AZPR EvBoard

取扱説明書

http://respon.org/

目次

1. 概要

1.1. 更新履歴

2012/08/11 初版作成

2012/10/20 2 版・本文中の誤りを訂正

1.2. 内容物

本製品のパッケージには下記の内容物が含まれます。

- AZPR EvBoard 本体
- ワンタッチ基板スタンド MPS-08
- 取扱説明書(本書)

なお、下記の製品は本製品のパッケージには含まれておらず、お客様自身で用意していただく必要があります。

- PC(OS は windows7 を推奨)
- AC アダプタ(出力電圧: 5[V], 出力電流:2[A]程度, 内径 2.1[mm])
- USB ケーブル(A-miniB)

また、ダイアグプログラムを動作させる場合には下記の製品が必要になります。

- 10 ピンフラットケーブル(5 ピン×2 列)

1.3. 注意事項

- 本製品をお使いになるには電子工作や電子回路についての一般的な知識、ザイリンクス社製 FPGA についての知識や開発環境などが必要です。
- 本製品をお使いになる前には、必ず IC のドキュメント類を参照してください。ザイリンクス社のホームページから、FPGA のドキュメントや開発ツールなどをダウンロードできます。
- 本製品は学習用の基板です。工業製品への組み込みなど、用途外の使用はご遠慮ください。また、本製品によって生じた損害などについては責任を負いかねます。
- 本製品の仕様は予告なく変更になる場合があります。ご注意ください。
- 本マニュアルは、内容について絶対の保証をするものではありません。

1.4. 書籍「CPU 自作入門」にて製作する基板との相違点

- 基板のバージョンを v.1.0 から v.2.0 に変更しています。
- FPGA の型番を XC3S250E から XC3S500E に変更しています。
- コンフィグ ROM の型番を XCF02S から XCF04S に変更しています。
- 電源回路を内蔵しています。
- USB-シリアル変換 IC 用の EEPROM を実装し、内部に設定を書き込んでいます。
  このため、デバイスマネージャからシリアルポートは 1ch のみ認識されます。

1.5. サポートページについて

下記のページにて、回路図、制約ファイル、ダイアグプログラムなどの配布などを行っています。

http://respon.org/

2. AZPR EvBoard の実装部品

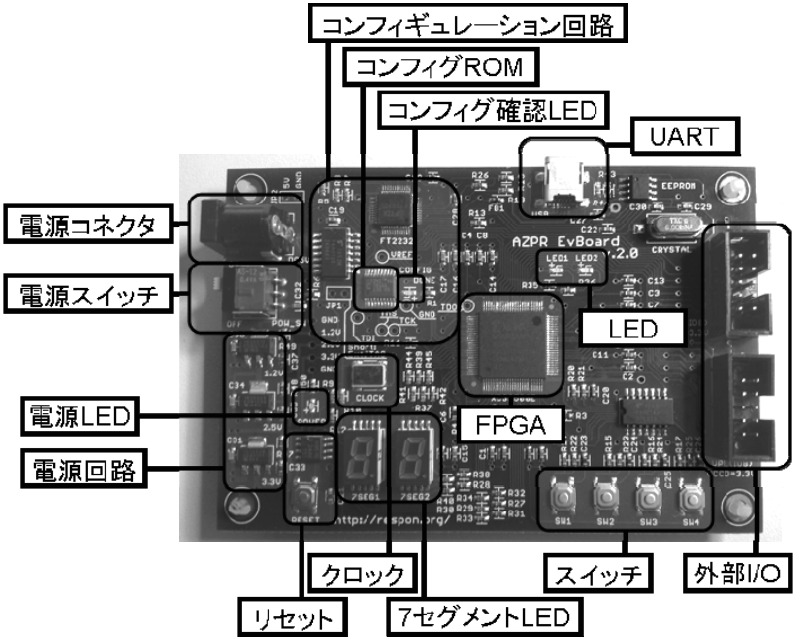


図 1: AZPR EvBoard の概観

- FPGA とコンフィグ ROM
  FPGA: Spartan3E(Device: XC3S500E, Package: VQG100, SpeedGrade: -4,)
  コンフィグ ROM: XCF04S

> AZPR EvBoard では USB によるコンフィギュレーションと外部ダウンロードケーブルによるコンフィギュレーションをサポートしています。
> 詳細は「5 FPGA のコンフィギュレーション方法」を参照してください。

- スイッチ
  4 個実装されています。正論理入力、チャタリング除去済みです。
- LED
  2 個実装されています。負論理出力です。

- 7 セグメント LED
  2 個実装されています。デコーダ回路はなく、FPGA から直結されています。負論理出力です。D.P, g, f, e, d, c, b, a と並べた場合のデコード表を表 1 に、端子配置と内部回路構成図を図 2 に示します。

表 1: 7 セグメント LED のデコード表

表示	2 進数	16 進数
0	11000000	0xC0
1	11111001	0xF9
2	10100100	0xA4
3	10110000	0xB0
4	10011001	0x99
5	10010010	0x92
6	10000010	0x82
7	11111000	0xF8
8	10000000	0x80
9	10010000	0x90

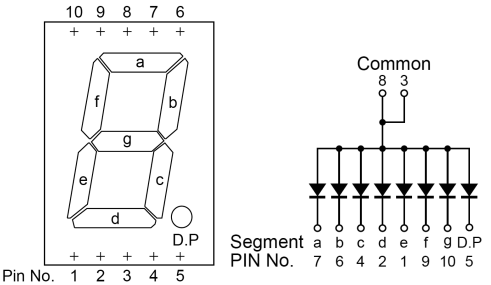


図 2: 7 セグメント LED の端子配置と内部回路構成図

- クロック
  10MHz の周波数が入力されています。FPGA 内部の DCM で周波数を変更することができます。
- リセット
  負論理入力です。リセット端子は、電圧降下に応じて自動的にリセットがかかります。
- UART
  USB-シリアル変換 IC による仮想 COM ポートです。
- 外部 I/O
  VPort 互換のボックスヘッダが 2 個実装されています。ピン割付を表 2 に、ピンアサインを表 3 に示します。

表 2: VPort のピン割付

ピン番号	用途
1～8	汎用 I/O
9	GND
10	Vcc

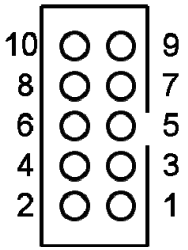
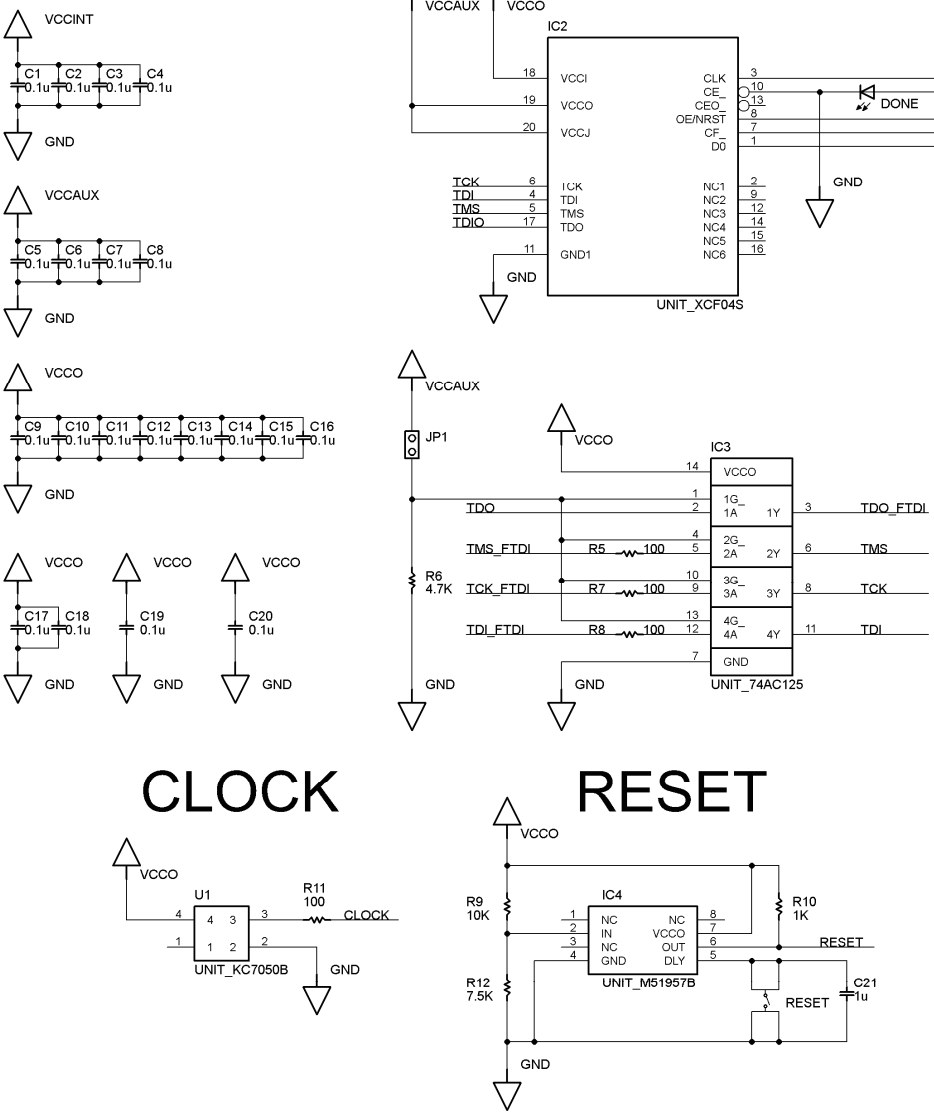
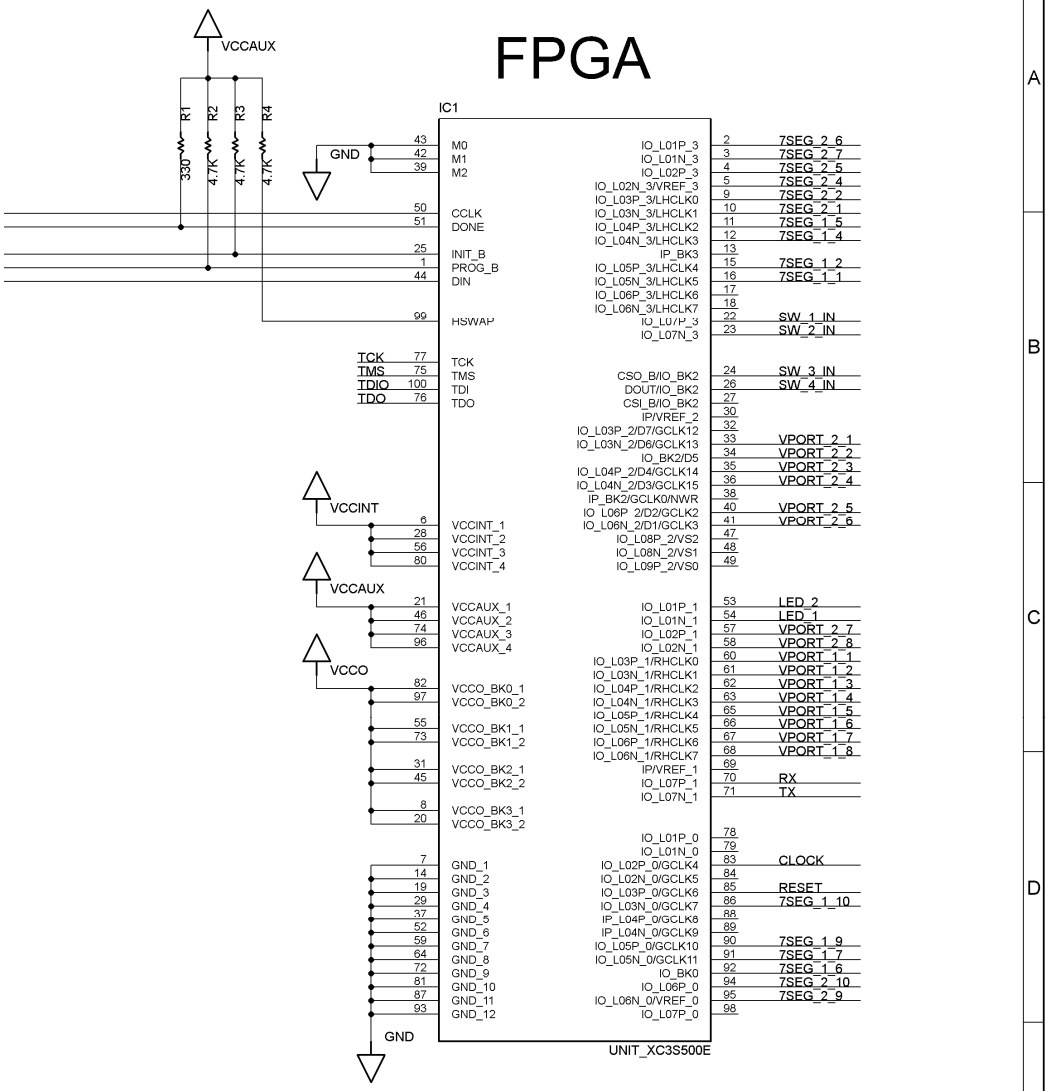


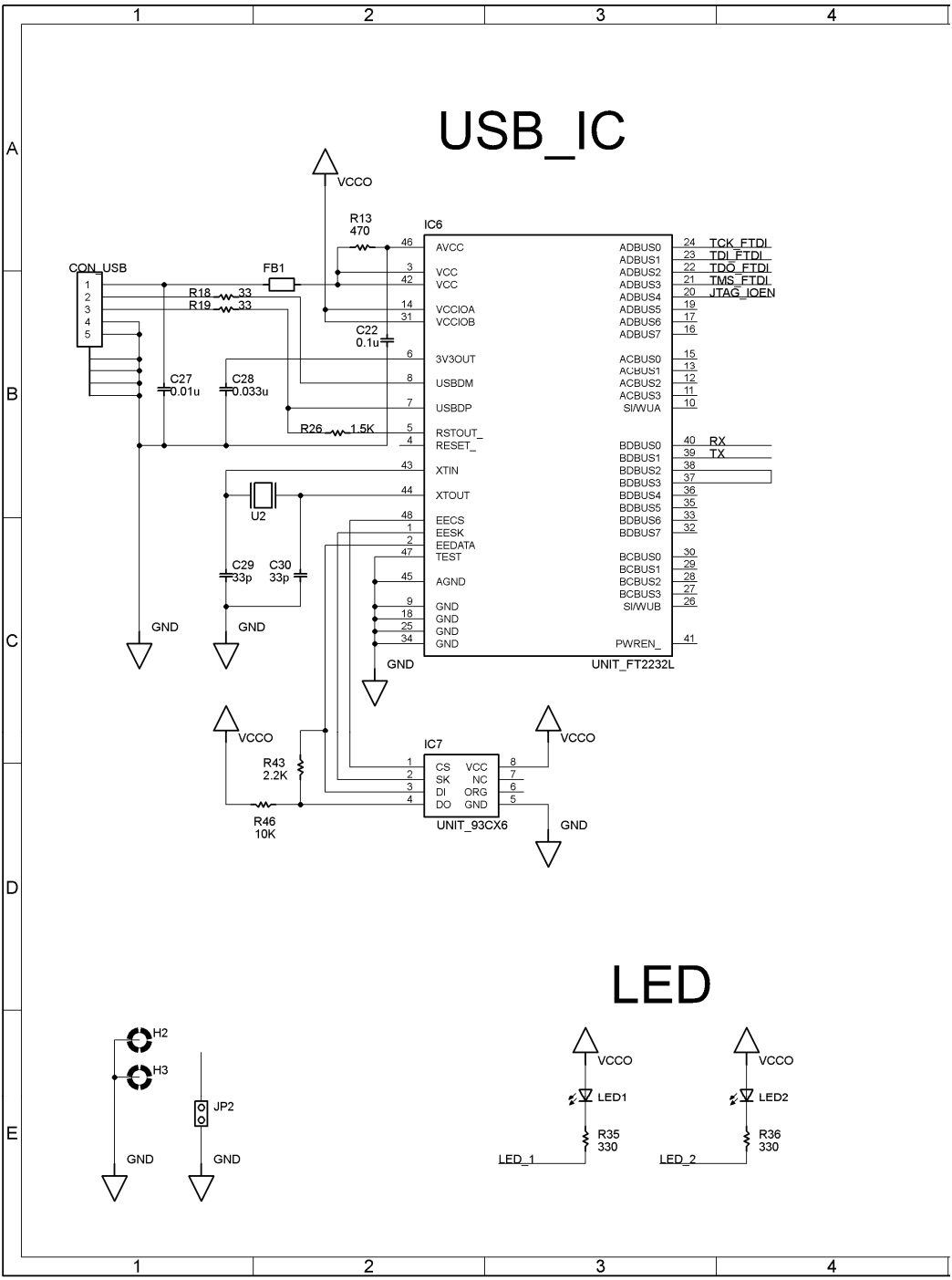
図 3: VPort 互換ボックスヘッダのピンアサイン

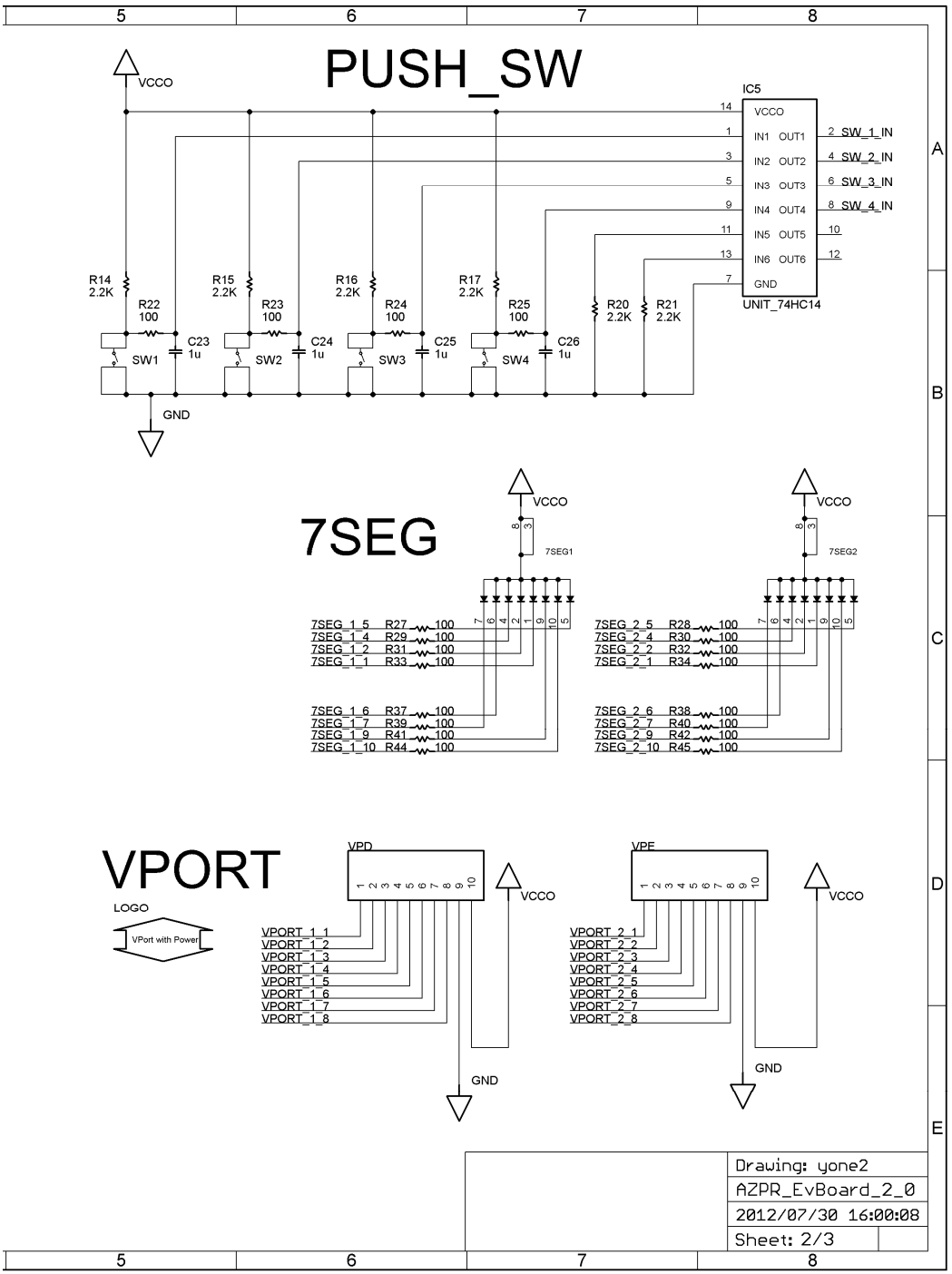
3. 回路図

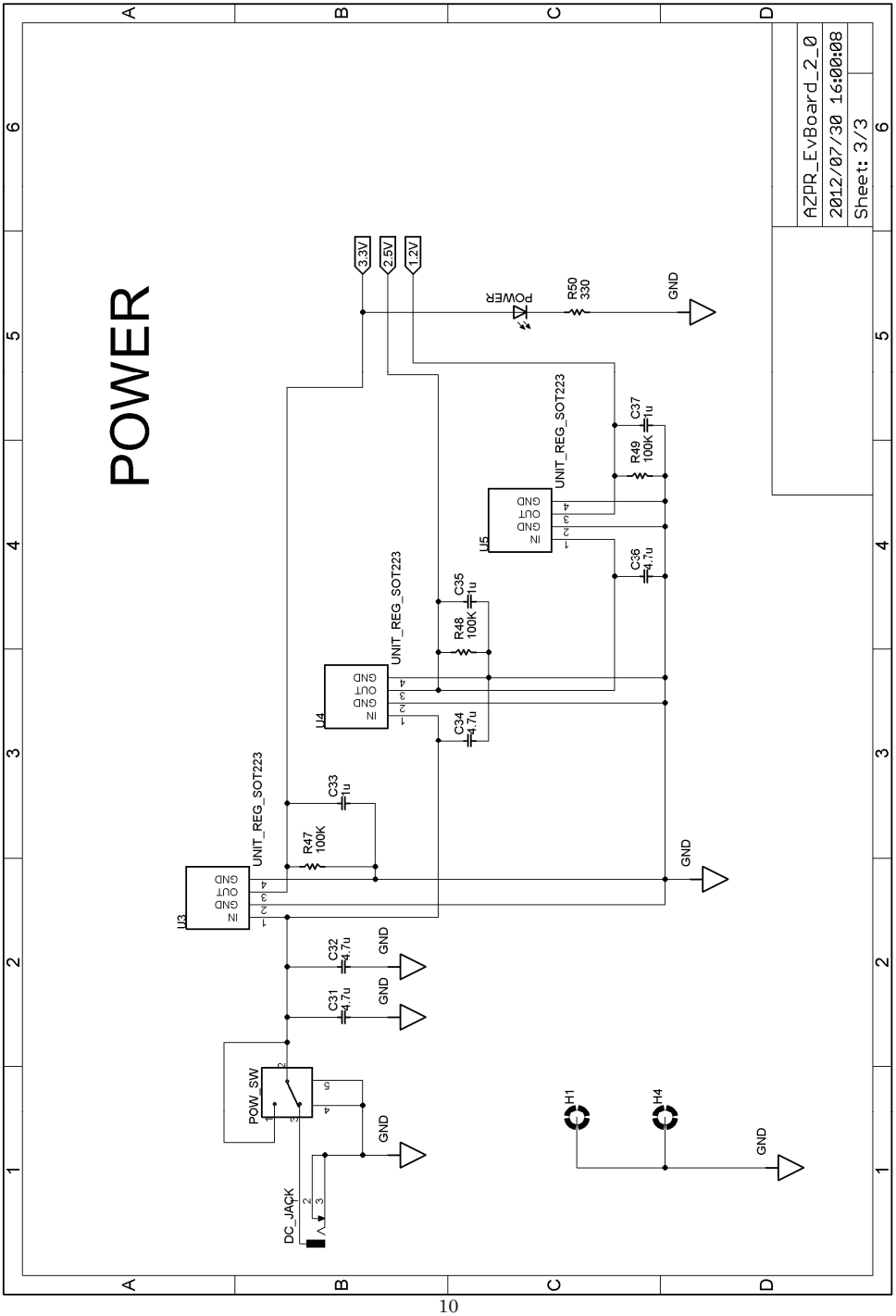
次ページ以降に示します。


VCCINT
C1 0.1u
C2 0.1u
C3 0.1u
C4 0.1u
GND
VCCAUX
C5 0.1u
C6 0.1u
C7 0.1u
C8 0.1u
GND
VCCO
C9 0.1u
C10 0.1u
C11 0.1u
C12 0.1u
C13 0.1u
C14 0.1u
C15 0.1u
C16 0.1u
GND
VCCO
C17 0.1u
C18 0.1u
GND
VCCO
C19 0.1u
GND
VCCO
C20 0.1u
GND
VCCAUX
VCCO
IC2
18 VCCI
19 VCCO
20 VCCJ
TCK 6
TDI 4
TMS 5
TDIO 17
11 GND1
GND
CLK 3
CE_ 10
CEO_ 13
OE/NRST 8
CF_ 7
D0 1
DONE
GND
NC1 2
NC2 9
NC3 12
NC4 14
NC5 15
NC6 16
UNIT_XCF04S
VCCAUX
JP1
R6 4.7K
GND
VCCO
IC3
14 VCCO
TDO 1 2
TMS_FTDI 4 5
TCK_FTDI 10 9
TDI_FTDI 13 12
7 GND
1G_ 1A 1Y 3 TDO_FTDI
2G_ 2A 2Y 6 TMS
3G_ 3A 3Y 8 TCK
4G_ 4A 4Y 11 TDI
R5 100
R7 100
R8 100
GND
UNIT_74AC125
CLOCK
VCCO
U1
R11 100
CLOCK
UNIT_KC7050B
GND
RESET
VCCO
R9 10K
IC4
1 NC
2 IN
3 NC
4 GND
NC 8
VCCO 7
OUT 6
DLY 5
R10 1K
RESET
R12 7.5K
C21 1u
RESET
GND
UNIT_M51957B


FPGA
VCCAUX
R1
R2
R3
R4
330
4.7K
4.7K
4.7K
GND
IC1
M0
M1
M2
CCLK
DONE
INIT_B
PROG_B
DIN
HSWAP
TCK
TMS
TDI
TDO
VCCINT
VCCINT_1
VCCINT_2
VCCINT_3
VCCINT_4
VCCAUX
VCCAUX_1
VCCAUX_2
VCCAUX_3
VCCAUX_4
VCCO
VCCO_BK0_1
VCCO_BK0_2
VCCO_BK1_1
VCCO_BK1_2
VCCO_BK2_1
VCCO_BK2_2
VCCO_BK3_1
VCCO_BK3_2
GND_1
GND_2
GND_3
GND_4
GND_5
GND_6
GND_7
GND_8
GND_9
GND_10
GND_11
GND_12
GND
IO_L01P_3
IO_L01N_3
IO_L02P_3
IO_L02N_3/VREF_3
IO_L03P_3/LHCLK0
IO_L03N_3/LHCLK1
IO_L04P_3/LHCLK2
IO_L04N_3/LHCLK3
IP_BK3
IO_L05P_3/LHCLK4
IO_L05N_3/LHCLK5
IO_L06P_3/LHCLK6
IO_L06N_3/LHCLK7
IO_L07P_3
IO_L07N_3
CSO_B/IO_BK2
DOUT/IO_BK2
CSI_B/IO_BK2
IP/VREF_2
IO_L03P_2/D7/GCLK12
IO_L03N_2/D6/GCLK13
IO_BK2/D5
IO_L04P_2/D4/GCLK14
IO_L04N_2/D3/GCLK15
IP_BK2/GCLK0/NWR
IO_L06P_2/D2/GCLK2
IO_L06N_2/D1/GCLK3
IO_L08P_2/VS2
IO_L08N_2/VS1
IO_L09P_2/VS0
IO_L01P_1
IO_L01N_1
IO_L02P_1
IO_L02N_1
IO_L03P_1/RHCLK0
IO_L03N_1/RHCLK1
IO_L04P_1/RHCLK2
IO_L04N_1/RHCLK3
IO_L05P_1/RHCLK4
IO_L05N_1/RHCLK5
IO_L06P_1/RHCLK6
IO_L06N_1/RHCLK7
IP/VREF_1
IO_L07P_1
IO_L07N_1
IO_L01P_0
IO_L01N_0
IO_L02P_0/GCLK4
IO_L02N_0/GCLK5
IO_L03P_0/GCLK6
IO_L03N_0/GCLK7
IP_L04P_0/GCLK8
IP_L04N_0/GCLK9
IO_L05P_0/GCLK10
IO_L05N_0/GCLK11
IO_BK0
IO_L06P_0
IO_L06N_0/VREF_0
IO_L07P_0
7SEG_2_6
7SEG_2_7
7SEG_2_5
7SEG_2_4
7SEG_2_2
7SEG_2_1
7SEG_1_5
7SEG_1_4
7SEG_1_2
7SEG_1_1
SW_1_IN
SW_2_IN
SW_3_IN
SW_4_IN
VPORT_2_1
VPORT_2_2
VPORT_2_3
VPORT_2_4
VPORT_2_5
VPORT_2_6
LED_2
LED_1
VPORT_2_7
VPORT_2_8
VPORT_1_1
VPORT_1_2
VPORT_1_3
VPORT_1_4
VPORT_1_5
VPORT_1_6
VPORT_1_7
VPORT_1_8
RX
TX
CLOCK
RESET
7SEG_1_10
7SEG_1_9
7SEG_1_7
7SEG_1_6
7SEG_2_10
7SEG_2_9
UNIT_XC3S500E
Drawing: yone2
AZPR_EvBoard_2_0
2012/07/30 16:00:08
Sheet: 1/3


USB_IC
VCCO
R13
470
IC6
CON_USB
FB1
R18 33
R19 33
C22
0.1u
C27
0.01u
C28
0.033u
R26 1.5K
U2
C29 33p
C30 33p
GND
AVCC
VCC
VCC
VCCIOA
VCCIOB
3V3OUT
USBDM
USBDP
RSTOUT_
RESET_
XTIN
XTOUT
EECS
EESK
EEDATA
TEST
AGND
GND
GND
GND
GND
ADBUS0
ADBUS1
ADBUS2
ADBUS3
ADBUS4
ADBUS5
ADBUS6
ADBUS7
ACBUS0
ACBUS1
ACBUS2
ACBUS3
SI/WUA
BDBUS0
BDBUS1
BDBUS2
BDBUS3
BDBUS4
BDBUS5
BDBUS6
BDBUS7
BCBUS0
BCBUS1
BCBUS2
BCBUS3
SI/WUB
PWREN_
TCK_FTDI
TDI_FTDI
TDO_FTDI
TMS_FTDI
JTAG_IOEN
RX
TX
UNIT_FT2232L
IC7
R43
2.2K
R46
10K
CS
SK
DI
DO
VCC
NC
ORG
GND
UNIT_93CX6
LED
H2
H3
JP2
LED1
LED2
R35
330
R36
330
LED_1
LED_2


PUSH_SW
VCCO
IC5
VCCO
IN1 OUT1
IN2 OUT2
IN3 OUT3
IN4 OUT4
IN5 OUT5
IN6 OUT6
GND
UNIT_74HC14
SW_1_IN
SW_2_IN
SW_3_IN
SW_4_IN
R14 2.2K
R15 2.2K
R16 2.2K
R17 2.2K
R20 2.2K
R21 2.2K
R22 100
R23 100
R24 100
R25 100
C23 1u
C24 1u
C25 1u
C26 1u
SW1
SW2
SW3
SW4
GND
7SEG
VCCO
7SEG1
7SEG2
7SEG_1_5 R27 100
7SEG_1_4 R29 100
7SEG_1_2 R31 100
7SEG_1_1 R33 100
7SEG_1_6 R37 100
7SEG_1_7 R39 100
7SEG_1_9 R41 100
7SEG_1_10 R44 100
7SEG_2_5 R28 100
7SEG_2_4 R30 100
7SEG_2_2 R32 100
7SEG_2_1 R34 100
7SEG_2_6 R38 100
7SEG_2_7 R40 100
7SEG_2_9 R42 100
7SEG_2_10 R45 100
VPORT
LOGO
VPort with Power
VPD
VPE
VCCO
GND
VPORT_1_1
VPORT_1_2
VPORT_1_3
VPORT_1_4
VPORT_1_5
VPORT_1_6
VPORT_1_7
VPORT_1_8
VPORT_2_1
VPORT_2_2
VPORT_2_3
VPORT_2_4
VPORT_2_5
VPORT_2_6
VPORT_2_7
VPORT_2_8
Drawing: yone2
AZPR_EvBoard_2_0
2012/07/30 16:00:08
Sheet: 2/3


POWER
3.3V
2.5V
1.2V
POWER
R50
330
GND
U5
UNIT_REG_SOT223
IN
GND
OUT
GND
C37
1u
R49
100K
C36
4.7u
U4
UNIT_REG_SOT223
C35
1u
R48
100K
C34
4.7u
U3
UNIT_REG_SOT223
C33
1u
R47
100K
C32
4.7u
C31
4.7u
POW_SW
DC_JACK
H1
H4
AZPR_EvBoard_2_0
2012/07/30 16:00:08
Sheet: 3/3

4. 制約ファイル

制約ファイルの作成時に必要な FPGA の接続情報を表 3 に示します。制約に関する詳細は、ザイリンクス社の「制約ガイド」(http://japan.xilinx.com/support/documentation/dt_ise.htm)を参照してください。

表 3: FPGA の接続情報

部品名	ピン	XC3S500E_PIN	TYPE	ucf ファイル例
clock	P83	IO_L02P_0/GCLK4	GCLK	NET "clock" LOC = "P83";
reset	P85	IO_L03P_0/GCLK6	GCLK	NET "reset" LOC = "P85";
led_1	P54	IO_L01N_1	I/O	NET "led_1" LOC = "P54";
led_2	P53	IO_L01P_1	I/O	NET "led_2" LOC = "P53";
push_sw_1	P22	IO_L07P_3	I/O	NET "push_sw_1" LOC = "P22";
push_sw_2	P23	IO_L07N_3	I/O	NET "push_sw_2" LOC = "P23";
push_sw_3	P24	IO_L01P_2/CSO_B	DUAL	NET "push_sw_3" LOC = "P24";
push_sw_4	P26	IO_L02P_2/DOUT/BUSY	DUAL	NET "push_sw_4" LOC = "P26";
seg_1_a	P91	IO_L05N_0/GCLK11	GCLK	NET "seg_1_a" LOC = "P91";
seg_1_b	P92	IO	I/O	NET "seg_1_b" LOC = "P92";
seg_1_c	P12	IO_L04N_3/LHCLK3/IRDY2	GCLK	NET "seg_1_c" LOC = "P12";
seg_1_d	P15	IO_L05P_3/LHCLK4/TRDY2	GCLK	NET "seg_1_d" LOC = "P15";
seg_1_e	P16	IO_L05N_3/LHCLK5	GCLK	NET "seg_1_e" LOC = "P16";
seg_1_f	P90	IO_L05P_0/GCLK10	GCLK	NET "seg_1_f" LOC = "P90";
seg_1_g	P86	IO_L03N_0/GCLK7	GCLK	NET "seg_1_g" LOC = "P86";
seg_1_dp	P11	IO_L04P_3/LHCLK2	GCLK	NET "seg_1_dp" LOC = "P11";
seg_2_a	P3	IO_L01N_3	I/O	NET "seg_2_a" LOC = "P3";
seg_2_b	P2	IO_L01P_3	I/O	NET "seg_2_b" LOC = "P2";
seg_2_c	P5	IO_L02N_3/VREF_3	VREF	NET "seg_2_c" LOC = "P5";
seg_2_d	P9	IO_L03P_3/LHCLK0	GCLK	NET "seg_2_d" LOC = "P9";
seg_2_e	P10	IO_L03N_3/LHCLK1	GCLK	NET "seg_2_e" LOC = "P10";
seg_2_f	P95	IO_L06N_0/VREF_0	VREF	NET "seg_2_f" LOC = "P95";
seg_2_g	P94	IO_L06P_0	I/O	NET "seg_2_g" LOC = "P94";
seg_2_dp	P4	IO_L02N_3/VREF_3	VREF	NET "seg_2_dp" LOC = "P4";
vport_1_1	P60	IO_L03P_1/RHCLK0	DUAL	NET "vport_1_1" LOC = "P60";
vport_1_2	P61	IO_L03N_1/RHCLK1	DUAL	NET "vport_1_2" LOC = "P61";
vport_1_3	P62	IO_L04P_1/RHCLK2	DUAL	NET "vport_1_3" LOC = "P62";
vport_1_4	P63	IO_L04N_1/RHCLK3/TRDY1	DUAL	NET "vport_1_4" LOC = "P63";
vport_1_5	P65	IO_L05P_1/RHCLK4/IRDY1	DUAL	NET "vport_1_5" LOC = "P65";
vport_1_6	P66	IO_L05N_1/RHCLK5	DUAL	NET "vport_1_6" LOC = "P66";
vport_1_7	P67	IO_L06P_1/RHCLK6	DUAL	NET "vport_1_7" LOC = "P67";
vport_1_8	P68	IO_L06N_1/RHCLK7	DUAL	NET "vport_1_8" LOC = "P68";
vport_2_1	P33	IO_L03N_2/D6/GCLK13	DUAL/GCLK	NET "vport_2_1" LOC = "P33";
vport_2_2	P34	IO/D5	DUAL	NET "vport_2_2" LOC = "P34";
vport_2_3	P35	IO_L04P_2/D4/GCLK14	DUAL/GCLK	NET "vport_2_3" LOC = "P35";
vport_2_4	P36	IO_L04N_2/D3/GCLK15	DUAL/GCLK	NET "vport_2_4" LOC = "P36";
vport_2_5	P40	IO_L06P_2/D2/GCLK2	DUAL/GCLK	NET "vport_2_5" LOC = "P40";
vport_2_6	P41	IO_L06N_2/D1/GCLK3	DUAL/GCLK	NET "vport_2_6" LOC = "P41";
vport_2_7	P57	IO_L02P_1	I/O	NET "vport_2_7" LOC = "P57";
vport_2_8	P58	IO_L02N_1	I/O	NET "vport_2_8" LOC = "P58";

5. FPGA のコンフィギュレーション方法

AZPR EvBoard では、3 種類のコンフィギュレーション方法をサポートしています。なお、本取扱説明書はコンフィギュレーション情報を格納したファイル(BIT ファイルや MCS ファイル)が準備してあるという前提で書かれています。ザイリンクス社のツールの基本的な使い方は、ザイリンクス社のホームページなどを参照してください。

5.1. UrJTAG を利用したコンフィギュレーション

UrJTAG を利用したコンフィギュレーションの流れを図 4 に示します。

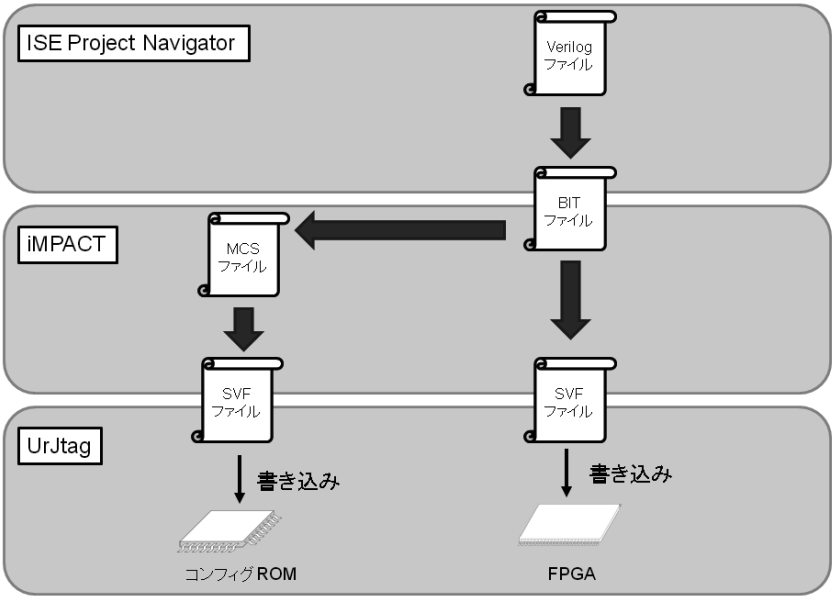


図 4: UrJTAG を利用したコンフィギュレーションの流れ

5.1.1. SVF ファイルの生成

SVF は Serial Vector Format の略で、JTAG 操作を記述したファイルです。コンフィギュレーションデータを SVF 形式でファイルに出力し、UrJtag というツールで使用します。UrJtag で SVF ファイルを再生することで、デバイスに対して JTAG 操作を行います。コンフィギュレーション情報を格納したファイル(BIT ファイルや MCS ファイル)が準備してあるという前提で、ここでは SVF ファイル作成の手順を説明します。

まず、iMPACT を起動します。図 5 に「ISE iMPACT Project」ウィンドウを示します。

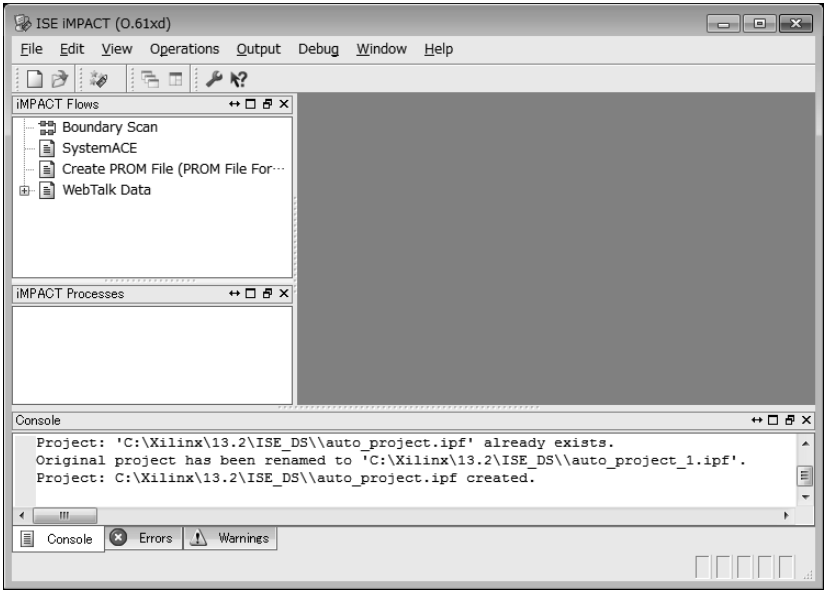


図 5: iMPACT 起動時の画面

「ISE iMPACT」ウィンドウの左上の領域から「Boundary Scan」をダブルクリックします。

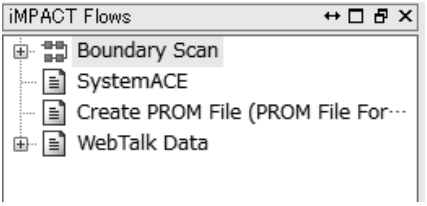


図 6: Boundary Scan が表示されている領域

Boundary Scan 実行後の画面を図 7 に示します。

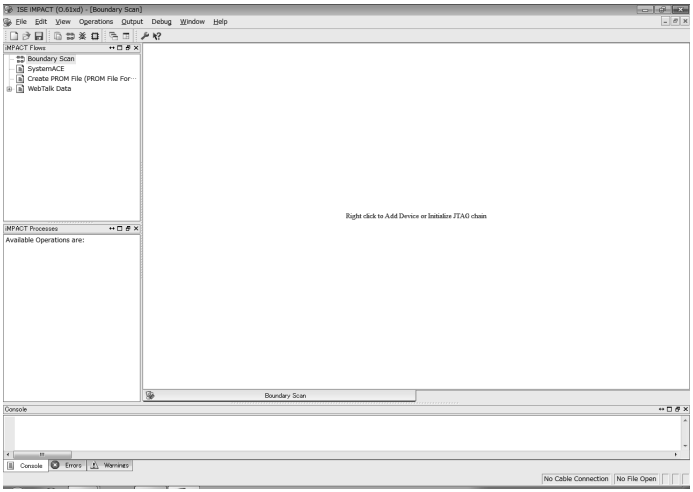


図 7: Boundary Scan 後の画面

「Right click to Add Device or Initialize JTAG chain」と表示された領域を右クリックし、図 8 に示すように [Output File Type]→[SVF File]→[Create SVF File]を選択し、SVF ファイル作成を開始します。

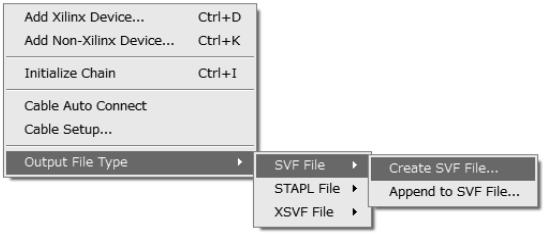


図 8: Create SVF File の選択

「Add Device」ダイアログで書きこむファイルを選択します。ここで BIT ファイルを選択する場合と MCS ファイルを選択する場合で手順が別れます。BIT ファイルを選択すると、図 9 のように「xc3s500e」が追加されます。

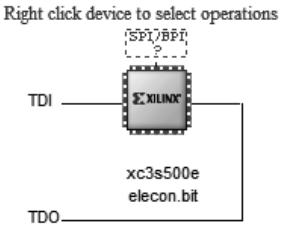


図 9: BIT ファイル追加後の画面

MCS ファイルを選択すると、**図 10** のように「Select Device Part Name」ダイアログが表示されます。ここで、PROM デバイスとして、「xcf04s」を選択します。[OK]ボタンをクリックすると、図 11 のように「xcf04s」が追加されます。

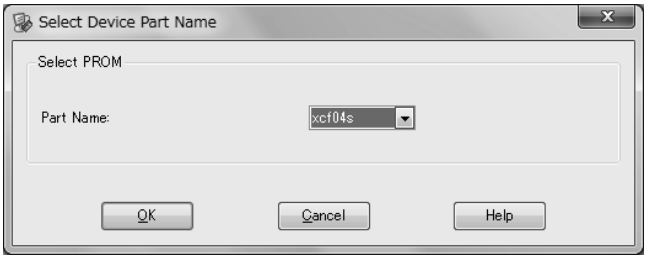


図 10: Select Device Part Name

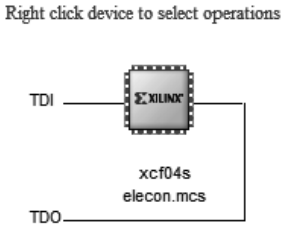


図 11: MCS ファイル追加後の画面

書き込むファイルを選択した後、追加されたデバイス上で右クリックし、「Program」を選択します。図 12 で示す「Device Programming Properties」ダイアログが開くので、そのまま[OK]ボタンを押します。

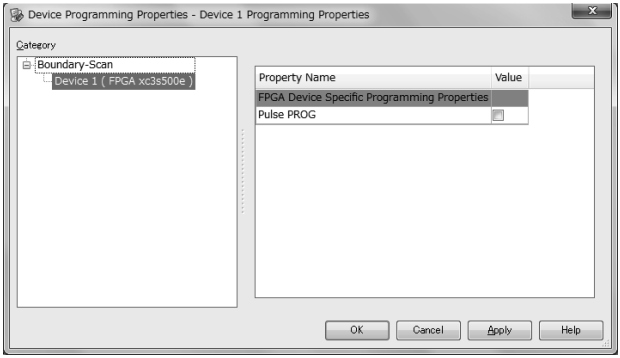


図 12: Device Programming Properties

最後に、図 13 のように「Program Succeeded」と表示されたのを確認後、メニューバーから[Output]→[SVF File]→[Stop Writing to File]を選択し、SVF ファイルの作成を終了します。

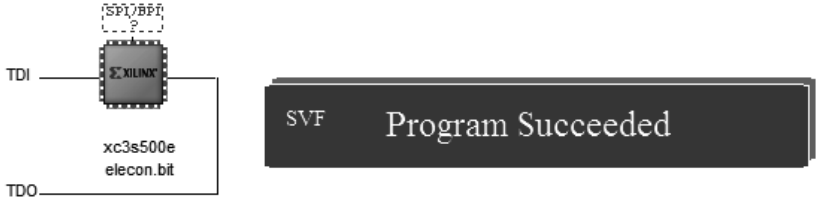


図 13: SVF ファイル作成完了

以上の手順で SVF ファイルが生成されます。

5.1.2. UrJTAG の実行

- 準備

UrJTAG と FT2232 ドライバと libusb-win32 が必要になります。ここでは、それぞれのインストールについて説明します。

➢ UrJTAG

UrJTAG は以下のページからインストーラをダンロードしてください。インストーラを実行する際は、インストーラを右クリックし「管理者として実行」を選択して、インストールしてください。

```
http://urjtag.org/
```

➢ FT2232 ドライバ

Windows 7 では、FT2232 を接続すると FT2232 ドライバは自動でインストールされます。USB ケーブルで PC と AZPR EvBoard を接続し、電源を ON にすると FT2232 ドライバのインストールが始まります。

➢ libusb-win32

libusb-win32 は以下のページからダウンロードしてください。

```
http://sourceforge.net/apps/trac/libusb-win32/wiki/
```

PC と AZPR EvBoard を接続したまま、ダウンロードしたファイルを解凍し、「bin」フォルダ内の「inf-wizard.exe」を実行します。図 14 で示す「libusb-win32 Inf-Wizard」ダイアログが表示されます。

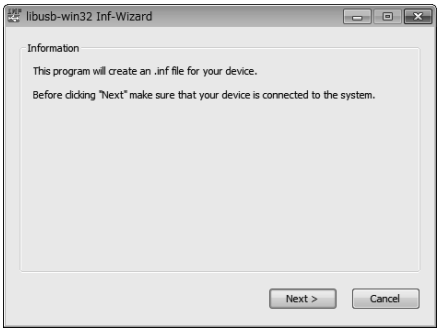


図 14: libusb-win32 Inf-Wizard (1/3)

[Next]ボタンをクリックすると、図 15 で示す画面が表示されます。

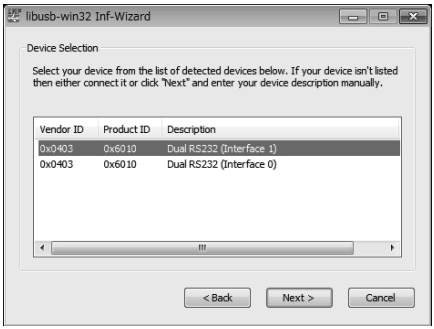


図 15: libusb-win32 Inf-Wizard (2/3)

ここで、どれか一つデバイスを選択し、[Next]ボタンを押します。図 15 では、「Description」が「Dual RS232 (Interface 1)」のデバイスを選択しています。ダイアログを進めると、図 16 で示す画面が表示されます。

図 16: libusb-win32 Inf-Wizard (3/3)

ここで[Install Now]ボタンを押し、ドライバをインストールします。

- 設定

UrJTAG のインストールディレクトリ(C:¥Program Files (x86)¥UrJTAG¥data¥xilinx)に以下のファイルを上書きして配置します。ファイルはサポートページからダウンロードしてください。

```
C:¥Program Files (x86)¥UrJTAG¥data¥xilinx
+ PARTS
+ xc3s500e
+ xcf04s
```

・ FPGA の認識

JTAG Shell を起動し、以下のコマンドを入力します。

```
jtag> cable jtagkey
jtag> detect
```

JTAG Shell に以下のように表示されれば、FPGA が認識されています。

```
jtag> detect
IR length: 14
Chain length: 2
Device Id: 01000001110000100010000010010011 (0x0000000041C22093)
  Manufacturer: Xilinx
  Part(0):      xc3s500e
  Stepping:     4
  Filename:     c:¥program files (x86)¥urjtag¥data/xilinx/xc3s500e/xc3s500e
Device Id: 11010101000001000110000010010011 (0x00000000D5046093)
  Manufacturer: Xilinx
  Part(1):      xcf04s
  Stepping:     0
  Filename:     c:¥program files (x86)¥urjtag¥data/xilinx/xcf04s/xcf04s
```

・ コンフィギュレーション

FPGA の認識で表示されたように、xc3s500e が part 0、xcf04s が part 1 になります。

BIT ファイルを選択して SVF ファイルを作成したときは、以下のコマンドを実行して、xc3s500e を選択します。

```
jtag> part 0
```

MCS ファイルを選択して SVF ファイルを作成したときは、以下のコマンドを実行して、xcf04s を選択します。

```
jtag> part 1
```

以下に、xcf04s のコンフィギュレーションを行うときのコマンドの入力例を示します。「D:¥sample.svf」は「5.1.1 SVF ファイルの生成」で生成した SVF ファイルのファイルパスになります。

```
jtag> part 1
jtag> svf D:¥sample.svf progress
Parsing   4690/4691 ( 99%)
Scanned device output matched expected TDO values.
jtag>
```

5.2. cblsrv-0.1_ft2232 を利用したコンフィギュレーション

cblsrv-0.1_ft2232 は fenrir 氏によって作成されたツールです。Xilinx 社のダウンロードツールである iMPACT と組み合わせて、AZPR EvBoard に搭載されている FT2232 を経由でコンフィギュレーションを行うことができます。この方法では、BIT ファイルや MCS ファイルを使うため、SVF ファイルを作る必要はありません。

注意：cblsrv-0.1_ft2232 の使用によっていかなる損害が生じても fenrir 氏及びれすぽんは一切の責任は負いませんのでご了承ください。また、cblsrv-0.1_ft2232 は ISE11 までしか動作確認できていません。最新版の ISE では動作しない可能性があります。Windows 7 は ISE 12 からの対応のため、ここでは Windows XP(32bit 版)での手順を示します。

- 準備

cblsrv-0.1_ft2232 と Microsoft Visual C++ 2008 再頒布可能パッケージ(x86)と FT2232 のドライバが必要になります。

- cblsrv-0.1_ft2232

cblsrv-0.1_ft2232 は fenrir 氏のホームページからダンロードできます。

```
http://fenrir.naruoka.org/archives/000644.html
```

バグの修正が行われているので、なるべく最新のバージョンを使ってください。(執筆時点では cblsrv-0.1_ft2232_r4804.zip が最新のバージョンです。)ダウンロードしたファイルを解凍します。以降は、解凍先のフォルダを「C:¥AZPR¥cblsrv-0.1_ft2232_r4804」として説明します。

- Microsoft Visual C++ 2008 再頒布可能パッケージ(x86)

以下のページから Microsoft Visual C++ 2008 再頒布可能パッケージ(x86)をダウンロードしてインストールしてください。

```
http://www.microsoft.com/downloads/details.aspx?familyid=9B2DA534-3E03-4391-8A4D-074B9F2BC1BF&displaylang=ja
```

- FT2232 ドライバ

Windows 7 に接続した場合、自動的にドライバがインストールされますが、Windows XP では以下のページから FT2232 ドライバをダウンロードしてください。Virtual COM Port Driver という名前で公開されています。

```
http://www.ftdichip.com/
```

ダウンロードしたファイルを解凍し、USB ケーブルで PC と AZPR EvBoard を接続してください。電源を ON にすると新しいハードウェアの検出ウィザードが表示されるので、先ほど解凍したフォルダから「i386」フォルダを指定して、ドライバをインストールしてください。

• コンフィギュレーション方法

ここでは、MCS ファイルを使って xcf04s をコンフィギュレーションする方法を紹介します。BIT ファイルを使って xc3s500e をコンフィギュレーションも同じ方法で行えるので、適宜読み替えてください。

まず、AZPR EvBoard とパソコンを USB ケーブルで接続し、電源スイッチを ON にします。次に、コマンドプロンプトを起動し、cblsrv-0.1_ft2232 の実行ファイル「cblsrv.exe」が「C:¥AZPR¥cblsrv-0.1_ft2232_r4804¥build¥win32¥Release」へ移動し、以下のようにコマンドを入力します。(※「Windows セキュリティの重要な警告」が表示された場合は、「ブロックを解除する」ボタンを押してください。)

```
C:¥AZPR¥cblsrv-0.1_ft2232_r4804¥build¥win32¥Release>cblsrv.exe -c amontec -p 1234
```

iMPACT を起動します。(図 17)

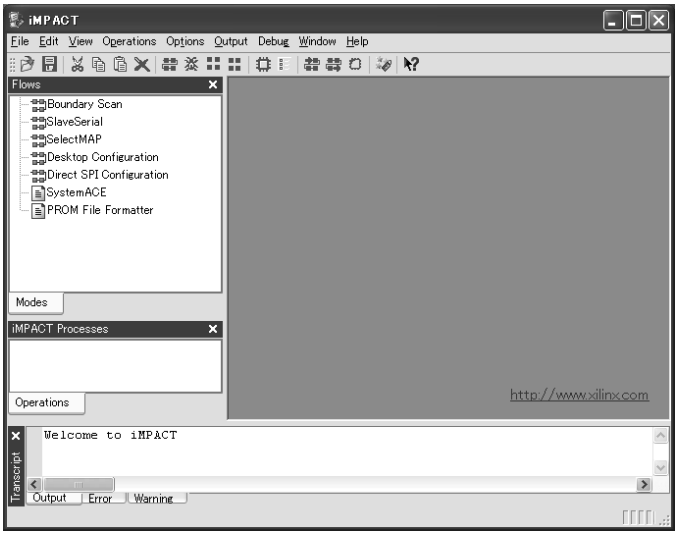


図 17: iMPACT の画面

ウィンドウ左上の領域から「Boundary Scan」をダブルクリックし、右のフレーム内に「Right click to Add Device or Initialize JTAG chain」という文字を表示させます。

その後、メニューから[Output]→[Cable Setup]を選択し、Cable Communication Setup ウィンドウを表示させます。図 18 のように「Communication Mode」の項目の Parallel Ⅲを選択、「Cable Location」の項目の「Remote」を選択、「Host Name」に[localhost:1234]と入力し、OK ボタンを押します。ここで指定する数字は、「cblsrv.exe」の-p オプションで指定したポート番号と一致させます。

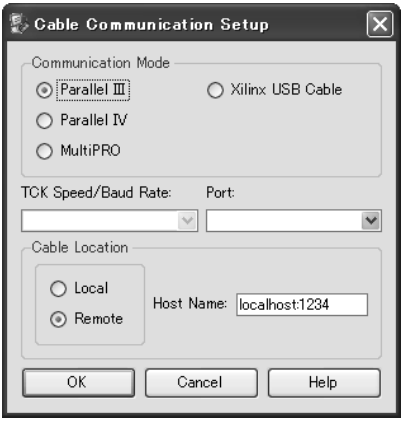


図 18: Cable Communication Setup ウィンドウ

「Right click to Add Device or Initialize JTAG chain」と表示されているフレーム内を右クリックし、「Initialize Chain」を選択します。図 19 のように xcf04s と xc3s500e を認識します。

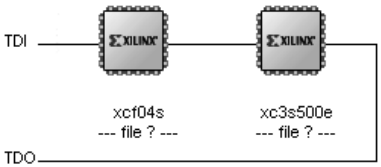


図 19: xcf04s と xc3s500e を認識した画面

xcf04s を右クリックし、「Assign New Configuration File」を選択するとダイアログが表示されます。ここで MCS ファイルを選択します。再び xcf04s を右クリックし、Program を選択し、OK ボタンを押してコンフィギュレーションが成功すると、図 20 のように「Program Succeeded」と表示されます。

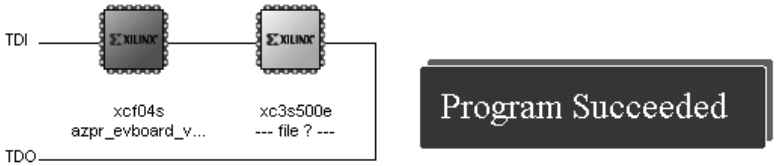


図 20: 「Program Succeeded」と表示された画面

これで、コンフィギュレーションは完了です。

iMPACT 実行中に問題が発生したときは、コマンドプロンプトで CTRL+C を押して「cblsrv.exe」を終了し、iMAPCT を閉じてください。その後、コンフィギュレーション方法の最初からやり直してください。

5.3. 外部ダウンロードケーブルからのコンフィギュレーション

AZPR EvBoard は、外部ダウンロードケーブルによるコンフィギュレーションが可能です。外部ダウンロードケーブルを利用することで、IMPACT から直接 FPGA をコンフィギュレーションすることができます。

外部ダウンロードケーブルを使用する際は、JP1 にジャンパーを接続し、基板上のシルクにしたがって JTAG ピンを接続します。このとき、サンハヤト社製の「テストワイヤ TTW-200」を利用するとスルーホールへのアクセスが可能になります。

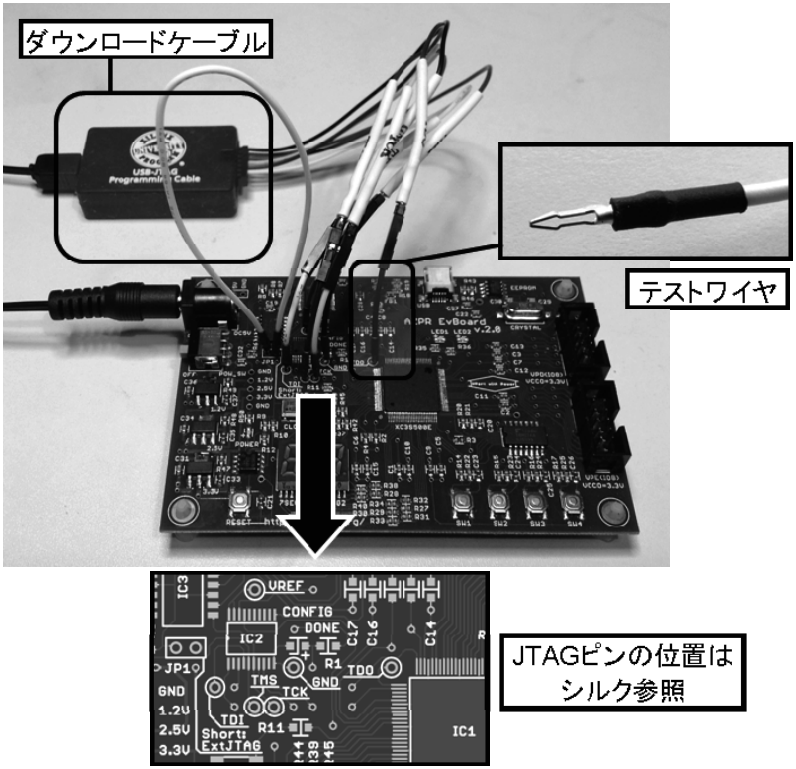


図 21: 外部ダウンロードケーブルの接続

6. ダイアグプログラム

AZPR EvBoard の出荷時のコンフィグ ROM には、FPGA-部品間の接続チェックを行うためのダイアグプログラムが格納されています。AZPR EvBoard の電源を入れることで、ダイアグプログラムを実行します。また、ダイアグプログラムはサポートページからダウンロードすることが可能です。

6.1. 事前準備

ダイアグプログラムの実行時には、下記のようにケーブル類を接続します。

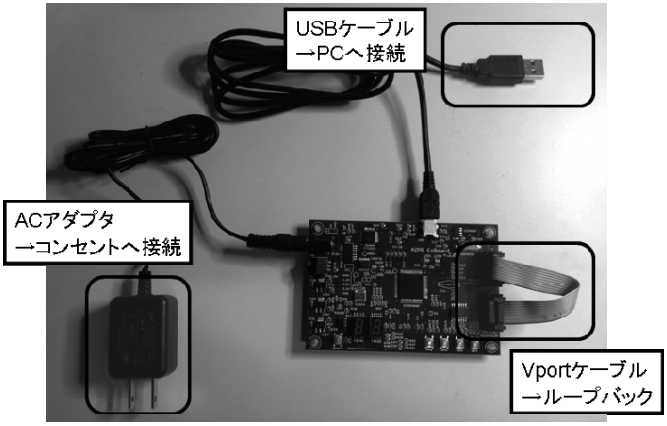


図 22: ダイアグプログラム実行用の接続

PC 上で端末エミュレータである TeraTerm を起動し、AZPR EvBoard の COM ポートと接続します。PC に TeraTerm がインストールされていない場合は、以下のページからダウンロードして、インストールしてください。

```
http://sourceforge.jp/projects/ttssh2/releases/
```

USB Serial Port に接続し、TeraTerm を下記のように設定します。

- 受信改行コードの設定: LF
- シリアルポートのボーレードの設定: 38400

6.2. ダイアグプログラムの実行

- LED テスト

起動直後、LED1 と LED2 が交互に 3 回ずつ点滅します。LED の点灯に問題がないことを確認してください。

- 7 セグテスト

7 セグメント LED が「11」→「22」→「33」→･･･→「77」→「00」と表示します。すべての数字が問題なく出力されていることを確認してください。

・ UART テスト

TeraTerm 上に以下の文字列が表示されるので、キーボードの"n"キーを押してください。

```
Press "n" key
```

「UART test: Pass」の表示が出力されたら、UART テストは Pass となります。

```
Press "n" key
n
UART test: Pass
```

・ VPort テスト

UART テストに続けて、VPort テストを行います。「VPort test: Pass」の表示が出力されたら、VPort のテストは Pass となります。

```
VPort connect...
VPort test: Pass
```

・ SW テスト

VPort テストに続けて、スイッチのテストを行います。SW1 から SW4 までを順番に押してください。TerTerm 上に以下の表示が出力されます。「SW test: Pass」の表示が出力されたら、SW テストは Pass となります。

```
SW test:
SW1: ON
SW1: OFF
SW2: ON
SW2: OFF
SW3: ON
SW3: OFF
SW4: ON
SW4: OFF
SW test: Pass
```

・ テスト終了

上記のテストがすべて終わると、以下の表示が出力されます。

```
All test Passed!
-----
UART test: Pass
VPort test: Pass
SW test: Pass
```

以上でダイアグプログラムのすべてのテストは終了です。

7. Appendix-A 電子サイコロの実装

サンプルアプリケーションとして、電子サイコロの実装を行います。電子サイコロは下記の仕様で設計します。

- リセット後、停止状態に遷移する
- スイッチ 1 を押すと回転状態に遷移する
- 再びスイッチ 1 を押すと停止状態に遷移する
- 内部にカウンタを持ち、カウンタの状態に応じてサイコロの値を表示する

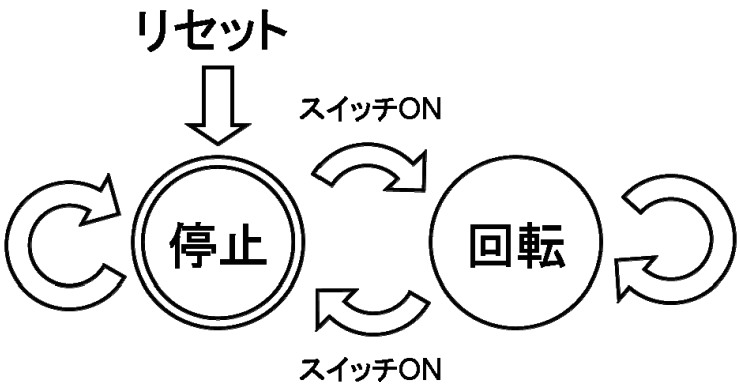


図 23: 電子サイコロの状態遷移図

リスト 1 に電子サイコロのソースコードを示します。

リスト 1　　電子サイコロのソースコード

[I]モジュール宣言

```
11 module dice (
12     input  wire       clk,                    // クロック
13     input  wire       reset_,                 // 非同期リセット
14     input  wire       sw_,                    // プッシュスイッチ
15     output reg  [6:0] led_                    // 7セグメントLED
16 );
17
```

[II]内部信号の定義

```
18     /********** 内部信号 **********/
19     // 汎用パラメータ
20     localparam        LOW        = 1'b0;      // Low Level
21     localparam        HIGH       = 1'b1;      // High Level
22     // LED点灯パラメータ (gfedcba)
23     localparam        LED_INIT   = 7'b111_1111; // 全消灯
24     localparam        LED_1      = 7'b111_1001; // 1 :     cb  を点灯
25     localparam        LED_2      = 7'b010_0100; // 2 : g ed ba を点灯
26     localparam        LED_3      = 7'b011_0000; // 3 : g  dcba を点灯
27     localparam        LED_4      = 7'b001_1001; // 4 : gf  cb  を点灯
28     localparam        LED_5      = 7'b001_0010; // 5 : gf dc a を点灯
29     localparam        LED_6      = 7'b000_0010; // 6 : gfedc a を点灯
30     // サイコロの状態
31     localparam        STATE_STOP = 1'b0;      // 停止状態
32     localparam        STATE_ROLL = 1'b1;      // 回転状態
33     reg               state;                  // 状態変数
34     // 表示用の分周カウンタ
35     localparam        DIV_RATIO  = 200000 - 1;  // 分周比 = 10MHz / 50 Hz - 1
36     reg [17:0]        div_cnt;                // 分周カウンタ
37     // サイコロ用カウンタ
38     reg [2:0]         dice_cnt;               // カウンタ
39     // スイッチ用ラッチ
40     reg               sw_l;                   // ラッチ
41     reg               sw_dl;                  // ダブルラッチ
```

(1)汎用パラメータ
(2)LED点灯パラメータ
(3)サイコロの状態
(4)表示用分周カウンタ
(5)サイコロ用カウンタ
(6)スイッチ用ラッチ

[III]LED点灯制御

```
42
43     /********** LED点灯制御 **********/
44     always @(*) begin
45         case (dice_cnt)
46             3'd1    : led_ = LED_1;    // 1
47             3'd2    : led_ = LED_2;    // 2
48             3'd3    : led_ = LED_3;    // 3
49             3'd4    : led_ = LED_4;    // 4
50             3'd5    : led_ = LED_5;    // 5
51             3'd6    : led_ = LED_6;    // 6
52             default : led_ = LED_INIT; // 全消灯
53         endcase
54     end
55
```

[IV]ラッチ制御

```
56      /********** スイッチ用ラッチ制御 **********/
57      always @ (posedge clk or negedge reset_) begin
58          if (reset_ == LOW) begin // 非同期リセット
59              sw_l  <= #1 HIGH;
60              sw_dl <= #1 HIGH;
61          end else begin
62              sw_l  <= #1 sw_;  // ラッチ
63              sw_dl <= #1 sw_l; // ダブルラッチ
64          end
65      end
```

[V]状態制御

```
66
67      /********** 状態制御 **********/
68      always @ (posedge clk or negedge reset_) begin
69          if (reset_ == LOW) begin // 非同期リセット
70              state    <= #1 STATE_STOP;
71              dice_cnt <= #1 3'd1;
72              div_cnt  <= #1 'h0;
73          end else begin
74              case (state)
75                  STATE_STOP : begin // 停止状態
76                      if ((sw_dl == HIGH) && (sw_l == LOW)) begin // スイッチON
77                          state    <= #1 STATE_ROLL;
78                      end
79                  end
80                  STATE_ROLL : begin // 回転状態
81                      if ((sw_dl == HIGH) && (sw_l == LOW)) begin // スイッチON
82                          state    <= #1 STATE_STOP;
83                      end else begin
84                          /* 表示用分周カウンタの制御 */
85                          if (div_cnt == DIV_RATIO) begin // カウンタ満了
86                              div_cnt  <= #1 'h0;
87                              /* サイコロ用カウンタの制御 */
88                              if (dice_cnt == 3'd6) begin // カウンタ満了
89                                  dice_cnt <= #1 3'd1;
90                              end else begin              // カウントアップ
91                                  dice_cnt <= #1 dice_cnt + 1'b1;
92                              end
93                          end else begin                  // カウントアップ
94                              div_cnt  <= #1 div_cnt + 1'b1;
95                          end
96                      end
97                  end
98              endcase
99          end
100     end
101
102 endmodule
```

(7)非同期リセット
(8)停止状態
(9)回転状態
(10)スイッチの判定
(11)表示用分周カウンタの制御
(12)サイコロ用カウンタの制御

[I]　モジュール宣言

「dice」というモジュールを宣言しています。入力はクロック(clk)とリセット(reset_)とプッシュスイッチ(sw_)。出力は 7 セグメント LED(led_)です。

[II]　内部信号の定義

(1)で汎用パラメータの LOW と HIGH を定義しています。

(2)で LED 点灯用のパラメータを定義しています。 (3)でサイコロの状態用パラメータと信号を定義しています。電子サイコロは「停止状態」と「回転状態」の 2 つの状態を取ります。

(4)で表示用の分周カウンタのパラメータと信号を定義しています。電子サイコロが「回転状態」の時は、分周カウンタが満了する度に表示される値が変わります。今回は表示変更の周期を 50Hz としました。そのため、分周比は AZPR EvBoard のオシレータの周波数 10MHz を 50Hz で割った値となります。実際には分周カウンタは 0 からカウントするため、マイナス 1 した値となります。199,999 をカウントするのに必要なビット数は 18 ビット(2^18 = 262,144 < 199,999)となるので、分周カウンタ(div_cnt)は 18 ビットになります。

(5)でサイコロ用のカウンタを定義しています。サイコロは 1～6 の値をカウントするため、3 ビット(2^3 = 8 < 6)となります。

(6)でスイッチ用のラッチ(レジスタ)を定義しています。スイッチ入力のエッジ検出を行うため、ラッチを 2 個使用します。

[III]　LED の点灯制御

サイコロ用カウンタ(dice_cnt)に応じて LED(led_)に点灯パターンを出力しています。

[IV]　スイッチ用ラッチ制御

プッシュスイッチの入力(sw_)を「sw_l」でラッチし、さらにそれを「sw_dl」でダブルラッチしています。状態制御の部分で「sw_l」と「sw_dl」を比較し、でエッジの検出を行います。

[V]　状態制御

(7)でリセットスイッチが押されて LOW になったら回路を非同期リセットします。

(8)で停止状態の制御を行っています。停止状態では現在の値を保持し続けます。プッシュスイッチのダウンエッジを検出(「sw_l」が HIGH、「sw_dl」が LOW)した場合、回転状態に遷移します。

(9)で回転状態の制御を行っています。

(10)でプッシュスイッチのダウンエッジを検出した場合、停止状態に遷移します。

(11)で表示用分周カウンタの制御を行っています。表示用文集カウンタ(div_cnt)が満了した場合、(12)にてサイコロ用のカウンタを更新しています。