

「さわれる」遠隔学習システム

利用マニュアル

藤枝 直輝（愛知工業大学）

Version 1.0.0

目次

第 1 章	「さわれる」遠隔学習システムの概要	1
1.1	システムの目的	1
1.2	必要な機材・ソフトウェア一式	1
1.3	遠隔サーバの構成	2
1.4	FPGA 内の回路構成	3
1.5	サポートする FPGA ボード	4
1.5.1	Nexys A7-100T	4
1.5.2	Arty A7-35T	5
1.5.3	CMod A7-35T	5
第 2 章	クイックスタート・ガイド	7
2.1	アカウントの確認とパスワードの変更	7
2.2	回路設計と必要なファイルの作成	8
2.3	必要なファイルの転送	10
2.4	遠隔サーバへの接続	10
2.5	リモートデスクトップ接続と Vivado の起動	11
2.6	回路の論理合成と実装	12
2.6.1	作成した回路の論理合成	12
2.6.2	ベース設計との統合と実装	13
2.7	FPGA への書き込み	14
2.8	Connector アプリ（サーバ側）の準備	15
2.9	Connector アプリ（PC 側）の準備と動作確認	16
2.10	接続を終了する前に	16
第 3 章	コマンド体系	18
3.1	コマンドの概要	18
3.2	ボードの認識に関するコマンド	18
3.3	出力に関するコマンド	19
3.4	入力に関するコマンド	19
第 4 章	入出力制御回路	21

4.1	ベース設計の内部構造	21
4.2	ベース設計の論理合成	22
第 5 章	コントローラボード	24
5.1	ボードの基本仕様	24
5.2	制御プログラムの書き込み	24
参考文献		28

第 1 章

「さわれる」遠隔学習システムの概要

1.1 システムの目的

2019 年末に発生した新型コロナウイルス感染症 (COVID-19) の拡大により、人同士の接触を避ける観点から、本邦でもほとんどの大学で講義・実験の遠隔化、オンライン化を迫られました。特に実験・実習科目においては、いかに実地での実験・実習と同等の教育効果を遠隔環境で達成するかが、重要な課題となります。また、もしもそのようなシステムがあれば、たとえ実地での実験・実習が可能である場合でも、実験時間外の自主学習に活用できます。

しかしながら、デジタル回路や FPGA の学習におけるこれまでの遠隔学習システムには、その操作から実際にハードウェアに触れているとの実感を得ることが難しい、という問題点がありました。例えば、ACRi ルーム [1] 上の仮想マシンには FPGA ボードが接続されているものの、そのスイッチや LED 等を直接操作・確認することはできませんでした。

我々（愛知工業大学 工学部電気学科 デジタルシステム研究室）がこの問題点に対する解決法として開発したのが、本マニュアルで説明する、デジタル回路の「さわれる」遠隔学習システムです。以下、本マニュアルではこのシステムを SawareruSys と表記します。詳しい学術的背景や、システムの設計・評価については、文献 [2] を参照してください。

1.2 必要な機材・ソフトウェア一式

遠隔実験の実施には、FPGA ボードの入出力の変化を手元で確認するためのコントローラボードが必要です。2024 年 3 月現在、コントローラボードには以下の 3 つのバリエーションがあります。

1. FPGA リモコンボード V2
2. FPGA リモコンボード V4
3. SawareruBoard V1

このうち 1 と 3 のボードの外景を、図 1.1 に示します。1 のボードは、左・中央・右の 3 つのタクトスイッチを備え、PC との接続端子は USB mini-B です。2 と 3 のボードは機能的・形状的に同一で、十字型の 5 つのタクトスイッチを備え、接続端子は USB Type-C です。

また、SawareruSys の配布パッケージには、以下のファイル一式が含まれています。

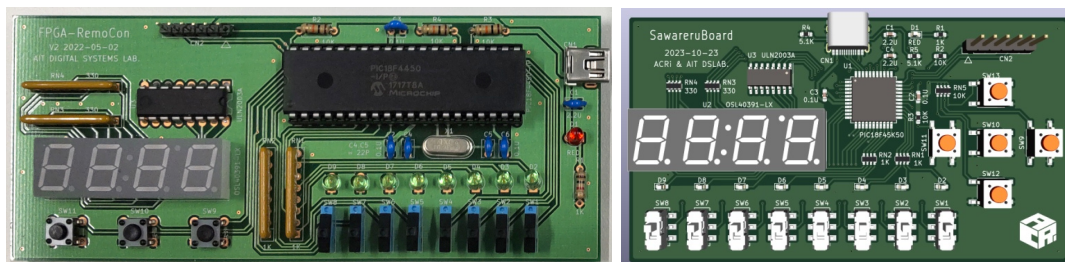


図 1.1 コントローラボードの例 (左: FPGA リモコンボード V2, 右: SawareruBoard V1).

- DRFront: SawareruSys での開発を支援するためのフロントエンドツール
- Connector アプリ: コントローラボードを遠隔サーバに接続するときに使用
- WinSCP: 遠隔サーバへファイルを転送するときに使用

加えて, Windows 標準の以下のアプリケーションも使用します.

- Windows PowerShell
- リモートデスクトップ接続

なお, WinSCP は Martin Prikryl 氏による著作物であり, GPL (version 3) ライセンスに従って, ポータブル版の実行ファイルを再配布しています. ライセンスに関する詳細は, 配布パッケージの COPYING ファイルを参照してください.

1.3 遠隔サーバの構成

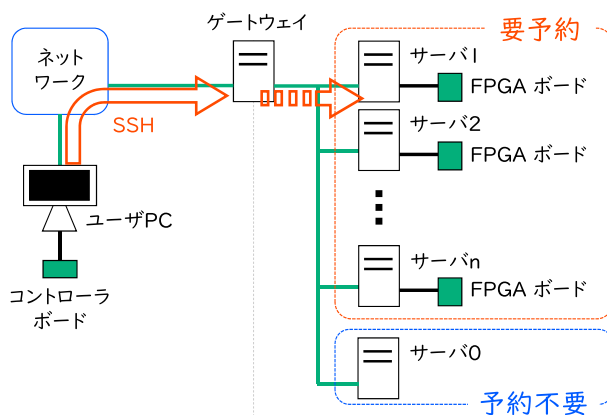


図 1.2 遠隔サーバの構成図.

SawareruSys における遠隔サーバの構成図を, 図 1.2 に示します. 遠隔サーバへの入口としてゲートウェイサーバが設置されており, ユーザは SSH 接続でゲートウェイサーバに接続します. 開発用のサーバへのアクセスは, SSH ポートフォワーディング機能により間接的に行います. 開発用のサーバには, FPGA ボードが 1 台ずつ設置された予約の必要なサーバと, FPGA ボードが接続されていない予約不要のサーバとがあります.

2024 年 3 月時点の ACRi ルームでは、ゲートウェイサーバには fserv4, 開発用のサーバには vs000～vs910 の名前がつけられています。このうち末尾 2 桁が 00 であるサーバは予約不要のサーバです。愛工大の学内環境では、ゲートウェイサーバには gs1, 開発用のサーバには ns0～ns5 の名前がつけられています。ns0 は予約不要です。予約が必要なサーバは、Web ブラウザで予約を行います。予約について詳しくは、各環境のドキュメントまたは利用説明ページを確認してください。

PC と開発用のサーバでは、それぞれ別々の Connector アプリを起動して、通信の中継を行います。PC とコントローラボード、サーバと FPGA ボードの通信には、それぞれで USB-UART (USB を介したシリアル通信) を使用します。一方、PC とサーバとの通信は、SSH ポートフォワーディングを介した TCP/IP 通信によります。このプロトコルの違いを吸収し、通信の中継を行うのが、Connector アプリです。

1.4 FPGA 内の回路構成

SawareruSys では、入出力の遠隔操作を実現するために、FPGA 内部にあらかじめ入出力制御回路を作りこんでおきます。ユーザは、入出力制御回路と接続されるユーザ回路部分のみを開発します。

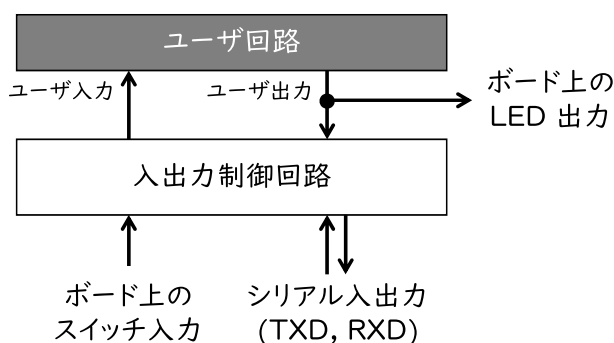


図 1.3 FPGA の内部構造の概要。

入出力制御回路とユーザ回路の関係の概要を、図 1.3 に示します。入出力制御回路はシリアル通信の入出力 (TXD と RXD) をもっており、サーバの Connector アプリと接続されたかどうかや、入出力の変化の状況をチェックしています。FPGA ボードのスイッチ入力は、まず入出力制御回路に入ります。サーバと接続されていない場合は、単にスイッチ入力の情報がユーザ回路の入力として渡されます。サーバと1度でも接続された場合は、入出力制御回路はコントローラボードのスイッチの状態を受け取って保持し、その結果をユーザ回路の入力とします。ユーザ回路の出力は、ボードの LED (または 7 セグメント LED) 出力として使われるほか、入出力制御回路にも渡されます。サーバと接続されている間、入出力制御回路はユーザ回路の出力の変化を監視し、その結果をサーバに送ります。

SawareruSys では、ユーザ回路の入出力として、以下のものをサポートしています。

- CLK: 100 MHz クロック
- RST: リセットスイッチ (正論理) 1 個
- SW(15)～SW(0): スライドスイッチ 16 個
- BTNU, L, C, R, D: タクトスイッチ 5 個 (それぞれ, 上, 左, 中央, 右, 下)
- LD(15)～LD(0): LED 16 個

- AN(7)～AN(0), CA～CG, DP: 7セグメント LED（負論理, 8桁のダイナミック点灯）1個

ただし, SW(15)～SW(8) はコントローラボードから操作できず, LD(15)～LD(8) と 7セグメント LED の左側 4桁（AN(7)～AN(4) に対応）はコントローラボードから確認できません. リセットスイッチは, ユーザ回路と入出力制御回路の両方をリセットします. サーバと接続されていた場合には, 接続されていない状態に戻りますので, ボードのスイッチ入力が再び使えるようになります.

FPGA において, あらかじめ作りこまれた回路に自由に書き換えられるユーザ回路を組み込んで使うための 1つの方法に, 動的部分再構成 (DPR; Dynamic Partial Reconfiguration) があります. AMD 社の FPGA における DPR 機能は, DFX (Dynamic Function eXchange) とよばれます. DFX 機能を使うためには, 同社の FPGA 開発環境である Vivado 上で, 多数のステップからなるスクリプトを実行する必要があります. DRFront は, SawareruSys での DFX に必要な手順を自動化・効率化するために開発されたフロントエンドツールです.

1.5 サポートする FPGA ボード

2024 年 3 月現在, SawareruSys では Digilent 社の Nexys A7-100T, Arty A7-35T, CMod A7-35T の 3 種類の FPGA ボードをサポートしています.

1.5.1 Nexys A7-100T

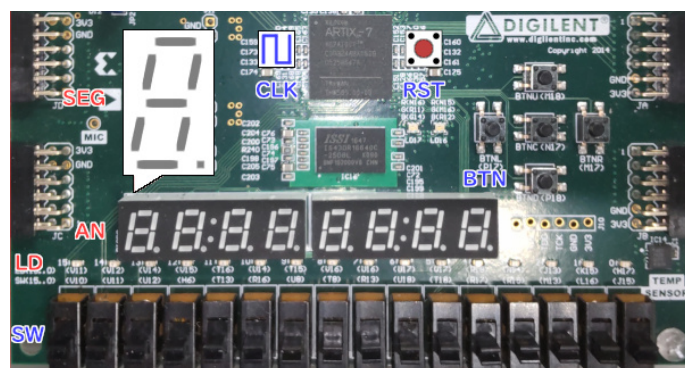


図 1.4 Nexys A7-100T で使用できる入出力.

Nexys A7-100T で物理的にアクセスできる入出力には, 図 1.4 に示す 16 個のスライドスイッチ, 5 個のタクトスイッチ, 16 個の LED, 8 桁の 7セグメント LED が含まれます. これらすべてをユーザ回路上で扱えますが, スライドスイッチ・LED・7セグメント LED のうち, 左半分はコントローラボードでの確認・操作に対応していないことに注意してください. リセットスイッチは, ボード右半分に 2 つ並んだ赤色のタクトスイッチのうち, 左側の CPU RESET と書かれたスイッチです.

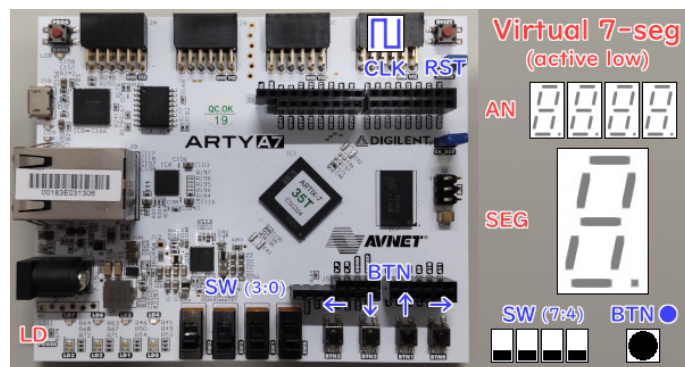


図 1.5 Arty A7-35T で使用できる入出力および仮想入出力。

1.5.2 Arty A7-35T

Arty A7-35T は、Nexys A7-100T よりは安価なボードで、図 1.5 に示す 4 個のスライドスイッチ、4 個のタクトスイッチ（上下左右）、8 個の LED に物理的にアクセスできます。LED のうち 4 個は RGB LED ですが、SawareruSys では緑色の表示にのみ対応します。コントローラボードを使って遠隔接続すると、仮想的に 4 桁の 7 セグメント LED と、追加の 4 個のスライドスイッチ、1 個のタクトスイッチ（中央）が利用できます。リセットスイッチは、ボード右上の RESET と書かれたスイッチです。

1.5.3 CMod A7-35T

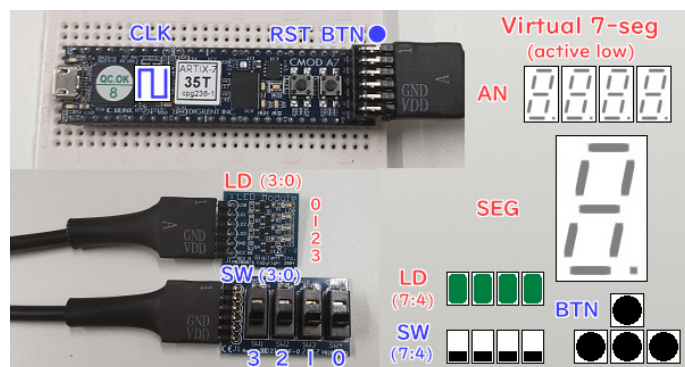


図 1.6 CMod A7-35T で使用できる入出力および仮想入出力。

CMod A7-35T は、ブレッドボードに差し込んで利用できる FPGA ボードで、サポートされているボードの中では最も安価なボードですが、最小限の入出力装置しか備えていません。そのため、Pmod ケーブルキットの 12 ピン-デュアル 6 ピンケーブルを使用し、Pmod 端子のうち 1~6 ピンに PMod LED を、7~12 ピンに PMod SWT を、それぞれ接続する使い方を仮定します。接続例を図 1.6 に示します。あるいは、Pmod 端子を使わずに、ほぼすべてをコントローラボードから操作・確認する仮想入出力とみなすのもよいでしょう。

Pmod 端子を使用した場合、CMod A7-35T を含む機材一式には、4 個のスライドスイッチ、1 個のタク

トスイッチ（中央）、4 個の LED が含まれます。ボード上の 2 個のタクトスイッチのうち、ユーザ回路に接続されるのは右側のスイッチです。左側はリセットスイッチとして使用します。コントローラボードを使って遠隔接続すると、4 桁の 7 セグメント LED と、追加の 4 個のスライドスイッチ、4 個のタクトスイッチ（上下左右）、4 個の LED が仮想的に利用できます。

PMod 端子を使用しない場合、CMod A7-35T には、1 個のタクトスイッチ（中央）と 3 個の LED が含まれます。ボード上の 2 個のタクトスイッチは、右側がユーザ回路に接続され、左側がリセットスイッチとして使われます。LED のうち 1 個は RGB LED ですが、SawareruSys では緑色の表示にのみ対応します。コントローラボードを使うと、4 桁の 7 セグメント LED、8 個のスライドスイッチ、追加の 4 個のタクトスイッチ（上下左右）、5 個の LED が仮想的に利用できます。

CMod A7-35T のボードに搭載されたオシレータの出力周波数は 12 MHz ですが、SawareruSys ではこれをクロック生成回路（MMCM）で 100 MHz にしてから、ユーザ回路および入出力制御回路に与えています。

第 2 章

クイックスタート・ガイド

本章では、コントローラボードと遠隔サーバのアカウントを入手した人が、コントローラボードを使って回路の動作確認を最初に行うまでの間に必要な手順を説明します。

2.1 アカウントの確認とパスワードの変更

注: 既にパスワードの変更を済ませている場合、本節の作業は不要です。

遠隔サーバのアカウントが発行されたら、まずは遠隔サーバにログインできるか確認し、仮パスワードを自分で決めたパスワードに変更してください。遠隔サーバへのログインには PowerShell を使用しますが、PowerShell はデフォルト設定ではフォントの問題で日本語が文字化けする場合があります。そのため、その対処を行ってから作業することをおすすめします。

Windows PowerShell をスタートメニューから開き、タイトルバーを右クリックして、「プロパティ」を選択します。ダイアログの「フォント」タブから好みの日本語フォントを選択して、OK ボタンを押すと、設定が反映されます。

その後、ネットワーク（愛工大の学内環境を使う場合は、学内 LAN）に接続された PC で、PowerShell から以下のコマンドを入力します。

```
ssh <ユーザ名>@<接続先>
```

ssh のあとには半角スペースが必要であることを注意してください。例えばユーザ名が `tu_fujieda` であり、接続先が `gw.acri.c.titech.ac.jp` であれば、入力するコマンドは、

```
ssh tu_fujieda@gw.acri.c.titech.ac.jp
```

となります。

初回に限り、接続しようとしているサーバの確認を求められるので、`yes` と入力します。その後、パスワードの入力を求められるので、発行された仮パスワードを入力します。このとき文字を入力しても画面上には何の反応もありますが、それで正常です。

ログインに成功し、画面上に `<ユーザ名>: $` などといったプロンプトが表示されたら、`passwd`（環境によっては `yppasswd`）と入力してパスワードの変更を行います。まず仮パスワードを 1 回入力し、自分で決

めたパスワードを2回続けて入力します。このときもやはり入力内容は画面に反映されませんので、打ち間違いに気をつけてください。「パスワードは正しく更新されました」、あるいはそれに相当する英語のメッセージが表示されたら、この段階では PowerShell は閉じて構いません。

2.2 回路設計と必要なファイルの作成

SawareruSys では、あらかじめ遠隔操作のためのベース設計の回路が用意されています。ユーザは、その一部を各自で作成した回路に書き換えて使用します。作成した回路をベース設計に接続するための回路は、SawareruSys の一部である DRFront を用いて、自動的に生成します。回路を作成するためには、作成した回路の各入出力ポートが、FPGA ボードのどの信号に接続されるかを指定する必要があります。この設定も DRFront の画面上で行います。

まずは配布パッケージの DRFront フォルダにある DRFront.exe 実行ファイルをダブルクリックして、DRFront を起動します。デフォルトではハードウェア記述言語 (HDL) として VHDL を、FPGA ボードとして Nexys A7-100T を使用する設定になっていますので、これ以外の言語やボードを使う場合は、Settings ボタンから使用する言語を変更してください。また、自分の PC に Vivado をインストールしている場合、インストール先とバージョンを指定すると、このあとの論理合成と実装までの作業を自分の PC で行えます。これらの設定は最初の起動時に1回だけ必要です。

作成した回路の HDL 記述を1つのフォルダにまとめておきます。このとき、フォルダ名に日本語や半角スペースが含まれていると、Vivado の実行時に不具合が発生することがありますので、フォルダ名は英数字で作成するようにしてください。自分の PC で Vivado を動作させる場合、親フォルダについても同様です。

最上段の「Source Dir.」の入力欄の横に「...」と書かれたアイコンがあるので、それをクリックし、HDL 記述をまとめたフォルダを指定します。記述やディレクトリ名に問題がなければ、その下の段の「Create Project」というボタンが押せるようになっているので、このボタンをクリックします。すると、Project の項目が「project_1」に変わり、ボード画像の下テーブルに、回路の入出力の一覧が表示されます。このときの様子を図 2.1 に示します。

なお、ファイルやディレクトリに問題があり入出力の信号が検出できなかった場合、「Source Dir.」の入力欄が赤文字で表示されます。また、ディレクトリ名に日本語を含んでいるなど、後々問題になりうる場合には、入力欄が黄色の背景で表示されます。いずれもマウスカーソルを当てると問題点が表示されますので、それを参考にファイルやフォルダ名、保存場所などを適切に修正してください。

入出力の設定には3通りの方法があります。

1. 各信号の横の「Assign to」の列から、割当て先の信号を指定する。
2. 信号名をドラッグして、ボード画像内の割当てたい部品へとドロップする。
3. 自動割当てを使う。

第1の方法では、「Assign to」の列をクリックすると、割当て可能な信号（入力であればスイッチやボタン、出力であれば LED など）の一覧が表示されます。その中から割当てたい信号名を選ぶと、表の上側のボード画像の対応する部品を囲む枠が水色で表示されます。これで割当ては完了です。例として、入力信号

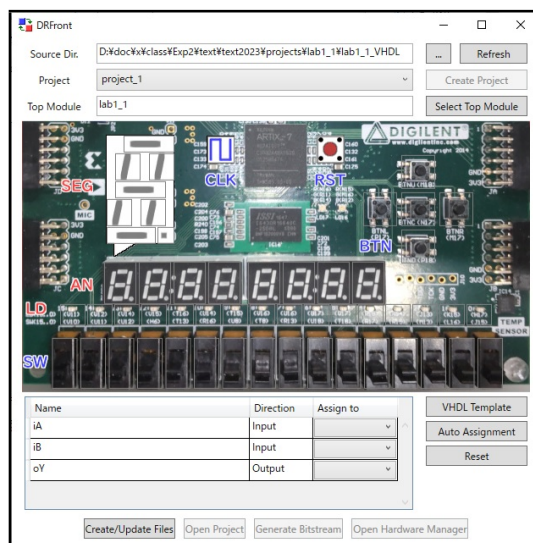


図 2.1 DRFront 上でプロジェクトを作成したときの様子。

iA を右端のスイッチ入力 SW(0) に接続したときの様子を図 2.2 に示します。

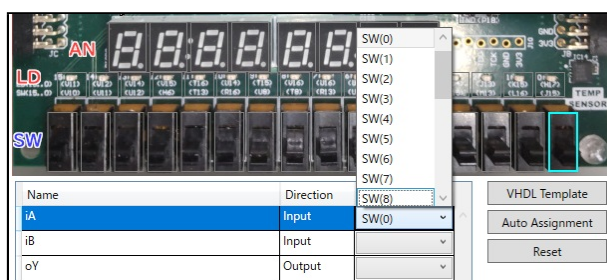


図 2.2 DRFront 上で入力信号を割当てたときの様子。

第 2 の方法では、割当てを行いたい信号の行をクリックして選択状態にしたあと、マウスをドラッグしてボード画像上の割当てたい部品のところへと移動させます。割当て可能であればマウスカーソルが切り替わり、枠が黄色で表示されるので、そのままマウスボタンを離します（ドロップします）。枠が水色で表示されれば、割当て完了です。

第 3 の方法では、表の右側にある「Auto Assignment」ボタンをクリックします。この方法では、まだ割当てが行われていない信号に対して、入力ならば SW(0), SW(1), ... が、出力ならば LD(0), LD(1), ... を信号名の順に割当てます。いくつかの信号を手動で割当てたあとで、自動割当てを行うこともできます。各自の好みの方法で信号の割当てを行ってください。

すべての入出力への割当てが完了したら、DRFront の左下の「Create/Update Files」ボタンを押します。すると、Vivado の実行に必要なファイルが生成された旨が表示され、「Open Project」ボタンが押せるようになります。

2.3 必要なファイルの転送

注: 自分の PC に Vivado をインストールしている場合、先に 2.6 節の「回路の論理合成と実装」の作業を行ってください。

DRFront で必要なファイルが生成できたら、WinSCP を使ってそれらを遠隔サーバに転送します。WinSCP を起動すると、セッション設定画面が表示されます。転送プロトコルには SFTP または SCP、ホスト名には接続先のホスト名または IP アドレス、ユーザ名には遠隔サーバのユーザ名を入力します。その後、必要に応じて保存ボタンで設定を保存してから、ログインボタンを押します。SSH の場合と同様、ここでも最初に 1 回だけサーバの確認があるので、Yes と回答します。パスワードの入力画面でパスワードを入力して、ログインしてください。

WinSCP のログイン後の画面では、通常左半分には自分の PC のファイル、右半分に遠隔サーバ上のファイルが一覧表示されます。ただし設定によっては、遠隔サーバ上のファイルのみが一覧表示される場合もあります。

まずは、遠隔サーバ上の適当な場所にフォルダを作成します。このとき、Desktop フォルダ上で行うと、あとでリモートデスクトップ接続したときにデスクトップにフォルダが表示されるので、見つけやすくなるかもしれません。右半分の画面の適当な空欄を右クリックし、「新規」→「ディレクトリ」でフォルダを作成します。次に、HDL 記述のあるフォルダを WinSCP の左半分の画面か、エクスプローラで開きます。フォルダ内にあるすべてのファイルとフォルダをドラッグし、WinSCP の右半分の画面上で作成したフォルダにドロップすると、ファイル一式が転送されます。

ACRi ルーム上で初めてコントローラボードを使う場合には、2.9 節で使用する Connector アプリ（サーバ）も、このとき遠隔サーバに転送しておきます。配布パッケージの Connector/Server フォルダにある `connector_serv.py` を、遠隔サーバの適当な場所に転送してください。愛工大の学内環境を使う場合はこの作業は不要です。

この時点で WinSCP は閉じて構いません。もし Vivado 実行後のプロジェクトを自分の PC にダウンロードしたければ、WinSCP を開いたまま次の作業に進むこともできます。遠隔サーバ上のファイル一覧は自動では更新されないので、Vivado を実行するなどしてファイルが更新されたと考えられる場合には、手動で更新ボタンを押してください。

また、自分の PC 上のファイルと遠隔サーバ上のファイルは、自動では同期しないことにも注意してください。もし記述ミスを見つけるなどして、自分の PC 上のファイルを変更したのであれば、そのファイルは忘れずに遠隔サーバにアップロードしてください。逆に、遠隔サーバ上のリモートデスクトップでファイルを変更した際には、そのファイルを遠隔サーバからダウンロードするのも忘れないでください。

ここまでの作業は、開発サーバを予約せずに進めることができます。そのため、予約した時間が来るまでにここまでの作業を済ませておくと、動作確認が円滑に進められるでしょう。

2.4 遠隔サーバへの接続

予約した時間帯になったら、PowerShell を開き、ssh コマンドを以下のとおり実行します。

```
ssh -L 13389:〈サーバ名〉:3389 -L 13399:〈サーバ名〉:3399 〈ユーザ名〉@〈接続先〉
```

ただし、サーバ名は予約したサーバの名前、つまり vs + 数字 3 桁 (ACRi ルーム)、または ns + 数字 1 桁 (愛工大の学内環境) となります。パスワードの入力を求められるので、遠隔サーバのパスワードを入力してログインします。これにより、ゲートウェイサーバに接続するとともに、開発サーバへのポートフォワーディングの設定が行われます。

ログインが完了したら、リモートデスクトップ接続やコントローラボードの接続をしている間は PowerShell は開いたままにしておいてください。PowerShell を閉じると、ポートフォワーディングの設定が解除されるため、リモートデスクトップやコントローラボードも同時に切断されます。

毎回これを打ち込むのが面倒な場合には、上記のコマンドをメモ帳などに書き入れて、拡張子 .ps1 で保存します。そのファイルを右クリックし「PowerShell で開く」を選択すれば、自動でコマンドが実行されます。ただし、.ps1 ファイルを実行できない設定になっている場合は、以下のコマンドを PowerShell で実行して、実行可能な設定に変更してください。

```
Set-ExecutionPolicy RemoteSigned -Scope CurrentUser -Force
```

なお、自分で作成したものでない .ps1 ファイルを安易に実行することはセキュリティリスクになりますので、十分注意してください。

2.5 リモートデスクトップ接続と Vivado の起動

開発サーバ上で Vivado を実行したい場合には、Windows 標準のリモートデスクトップ接続アプリケーションを使います。スタートメニューの「Windows アクセサリ」からリモートデスクトップ接続を起動します。起動画面で「オプションの表示」をクリックし、必要に応じて「画面」タブで画面サイズの設定を行ってください。「全般」タブで「コンピューター」の欄に localhost:13389 と入力して、「接続」ボタンを押します。初回のみ「このリモートコンピュータの ID を識別できません」というウインドウが表示されるので、「このコンピュータへの接続について今後確認しない」をチェックして、「はい」ボタンを押してください。

リモートデスクトップの画面では、最初に図 2.3 に示すログイン画面が表示されます (ACRi ルームではユーザ名は自動入力されます)。遠隔サーバのユーザ名とパスワードを入力してログインします。正しくログインできれば、Ubuntu のデスクトップ画面が表示されます。デスクトップ画面が表示されたら、Ctrl+Alt+T キーを押すとターミナルが開きます。開いたターミナル画面で以下のコマンドを入力すると、Vivado が起動します。

```
source /tools/Xilinx/Vivado/2020.2/settings64.sh
vivado &
```

ただし、2020.2 のところは使いたいバージョンに合わせてください。愛工大の学内環境では、この作業のかわりに、デスクトップに用意されたスクリプト start-vivado.sh をダブルクリックすることでも Vivado を起動できます。

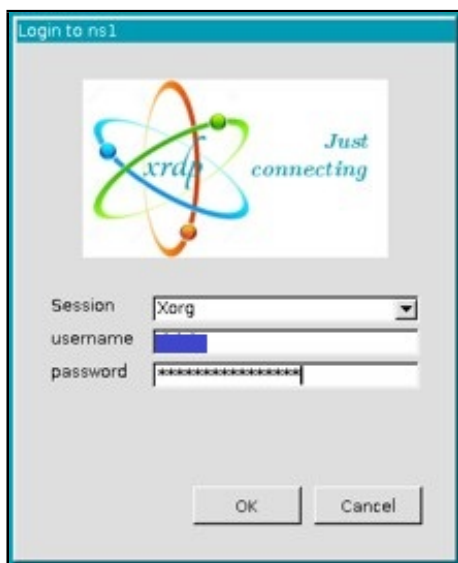


図 2.3 開発サーバのログイン画面.

2.6 回路の論理合成と実装

本節の手順で，作成した回路をベース設計の回路と組み合わせて合成し，FPGA に書き込むビットストリームファイルを作成します．この作業は標準的には開発サーバ上の Vivado を用いて行いますが，自分の PC に Vivado がインストールされている場合は，そこで作業することもできます．

開発サーバ上の Vivado を用いる場合は，Vivado を起動したあと，必要な手順に対応するスクリプトを読み込ませます．Vivado のメニューの「Tools」→「Run Tcl Script...」をクリックし，プロジェクトのフォルダ（ソースのあるフォルダの直下に作成されている）の中にある拡張子 .tcl のファイルを選択します．自分の PC にインストールされた Vivado を用いる場合は，必要な手順に対応する DRFront のボタンを押し，DRFront から Vivado を起動します．

2.6.1 作成した回路の論理合成

まずは，作成した回路そのものを論理合成します．開発サーバ上では，OpenProject.tcl を読み込ませます．自分の PC では，DRFront の「Open Project」ボタンを押します．10～20 秒ほどすると，VHDL ファイルのチェックが完了し，回路の階層関係が画面右上の Sources タブに表示されます．Design Sources の下にある DR_TOP が，DRFront によって作成されたベース設計との接続用の回路です．その横の「>」をクリックすると，この回路に含まれる回路も表示されます．

階層関係の一例を，図 2.4 に示します．この図は，DR_TOP の中に Deeds-DcS で作成した回路 lab1_1 があり，さらにその回路が AND ゲート (AND2_gate) を使用していることを示しています．

回路が無事認識されていることを確認したら，論理合成を行います．メニューの「Flow」→「Run Synthesis」，または画面左側 Flow Navigator の「SYNTHESIS」→「Run Synthesis」をクリックします．設定によっては論理合成の設定ダイアログが表示されますが，そのまま OK ボタンを押してください．数十



図 2.4 回路の階層関係表示の例.

秒ほど待つと論理合成が終了し、図 2.5 で示すダイアログが表示されます。ここでは、Open Synthesized Design を選択して OK ボタンを押し、論理合成結果を確認する画面を開いてください。

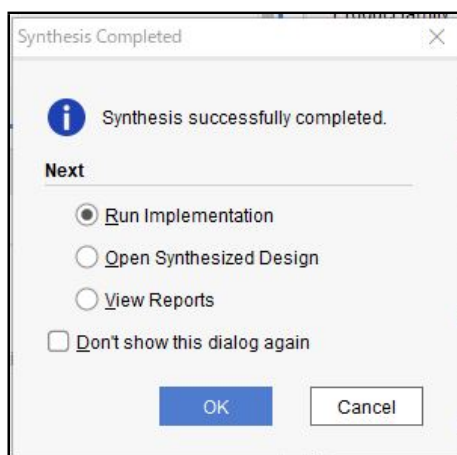


図 2.5 論理合成成功のダイアログ.

なお、ダイアログを閉じてしまった場合や、論理合成結果を後から確認する必要がある場合には、メニューの「Flow」→「Open Synthesized Design」または Flow Navigator の「SYNTHESIS」→「Open Synthesized Design」をクリックすると、この画面に戻ってくることができます。

最後に、論理合成の結果をチェックポイントファイルとして保存します。メニューの「File」→「Checkpoint」→「Write」から「Write Checkpoint」のダイアログを開き、ファイル名を変更せずに OK を押します。自分の PC から Vivado を使用している場合、この時点で Vivado は一旦閉じてしまっても構いません。開発サーバ上の Vivado では、次のスクリプトを読み込ませると現在の作業内容は自動的に閉じられますので、Vivado は開いたままで構いません。

2.6.2 ベース設計との統合と実装

ここから行う作業は、ベース設計に先ほど作成した回路を組み込み、配置配線を行い、ビットストリームとよばれる FPGA に書き込むためのデータファイルの作成を行う、というものです。これらの一連の作業はすべてスクリプトで自動的に実行されます。

開発サーバ上では、GenerateBitStream.tcl を読み込ませます。自分の PC では、DRFront の「Generate BitStream」ボタンを押します。すべての作業が終了するまでに 1~2 分程度かかります。もしも途中でエラーが発生した場合には、Vivado はエラーの内容を表示して停止します。問題なく全ての手順が完了した

場合には、Vivado は自動的に閉じ（あるいは起動直後の状態に戻り）ます。

本節で自分の PC を使って作業をしていた場合、ここで 2.3 節に戻り、必要なファイルの転送とリモートデスクトップ上での Vivado の起動を行うことになります。次の手順のために最低限転送する必要のあるファイルは、プロジェクトのフォルダの中にある OpenHW.tcl と（回路名）__（プロジェクト名）.bit の 2 つのファイルです。

2.7 FPGA への書き込み

FPGA に書き込むためのビットストリームファイルは、プロジェクトのフォルダの中に、（回路名）__（プロジェクト名）.bit という名前で用意されます。動作試験前の最後の手順は、FPGA へこのファイルを書き込むことです。開発サーバ上で起動している Vivado に OpenHW.tcl を読み込ませます。Vivado が Hardware Manager 画面に切り替わります。Hardware Manager では、図 2.6 に示す、まだハードウェアを開いていないという旨のメッセージが表示されます。

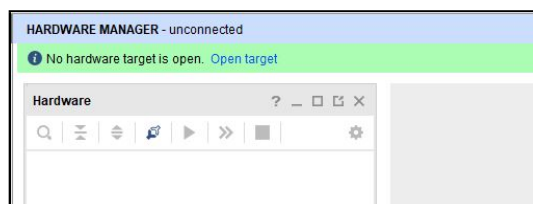


図 2.6 Hardware Manager 画面.

メッセージの横にある Open target のリンクをクリックし、Auto Connect を選択して、サーバと FPGA との間の接続を行います。正しく接続できていれば、左上の Hardware タブに（チップ名）__0 というデバイスが表示され、画面が図 2.7 に示す通りに切り替わります。チップ名には、例えば xc7a100t などといった文字列が入ります。

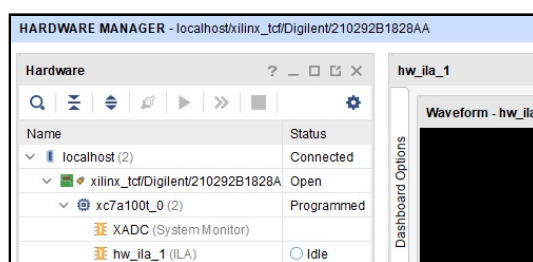


図 2.7 Hardware Manager で FPGA が認識されているときの様子.

（チップ名）__0 を右クリックし、出てきたメニューから Program Device をクリックすると、書き込むファイルを選択するダイアログが表示されます。「Bitstream file」の入力欄の横にある「...」ボタンをクリックすると、図 2.8 に示すファイル指定のダイアログが表示されます。

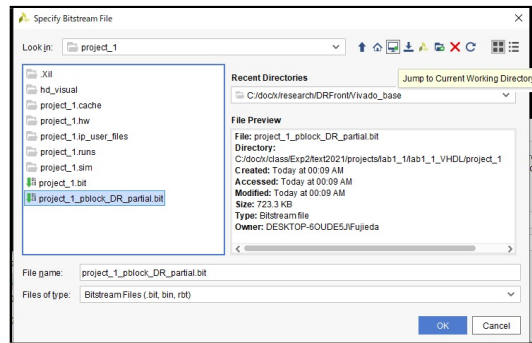



図 2.8 書き込むビットストリームの選択画面。

ダイアログの右上の  のアイコンをクリックするとプロジェクトのディレクトリに移動します。そうすると、書き込むべき .bit ファイルがファイルリストに現れますので、これを選択して OK ボタンを押します。最後に Program ボタンを押すと、ビットストリームが FPGA に書き込まれ、作成した回路が動作を開始します。

ここまで開発サーバ上で Vivado を操作していた場合、Vivado 起動から FPGA への書き込みまでの間に必要な手順は以下の通りとなります。

1. Tool → Run Tcl Script で OpenProject.tcl を実行する
2. Run Synthesis で論理合成を行う
3. Open Synthesized Design で合成結果を開く
4. File → Checkpoint → Write でチェックポイントを保存
5. Tool → Run Tcl Script で GenerateBitstream.tcl を実行する
6. Tool → Run Tcl Script で OpenHW.tcl を実行する
7. Open target → Auto Connect でボードと接続する
8. (チップ名) _0 を右クリックして、Program Device を行う

2.8 Connector アプリ（サーバ側）の準備

注：愛工大の学内環境ではこの作業は不要です。Connector アプリが常時起動する設定になっています。

コントローラボードを使った動作確認のために、コントローラボードと FPGA ボードとの間の通信を中継する Connector アプリを、サーバ側、PC 側の両方で起動します。本節ではまずサーバ側の Connector アプリを起動します。

ターミナル上で 2.3 節で転送した connector_serv.py のあるフォルダに移動します。その上で、以下のコマンドで Connector アプリを起動します。

```
python3 connector_serv.py /dev/ttyUSB1 3399
```

Connector アプリが起動すると、動作ログがターミナル上に表示されます。FPGA ボードが正しく認識できている場合、動作ログは例えば以下のような内容となります。

[2024-04-01 13:41:37] Connector for SawareruSys started.

[2024-04-01 13:41:37] Serial port opened.

[2024-04-01 13:41:37] Server started listening.

Connector アプリを終了させたい場合には、Ctrl+C キーを押してください。

2.9 Connector アプリ（PC 側）の準備と動作確認

つづいて、PC 側でも同様に通信の中継を設定します。PC 側で使用する Connector アプリは、配布パッケージの Connector/Client フォルダにある、Connector.exe 実行ファイルです。

コントローラボードと PC を USB ケーブルで接続したあと、Connector アプリを起動します。Controller Board の欄の下に None をクリックすると、その時点で接続されているシリアル通信デバイスの一覧が表示されます。通常、コントローラボードは「USB シリアル デバイス」という名前で認識されますので、その名前のついたデバイスを選択してください。そうすると、Controller Board の欄下部の表示が、選択したデバイスのポート番号（COM + 数字）に切り替わります。FPGA ボード側のポート番号は、通常変更する必要はありません。2.4 節で遠隔サーバに接続したときに設定したポート番号（13399）が自動で入力されています。Connector アプリでシリアル通信デバイスの一覧が表示されているときの様子を、図 2.9 に示します。



図 2.9 Connector アプリのポート設定の例。

ポート設定が済んだら、Connector アプリの両サイドの Connect ボタンを両方クリックします。ボタン上の文字の色がともに青色に変わり、中央のバーの色が青色に変われば、手元のコントローラボード、遠隔の FPGA ボードともに正しく認識されています。この状態になれば、コントローラボードの入力を操作することで、FPGA ボードの入力を遠隔で操作できるようになっているはずです。また、FPGA ボードの出力が、コントローラボード上の LED や 7 セグメント LED で確認できるようになっているはずです。

2.10 接続を終了する前に

動作確認をやめる場合には、まず PC 側の Connector アプリを終了して、コントローラボードに接続された USB ケーブルを抜きます。リモートデスクトップ上では、Vivado とサーバ側の Connector アプリを終了し、デスクトップ右上の電源マークのアイコンから、「電源オフ/ログアウト」→「ログアウト」を選択して、開発サーバからログアウトします。

これらを確認してから、PowerShell を閉じて、遠隔サーバとの接続を終了してください。PowerShell を先に閉じてしまうと、次回以降のログイン時に問題が生じる場合があります。

第 3 章

コマンド体系

本章では，コントローラボードと FPGA ボードとの間の通信に用いるコマンドについて説明します．

3.1 コマンドの概要

SawareruSys におけるコマンドは 2 つの ASCII 文字で 1 セットになっています．コマンドには，(1) ボードの認識，(2) 出力，(3) 入力 の 3 種類があります．(1) は，コントローラボードや FPGA ボードと，対応する Connector アプリとの間で主に使用します．(2) は FPGA ボードからコントローラボードへと送信され，(3) はコントローラボードから FPGA ボードに送信されます．Connector アプリは，(2) と (3) のコマンドに対しては中継のみ行います．

表 3.1 に，現在定義されているコマンドを列挙します．コマンドの詳細は 3.2 節以降を参照してください．

表 3.1 コマンド定義の一覧表．

	コマンド	定義
(1)	V のあとに X	Connector アプリからのボード種別応答要求
	W のあとに Y	コントローラボードの種別応答
	v のあとに x	FPGA ボードの種別応答
	V のあとに Z	すべての入出力状態の再送要求
(2)	0-4	出力グループの指定
	A-H/a-h	対応する出力をオン/オフにする
(3)	I-S, q-r	入力の指定
	U/u	対応する入力をオン/オフにする

3.2 ボードの認識に関するコマンド

ボードの認識に関するコマンドには 4 種類あります．第 1 のコマンドは “VX” の 2 文字で，Connector アプリが接続されたボードに対して，ボードの種類を認識するために送信します．コントローラボードは，コマンド “VX” を受け取ったら，第 2 のコマンドである “WY” で応答します．また，FPGA ボードは，

第 3 のコマンドである “vx” で応答します。

Connector アプリはまた、ボードが動作しているかどうかを認識するために、“VX” を定期的送信します。PC 側の Connector アプリはコントローラボードからの応答 (“WY”) を、サーバ側の Connector アプリは FPGA ボードからの応答 (“vx”) を、それぞれ一定時間内に受信できない場合は、タイムアウトとして接続を切断してもよいものとします。

第 4 のコマンドである “VZ” は、Connector アプリが、コントローラボードのすべての入力、または FPGA ボードのすべての出力の状態を再送してほしいときに送信します。各ボードは、すべての入力または出力に対してオン・オフのいずれかのコマンドを送信することで、このコマンドに応答します。

3.3 出力に関するコマンド

コントローラボードには、8 個の LED と 4 桁の 7 セグメント LED（小数点を含む）が搭載されていますので、制御すべき LED の個数は $8 + (7 + 1) \times 4 = 40$ 個になります。出力に関するコマンドでは、これを 8 つのセグメントからなるグループ 5 つで管理します。コマンド文字列の 1 文字目では 5 つのグループいずれかを選択し、2 文字目で 8 つのセグメントのいずれかをオンまたはオフにします。

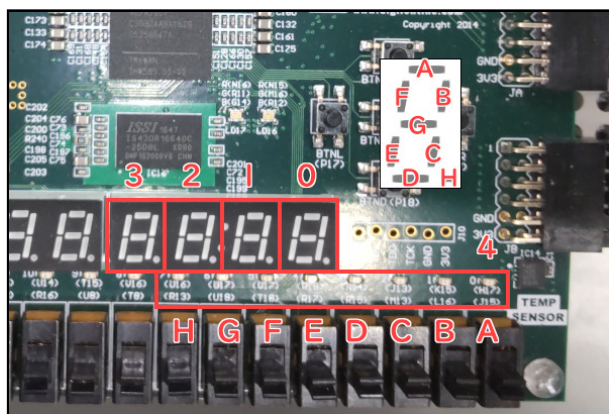


図 3.1 出力グループおよびセグメントとコマンド文字の対応。

図 3.1 に、グループおよびセグメントと、対応するコマンド文字との対応を示します。コントローラボードに図中に記載した文字 A–H のいずれかを送信すると、対応するセグメントがオンになります。セグメントをオフにする場合には、小文字の a–h を使用します。例えば、コマンド文字列 “3G” は、7 セグメント LED の左端の桁の中央のセグメントを点灯させる、という意味になります。

LED は、PWM 出力やダイナミック点灯により、しばしば頻繁にオン・オフされます。これらによって大量の通信が発生することを防ぐため、出力はサンプリングされています。詳細は 4 章を参照してください。

3.4 入力に関するコマンド

コントローラボードには、8 個のスライドスイッチと 5 個（または 3 個）のタクトスイッチが搭載されています。入力に関するコマンドではこれらを個別に管理します。コマンド文字列の 1 文字目では 13 個のス

イッチいずれかを選択し、2 文字目でそのスイッチをオンまたはオフにします。

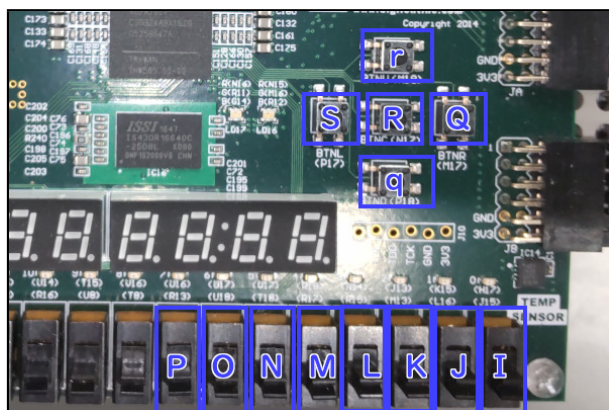


図 3.2 入力とコマンド文字の対応.

図 3.2 に、スイッチと対応するコマンド文字との対応を示します。FPGA ボードに図中に記載した文字 I-S または q, r のいずれかを送信したあと、U を送信すると、対応するスイッチが仮想的にオンになります。スイッチをオフにする場合には、小文字の u を使用します。例えば、コマンド文字列 “SU” は、左端のスライドスイッチをオンにする、という意味になります。

第 4 章

入出力制御回路

4.1 ベース設計の内部構造

図 4.1 に、より詳細なベース設計のブロック図を示します。ソースコードは、パッケージの DRFront/BaseDesign_RC フォルダに置かれており、図中の青字は回路名またはソースコードのファイル名を示しています。青色白抜きで示している DR_TOP がユーザ回路で、それ以外が入出力制御回路です。

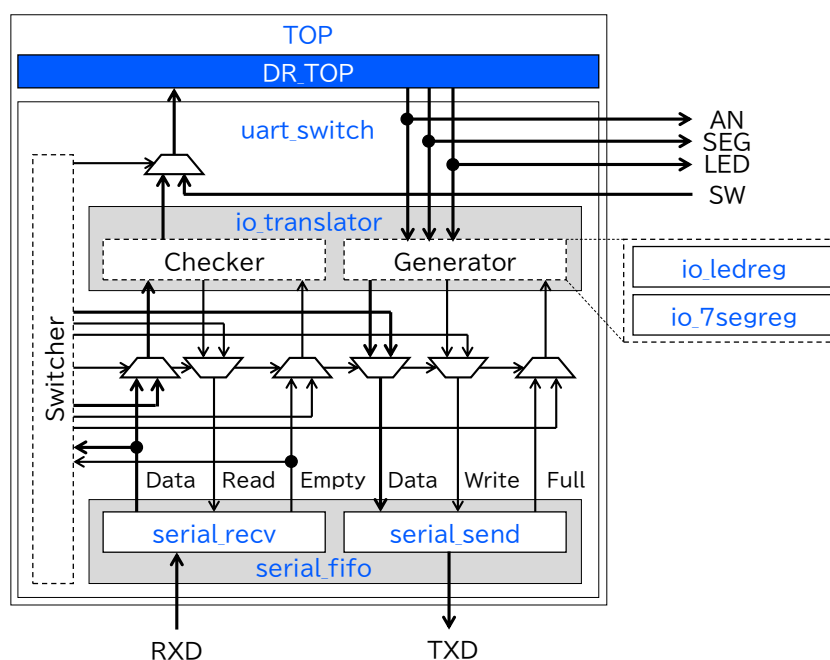


図 4.1 ベース設計の内部構造.

入出力制御回路には 3 つの主要な回路が組み込まれています。1 つは UART コントローラ (serial_fifo), 1 つは I/O トランスレータ (io_translator), もう 1 つはスイッチャー回路 (uart_switch の内部回路) です。UART コントローラは、FPGA ボードの USB-UART 入出力に対するシンプルな FIFO インタフェースを提供します。I/O トランスレータは、出力の生成回路 (Generator) 部分と入力チェック回路 (Checker) 部分とに分かれています。

出力の生成回路部分では、コントローラボードが知っているであろう LED 出力の値を保持し、現在の

ユーザ回路の出力と比較します。もし差異があれば、その中から 1 つを選択して、コマンド文字列を生成するとともに、保持している LED 出力の値を更新します。

生成回路部分ではまた、出力のサンプリングも行っています。この目的は 3 つです。第 1 の目的は、コマンドの生成される頻度を減らすことです。そのために、7 セグメント LED では 100 Hz で、通常の LED では 500 Hz で、それぞれ出力をサンプリングしています。第 2 の目的は、通常ダイナミック点灯により制御される 7 セグメント LED の表示を安定させるためです。そのために、サンプラーは 7 セグメント LED がどのように見えるかを判定します。見た目が変わらない限りは生成回路部分からコマンドが送信されることはありません。第 3 の目的は、PWM 出力をおおまかに再現することです。ユーザが LED に PWM 信号を与えていた場合、その周期がサンプリング周期と一致してしまうと、意図した出力がコントローラボード上に得られないかもしれません。これを防ぐため、通常の LED 用のサンプラーは、単なるサンプリングではなく '1' が出力されていた割合をチェックします。これが周期的に変化するしきい値よりも大きかった場合に、その LED 出力は '1' であると判定します。

入力のチェック回路部分では、現在のスイッチ入力の値を保持し、コマンド文字列を待ちます。スイッチをオン・オフするコマンド ('U' または 'u') を受信したら、対応するスイッチ入力の値を '1' または '0' に変更します。

スイッチャー回路は、リセット直後は Connector アプリからの応答要求 ("VX") が送信されるまで待機します。またこの間は、ボードのスイッチ入力をそのままユーザ回路へと渡すとともに、I/O トランスレータからの信号をブロックします。これにより、FPGA ボードが単体で動作している場合には、あたかもユーザ回路だけが動作しているかのように見えます。応答要求を受信したら、I/O トランスレータからの信号のブロックを解除します。

4.2 ベース設計の論理合成

ベース設計の論理合成は、AMD FPGA の動的部分再構成 (DFX) 機能を使って行うため、通常の論理合成とは方法が異なります。本節では、ソースコードからベース設計を論理合成する手順を説明します。なお、ソースコードの一部は対象とする FPGA ボードによって異なります。以降の説明では、これを〈ボード名〉と表記します。ボード名は、Nexys, Arty, CMod のいずれかです。

1. プロジェクトを新規作成し、ユーザ回路として空の回路を指定して、回路全体を論理合成します。設計のソースとして `dr_top.vhdl` と〈ボード名〉/`dr_base.vhdl` 以外のすべての VHDL ファイル、制約ファイルとして〈ボード名〉/`<ボード名>.xdc` を指定します。ロジックアナライザ (ILA) の IP コア設定として〈ボード名〉/`ila_0.xci` も設計のソースに追加します。
2. 論理合成が終了したら、Open Synthesized Design で合成後の設計を開き、「File」→「Checkpoint」→「Write」でチェックポイントを保存します。保存したチェックポイントファイルを、以後 `step_1.dcp` とします。合成後の設計を開いた際に警告 (Project 1-486) が表示されますが、無視して構いません。
3. 別のプロジェクトを新規作成し、`dr_top.vhdl` と〈ボード名〉/`dr_base.vhdl` を使って、デフォルトのユーザ回路を論理合成します。このとき、SYNTHESIS を右クリックして Synthesis settings を開き、More Options の欄に `-mode out_of_context` と入力します。これにより、不要な I/O バッファの挿入を抑止できます。

4. 論理合成が終了したら、ステップ 2 と同様にチェックポイントを保存します。保存したチェックポイントファイルを、以後 step_3.dcp とします。
5. 「File」→「Checkpoint」→「Open」で step_1.dcp を開きます。警告（Project 1-486）は無視して構いません。
6. Tcl Console に以下のコマンドを打ち込み、ベース設計にデフォルトのユーザ回路を組み込みます。ただし、/PATH/TO/CHECKPOINT はチェックポイントを保存したフォルダのパス名とします。

```
set_property HD.RECONFIGURABLE true [get_cells DR]
cd /PATH/TO/CHECKPOINT
read_checkpoint -cell [get_cells DR] step_3.dcp
```

7. ユーザ回路の配置される範囲（Pblock）を指定します。Vivado の画面右上 Device タブで、Draw Pblock のボタン（P+ と書かれたアイコン）をクリックして、スライスや DSP、BRAM を含む適当な範囲を囲みます。
8. 指定した Pblock の範囲に問題がないかチェックします。「Reports」→「Report DRC」を選択し、Rules で Dynamic Function Exchange にのみチェックを入れた状態で、DRC を実行します。もし DRC に失敗したら、エラーメッセージに従って Pblock の範囲を修正して、やり直してください。
9. Tcl Console に以下のコマンドを打ち込み、作成した Pblock の範囲の設定を保存します。

```
write_xdc -force PBLOCK.xdc
```

2 回目以降は Pblock の範囲指定は read_xdc PBLOCK.xdc で行えます。

10. Tcl Console に以下のコマンドを打ち込み、実装とビットストリームファイルの作成を行います。

```
opt_design
place_design
route_design
write_bitstream -force -bin_file base.bit
write_debug_probes -force base.ltx
```

11. Tcl Console に以下のコマンドを打ち込み、ユーザ回路を空の回路に戻した状態でチェックポイントを作成します。ここで作成したチェックポイントファイルを DRFront で使用します。

```
update_design -cell DR -black_box
lock_design -level routing
write_checkpoint -force base.dcp
```

第 5 章

コントローラボード

本章では、コントローラボードの仕様とコントローラボードのマイコン向けの制御プログラムのビルド方法、制御プログラムの書き込みと動作確認の方法について説明します。なお本章では、「FPGA リモコンボード V2」を V2 ボード、「FPGA リモコンボード V4」および「SawareruBoard V1」を V4 ボードと呼称します。

5.1 ボードの基本仕様

コントローラボードは、USB に対応した PIC18F マイコンを搭載しています。マイコンの型番は、V2 ボードが PIC18F4450-I/P (40 ピン DIP)、V4 ボードが PIC18F45K50/I-PT (44 ピン QFP) です。マイコンの汎用 I/O ピンとして利用できる全てのピンが、ボードの入出力または USB の D+/D- 信号に割当てられています。表 5.1 に、入出力の割当ての一覧を示します。入出力（名前は DRFront における表記に準拠）のそれぞれに対して、割当て先となるマイコンのピン番号とポート名を記載しています。

電源 5 V は CN1 の USB ポート（mini B または Type-C）USB バスパワーで供給されます。電源が供給されている間は、ボード上の D1 LED が赤色で点灯します。マイコンへの制御プログラムの書き込みは、CN2 の ICSP 端子を介して行います。

コントローラボードのすべての入力は負論理です。またそのうち、マイコンの RB ポートおよび RE3 ポートに接続されている入力については、マイコンの内部プルアップを使用します。それ以外の入力は、10 k Ω の抵抗で明示的にプルアップされます。一方で、コントローラボードのすべての出力は正論理です。通常の LED は 1 k Ω の、7 セグメント LED は 330 Ω の抵抗により電流が制限されます。

5.2 制御プログラムの書き込み

制御プログラムのソースコードおよびプロジェクト一式は、PIC/RemoCon_v?.x に置かれています。プログラムは MicroChip 社 MPLAB X IDE v6.10, MPLAB Code Configurator (MCC) v5.3.7, および XC8 v2.4.1 を用いてビルドできることを確認しています。

V4 ボード向けの制御プログラムをビルドするためには、あらかじめ MCC によるコード再生成が必要です。PIC/RemoCon_v4.x フォルダを MPLAB X IDE 上でプロジェクトとして開くと、プロジェクトに必要なファイルが見つからない旨のエラーが表示されます。このエラーは一旦無視して、「Window」→

表 5.1 コントローラボードの I/O のマイコンへの割当て.

入力	V2 ボード		V4 ボード		出力	V2 ボード		V4 ボード	
SW(0)	25	RC6	17	RB7	LD(0)	9	RE1	26	RE1
SW(1)	26	RC7	16	RB6	LD(1)	8	RE0	25	RE0
SW(2)	33	RB0	15	RB5	LD(2)	7	RA5	24	RA5
SW(3)	34	RB1	14	RB4	LD(3)	6	RA4	23	RA4
SW(4)	35	RB2	11	RB3	LD(4)	5	RA3	22	RA3
SW(5)	36	RB3	10	RB2	LD(5)	4	RA2	21	RA2
SW(6)	37	RB4	9	RB1	LD(6)	3	RA1	20	RA1
SW(7)	38	RB5	8	RB0	LD(7)	2	RA0	19	RA0
BTNR	39	RB6	32	RC0	AN(0)	17	RC2	5	RD7
BTNC	40	RB7	30	RA7	AN(1)	16	RC1	4	RD6
BTNL	1	RE3	18	RE3	AN(2)	15	RC0	3	RD5
BTND	n/a	n/a	27	RE2	AN(3)	10	RE2	2	RD4
BTNU	n/a	n/a	31	RA6	CA	19	RD0	41	RD3
					CB	20	RD1	39	RD1
					CC	21	RD2	44	RC6
					CD	22	RD3	38	RD0
					CE	27	RD4	36	RC2
					CF	28	RD5	1	RC7
					CG	29	RD6	40	RD2
					DP	30	RD7	35	RC1

「MPLAB Code Configurator v5」→「MPLAB Code Configurator v5: Open/Close」で MCC を開きます。MCC で画面左上の Project Resources から、Generate ボタンをクリックします。これにより、必要なファイルが自動的に生成されます。

制御プログラムがビルドできたら、PICkit などの PIC ライタを CN2 に三角の印を合わせて接続して、プログラムをマイコンに書き込んでください。

書き込んだプログラムを簡易的に動作確認するためには、Tera Term などの UART に対応したターミナルアプリから、直接コマンドを打ち込みます。以下の手順で動作確認を行います。

1. PC とコントローラボードを USB で接続します。
2. ターミナルアプリでコントローラボードのシリアルポートを開きます。デバイス名は通常「USB シリアル デバイス」となります。転送レートは 115,200 bps, データは 8 bit, ストップビットは 1 bit, パリティとフロー制御はいずれもなしに設定します。
3. ターミナルアプリ上で「VX」と入力すると、画面上に「WY」と表示されます。
4. コントローラボードの左端のスライドスイッチをオン、オフと操作します。スイッチをオンにしたときとオフにしたときに、画面上にそれぞれ「PU」「Pu」と表示されます。

5. ターミナルアプリ上で「3g」「3H」と入力します。7 セグメント LED の左端の桁が消灯したあと、その桁の小数点が点灯します。
6. ターミナルアプリ上で「4G」と入力すると、左端から 2 番目の LED が点灯します。

謝辞

SawareruSys は，日本学術振興会 科学研究費助成事業 基盤研究 (C) 課題番号 21K12164 “デジタル回路の「さわれる」遠隔学習システムに関する研究” の支援により開発され，その成果物として公開しているものです。

SawareruSys の開発および愛工大の学内でのテスト運用にあたっては，ACRi ルームの運用ノウハウを大いに参考にしました。ACRi ルームの立ち上げに尽力された皆様，特に東京工業大学の吉瀬謙二教授とイーツリーズ・ジャパンの三好健文氏に，厚く御礼申し上げます。

また，いくつかの構成要素の開発にあたっては，愛工大デジタルシステム研究室の卒業生が下記の通り貢献しています。感謝申し上げます。

- 原 瑛さん: 入出力制御回路の初期の開発
- 奥地 充稀 さん: Connector アプリのプロトタイプの構築
- 吉川 恵立 さん: Connector アプリのレイテンシ改善

参考文献

- [1] アダプティブコンピューティング研究推進体. ACRi ルーム. <https://gw.acri.c.titech.ac.jp/wp/>, accessed 2024-03-28.
- [2] N. Fujieda and A. Okuchi. A Novel Remote FPGA Lab Platform Using MCU-based Controller Board. In *12th International Conference on Teaching, Assessment and Learning for Engineering (TALE 2023)*, pp. 188–193, 2023.