altera Corporation
# Your use of Altera Corporation's design tools, logic functions
# and other software and tools, and its AMPP partner logic
# functions, and any output files from any of the foregoing
# (including device programming or simulation files), and any
# associated documentation or information are expressly subject
# to the terms and conditions of the Altera Program License
# Subscription Agreement, Altera MegaCore Function License
# Agreement, or other applicable license agreement, including,
# without limitation, that your use is for the sole purpose of
# programming logic devices manufactured by Altera and sold by
# Altera or its authorized distributors. Please refer to the
# applicable agreement for further details.
#
# -------------------------------------------------------------------------- #
#
# Quartus II 64-Bit
# Version 13.0.1 Build 232 06/12/2013 Service Pack 1 SJ Web Edition
# Date created = 05:22:27 July 28, 2014
#
# -------------------------------------------------------------------------- #

QUARTUS_VERSION = "13.0"
DATE = "05:22:27 July 28, 2014"

# Revisions

PROJECT_REVISION = "adder16s"
```

ソースコード 14　adder16s.qsf

```
# Project-Wide Assignments
# ========================
set_global_assignment -name LAST_QUARTUS_VERSION "19.1.0 Lite Edition"

# Pin & Location Assignments
# ==========================
set_location_assignment PIN_C10 -to S1
set_location_assignment PIN_B8 -to D0
```

```
set_location_assignment PIN_A7 -to D1
set_location_assignment PIN_B11 -to Y

# Analysis & Synthesis Assignments
# ================================
set_global_assignment -name FAMILY "MAX 10"
set_global_assignment -name TOP_LEVEL_ENTITY adder16s

# Fitter Assignments
# ==================
set_global_assignment -name DEVICE 10M50DAF484C6GES
set_global_assignment -name CYCLONEII_RESERVE_NCEO_AFTER_CONFIGURATION "USE AS REGULAR IO
    "
set_global_assignment -name RESERVE_ALL_UNUSED_PINS "AS INPUT TRI-STATED"

# ====================
set_global_assignment -name RESERVE_ALL_UNUSED_PINS_NO_OUTPUT_GND "AS INPUT TRI-STATED"

set_global_assignment -name SEED 1
set_global_assignment -name FIT_ONLY_ONE_ATTEMPT OFF
set_global_assignment -name OPTIMIZE_HOLD_TIMING "IO PATHS AND MINIMUM TPD PATHS"
set_global_assignment -name OPTIMIZE_MULTI_CORNER_TIMING OFF

set_global_assignment -name VERILOG_FILE adder16s.v

set_global_assignment -name PARTITION_NETLIST_TYPE SOURCE -section_id Top
set_global_assignment -name PARTITION_FITTER_PRESERVATION_LEVEL PLACEMENT_AND_ROUTING -
    section_id Top
set_global_assignment -name PARTITION_COLOR 16764057 -section_id Top


set_global_assignment -name EDA_SIMULATION_TOOL "ModelSim-Altera (Verilog)"
set_global_assignment -name EDA_OUTPUT_DATA_FORMAT "VERILOG HDL" -section_id
    eda_simulation
set_global_assignment -name BOARD "MAX 10 DE10 - Lite"

set_instance_assignment -name PARTITION_HIERARCHY root_partition -to | -section_id Top
```

作成した回路記述を Quartus でコンパイルし，論理合成とレイアウトを行う。回路構成やロジックエレメント数，遅延時間について確認する。

3.3 実験結果

3.3.1 機能レベルシュミレーション

ModelSim で波形を作成した結果，以下のような波形になった。

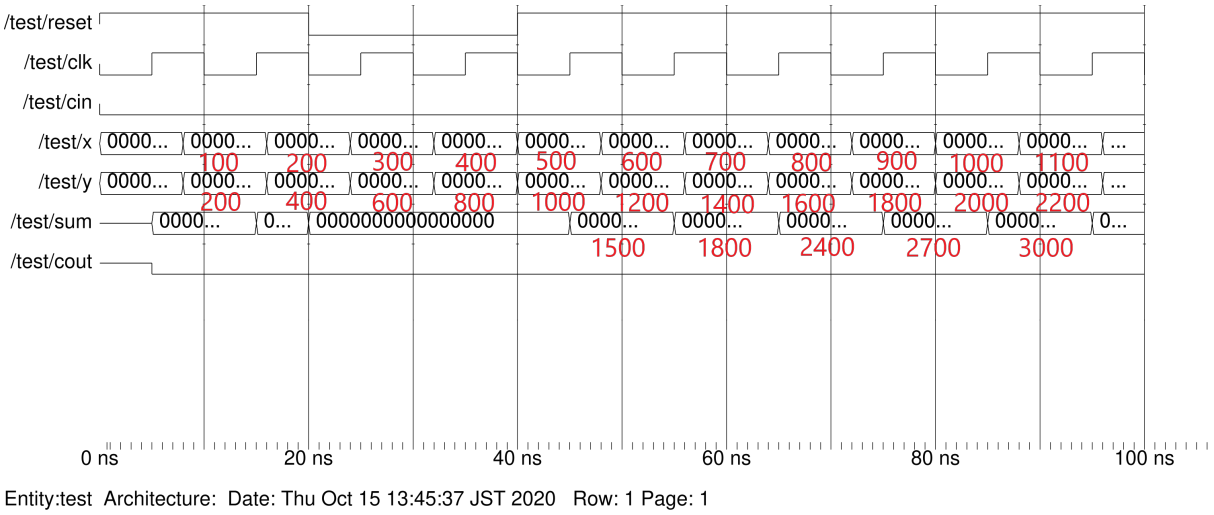

図 8　adder16s の波形

3.3.2　論理合成

論理合成の結果，以下のような回路が作られた。

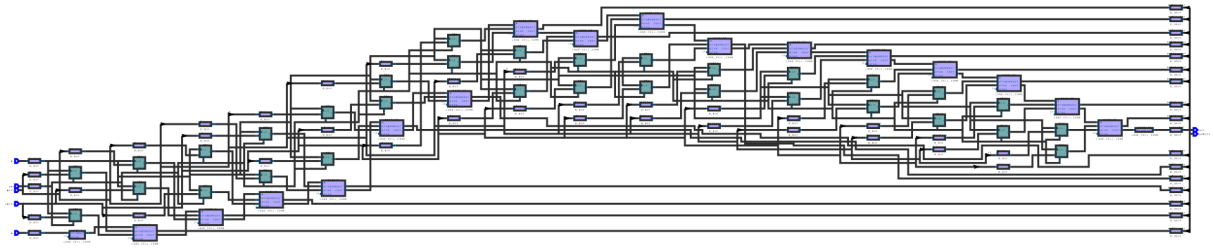

図 9　adder16s の回路

ロジックエレメント数は 35 だった。

回路全体の遅延時間は，以下のようになった。

	Total	Incr	RF	Type	Fanout	Location	Element
1	9.479	9.479					data path
1	0.000	0.000			1	PIN_H4	cin
2	0.000	0.000	FF	IC	1	IOIBUF_X0_Y35_N1	cin~input|i
3	0.860	0.860	FF	CELL	1	IOIBUF_X0_Y35_N1	cin~input|o
4	3.495	2.635	FF	IC	1	LCCOMB_X1_Y35_N14	Add0~1|datab
5	3.877	0.382	FF	CELL	1	LCCOMB_X1_Y35_N14	Add0~1|cout
6	3.877	0.000	FF	IC	2	LCCOMB_X1_Y35_N16	Add0~2|cin
7	3.924	0.047	FR	CELL	1	LCCOMB_X1_Y35_N16	Add0~2|cout
8	3.924	0.000	RR	IC	2	LCCOMB_X1_Y35_N18	Add0~4|cin
9	3.971	0.047	RF	CELL	1	LCCOMB_X1_Y35_N18	Add0~4|cout
10	3.971	0.000	FF	IC	2	LCCOMB_X1_Y35_N20	Add0~6|cin
11	4.018	0.047	FR	CELL	1	LCCOMB_X1_Y35_N20	Add0~6|cout
12	4.018	0.000	RR	IC	2	LCCOMB_X1_Y35_N22	Add0~8|cin
13	4.065	0.047	RF	CELL	1	LCCOMB_X1_Y35_N22	Add0~8|cout
14	4.065	0.000	FF	IC	2	LCCOMB_X1_Y35_N24	Add0~10|cin
15	4.112	0.047	FR	CELL	1	LCCOMB_X1_Y35_N24	Add0~10|cout
16	4.112	0.000	RR	IC	2	LCCOMB_X1_Y35_N26	Add0~12|cin
17	4.159	0.047	RF	CELL	1	LCCOMB_X1_Y35_N26	Add0~12|cout
18	4.159	0.000	FF	IC	2	LCCOMB_X1_Y35_N28	Add0~14|cin
19	4.206	0.047	FR	CELL	1	LCCOMB_X1_Y35_N28	Add0~14|cout
20	4.206	0.000	RR	IC	2	LCCOMB_X1_Y35_N30	Add0~16|cin
21	4.701	0.495	RR	CELL	1	LCCOMB_X1_Y35_N30	Add0~16|combout
22	7.008	2.307	RR	IC	1	IOOBUF_X24_Y39_N16	sum[7]~output|i
23	9.479	2.471	RR	CELL	1	IOOBUF_X24_Y39_N16	sum[7]~output|o
24	9.479	0.000	RR	CELL	0	PIN_F7	sum[7]

図 10　adder16s の遅延時間

3.4　考察

3.4.1　回路の HDL 記述

コード 11 では，クロックの立ち上がりで，定義した 2 つの 16bit レジスタに入力 x，y の値を代入している。その結果がコード 7 と同様に assign されているので，クロックに同期して動作すること以外は，7 と同じである。

また，リセット信号の立ち下がりでレジスタの値はリセットされる。

コード 12 ではテストベンチとして，5ns ごとにクロック信号を反転している。(10ns ごとに回路が動作する)

また，8ns ごとに x に 100 を加算し，y に 200 加算している。

3.4.2　機能レベルシュミレーション

クロック信号に対応して出力されていた。また，リセット信号も正しく動作している。

3.4.3 論理合成 (実験 2 との比較)

合成結果では，全加算器列と桁数*2 個のフリップフロップによって回路が作られていた。これによって，組み合わせ回路のときよりもロジックエレメント数は増加している。

一方で，全体の遅延時間は 10% 程度減少している。回路内で，全加算器の数は両方 16 個で，加算器のみの遅延は図 10 より 0.047 で，図 7 の遅延時間と同じである。

しかし，順序回路では入力の変化に関係なく，クロックによってフリップフロップに保存される予定の，その瞬間の入力値が読み取られる。つまり，クロックが入力されるまで，入力の値はフリップフロップのすぐ前に待機していると言える。その結果として，実験 3 では最長パスが 1 桁目の全加算器から始まっている。このような原因で，順序回路のほうが遅延は小さいと考えられる

4 実験課題 1

4.1 実験の目的，概要

1 桁の BCD コードを出力する BCD カウンタを設計し，それをもとに 2 桁の BCD コード出力する BCD カウンタを階層設計によって設計する。設計した 2 つの回路で，それぞれ機能レベルシュミレーションと論理合成を実行し，入出力の正しさと合成による結果を確認する。

ICE 計算機上の，ModelSim，Quartus を使用する。

この実験で，簡単な順序回路の設計方法と，それをもとにした階層設計の方法を実践し，習得することを目的にする。

4.2 実験方法

課題 1-1

4.2.1 回路の HDL 記述

以下のような回路記述を bcd1.v，テストベンチを test_bcd1.v として作成する。

ソースコード 15　bcd1.v

```
1  module bcd1 (
2      clk, reset, x, bcd1_out
3  );
4  input clk, reset, x;
5  output [3:0] bcd1_out;
6
7  reg [3:0] r0;
8
9  assign bcd1_out=r0;
10
11 always @(posedge clk or negedge reset) begin
12     if (reset == 1'b0) begin
13         r0<=4'b0000;
```

```
14      end else begin
15          if (x == 1'b1) begin
16              if (r0 < 9) begin
17                  r0<=r0+1'b1;
18              end else begin
19                  r0<=4'b0000;
20              end
21          end
22      end
23  end
24
25  endmodule
```

ソースコード 16　test_bcd1.v

```
1  /* *
2   * test_bcd1.v *
3   * 1桁のBCD カウンタのテストベンチ *
4   * */
5
6  `timescale 1ns / 1ns // シミュレーションの単位時間 / 精度
7                                                    // 1ns = 1/1000000000 sec
8  `include "bcd1.v" // bcd1.v の読み込み
9
10 module test ;
11   /*** bcd1 の入力値格納用のレジスタ ****/
12   reg reset, clk, x;
13
14   /*** bcd1 の出力観測用の信号線 ****/
15   wire [3:0] bcd1_out;
16
17   /*** bcd1 の実体化 ***/
18   bcd1 bcd1(clk, reset, x, bcd1_out);
19
20   always begin
21     #5 clk = ~clk;
22   end
23
24   always begin
25     #15 x = ~x;
26   end
27
28   initial begin
29     reset = 1; clk = 0; x = 0;
30
31     // 20 単位時間 (20 ns)後
32     #20 reset = 0;
```

```
33
34      // 更に 20 単位時間 (20 ns)後
35      #20 reset = 1;
36
37    end
38
39 endmodule // test
```

4.2.2 機能レベルシュミレーション

作成したテストベンチをもとに，ModelSim で信号波形を出力する。入出力の値が仕様通りの真理値表と一致することを確認する。

4.2.3 論理合成

以下の 2 ファイルを作成し，配置する。

ソースコード 17　bcd1.qpf

```
1  # -------------------------------------------------------------------------- #
2  #
3  # Copyright (C) 1991-2013 Altera Corporation
4  # Your use of Altera Corporation's design tools, logic functions
5  # and other software and tools, and its AMPP partner logic
6  # functions, and any output files from any of the foregoing
7  # (including device programming or simulation files), and any
8  # associated documentation or information are expressly subject
9  # to the terms and conditions of the Altera Program License
10 # Subscription Agreement, Altera MegaCore Function License
11 # Agreement, or other applicable license agreement, including,
12 # without limitation, that your use is for the sole purpose of
13 # programming logic devices manufactured by Altera and sold by
14 # Altera or its authorized distributors. Please refer to the
15 # applicable agreement for further details.
16 #
17 # -------------------------------------------------------------------------- #
18 #
19 # Quartus II 64-Bit
20 # Version 13.0.1 Build 232 06/12/2013 Service Pack 1 SJ Web Edition
21 # Date created = 05:22:27 July 28, 2014
22 #
23 # -------------------------------------------------------------------------- #
24
25 QUARTUS_VERSION = "13.0"
26 DATE = "05:22:27 July 28, 2014"
27
28 # Revisions
```

```
29
30 PROJECT_REVISION = "bcd1"
```

ソースコード 18　bcd1.qsf

```
1  # Project-Wide Assignments
2  # ========================
3  set_global_assignment -name LAST_QUARTUS_VERSION "19.1.0 Lite Edition"
4
5  # Pin & Location Assignments
6  # ==========================
7  set_location_assignment PIN_C10 -to S1
8  set_location_assignment PIN_B8 -to D0
9  set_location_assignment PIN_A7 -to D1
10 set_location_assignment PIN_B11 -to Y
11
12 # Analysis & Synthesis Assignments
13 # ================================
14 set_global_assignment -name FAMILY "MAX 10"
15 set_global_assignment -name TOP_LEVEL_ENTITY bcd1
16
17 # Fitter Assignments
18 # ==================
19 set_global_assignment -name DEVICE 10M50DAF484C6GES
20 set_global_assignment -name CYCLONEII_RESERVE_NCEO_AFTER_CONFIGURATION "USE AS REGULAR IO
       "
21 set_global_assignment -name RESERVE_ALL_UNUSED_PINS "AS INPUT TRI-STATED"
22
23 # ======================
24 set_global_assignment -name RESERVE_ALL_UNUSED_PINS_NO_OUTPUT_GND "AS INPUT TRI-STATED"
25
26 set_global_assignment -name SEED 1
27 set_global_assignment -name FIT_ONLY_ONE_ATTEMPT OFF
28 set_global_assignment -name OPTIMIZE_HOLD_TIMING "IO PATHS AND MINIMUM TPD PATHS"
29 set_global_assignment -name OPTIMIZE_MULTI_CORNER_TIMING OFF
30
31 set_global_assignment -name VERILOG_FILE bcd1.v
32
33 set_global_assignment -name PARTITION_NETLIST_TYPE SOURCE -section_id Top
34 set_global_assignment -name PARTITION_FITTER_PRESERVATION_LEVEL PLACEMENT_AND_ROUTING -
       section_id Top
35 set_global_assignment -name PARTITION_COLOR 16764057 -section_id Top
36
37
38 set_global_assignment -name EDA_SIMULATION_TOOL "ModelSim-Altera (Verilog)"
39 set_global_assignment -name EDA_OUTPUT_DATA_FORMAT "VERILOG HDL" -section_id
       eda_simulation
```

```
set_global_assignment -name BOARD "MAX 10 DE10 - Lite"
set_instance_assignment -name PARTITION_HIERARCHY root_partition -to | -section_id Top
```

作成した回路記述を Quartus でコンパイルし，論理合成とレイアウトを行う。回路構成やロジックエレメント数，遅延時間について確認する。

課題 1-2

以下のような回路記述を bcd2.v，テストベンチを test_bcd2.v として作成する。

ソースコード 19　bcd2.v

```
`include "bcd1.v"

module bcd2 (clk, reset, x, bcd2_out);
  input clk, reset, x;
  output [7:0] bcd2_out;

  /**** 1 桁目用の信号線 ****/
  wire bcd1a_clk;
  wire bcd1a_reset;
  wire bcd1a_x;
  wire [3:0] bcd1a_out;

  /**** 2 桁目用の信号線 ****/
  wire bcd1b_clk;
  wire bcd1b_reset;
  wire bcd1b_x;
  wire [3:0] bcd1b_out;

  /**** clk 信号の分配 ****/
  assign bcd1a_clk = clk;
  assign bcd1b_clk = clk;

  /**** reset 信号の分配 ****/
  assign bcd1a_reset = reset;
  assign bcd1b_reset = reset;

  /**** 1 桁目のカウントアップ信号 ****/
  assign bcd1a_x = x;

  /*** 各桁の BCD カウンタの実体化 ***/
  bcd1 bcd1a(bcd1a_clk, bcd1a_reset, bcd1a_x, bcd1a_out);
  bcd1 bcd1b(bcd1b_clk, bcd1b_reset, bcd1b_x, bcd1b_out);

  /**** 2 桁目のカウントアップ信号 ****/
  assign bcd1b_x = (bcd1a_out == 4'b1001) & x;
```

```
36
37   /**** 2 桁の BCD カウンタの出力のアサイン ****/
38   assign bcd2_out = {bcd1b_out, bcd1a_out};
39
40 endmodule // bcd2
```

ソースコード 20　test_bcd2.v

```
1  /* *
2   * test_bcd2.v *
3   * 2桁のBCD カウンタのテストベンチ *
4   * */
5
6  `timescale 1ns / 1ns // シミュレーションの単位時間 / 精度
7                                                    // 1ns = 1/1000000000 sec
8  `include "bcd2.v" // bcd2.v の読み込み
9
10 module test ;
11   /*** bcd2 の入力値格納用のレジスタ ****/
12   reg reset, clk, x;
13
14   /*** bcd2 の出力観測用の信号線 ****/
15   wire [7:0] bcd2_out;
16
17   /*** bcd2 の実体化 ***/
18   bcd2 bcd2(clk, reset, x, bcd2_out);
19
20   always begin
21     #5 clk = ~clk;
22   end
23
24   always begin
25     #15 x = ~x;
26   end
27
28   initial begin
29     reset = 1; clk = 0; x = 0;
30
31     // 20 単位時間 (20 ns)後
32     #20 reset = 0;
33
34     // 更に 20 単位時間 (20 ns)後
35     #20 reset = 1;
36
37   end
38
39 endmodule // test
```

4.2.4 機能レベルシュミレーション

作成したテストベンチをもとに，ModelSim で信号波形を出力する。入出力の値が仕様通りの真理値表と一致することを確認する。

4.2.5 論理合成

以下の 2 ファイルを作成し，配置する。

ソースコード 21　bcd2.qpf

```
1  # -------------------------------------------------------------------------- #
2  #
3  # Copyright (C) 1991-2013 Altera Corporation
4  # Your use of Altera Corporation's design tools, logic functions
5  # and other software and tools, and its AMPP partner logic
6  # functions, and any output files from any of the foregoing
7  # (including device programming or simulation files), and any
8  # associated documentation or information are expressly subject
9  # to the terms and conditions of the Altera Program License
10 # Subscription Agreement, Altera MegaCore Function License
11 # Agreement, or other applicable license agreement, including,
12 # without limitation, that your use is for the sole purpose of
13 # programming logic devices manufactured by Altera and sold by
14 # Altera or its authorized distributors. Please refer to the
15 # applicable agreement for further details.
16 #
17 # -------------------------------------------------------------------------- #
18 #
19 # Quartus II 64-Bit
20 # Version 13.0.1 Build 232 06/12/2013 Service Pack 1 SJ Web Edition
21 # Date created = 05:22:27 July 28, 2014
22 #
23 # -------------------------------------------------------------------------- #
24
25 QUARTUS_VERSION = "13.0"
26 DATE = "05:22:27 July 28, 2014"
27
28 # Revisions
29
30 PROJECT_REVISION = "bcd2"
```

ソースコード 22　bcd2.qsf

```
1  # Project-Wide Assignments
2  # ========================
```

```
3 set_global_assignment -name LAST_QUARTUS_VERSION "19.1.0 Lite Edition"
4
5 # Pin & Location Assignments
6 # ==========================
7 set_location_assignment PIN_C10 -to S1
8 set_location_assignment PIN_B8 -to D0
9 set_location_assignment PIN_A7 -to D1
10 set_location_assignment PIN_B11 -to Y
11
12 # Analysis & Synthesis Assignments
13 # ================================
14 set_global_assignment -name FAMILY "MAX 10"
15 set_global_assignment -name TOP_LEVEL_ENTITY bcd2
16
17 # Fitter Assignments
18 # ==================
19 set_global_assignment -name DEVICE 10M50DAF484C6GES
20 set_global_assignment -name CYCLONEII_RESERVE_NCEO_AFTER_CONFIGURATION "USE AS REGULAR IO
       "
21 set_global_assignment -name RESERVE_ALL_UNUSED_PINS "AS INPUT TRI-STATED"
22
23 # ======================
24 set_global_assignment -name RESERVE_ALL_UNUSED_PINS_NO_OUTPUT_GND "AS INPUT TRI-STATED"
25
26 set_global_assignment -name SEED 1
27 set_global_assignment -name FIT_ONLY_ONE_ATTEMPT OFF
28 set_global_assignment -name OPTIMIZE_HOLD_TIMING "IO PATHS AND MINIMUM TPD PATHS"
29 set_global_assignment -name OPTIMIZE_MULTI_CORNER_TIMING OFF
30
31 set_global_assignment -name VERILOG_FILE bcd2.v
32
33 set_global_assignment -name PARTITION_NETLIST_TYPE SOURCE -section_id Top
34 set_global_assignment -name PARTITION_FITTER_PRESERVATION_LEVEL PLACEMENT_AND_ROUTING -
       section_id Top
35 set_global_assignment -name PARTITION_COLOR 16764057 -section_id Top
36
37
38 set_global_assignment -name EDA_SIMULATION_TOOL "ModelSim-Altera (Verilog)"
39 set_global_assignment -name EDA_OUTPUT_DATA_FORMAT "VERILOG HDL" -section_id
       eda_simulation
40 set_global_assignment -name BOARD "MAX 10 DE10 - Lite"
41 set_instance_assignment -name PARTITION_HIERARCHY root_partition -to | -section_id Top
```

作成した回路記述を Quartus でコンパイルし，論理合成とレイアウトを行う。回路構成やロジックエレメント数，遅延時間について確認する。

4.3 実験結果

課題 1-1

4.3.1 機能レベルシュミレーション

ModelSim で波形を作成した結果，以下のような波形になった。

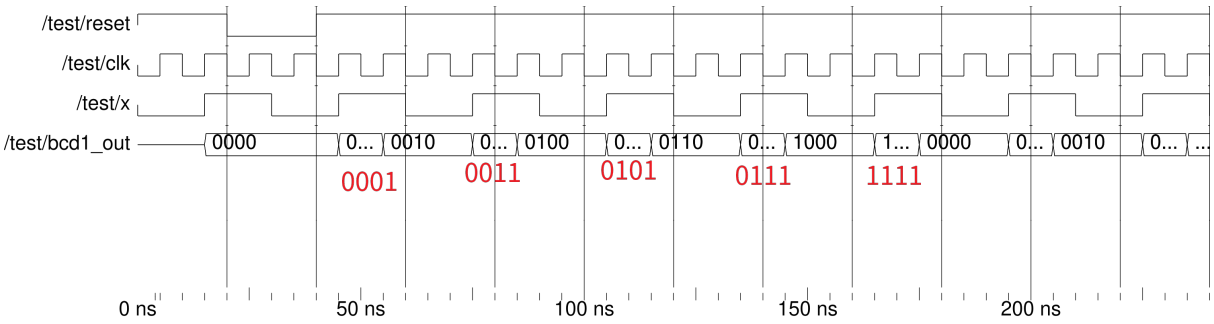

図 11　bcd1 の波形

bcd1_out の値は期待通り 10 進 1 桁 bcd カウンタとして機能している。

4.3.2 論理合成

論理合成の結果，以下のような回路が作られた。

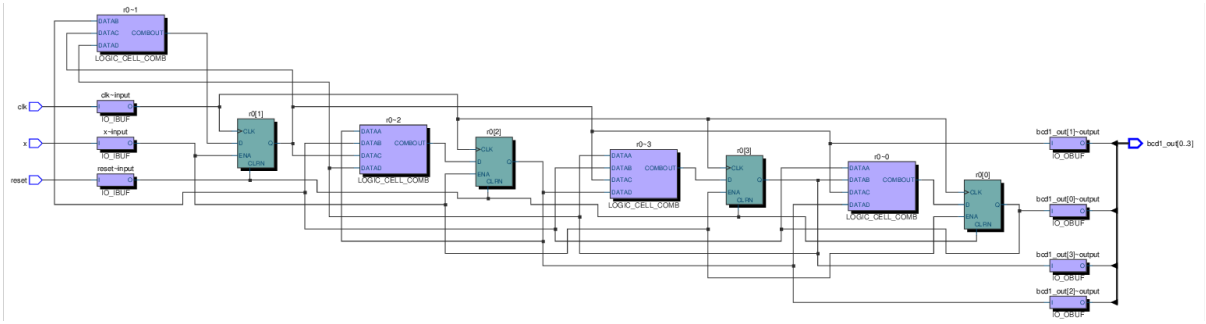

図 12　bcd1 の回路

ロジックエレメント数は 5 だった。

回路全体の遅延時間は，以下のようになった。

	Total	Incr	RF	Type	Fanout	Location	Element
1	4.349	4.349					data path
1	0.000	0.000			1	PIN_F7	x
2	0.000	0.000	FF	IC	1	IOIBUF_X24_Y39_N15	x~input|i
3	0.837	0.837	FF	CELL	4	IOIBUF_X24_Y39_N15	x~input|o
4	3.787	2.950	FF	IC	1	FF_X24_Y38_N13	r0[0]|ena
5	4.349	0.562	FF	CELL	1	FF_X24_Y38_N13	r0[0]

図 13　bcd1 の遅延時間

課題 1-2

4.3.3　機能レベルシュミレーション

ModelSim で波形を作成した結果，以下のような波形になった。

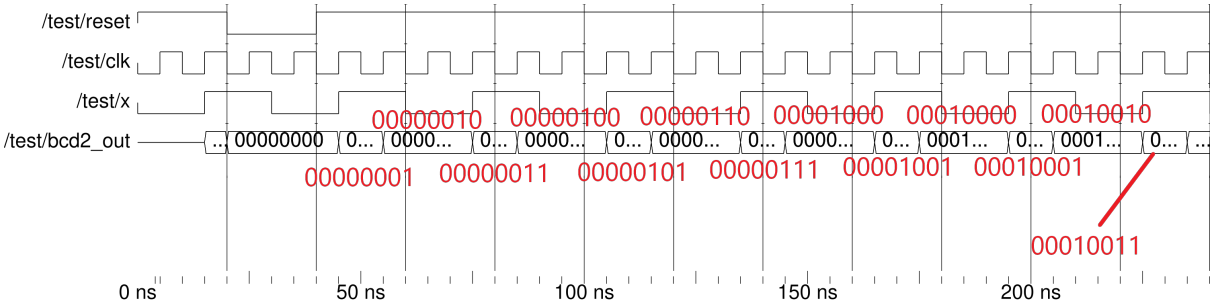

図 14　bcd2 の波形

bcd1_out の値は期待通り 10 進 2 桁 bcd カウンタとして機能している。

4.3.4　論理合成

論理合成の結果，以下のような回路が作られた。

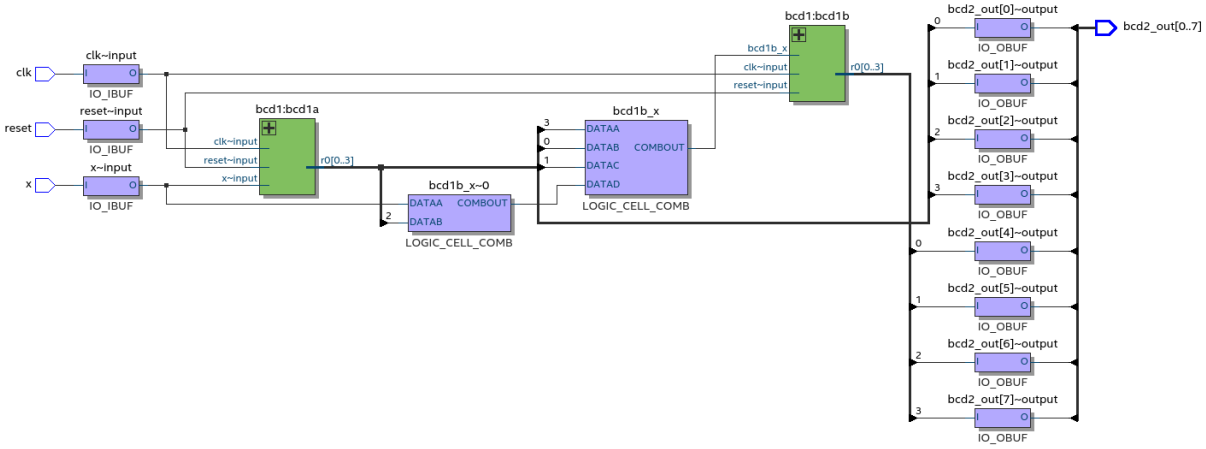

図 15　bcd2 の回路

bcd1 モジュールの他に以下のゲート構成があった。

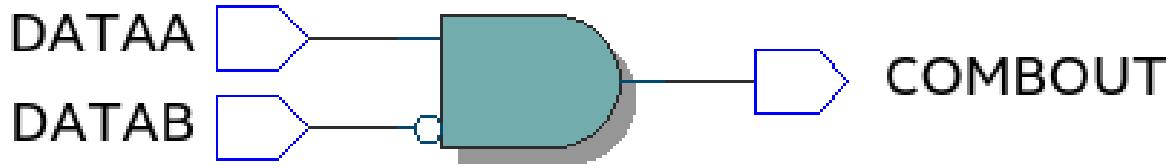

図 16　x と bcd1a[2]

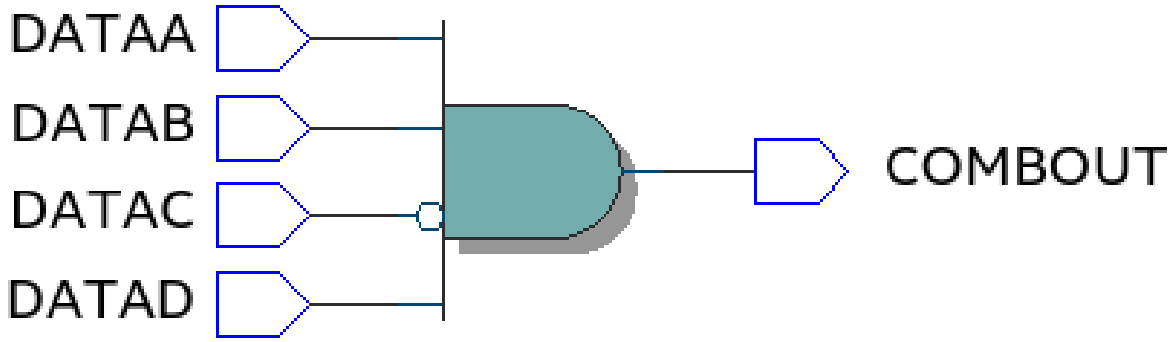

図 17　bcd2[3,0,1]

ロジックエレメント数は 11 だった。

回路全体の遅延時間は，以下のようになった。

	Total	Incr	RF	Type	Fanout	Location	Element
1	5.165	5.165					data path
1	0.000	0.000			1	PIN_W4	x
2	0.000	0.000	FF	IC	1	IOIBUF_X18_Y0_N15	x~input|i
3	0.874	0.874	FF	CELL	5	IOIBUF_X18_Y0_N15	x~input|o
4	3.789	2.915	FF	IC	1	LCCOMB_X18_Y1_N24	bcd1b_x~0|datac
5	4.013	0.224	FF	CELL	1	LCCOMB_X18_Y1_N24	bcd1b_x~0|combout
6	4.231	0.218	FF	IC	1	LCCOMB_X18_Y1_N26	bcd1b_x|datad
7	4.330	0.099	FF	CELL	4	LCCOMB_X18_Y1_N26	bcd1b_x|combout
8	4.603	0.273	FF	IC	1	FF_X18_Y1_N17	bcd1b|r0[0]|ena
9	5.165	0.562	FF	CELL	1	FF_X18_Y1_N17	bcd1:bcd1b|r0[0]

図 18　bcd2 の遅延時間

4.4　考察

課題 1-1

4.4.1　回路の HDL 記述

コード 15 ではクロックの立ち上がりで，x=1 のときに限ってインクリメントしている。また，レジスタの値が 9 以上ならば，ゼロを代入している。

リセット信号は，立ち下がりでレジスタにゼロを代入している。

テストベンチ (コード 16) では，5ns ごとのクロック反転と，15ns ごとの x の反転を実行している。

また，開始 20ns 後から 40ns 後までリセット信号を 0 にしている。

4.4.2　機能レベルシュミレーション

1 桁の bcd カウンタとして，クロックと x の値によって 0〜9 までの値をとっていた。

4.4.3　論理合成

回路では，4 桁分のレジスタ用フリップフロップと，各桁での論理演算回路が 4 桁分あることがわかる。

課題 1-2

4.4.4　回路の HDL 記述

コード 19 では，bcd1 のモジュールを利用して 2 桁分の BCD カウンタを実現している。具体的には，1 桁目が 9 で桁上りするときに限って，2 桁目の BCD カウンタに x を入力している。また，クロック信号は 2 つのモジュールに同じ信号を与えている。

リセット信号は，立ち下がりで 2 つの bcd1 モジュールにリセット信号を入力している。

テストベンチ (コード 20) では，5ns ごとのクロック反転と，15ns ごとの x の反転を実行している。

また，開始 20ns 後から 40ns 後までリセット信号を 0 にしている。

4.4.5 機能レベルシュミレーション

図 14 より，4bit を 10 進 1 桁分として，2 桁分の BCD カウンタが動作している。

4.4.6 論理合成

合成された回路では，1 桁目の 2bit 目と x をとって演算した結果を，他の bit と合成して，2 桁目の x として入力していることがわかる。この部分が，$(bcd1a\_out == 1001) \wedge x$ の結果として反映されていることが分かった。ほとんど設計したとおりにモジュールが配置されている。

このとき，bcd1 モジュールをインラインで展開した場合，回路の構成が異なる可能性がある。今回の結果より，モジュールの中身については各モジュールで最適化したのと同じ最適化が施されている。よって，階層設計による回路よりも，インラインで記述したほうがロジックエレメント数や，遅延時間が改善する可能性がある，と予想される。今回の実験ではわからなかったが，もしそのような差異があるならば，ボード規模などの制限によっては，記述しやすい階層設計と，インラインの記述を使い分ける必要があると予想した。

この実験で，順序回路の階層設計の手法と，階層化によって論理合成がどのように行われるかについて理解できた。

5 実験課題 2

5.1 実験の目的，概要

本実験では，0011 および 0010 が入力されるたびに，1 を出力する系列検出回路を設計する。このとき，状態器械による順序回路記述で設計する。作成した回路は，機能レベルシュミレーションと論理合成を行い，結果を確認する。

ICE 計算機上の，ModelSim，Quartus を使用する。

この実験で，オートマトンの設計を回路にどう反映させるかの方法について，習得することを目的にする。

5.2 実験方法

5.2.1 回路の HDL 記述

以下のような回路記述を m.v，テストベンチを test_m.v として作成する。

ソースコード 23　m.v

```
1 ‘define st0 2’b00
2 ‘define st1 2’b01
3 ‘define st2 2’b10
4 ‘define st3 2’b11
5
6 module m (clk, reset, x, y);
7   input clk, reset, x;
8   output y;
9
```

```
10  reg y_reg; // 1-bit 出力レジスタ
11  reg [1:0] st_reg; // 2-bit 状態レジスタ
12
13  assign y = y_reg;
14
15  always @(posedge clk or negedge reset) begin
16    if (reset == 1'b0) begin
17      /*** レジスタの初期値設定 ***/
18      y_reg<=0;
19      st_reg<=`st0;
20    end else begin
21      case (st_reg)
22        `st0: begin // 状態 0 にいる時
23          if (x == 1'b0) begin
24            st_reg<=`st1;
25          end else begin
26            st_reg<=`st0;
27          end
28          y_reg<=0;
29        end
30        `st1: begin // 状態 1 にいる時
31          if (x == 1'b0) begin
32            st_reg<=`st2;
33          end else begin
34            st_reg<=`st0;
35          end
36          y_reg<=0;
37        end
38        `st2: begin // 状態 2 にいる時
39          if (x == 1'b0) begin
40            st_reg<=`st2;
41          end else begin
42            st_reg<=`st3;
43          end
44          y_reg<=0;
45        end
46        `st3: begin // 状態 3 にいる時
47          if (x == 1'b0) begin
48            st_reg<=`st1;
49          end else begin
50            st_reg<=`st0;
51          end
52          y_reg<=1;
53        end
54      endcase
55    end
```

```
56   end
57 endmodule // m
```

ソースコード 24　test_m.v

```
1  /* *
2   * test_mv.v *
3   * 系列検出器のテストベンチ *
4   * */
5
6  `timescale 1ns / 1ns // シミュレーションの単位時間 / 精度
7                                                  // 1ns = 1/1000000000 sec
8  `include "m.v" // m.v の読み込み
9
10 module test ;
11   /*** m の入力値格納用のレジスタ ****/
12   reg reset, clk, x;
13
14   /*** y の出力観測用の信号線 ****/
15   wire y;
16
17   /*** m の実体化 ***/
18    m m(clk, reset, x, y);
19
20   always begin
21     #5 clk = ~clk;
22   end
23
24   initial begin
25     reset = 1; clk = 0; x = 0;
26
27     // 20 単位時間 (20 ns)後
28     #20 reset = 0;
29
30     // 更に 20 単位時間 (20 ns)後
31     #20 reset = 1;
32
33     #1 x = 1;
34     #10 x = 0;
35     #10 x = 0;
36     #10 x = 1;
37     #10 x = 0;
38     #10 x = 0;
39     #10 x = 1;
40     #10 x = 1;
41
42     // 更に 1000 単位時間 (1000 ns)後，終了
```

```
43      #1000 $finish;
44    end
45
46  endmodule // test
```

5.2.2 機能レベルシュミレーション

作成したテストベンチをもとに，ModelSim で信号波形を出力する。入出力の値が仕様通りの真理値表と一致することを確認する。

5.2.3 論理合成

以下の 2 ファイルを作成し，配置する。

ソースコード 25　m.qpf

```
1  # -------------------------------------------------------------------------- #
2  #
3  # Copyright (C) 1991-2013 Altera Corporation
4  # Your use of Altera Corporation's design tools, logic functions
5  # and other software and tools, and its AMPP partner logic
6  # functions, and any output files from any of the foregoing
7  # (including device programming or simulation files), and any
8  # associated documentation or information are expressly subject
9  # to the terms and conditions of the Altera Program License
10 # Subscription Agreement, Altera MegaCore Function License
11 # Agreement, or other applicable license agreement, including,
12 # without limitation, that your use is for the sole purpose of
13 # programming logic devices manufactured by Altera and sold by
14 # Altera or its authorized distributors. Please refer to the
15 # applicable agreement for further details.
16 #
17 # -------------------------------------------------------------------------- #
18 #
19 # Quartus II 64-Bit
20 # Version 13.0.1 Build 232 06/12/2013 Service Pack 1 SJ Web Edition
21 # Date created = 05:22:27 July 28, 2014
22 #
23 # -------------------------------------------------------------------------- #
24
25 QUARTUS_VERSION = "13.0"
26 DATE = "05:22:27 July 28, 2014"
27
28 # Revisions
29
30 PROJECT_REVISION = "m"
```

ソースコード 26　m.qsf

```
# Project-Wide Assignments
# ========================
set_global_assignment -name LAST_QUARTUS_VERSION "19.1.0 Lite Edition"

# Pin & Location Assignments
# ==========================
set_location_assignment PIN_C10 -to S1
set_location_assignment PIN_B8 -to D0
set_location_assignment PIN_A7 -to D1
set_location_assignment PIN_B11 -to Y

# Analysis & Synthesis Assignments
# ================================
set_global_assignment -name FAMILY "MAX 10"
set_global_assignment -name TOP_LEVEL_ENTITY m

# Fitter Assignments
# ==================
set_global_assignment -name DEVICE 10M50DAF484C6GES
set_global_assignment -name CYCLONEII_RESERVE_NCEO_AFTER_CONFIGURATION "USE AS REGULAR IO
    "
set_global_assignment -name RESERVE_ALL_UNUSED_PINS "AS INPUT TRI-STATED"

# ====================
set_global_assignment -name RESERVE_ALL_UNUSED_PINS_NO_OUTPUT_GND "AS INPUT TRI-STATED"

set_global_assignment -name SEED 1
set_global_assignment -name FIT_ONLY_ONE_ATTEMPT OFF
set_global_assignment -name OPTIMIZE_HOLD_TIMING "IO PATHS AND MINIMUM TPD PATHS"
set_global_assignment -name OPTIMIZE_MULTI_CORNER_TIMING OFF

set_global_assignment -name VERILOG_FILE m.v

set_global_assignment -name PARTITION_NETLIST_TYPE SOURCE -section_id Top
set_global_assignment -name PARTITION_FITTER_PRESERVATION_LEVEL PLACEMENT_AND_ROUTING -
    section_id Top
set_global_assignment -name PARTITION_COLOR 16764057 -section_id Top


set_global_assignment -name EDA_SIMULATION_TOOL "ModelSim-Altera (Verilog)"
set_global_assignment -name EDA_OUTPUT_DATA_FORMAT "VERILOG HDL" -section_id
    eda_simulation
set_global_assignment -name BOARD "MAX 10 DE10 - Lite"

set_instance_assignment -name PARTITION_HIERARCHY root_partition -to | -section_id Top
```

作成した回路記述を Quartus でコンパイルし，論理合成とレイアウトを行う。回路構成やロジックエレメント数，遅延時間について確認する。

5.3 実験結果

5.3.1 機能レベルシュミレーション

ModelSim で波形を作成した結果，以下のような波形になった。

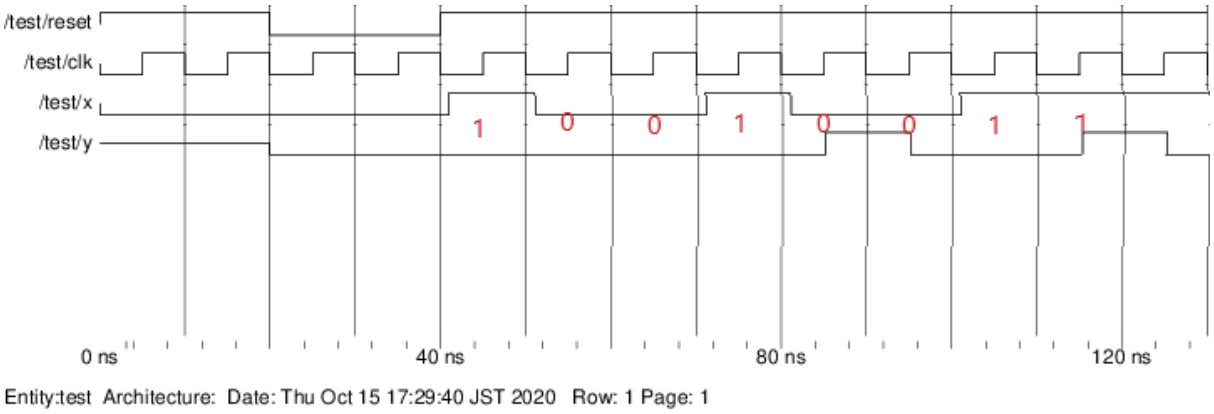

図 19　m の波形

出力 Y は，期待通り 2 回 1 を出力していた。

5.3.2 論理合成

論理合成の結果，以下のような回路が作られた。

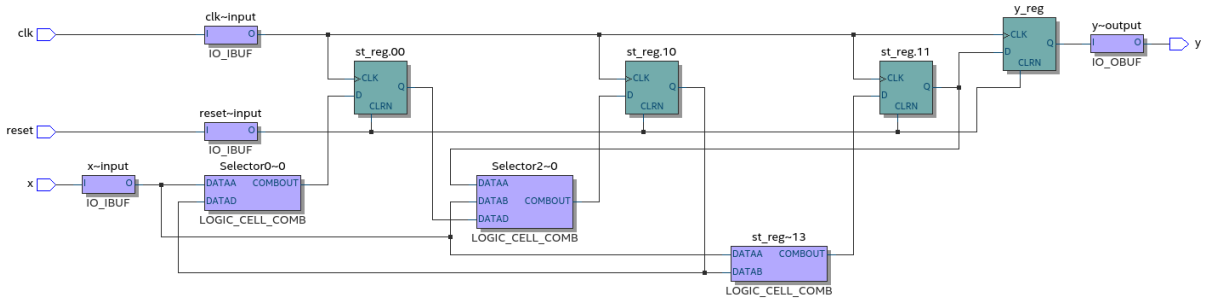

図 20　m の回路

ロジックエレメント数は 5 だった。

回路全体の遅延時間は，以下のようになった。

	Total	Incr	RF	Type	Fanout	Location	Element
1	4.206	4.206					data path
1	0.000	0.000			1	PIN_C9	x
2	0.000	0.000	FF	IC	1	IOIBUF_X46_Y54_N15	x~input|i
3	0.857	0.857	FF	CELL	3	IOIBUF_X46_Y54_N15	x~input|o
4	3.790	2.933	FF	IC	1	LCCOMB_X46_Y53_N0	Selector2~0|dataa
5	4.123	0.333	FR	CELL	1	LCCOMB_X46_Y53_N0	Selector2~0|combout
6	4.123	0.000	RR	IC	1	FF_X46_Y53_N1	st_reg.10|d
7	4.206	0.083	RR	CELL	1	FF_X46_Y53_N1	st_reg.10

図 21　m の遅延時間

5.4　考察

5.4.1　回路の HDL 記述

図 22 のような出力付きのオートマトンをもとにして，回路記述を行った。レジスタに状態を記憶し，入力に応じて状態を遷移，出力する。

今回は，0010011 のような系列に 2 回検出を行うオートマトンを設計した。

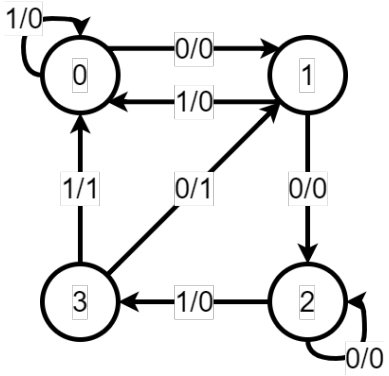

図 22　設計したオートマトン

5.4.2　機能レベルシュミレーション

テストベンチでは，これまでの順序回路と同様な設定の後に，1 → 0 → 0 → 1 → 0 → 0 → 1 → 1 という系列を 10ns おきに入力している。最初に 0010 が，次に 0011 が検出されることが期待されて，実際に 2 回検出されている。

5.4.3　論理合成

コード 23 では，4 状態のオートマトンと出力レジスタが宣言されているが，図 20 では，合成によって 3 つの状態と出力，論理演算のみで構成されていることがわかった。状態として持っていた情報を，現在の状態と入力の演算として再定義することで，状態数を削減しているのだと予想する。

この実験で，オートマトンの設計を順序回路として記述する方法がわかった。また，そのような回路は合成によって状態数が減ることがある，ということも確認できた。

6 論理合成の最適化について

6.1 具体的な手法

HDL で記述された回路は，まず論理演算等が率直に実装される。次に，各演算に対する最適化や，モジュールの共用が行われる。

具体的には，カルノー図による論理関数の簡単化や，1 ゲートで実行できる演算の置換が行われる (XOR など)。

モジュールの共用では，次のような最適化例がある。2 入力 1 出力の，入力をゲート A に通してゲート B に通す回路を考える。このとき，素の入力にゲート B を施してからゲート A を通す回路が，等価な回路として考えられる場合，ゲートの数を 1 つ削減することができる。

ZHANG, ZHIFEI[2] によると，HDL 記述の抽象度が高いほど，このような最適化を行う余地が生まれ，回路規模も遅延も小さくなる傾向があるとされている。実際，実験 2 で桁上りの先読みについて予想しているが，ZHANG, ZHIFEI[2] では回路規模の点でも配線長の点でも不利になることから，4bit 加算器の場合は，ナイーブな実装のほうが優れているとされている。

6.2 遅延時間と回路面積の関係

ALTERA[1] によると，回路合成による回路の分散と遅延時間はトレードオフになっていることがわかる。しかし，面積が小さい場合，配線の長さによる遅延時間も大きくなり，ZHANG, ZHIFEI[2] のキャリー読み加算回路のように，配線前後で遅延時間が大幅に変動する。

つまり，最適化の結果，段数を増やしてクリティカルパスを短くするような回路では，面積と遅延がトレードオフになる。しかし，演算による遅延に対して配線の影響が大きい場合，遅延は面積の大きさに伴って増加する。

参考文献

[1] AN 584: 高度な FPGA デザインにおける タイミング・クロージャ手法, ALTERA，2009，https://www.intel.co.jp/content/dam/altera-www/global/ja_JP/pdfs/literature/an/an584_j.pdf

[2] RTL とゲートレベルを混在させた最適な論理回路設計に関する研究，ZHANG, ZHIFEI，2014，https://dspace.jaist.ac.jp/dspace/handle/10119/12013

すべて 2020 年 10 月 22 日閲覧