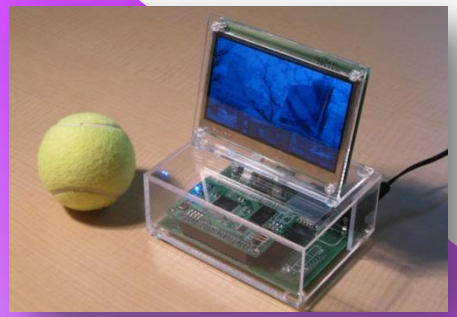
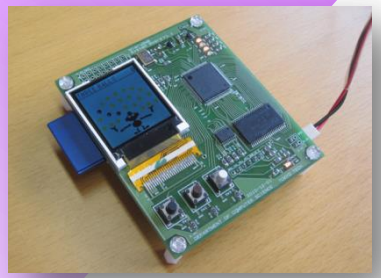


シンプルな計算機システムと組込みシステムの作り方と使い方
～ そこから見えてきた今時のハードとソフトの学び方 ～



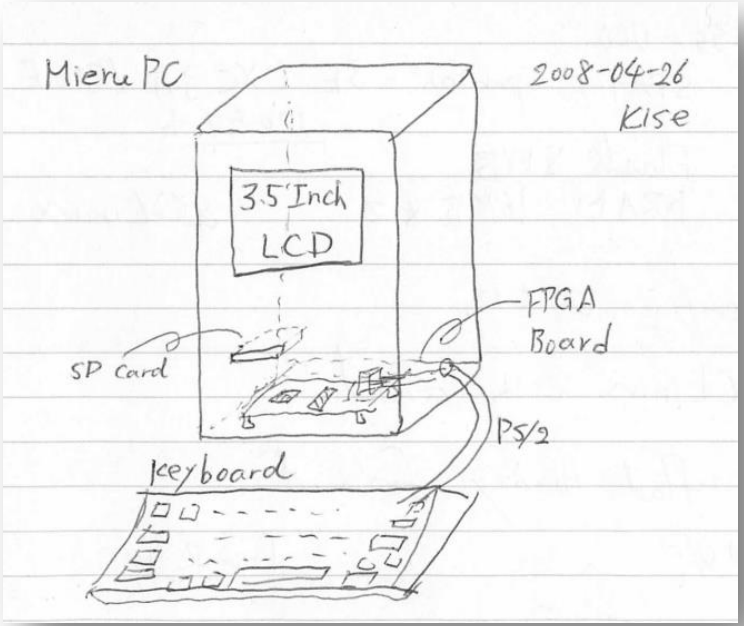
吉瀬謙二 (Kise Kenji)
東京工業大学

TOKYO TECH Pursuing Excellence

チュートリアルの内容

- 作ってきたハードウェア
  - シンプルな計算機システム MieruPC
  - シンプルな組み込みシステム MieruEMB
  - たくさんのボードによるアクセラレータ ScalableCore
- 作り方と使い方（少し）
  - 回路図設計とプリント基板設計
  - プリント基板製造
  - プリント基板への部品実装（はんだ付け）
- 議論

シンプルな計算機システム MieruPC-2010



製品仕様

- FPGACard (Spartan-3E XC3S250E搭載)
- 独自開発のマザーボード
- MeruLCD ― 独自開発のコマンドインタプリタ型液晶ユニット
  (4.3インチ 480×272画素,
  グラフィックコントローラとしてSpartan-3E XC3S250E搭載)
- 英語配列ミニキーボード(PS/2接続)
- 電源ケーブル (AC電源, USB給電)

★MeruPC2010の本体に搭載されるFPGAには，MIPS32に準拠した命令セットを持つ，独自開発のRISCプロセッサが組込まれています．
★マルチメディアカードからカーネルをロードし，独自のOSが動作します．カード内部のファイルシステム(FAT)にアクセスできます．

吉瀬謙二: シンプルな計算機システムの開発に向けた挑戦,
情報処理学会論文誌, Vol.54, No.7, pp1902-1912 (July 2013).

シンプルな計算機システム MieruPC-2010

MieruPCが販売の教材用PC
ザイリンクス製FPGAが採用
米ザイリンクス社の日本法人（東京都品川区、サムローガン社長）は9日、MieruPC（東京都目黒区、藤枝直輝代表）が販売している教材用PC「MieruPC」にザイリンクス製FPGAが採用されたと発表した。
採用されたのは「Spartan-3E」。MieruPCはハードとソフトのすべてが見える計算機システムをコンセプトに開発。主に大学生を想定した、PCを構成するチップの仕様や動作を把握し、変更を加えたり、機能を補足するプログラムを開発したりしながらコンピュータを学習できる。
ハードウエア記述言語「Verilog HDL」を用いハードをFPGA上に実装し、CPU内部を見たり、同言語やザイリンクスのISEやザインスイートソフトウエアを使用
Aug. 11, 2010 Dempa Shimbun Daily P.4

Aug. 11, 2010 Dempa Shimbun Daily

ディープな自作の世界へようこそ:
燃やせ！工作魂――PCを“ゼロ”から作る「MieruPC」(2/2)
前のページへ 1 2

OSから開発環境まで自家製可能

先ほども紹介したように、計算機システムを“ゼロ”から構築するMieruPCプロジェクトでは、ハードウエアの上で動作するソフトウエアも、ゲームやテキストエディタといったアプリケーションはもとより、コンパイラ、そしてOSまで、すべてをユーザー側で用意することも可能だ。なお、OSとコンパイラなど開発環境、いくつかのアプリケーションはMieruPCプロジェクトで用意して、ユーザーだけがアクセスできる限定Webページで配布している。

MieruPC2010に導入されるOSは、「ls」「cat」などUNIXで使われている同名のコマンドが使えるが、同プロジェクトですべて開発したもので、プログラムサイズは30Kバイトとコンパクト。CUIマンマシンインタフェースを使ったシェルの利用、FAT32によるファイル管理、システムコールを利用してアプリケーション側からOSの機能の利用が行えるほか、デバイスドライバも提供する。

同様に、MieruPCプロジェクトではアプリケーションとして、テキストエディタ、ビットマップローダー、テトリス相当、マインスイーパー相当といったシンプルなゲーム群をソースコードとともにすでに用意している。

MieruPC2008で動くテトリス（タイトルはTETRIS）相当のゲームと（写真＝左）、MieruPC2010で動く、ゲーム＆ウオッチのユーザーにはなつかしい「BALL」相当のゲーム（写真＝右）

日本の情報工学をMieruPCで活性化したい

+D PC USER

MIERUPC

中身の見える計算機システム MieruPC-2010

内部を参照および修正可能な教育向け計算機システムを Spartan FPGA で構築

組み込みシステムの広範囲化や大規模化、複雑化にともなって、プロセッサや OS、ソフトウェアといったさまざまな分野を包括的に学ぶためのプラットフォームが求められています。東京工業大学発のベンチャー企業である MieruPC 株式会社では、そのためのアプローチのひとつとして「中身が見える計算機システム」をキーワードとした教育向け計算機システム MieruPC-2010 を開発、販売しています。

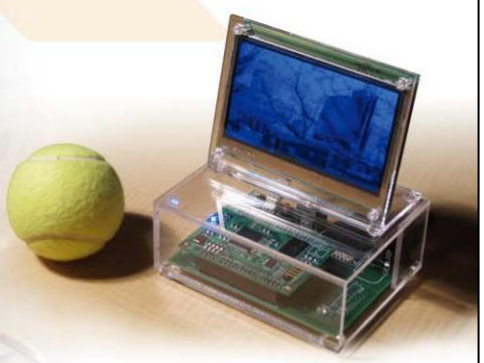
MieruPC プロジェクトは、東京工業大学大学院情報理工学研究科の吉瀬研究室を中心として 2008 年 4 月に立ち上げられた研究／教育プロジェクトです。シンプルかつ内部を参照および修正可能な計算機システムを構築すること、それを用いて学生や企業の新人エンジニアたちが自分自身で計算機を作り拡張する面白さを体験してもらうことを目的としています。MieruPC は主に大学 2～3 年生レベルの知識を前提としており、計算機を構成する回路、プロセッサ、I/O コントローラ、OS、アプリケーションといった要素のすべて、すなわち計算機の「中身」を容易に入手して参照し、必要に応じて修正も可能になることをコンセプトとして設計されています。

プロジェクトの研究成果を広めるために、MieruPC 株式会社は 2010 年 7 月より MieruPC-2010 の販売を開始していますが、主にマザーボードと FPGA ボード、液晶ユニットの 3 点から構成されています。FPGA ボードと液晶ユニットには Spartan®-3E を採用しています。ボードの FPGA には、独自開発された MIPS ライクのソフトプロセッサが組み込まれており、MIPS 命令セットで記述されたアプリケーションや OS を実行します。また、液晶ユニットにはグラフィックコントローラとシリアル転送のコマンド受信機が組み込まれており、MieruPC 本体から送信されたコマンドを解釈し、液晶パネルに表示する機能を備えています。

こうした機能を実現するコード群は Verilog HDL により記述されており、MieruPC を購入したユーザーに提供されます。ユーザーは授業や研究の範囲で自由にコードを参照し、ザイリンクスの ISE® WebPack ソフトウェアを使用して改変することができます。この他にも、MieruPC 上で動作する OS やアプリケーション、これらを構築するためのライブラリ、動作検証のためのシステム シミュレータといったソフトウェア資産をオープンソース ライセンスとして入手できます。

MieruPC を教育に利用することで、学習者はこれから自分が学んでいくものがどのように実を結ぶかをわかりやすく確認できます。また、読みやすさを重視したソースコードにより、それぞれの構成要素に対する理解を助けます。さらに、自分の興味に合わせて要素を拡張することで、興味のある分野に対するより深い理解を得ることができます。こうした体験を通じて、手のひらサイズの計算機システムが「計算機は自分でも作れる、拡張できる」という実感とモチベーションを学習者に与えます。

MieruPC プロジェクトの詳細はプロジェクト Web サイト http://www.arch.cs.titech.ac.jp/mieru/ を、MieruPC 製品の詳細は MieruPC 株式会社の Web サイト http://www.mierupc.com/ を参照してください。



ザイリンクスの Spartan®-3E を採用した、教育向け計算機システム MieruPC-2010

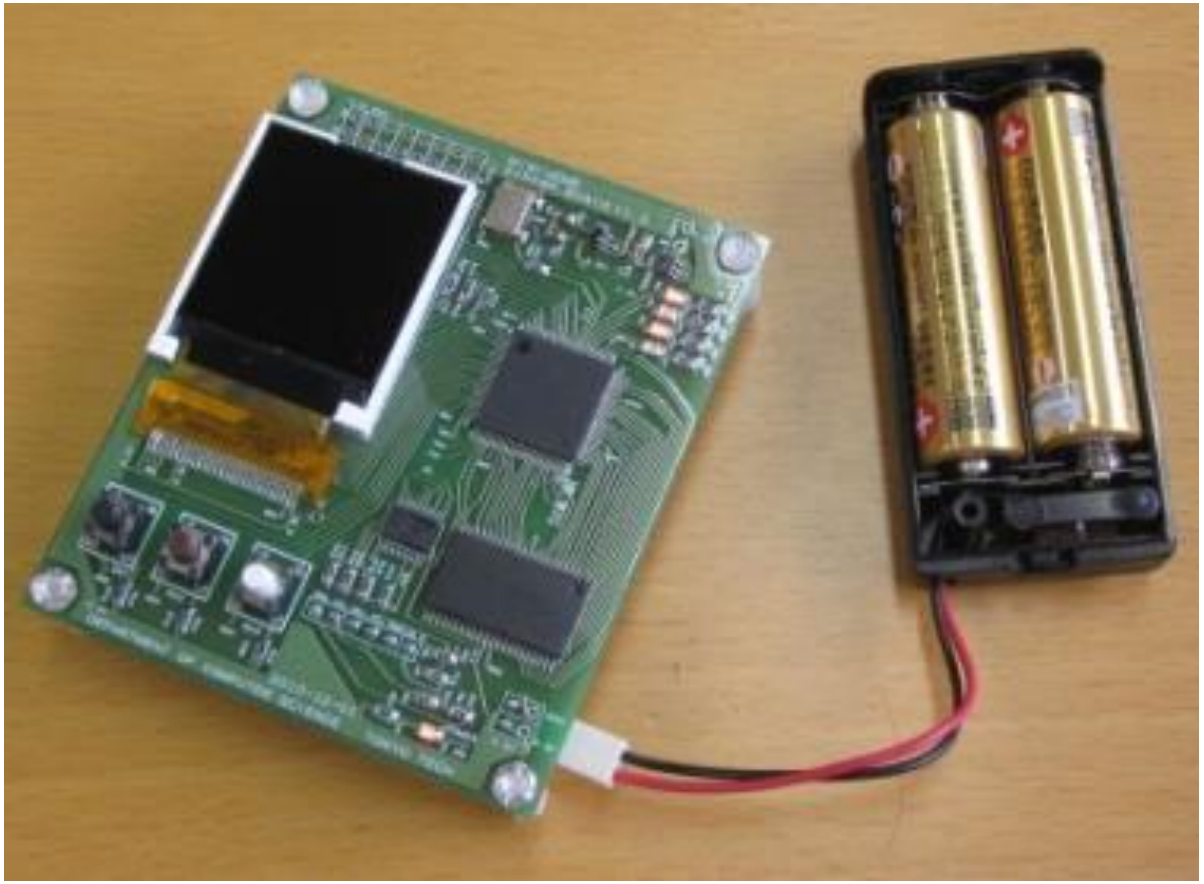
18 Xcell Journal 2010 カスタマ イノベーション 特別号

Xcell Journal 2010 カスタマ イノベーション 特別号

シンプルな組み込みシステム MieruEMB

目的

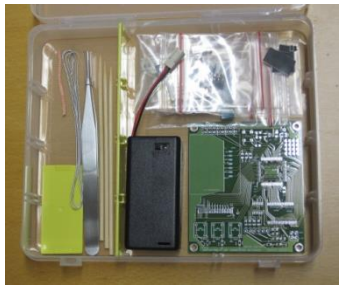
ハードウェアおよびソフトウェアからのアプローチを通じて，組み込みシステムに関する知識と技術を習得する.



シンプルな組み込みシステム MieruEMB

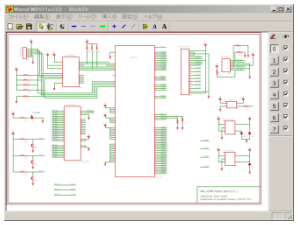
年間 40人，40セットを制作

サーバ計算機(Linux)
MIPSプロセッサのクロス開発環境



(3)基板実装，
はんだ付け

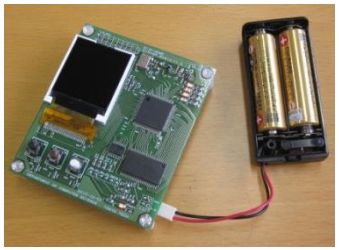
組み込みシステムHWキット
（1人1台を提供）



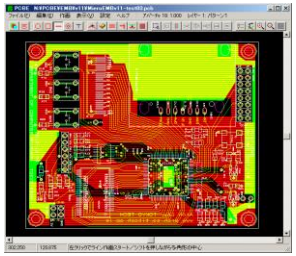
(1)回路図エディタの利用

(5)リモートログイン
MIPSアプリケーション開発

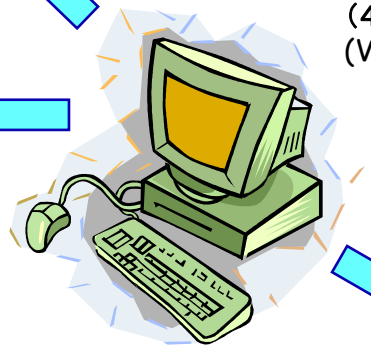
(4)FPGA開発，プロセッサ開発
(Verilog HDL)



組み込みシステムMieruEMB

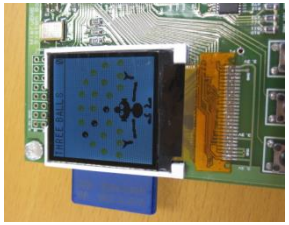


(2)プリント基板エディタの利用



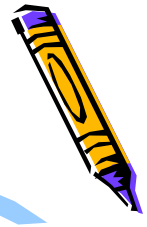
情報ネットワーク演習室 (Windows7)
FPGA開発，アプリケーション開発，CAD利用

(5)アプリケーション検証
組み込みシステム開発



(6)組み込みシステムコンテスト
優れたシステムを表彰

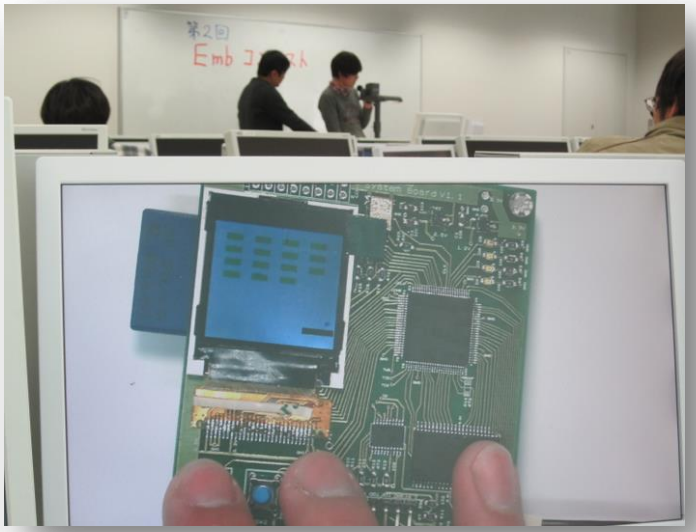
情報実験スケジュール(学部3年生 後期), 実施場所

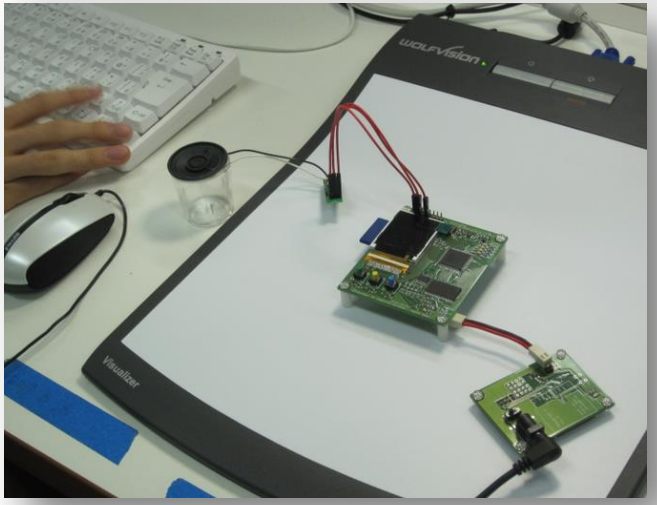


1. 実験の説明, セットアップ等 【A】
2. 組み込みシステムHWの制作と動作確認 【B】
3. 組み込みシステムHWの制作と動作確認 【B】
4. ハードウェア記述言語によるFPGA開発 【A】
5. FPGAへのプロセッサの実装 【A】
6. アセンブラによる組み込みアプリケーション開発 【A】
7. アセンブラによる組み込みアプリケーション開発 【A】
8. C言語による組み込みアプリケーション開発 【A】
9. C言語による組み込みアプリケーション開発 【A】
10. 組み込みシステム開発 【A】
11. 組み込みシステム開発 【A】
12. 組み込みシステムコンテスト 【A】

【A】は, 情報ネットワーク演習室 第1演習室(大岡山 南4号館 3階) で実施.
【B】は, VLSI設計室 http://www.vdc.ss.titech.ac.jp/ で実施.
9:40 に集合(実験時間 9:45～12:15, 2.5時間)してください. 開始時に出席をとります.

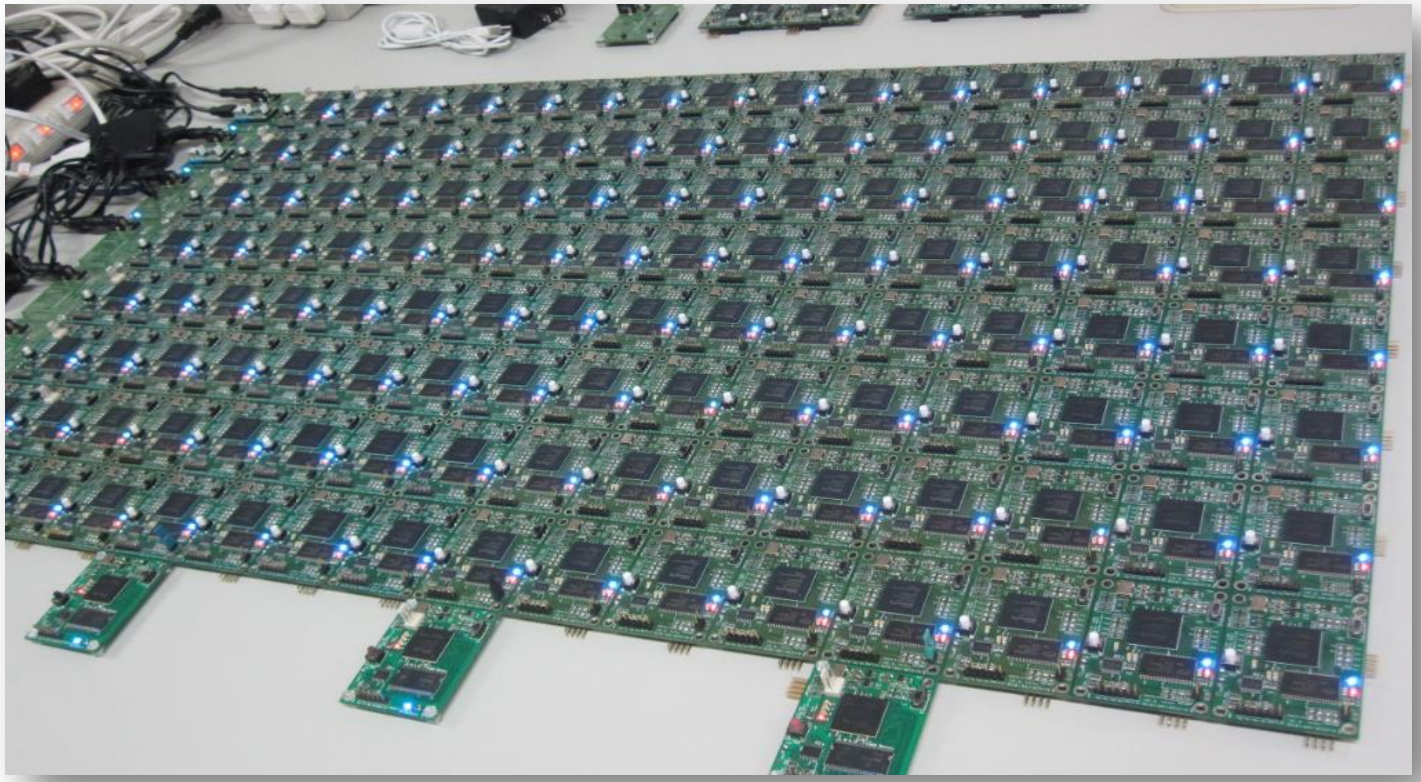
東工大 情報工学科(3年生) 情報実験の風景





FPGAアレーシステム ScalableCore

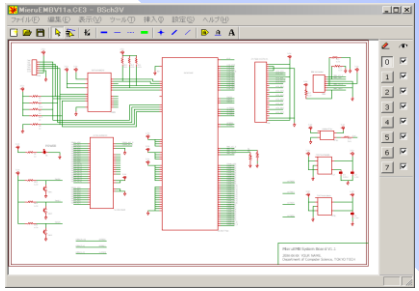
- 128枚のFPGAボードを接続した研究用のFPGAアレーシステム
  - メニーコアプロセッサの高速エミュレーション
  - 専用計算機（アクセラレータ）

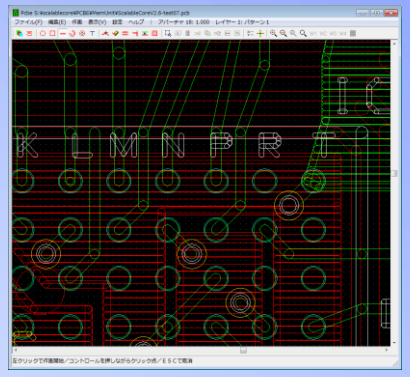


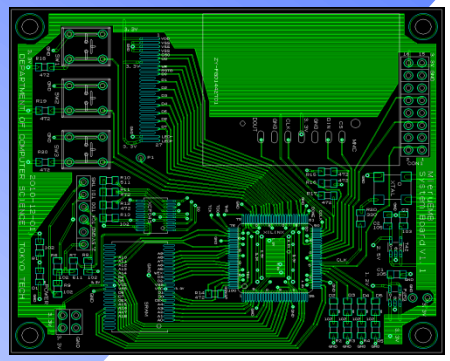
S. Takamaeda, S. Sano, Y. Sakaguchi, N. Fujieda, and K. Kise: ScalableCore System: A Scalable Many-core Simulator by Employing Over 100 FPGA, The 8th International Symposium on Applied Reconfigurable Computing (ARC2011), (March 2012).

回路図設計とプリント基板設計

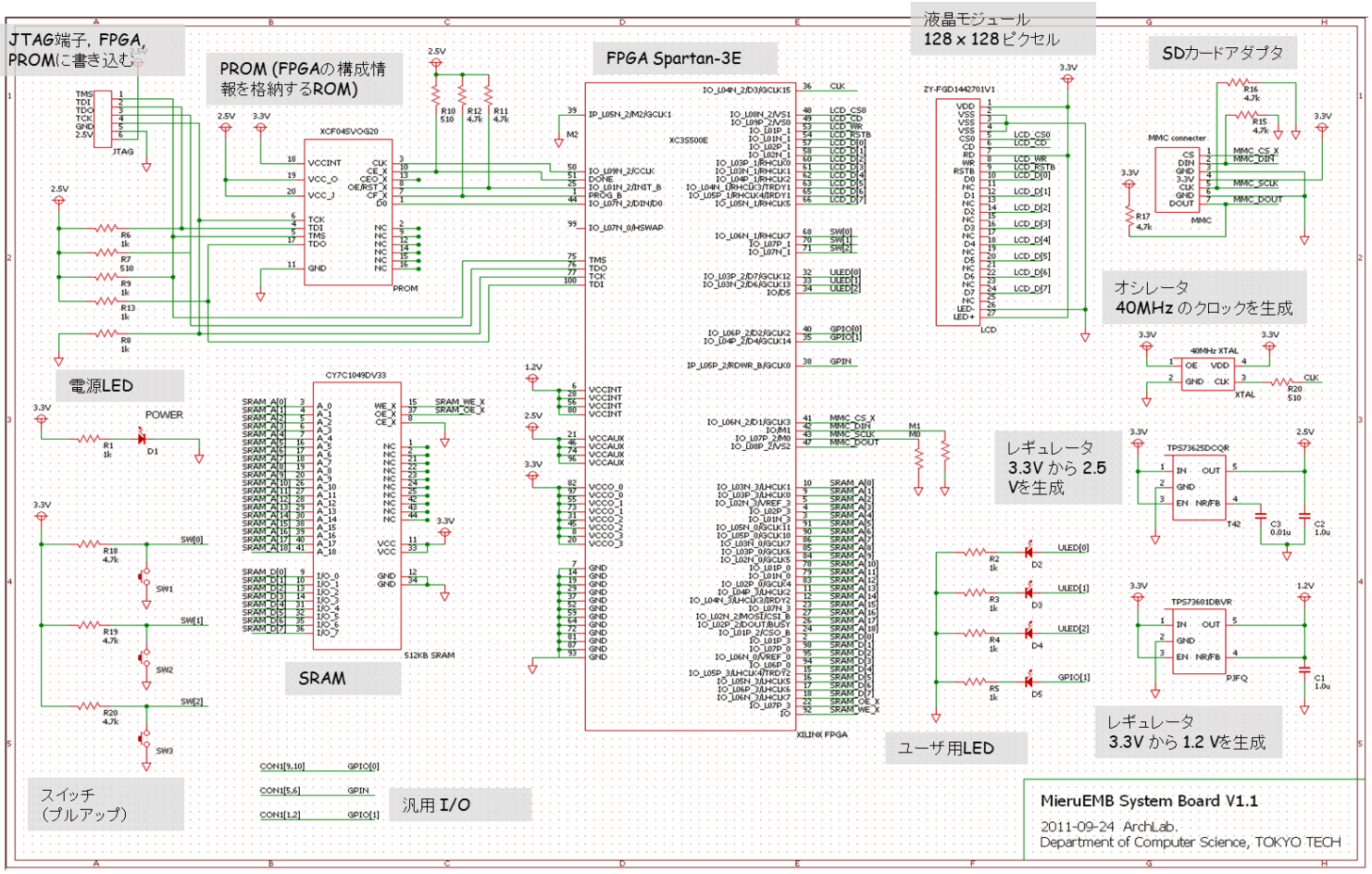
CADを用いたプリント基板の設計データ作成







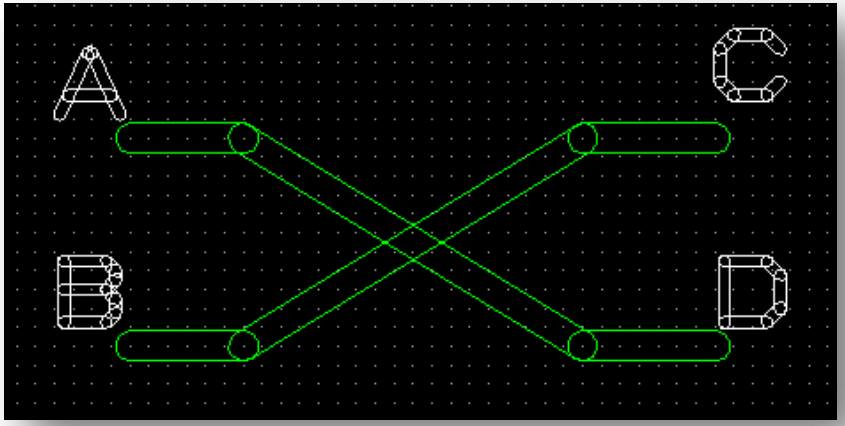
回路図設計



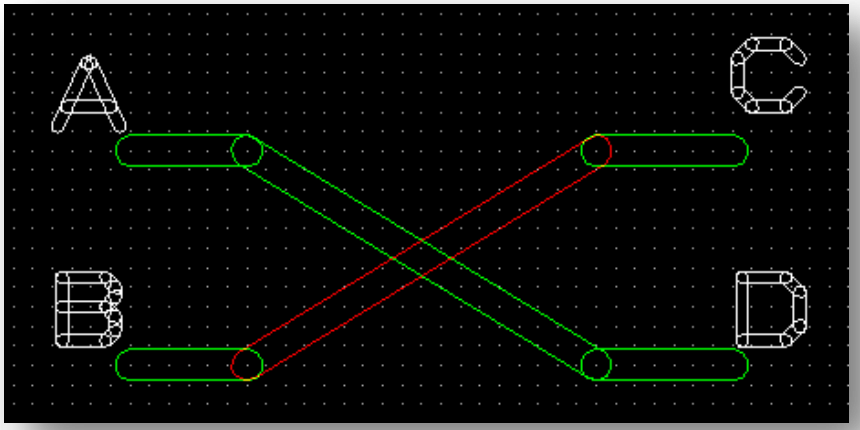
- 実際は（基板製造前には）ほとんど回路図は書いていません．

2層基板の設計

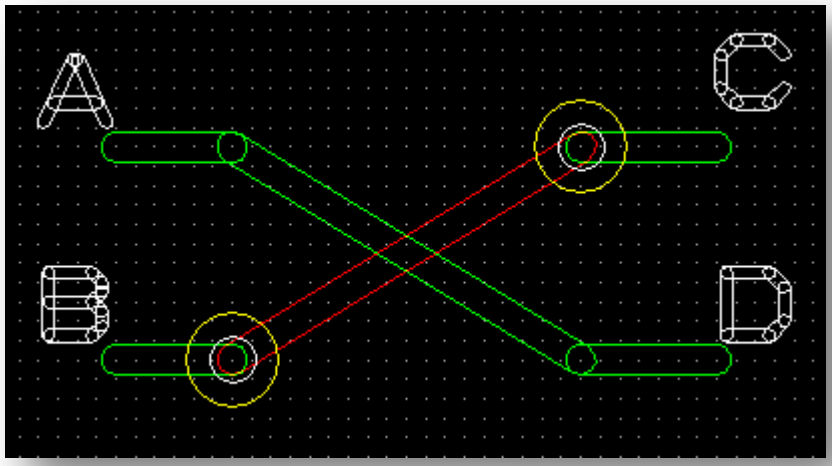
- 表面 パターン1：緑色
- 裏面 パターン2：赤色



1層基板では A-D, B-C を接続できない



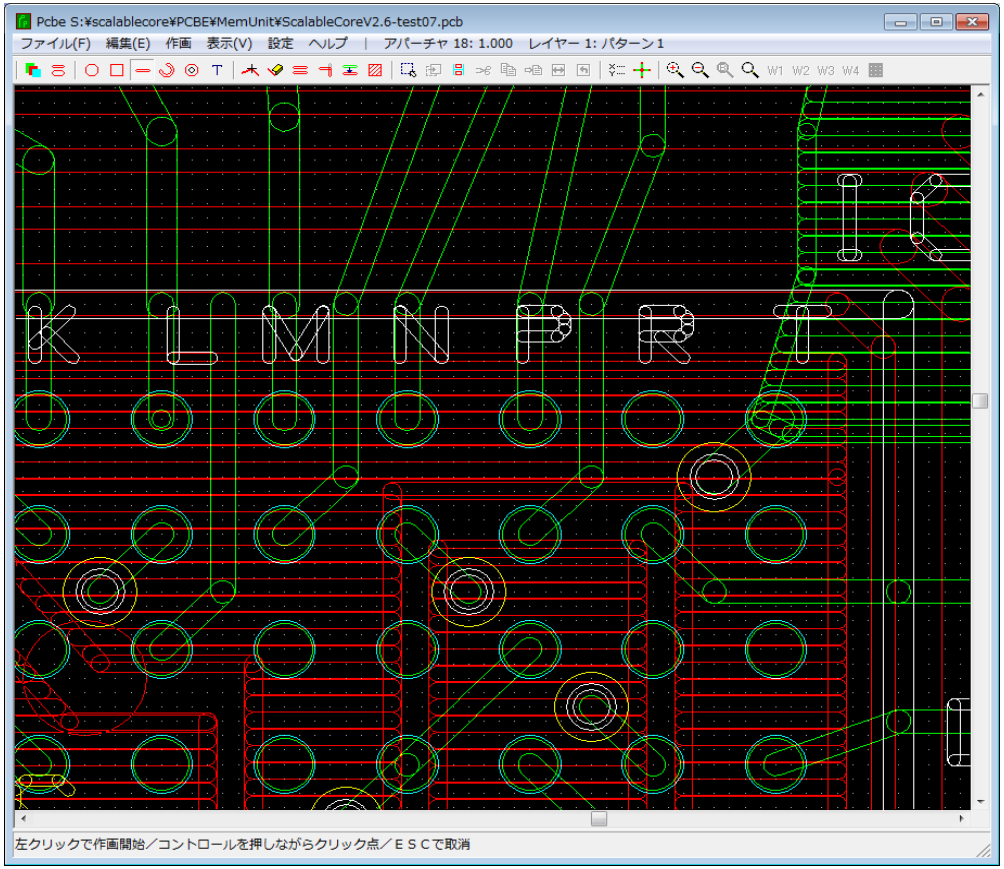
2層基板だとなんとかなりそう．表面と裏面の接続は？

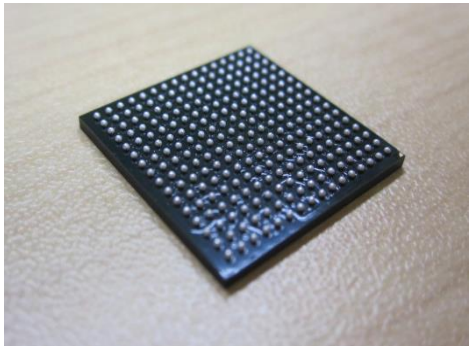


スルーホールで，表と裏を接続すれば大丈夫．

プリント基板設計

- CADを用いて，プリント基板を設計
- ほとんどはフリーソフトウェア PCBE を利用

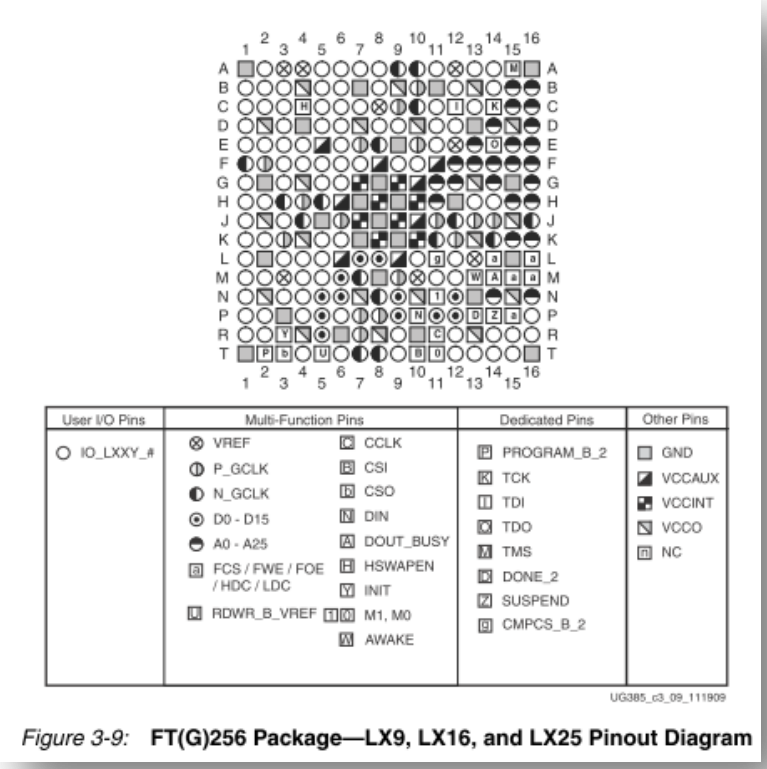




Ball Grid Array

FPGA(Xilinx Spartan-6)パターンのレイアウト例

- Ball Grid Array (BGA) のパターンはPowerPointで検討

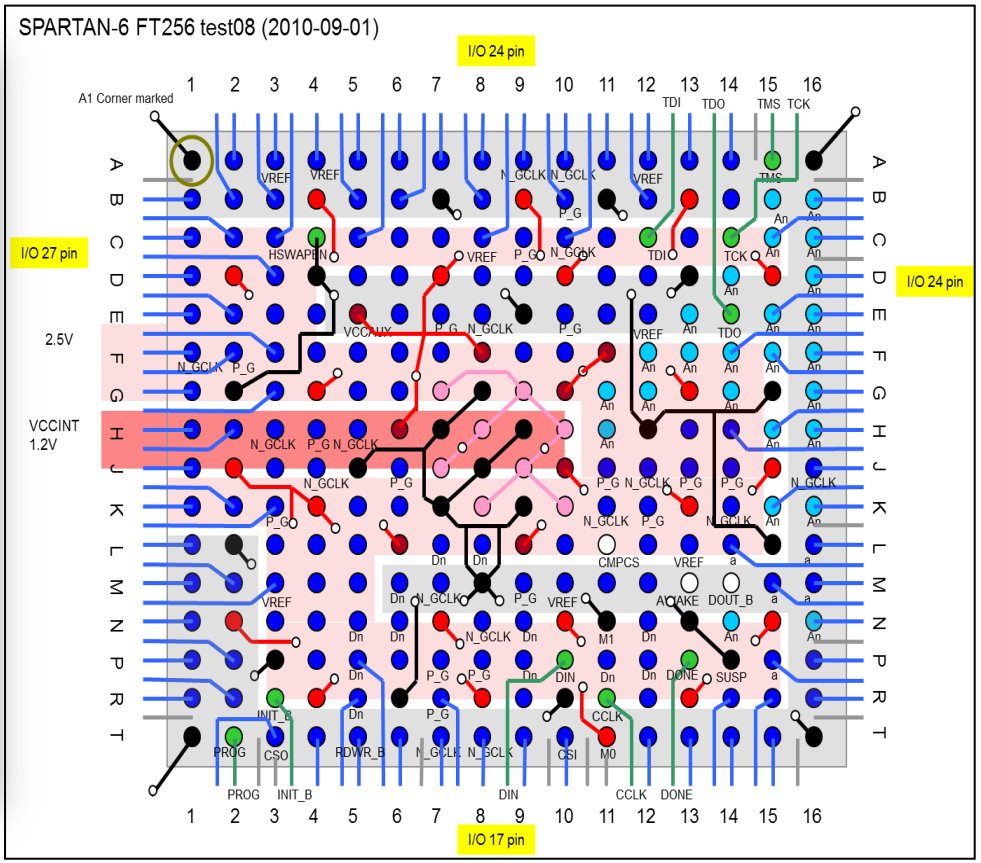


User I/O Pins	Multi-Function Pins		Dedicated Pins	Other Pins
IO_LXXY_#	VREF	CCLK	PROGRAM_B_2	GND
	P_GCLK	CSI	TCK	VCCAUX
	N_GCLK	CSO	TDI	VCCINT
	D0 - D15	DIN	TDO	VCCO
	A0 - A25	DOUT_BUSY	TMS	NC
	FCS / FWE / FOE / HDC / LDC	HSWAPEN	DONE_2	
	RDWR_B_VREF	INIT	SUSPEND	
		M1, M0	CMPCS_B_2	
		AWAKE		

UG385_c3_09_111909

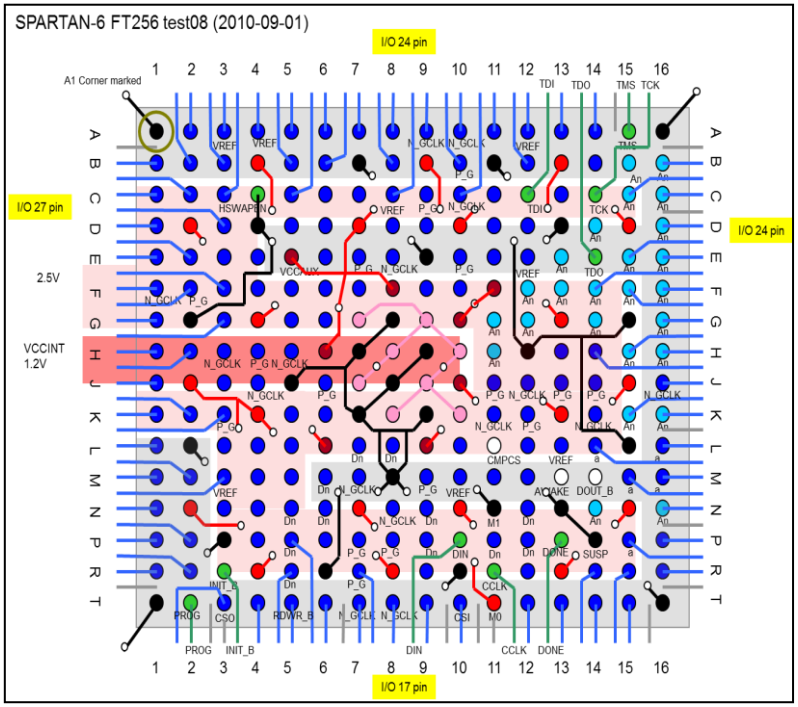
Figure 3-9: **FT(G)256 Package—LX9, LX16, and LX25 Pinout Diagram**

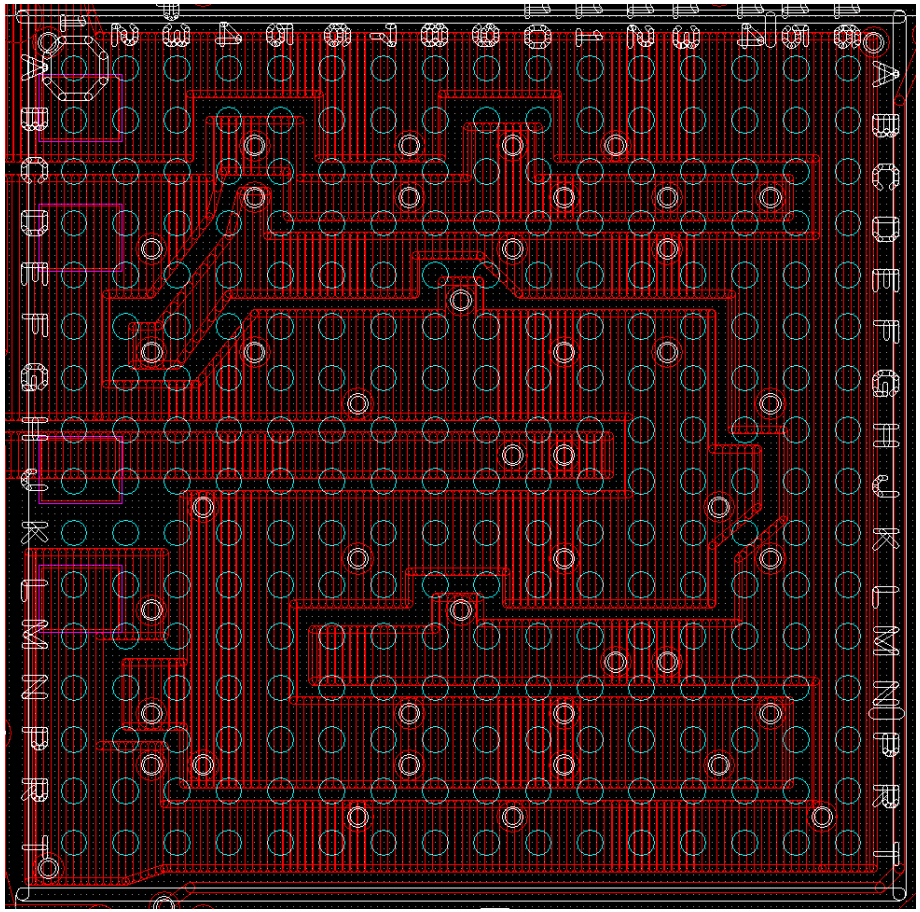
UG385 Spartan-6 FPGA Packaging and Pinouts



プリント基板設計ScalableCoreUnit2.3の例

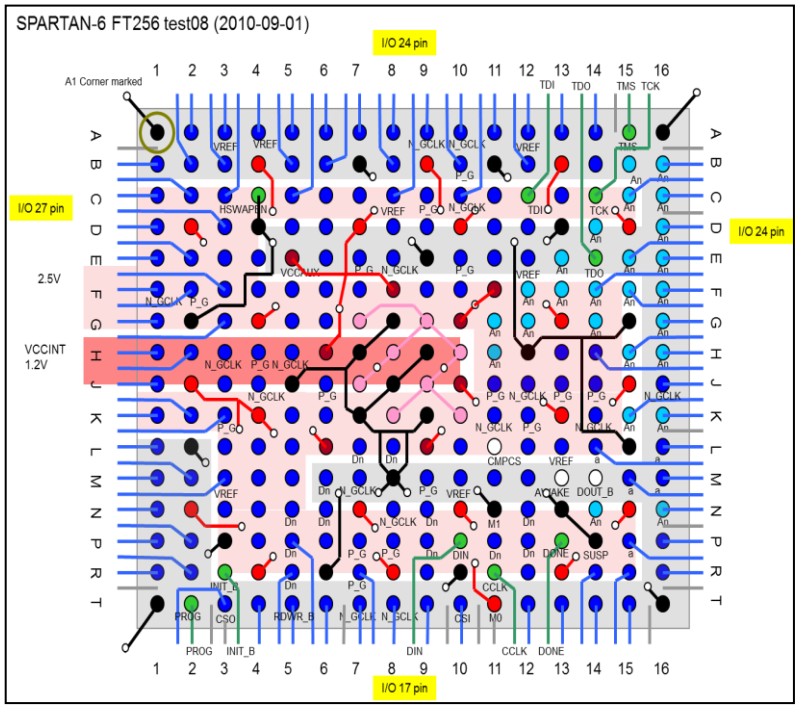
- 裏面 パターン2：赤色

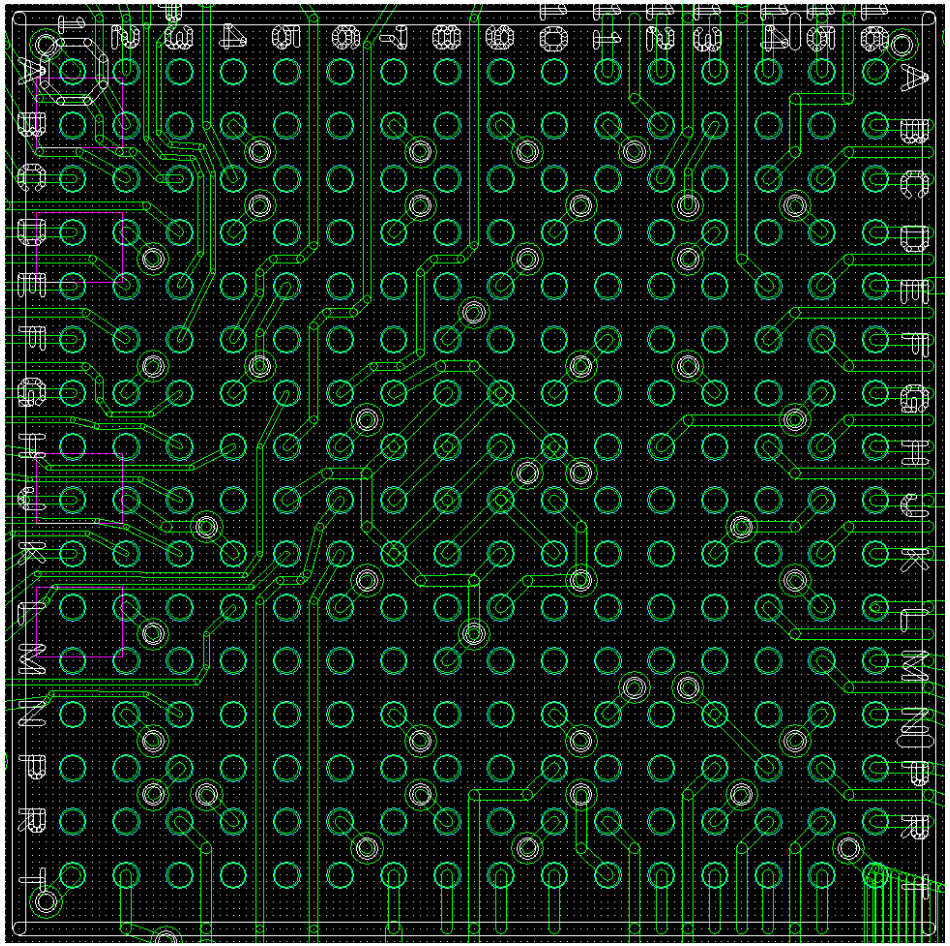




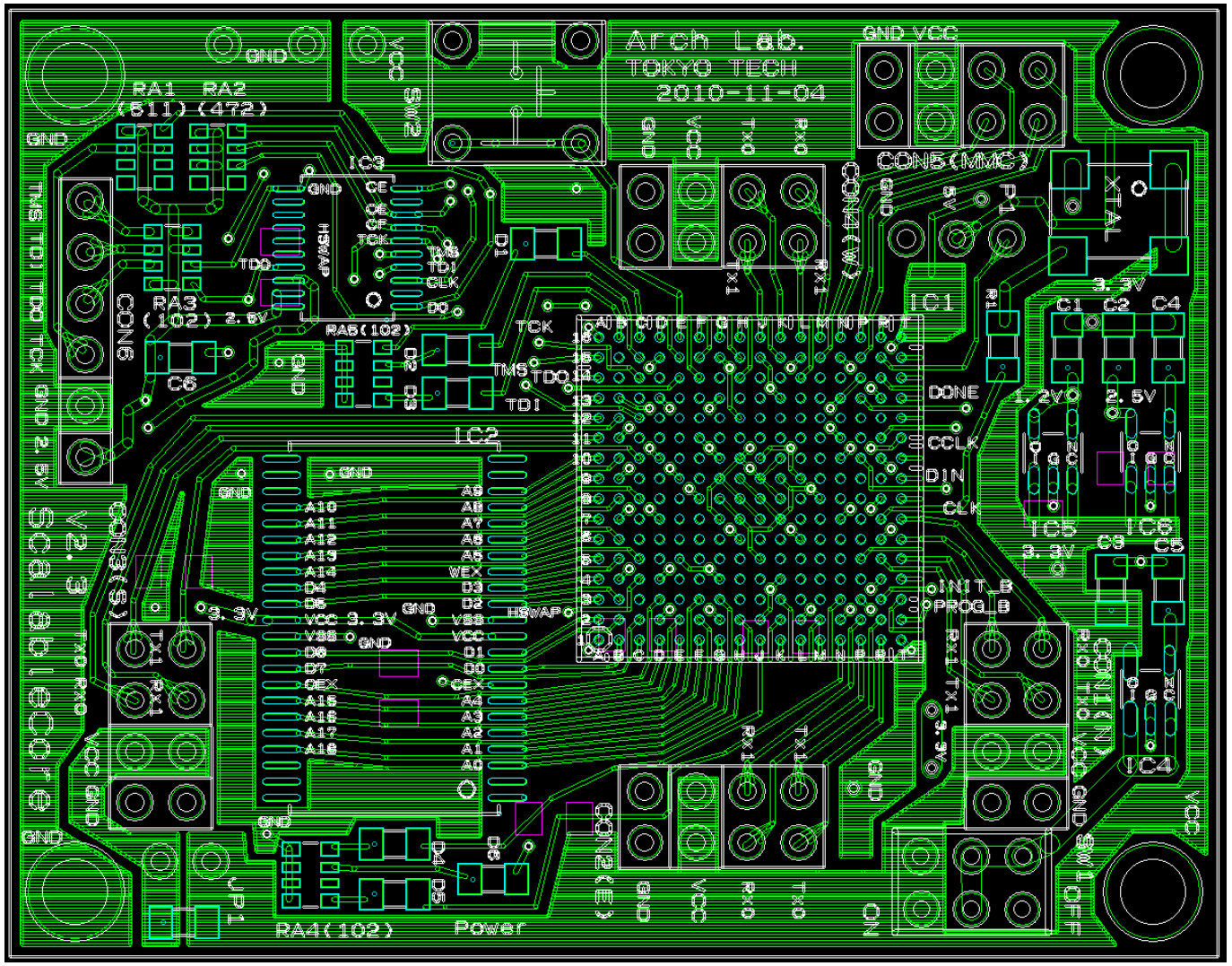
プリント基板設計ScalableCoreUnit2.3の例

- 表面 パターン1：緑色





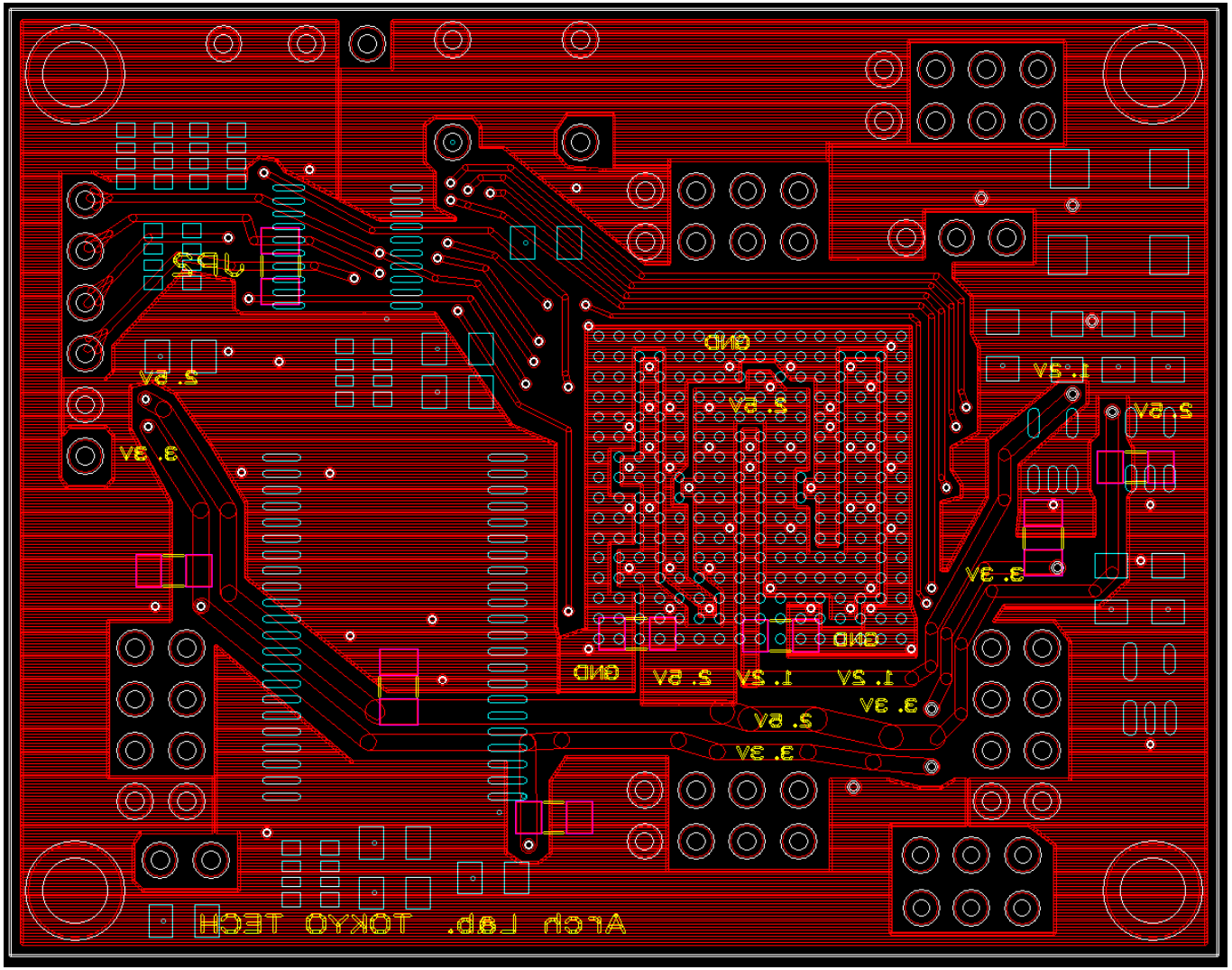
プリント基板設計ScalableCoreUnit2.3の例



46mm

60mm

プリント基板設計ScalableCoreUnit2.3の例

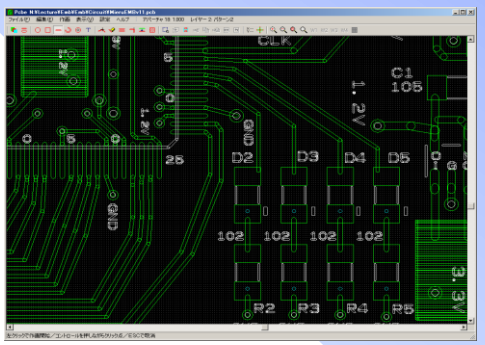


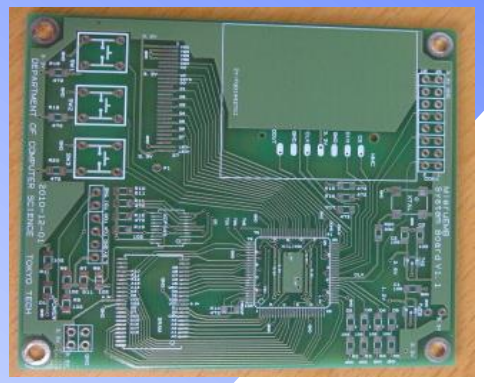
46mm

60mm

プリント基板製造

プリント基板の設計データから基板を製造



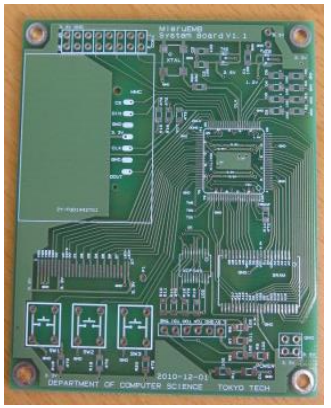


自分でエッジング，業者に依頼

- 業者に依頼しています．

プリント基板製造の見積もり例

- 70mm x 81mm, 2層, 20枚製造, ノーマル 5日
  - ¥27,195, ¥1,360 / board

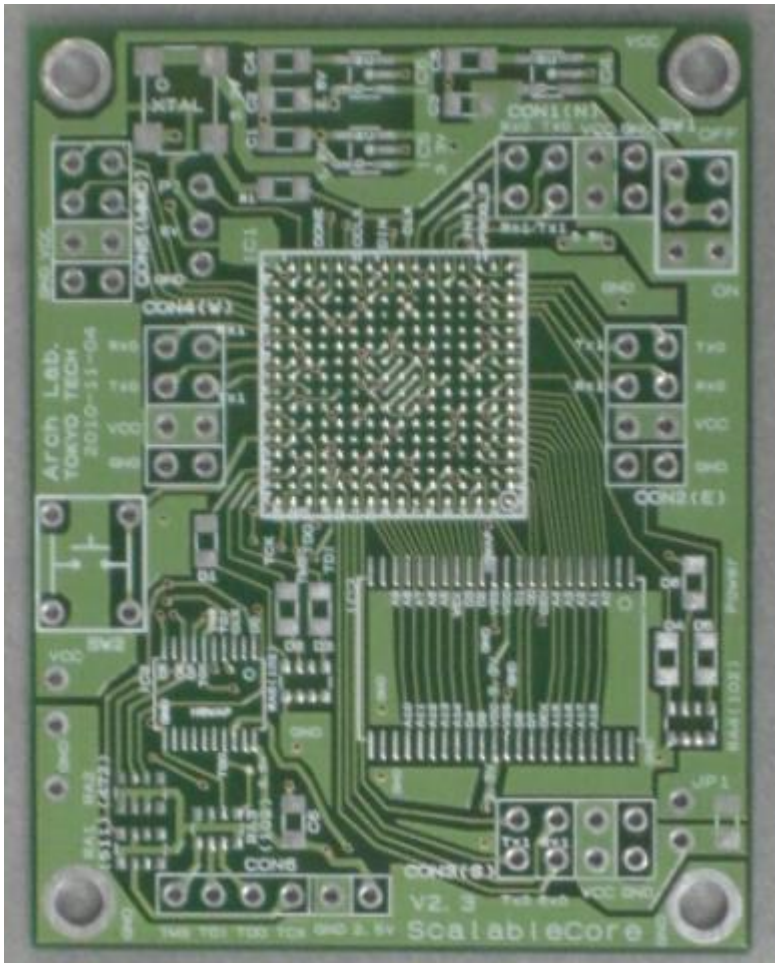


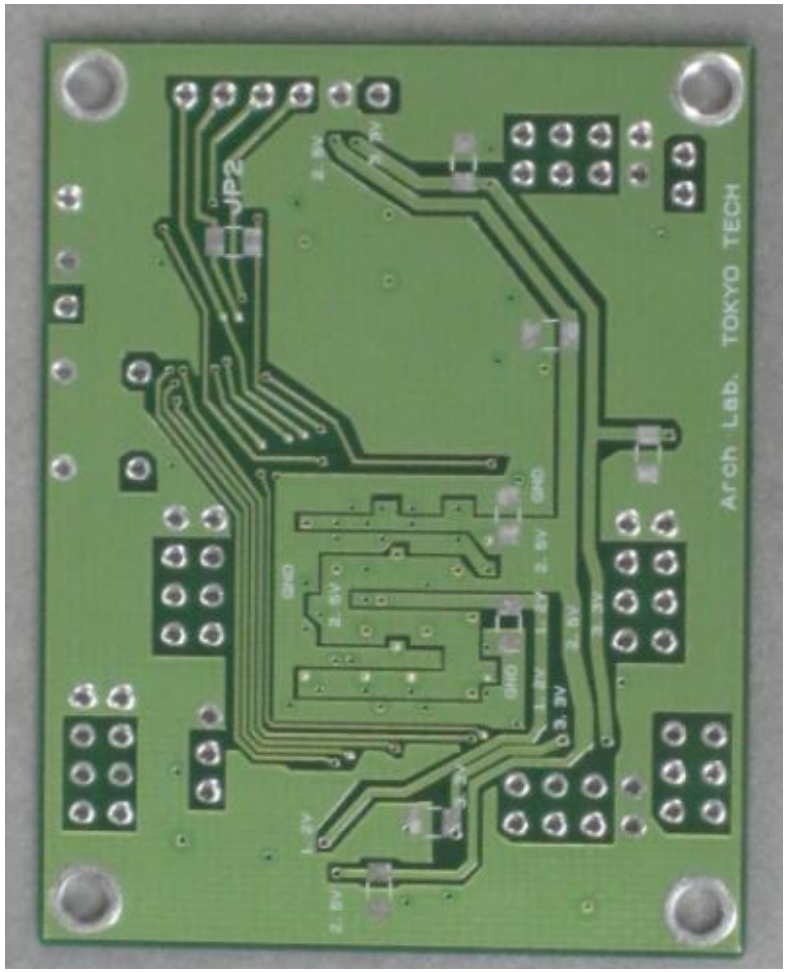
81mm

70mm

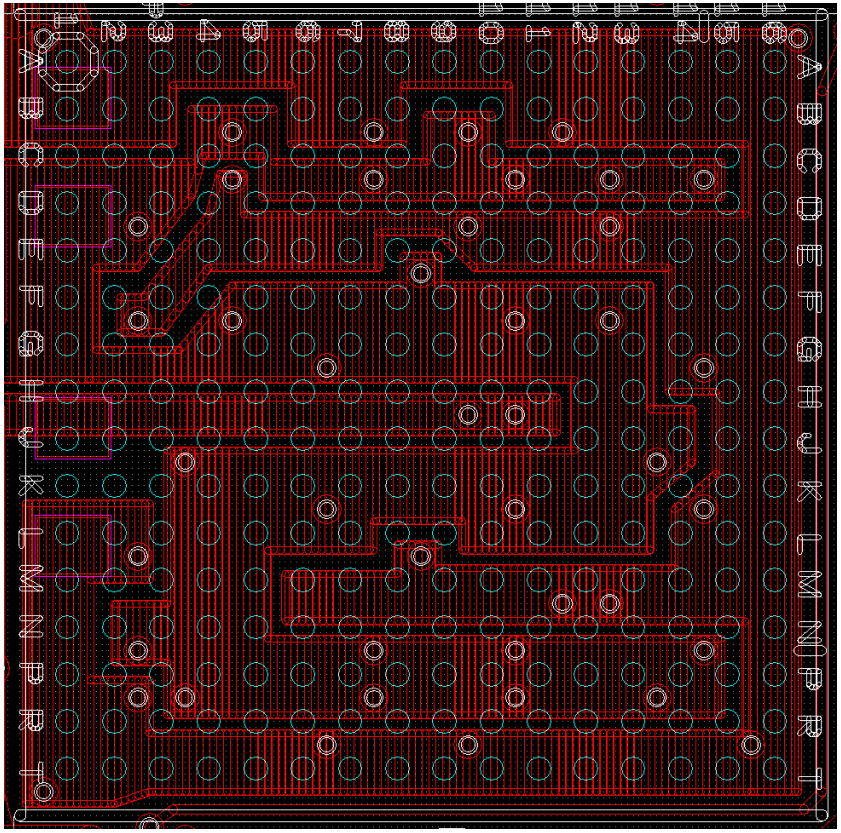
- 70mm x 81mm, 2層, 40枚製造, ノーマル 5日
  - ¥32,718, ¥818 / board

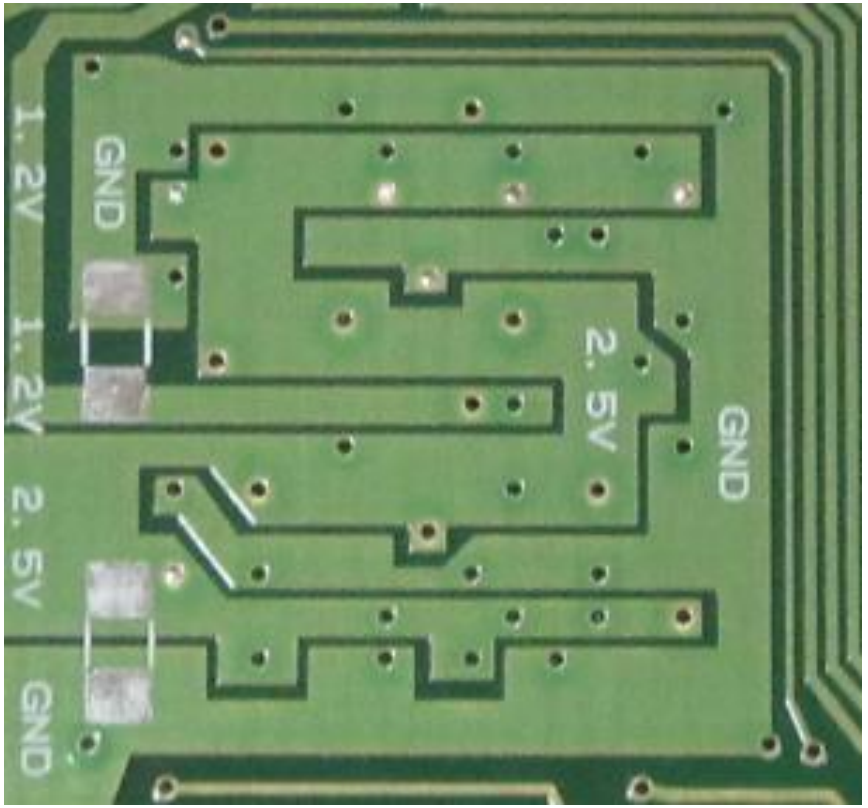
プリント基板ScalableCoreUnit2.3の例



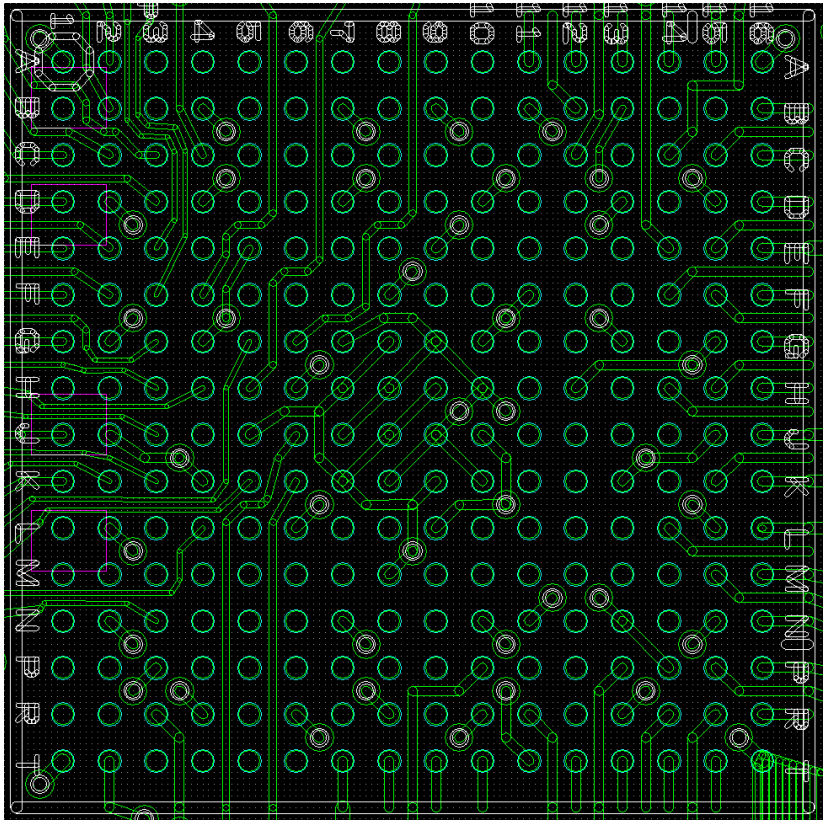


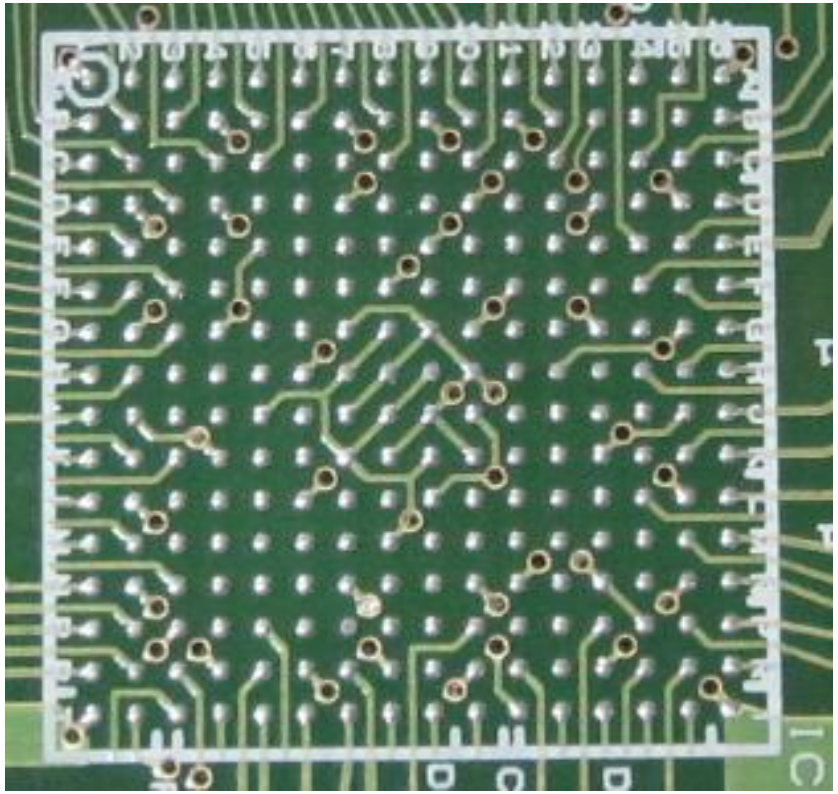
プリント基板ScalableCoreUnit2.3の例





プリント基板ScalableCoreUnit2.3の例



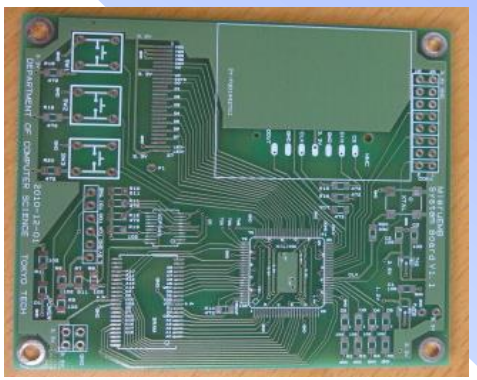


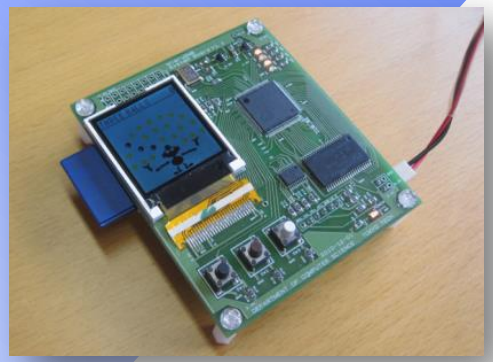
17mm

17mm

プリント基板への部品実装

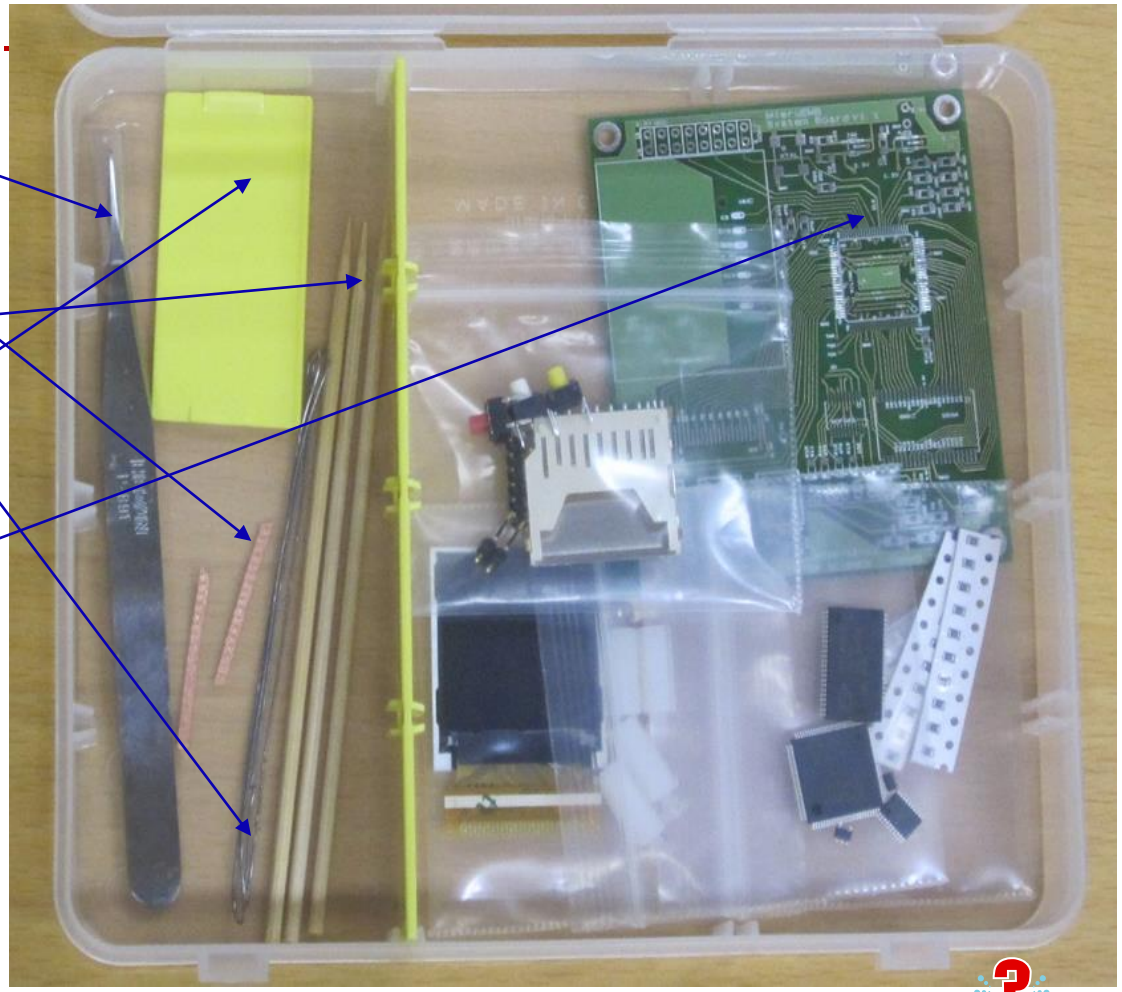
プリント基板に部品をはんだ付けして，動作検証





組み込みシステムHWキットの確認（1／2）

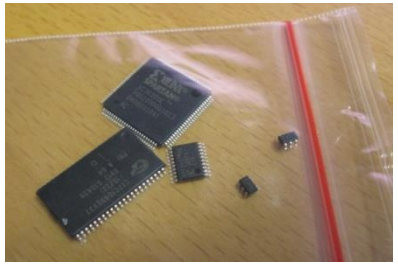
- 以下のすべての部品が揃っていることを確認する.
- 部品は丁寧に扱うこと.
  - ピンセット
  - はんだ
  - はんだ吸取線
  - 竹串（3本）
  - プラスチック片
  - プリント基板



不足部品がある場合にはTA/教員に尋ねる.

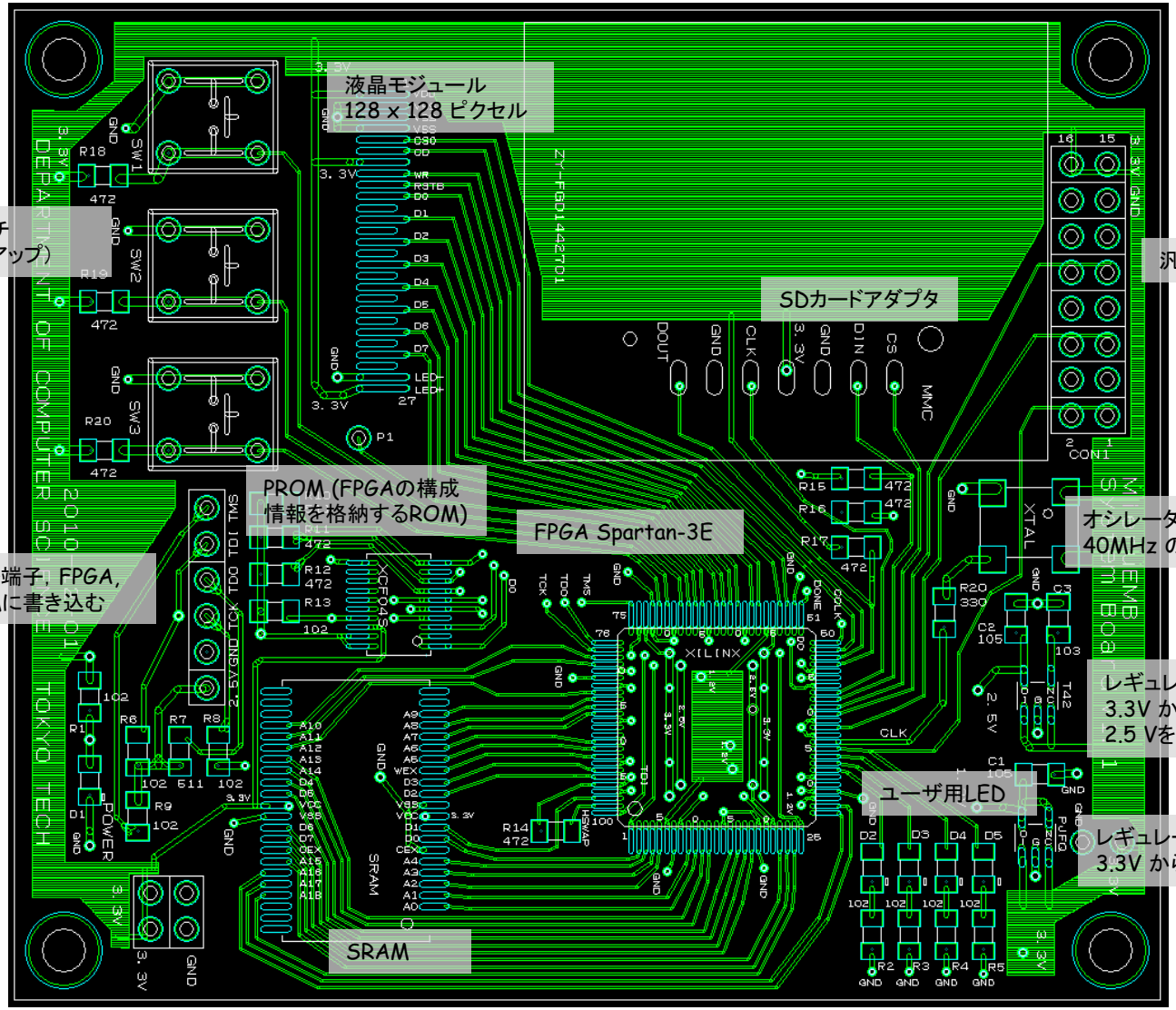
組み込みシステムキットの確認（2／2）

- 以下のすべての部品が揃っていることを確認する.
  - FPGA
  - SRAM
  - PROM
  - レギュレータ 2個(PQFJ, T42の刻印)
  - 液晶モジュール ZY-FGD1442701V1
    コントローラIC：ST7735
  - スペーサとネジ 4組
  - チップ抵抗 102 10個, 472 10個, 511 5個
  - チップコンデンサ 105 3個, 103 2個
    (チップコンデンサには刻印が無いので注意)
  - 発光ダイオード 5個(または6個)
  - 6ピンヘッダ, 2ピンヘッダ
  - スイッチ 3個
  - 40MHz クロックオシレータ
  - SDカードスロット



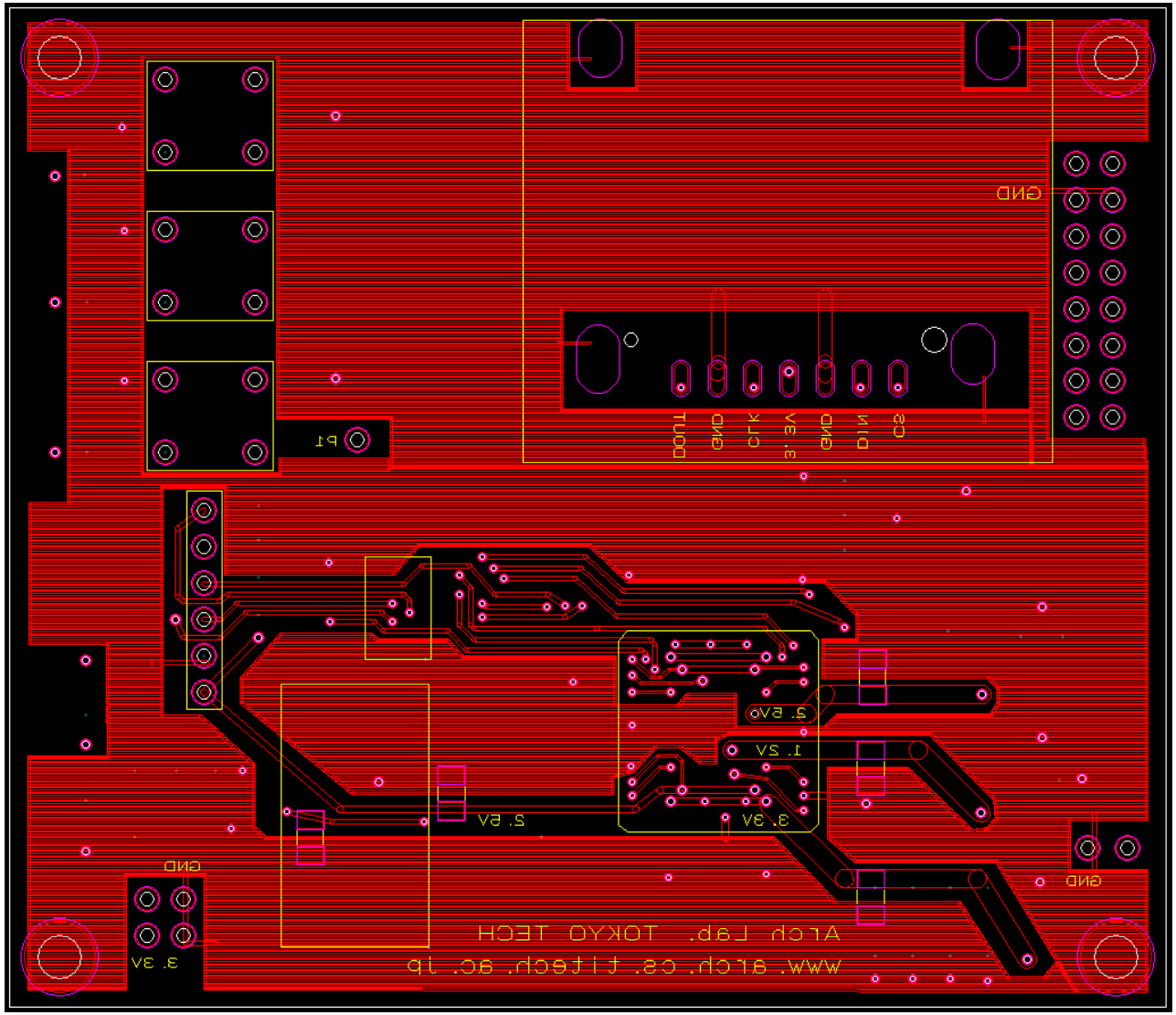
部品面（表）

- レイヤー 2, 4, 6, 7（部品面パターン，部品面シルク，部品面レジスト，外形）



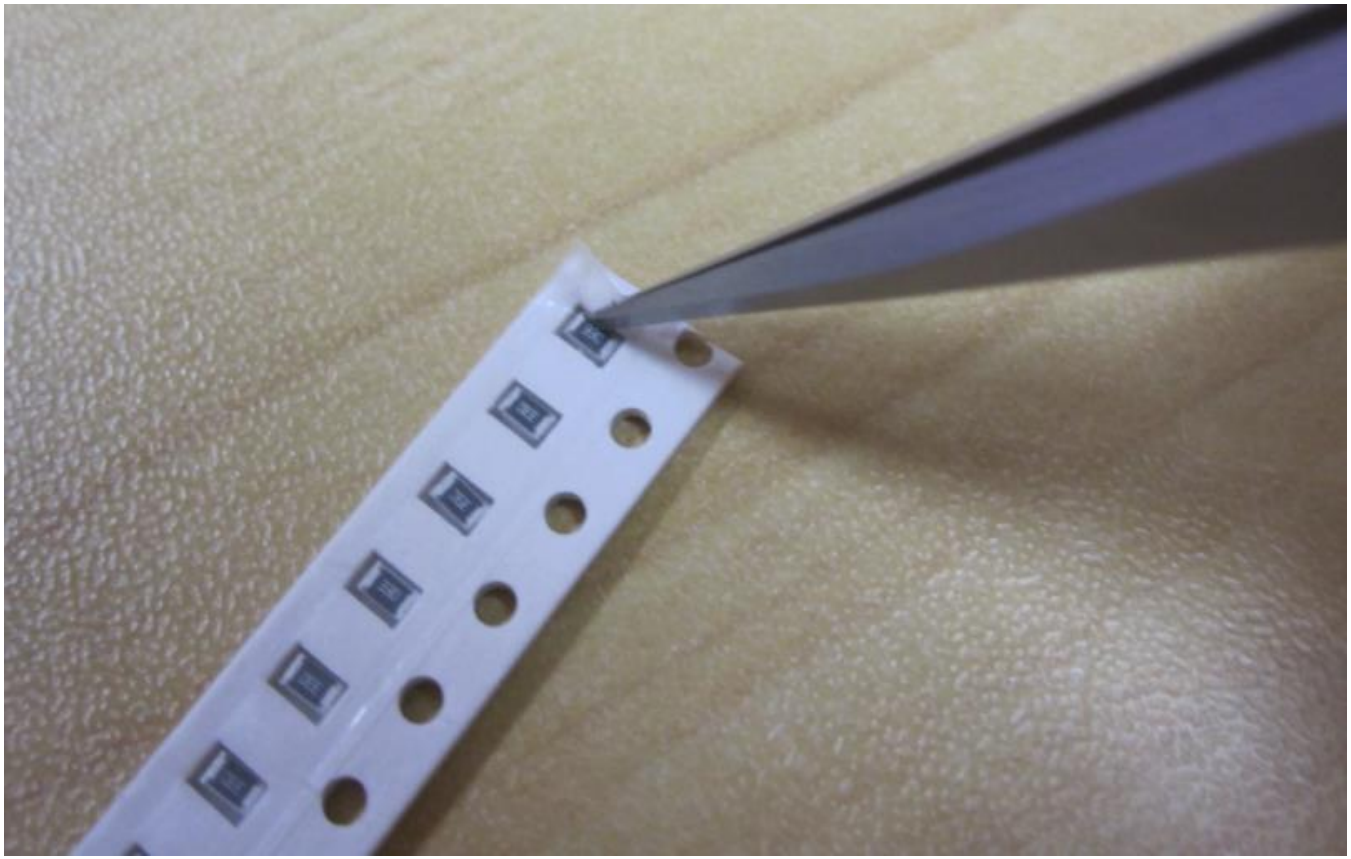
半田面（裏）

- レイヤー 1, 3, 5, 7（半田面パターン，半田面シルク，半田面レジスト，外形）



1. チップ抵抗，コンデンサ，ダイオードなどの取り出し方

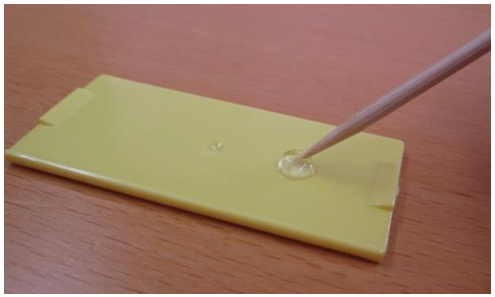
- ピンセットを使って，ビニールをゆっくり剥がす．
- 小さい部品なので飛び散らないように注意すること．

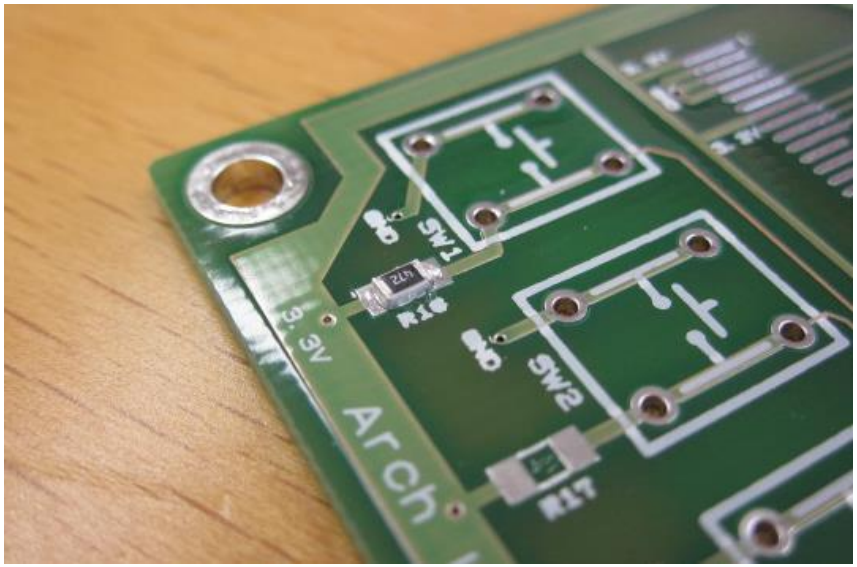


1. チップ抵抗，コンデンサ，ダイオードの固定方法

- プラスチック片に接着剤をのせる.

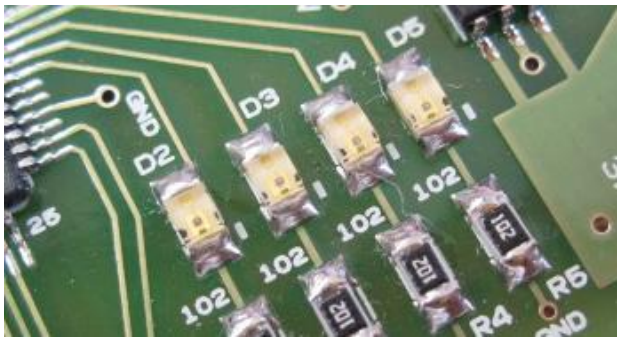
- 竹串の先に少量の接着剤をつける.
- 基板の部品固定部分に接着剤をぬる.
- ピンセットを使って部品を固定する.
  部品を固定場所に置いて，接着剤の付いていない別の竹串で抑えるとうまくいく.



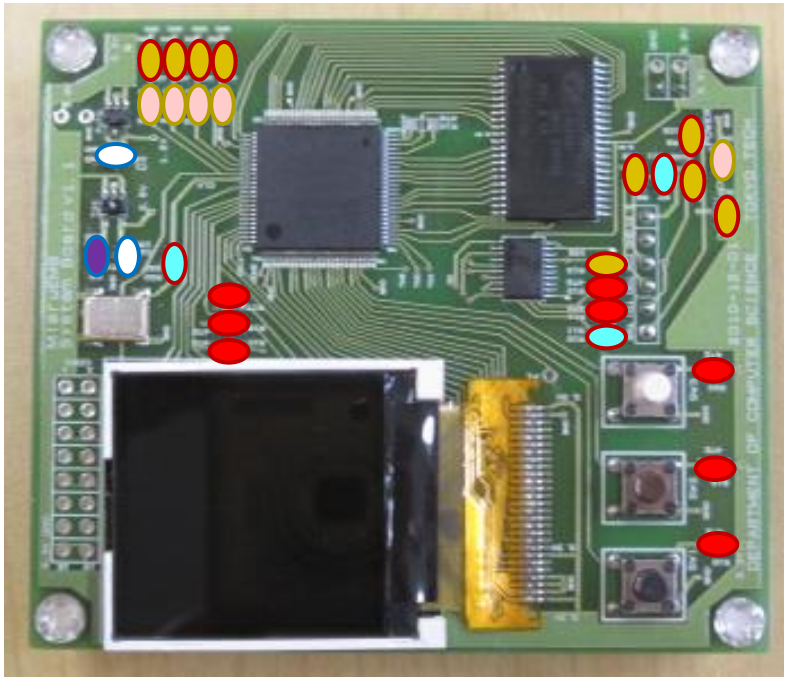


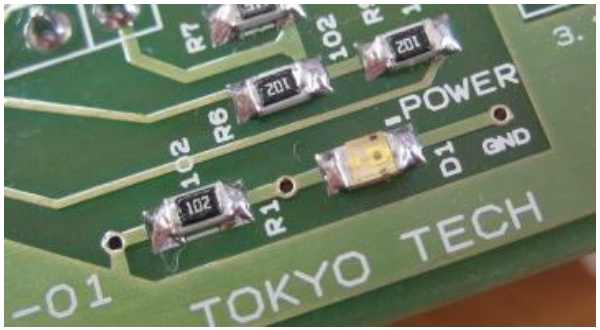
1. チップ抵抗，コンデンサ，ダイオードの固定（30～50分）

- (a) チップ抵抗を固定
  - 472 (4.7KΩ) 8個
  - 102 (1KΩ) 9個
  - 511 (510Ω) 3個
    - R20 330 にも 511(510Ω)を使う.
- (c) チップコンデンサの固定
  - C1 と C2 は 105 (3個入りのパッケージ) 2個
  - C3 は 103 (2個入りのパッケージ) 1個
  - を使う. 要注意.
- (b) ダイオードの固定 5個
  - ダイオードには極性がある（正しい方向で固定）.
  - 黒色のマークがある方をGND側にする.
- 全てのチップ抵抗，チップコンデンサ，ダイオードを固定(CP3)



写真ははんだ付け後のもの，ダイオードの向きに注意





写真ははんだ付け後のもの，ダイオードの向きに注意

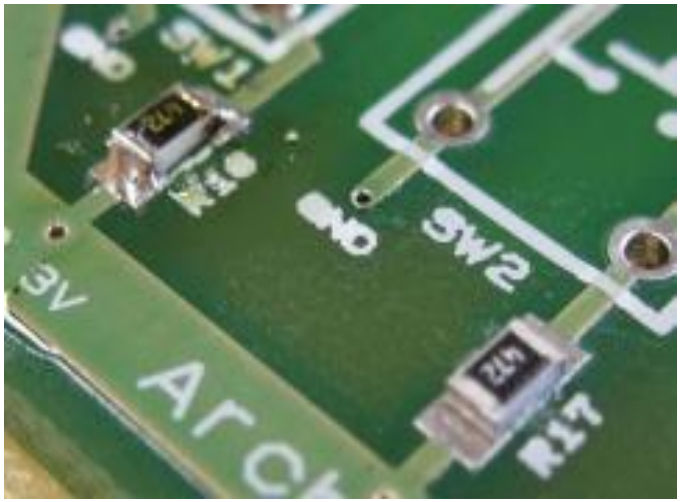
2. チップ抵抗，コンデンサ，ダイオードのはんだ付け方法

- 接着剤が乾くまで5分～10分ほど待つ.
- こて台にスポンジを入れ少量の水を注入.

- はんだ付けする部分にフラックスをぬる.
- 以下の手順ではんだ付けを行う:
  - Step1. はんだ付けする部分をこてで加熱
  - Step2. 加熱部にはんだ線を付ける
  - Step3. 加熱部からはんだ線を離す
  - Step4. 加熱部からこてを離す

はんだごて（青），こて台（黒+スポンジ），フラックス（右下）

- はんだごての先は熱いのでやけどに注意すること.
- 十分に換気すること.



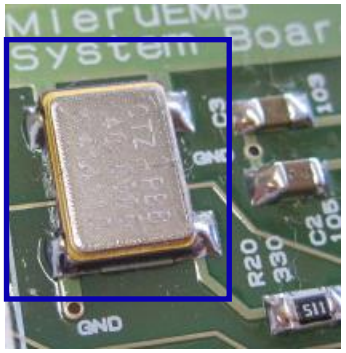
奥のチップ抵抗(R18)のはんだ付けをしたところ.
手前(R17)は固定したところ.

2. チップ抵抗，コンデンサ，ダイオードのはんだ付け（20～40分）

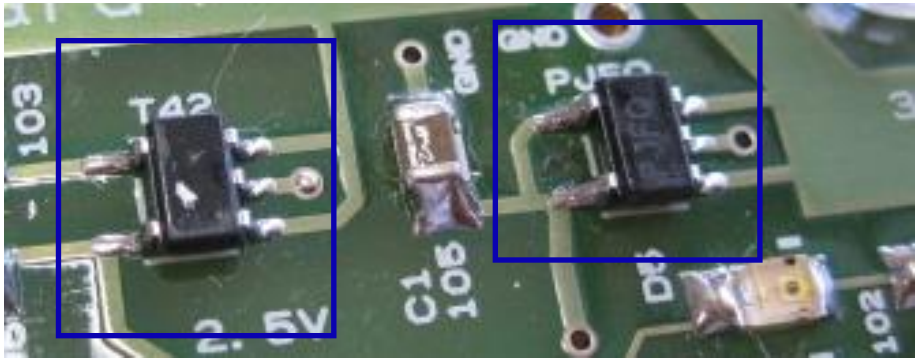
- 固定したチップ抵抗，コンデンサ，ダイオードのはんだ付け (CP4)

3.3Vの供給により，
POWER LED (D1)が点灯することを確認してもらう

3. オシレータ，レギュレータ，ICの固定（30～50分）



(a) オシレータ



(b) レギュレータ 2個（異なる部品なので固定場所に注意）

(c) PROM

(d) SRAM

(e) FPGA

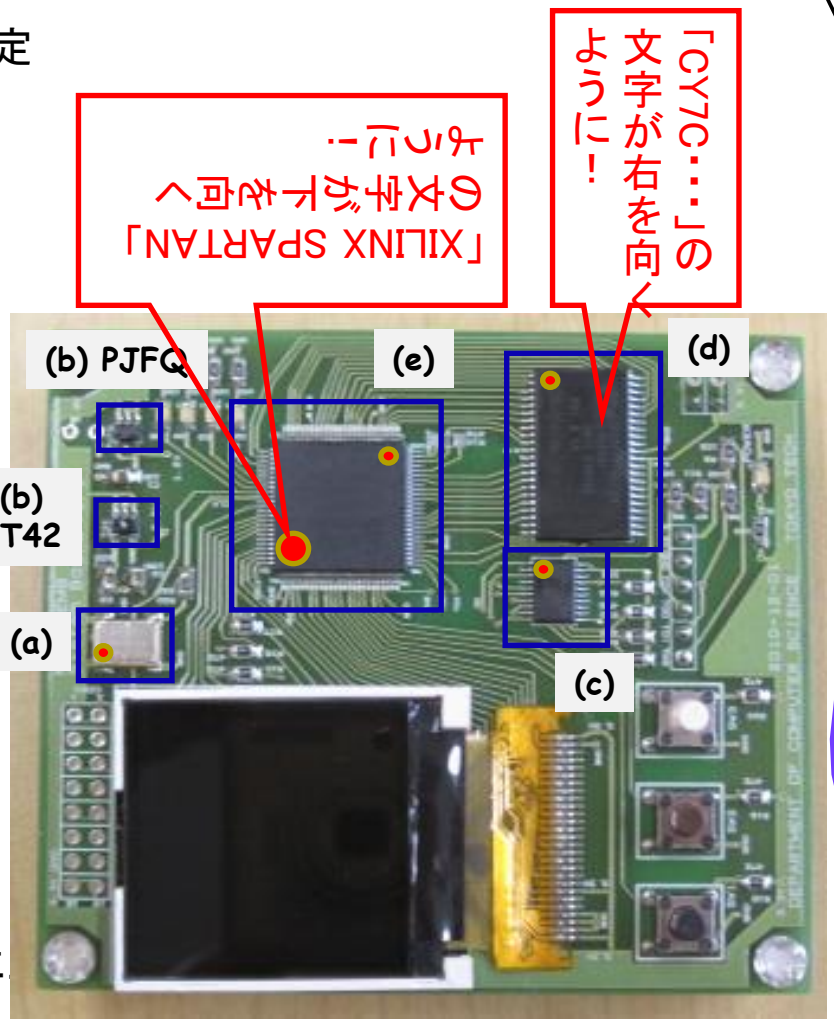
写真ははんだ付け後のもの

3. オシレータ，レギュレータ，ICの固定（30～50分）

- 悪い例
  - FPGAが右側に寄っている．
  - 2つのピンがショートするため，正しく動作しない．

3. オシレータ，レギュレータ，ICの固定（30～50分）

- 接着剤を用いて，オシレータ，レギュレータ，ICを固定
  - (a) クロックオシレータ
    - 方向に注意．右写真，丸印が左下になる．
  - (b) レギュレータ
    - PJFQの刻印のあるものを上，T42と刻印のあるものを下に．
  - (c) PROM
    - 丸印が左上になるように．
    - ピンを確実に接続するように位置調整．
  - (d) SRAM
    - 丸印が左上になるように．
    - CY7C･･･の文字が右を向くように．
    - ピンを確実に接続するように位置調整．
  - (e) FPGA
    - 丸印(大)が左下，丸印(小)が右上
    - Xilinx SPARTANの文字が下を向くように
    - ピンを確実に接続するように位置調整．



(CP5)

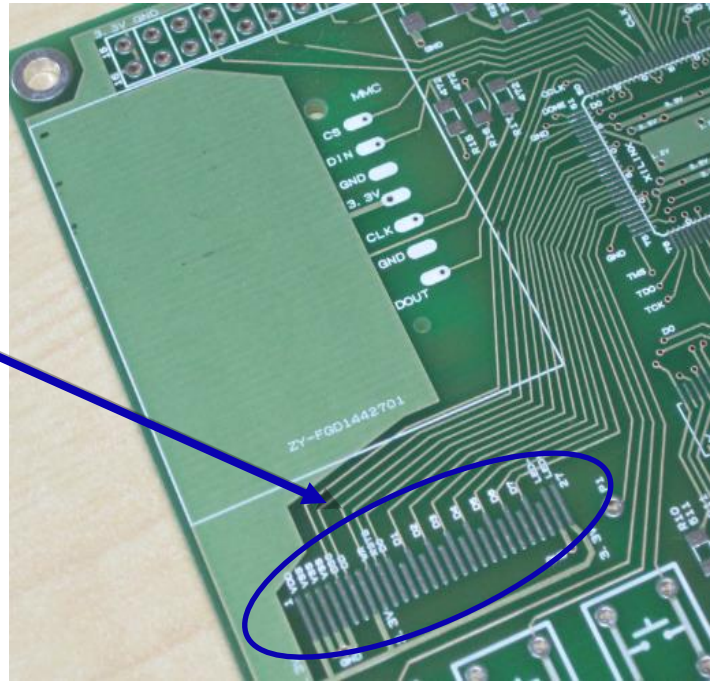
FPGA

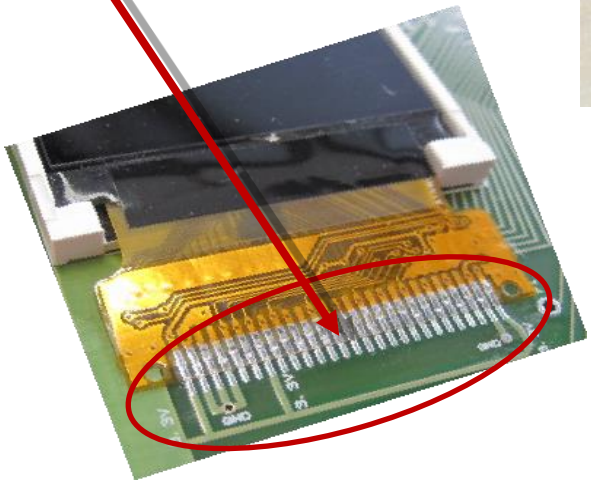
4. オシレータ，レギュレータ，ICのはんだ付け（30～50分）

- はんだ付けの前に，完全に固定するまで10分の休憩をとる！
  - 使わない時にははんだごての電源を切ること．
- オシレータ，レギュレータ，ICのはんだ付け
  - 多量のフラックスを利用する．
  - 2本のピンが接続されるブリッジがおきないように．
  - はんだが多すぎる場合には，はんだ吸取線を使う．
    または，TAに相談．
- (CP6)

5. 液晶モジュール等のはんだ付け（20～40分）

- 液晶モジュール等のはんだ付け（1／2）
  - (a) 液晶モジュール
    - 基板のこの部分にフラックスをぬる.
    - 基板のこの部分にはんだをもる.
    - さらに，その上にフラックスをぬる.
    - 位置を慎重に固定して，
      液晶モジュールの上からはんだ付け.
    - 基板と液晶モジュールの裏側が
      はんだ付けされる.

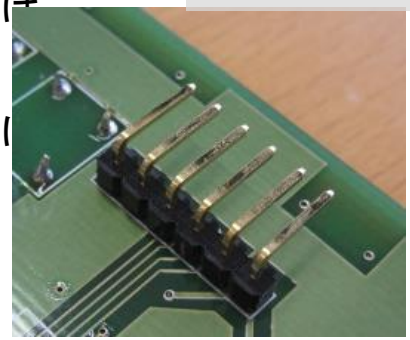




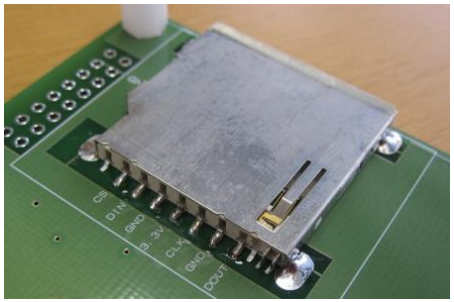
5. 液晶モジュール等のはんだ付け（20～40分）

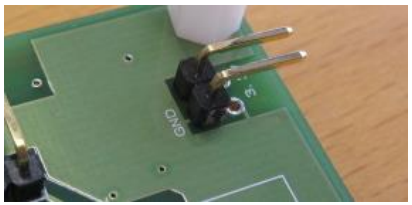
- 液晶モジュール等のはんだ付け（2／2）
  - (b) スイッチ
    - 3個のスイッチをはんだ付け.
  - (c) SDカードスロット
    - 基板裏面に，SDカードスロットをはんだ付け.
  - (d) 6ピンヘッダ
    - 基板裏面に，6ピンヘッダをはんだ付け
  - (e) 2ピンヘッダ
    - 基板裏面に，2ピンヘッダをはんだ付け
  - (d) スペーサ
    - ネジで4個のスペーサを固定する.
- (CP7)

裏面にはんだ付け

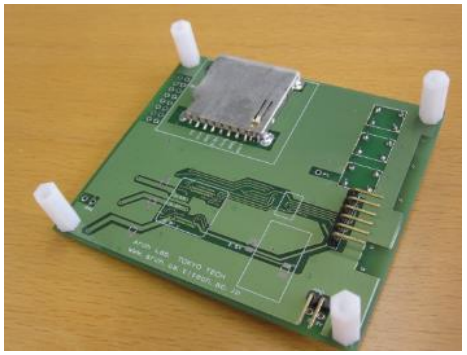


裏面にはんだ付け





裏面にはんだ付け



基板裏面

動作確認

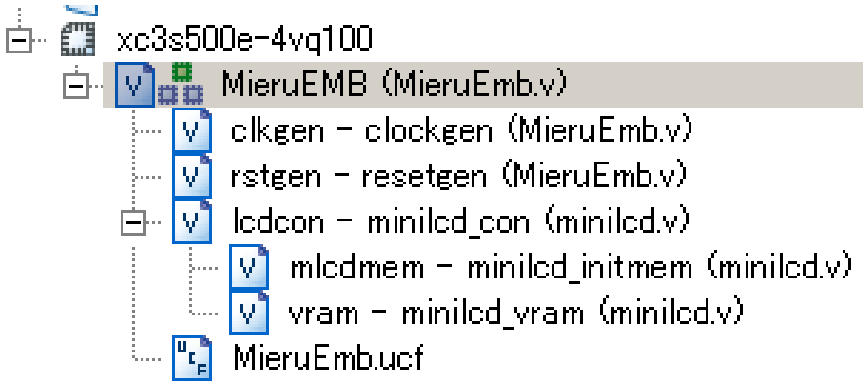
- テスターを用いて電圧を確認
  - 2.5V
  - 1.2V
- FPGAの動作確認
- PROMの動作確認
- スイッチの動作確認
- LEDの動作確認
- 液晶モジュールの動作確認
- SDカードの動作確認

- テスターの操作マニュアルは以下
  - http://akizukidenshi.com/download/P-10manual.pdf
- ハードウェアデバッグについては以下
  - Z:¥Emb¥Doc¥ Emb-HWDebug.pdf

液晶モジュールのサンプルプロジェクト lcd02

- File → Open Project
  - Z:¥Emb¥ISE¥lcd02¥fpga¥main.xise

- 論理合成して，FPGAに書き込み．



液晶モジュールのサンプルプロジェクト lcd02

- C言語の擬似コード

```
int x = 0;
int y = 33;
int color = 7;

while(1) {
   x++;
   draw_dot(x, y, color);
}
```

```
module MieruEMB(CLK, SW, ULED, LCD_CSO, LCD_CD, LCD_WR, LCD_RSTB, LCD_D);
    input         CLK;
    input [2:0]   SW;
    output [3:0]  ULED;
    output        LCD_CSO, LCD_CD, LCD_WR, LCD_RSTB;
    output [7:0]  LCD_D;

    assign ULED = 0;
    wire FCLK, RST_X, LOCKED;

    clockgen clkgen(CLK, FCLK, LOCKED);
    resetgen rstgen(FCLK, ((SW[0] | SW[1] | SW[2]) & LOCKED), RST_X);

    /*************************************************************************************
    reg [22:0] cnt;
    always @(posedge FCLK or negedge RST_X) begin
        if (!RST_X) cnt <= 0;
        else        cnt <= cnt + 1;
    end

    reg [6:0] x; // x location
    always @(posedge cnt[22] or negedge RST_X) begin
        if (!RST_X) begin
            x <= 0;
        end else begin
            x <= x + 1;
        end
    end

    wire [2:0] color = 3'b111;
    wire [6:0] y = 33;

    minilcd_con lcdcon(.CLK(FCLK), .RST_X(RST_X),
                       .VRAM_ADDR({y, x}), .VRAM_DATA({1'b0, color}), .VRAM_WE(1),
                       .LCD_CSO(LCD_CSO), .LCD_CD(LCD_CD),
                       .LCD_RSTB(LCD_RSTB), .LCD_D(LCD_D), .LCD_WR(LCD_WR));
endmodule
```

resetgen は，リセット信号 RST_X を生成．
3個のスイッチが押されるとリセットとしている．

cnt[22] をクロックとして
利用している点に注意．

ナイトライダー回路

- File → Open Project
  - Z:¥Emb¥ISE¥lcd03¥fpga¥main.xise

- lcd03 は，先のlcd02と同じ内容．これをベースに編集する．
- MieruEMB.v を編集して，画面上に，ドットが左右に反射しながら移動する回路を作成する．
- ナイトライダー回路の動作確認 (CP8)

001_dot: ドットを描くプログラム

~/Emb/SDK/asm/001_dot/main.S

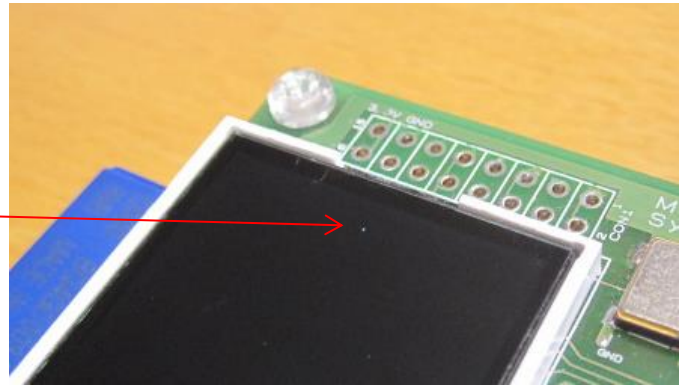
kterm

```
##################################################
# Sample Program for MieruEMB System v1.0
##################################################
        .text
        .align  2
        .globl  main
        .ent    main

main:
        .set    noreorder

        li      $3,0x900000     # $3 = vram address
        li      $2,7            # $2 = 7 (white)
        sw      $2,4288($3)     # vram[4288] = 7;
                                # (33 * 128 + 64) = 4288
$L1:
        j       $L1             # while(1);
        nop                     #

        .end    main
```

メモリへのストアにより
ドットを描く
sw ではなく sb を
使ってください.

main.S

ドット



kterm

```
##################################################
# Sample Program for MieruEMB System v1.0
##################################################
        .text
        .globl  _start
        .ent    _start
_start:
        .set    noreorder
        .set    noat

        nop
        move    $1, $0
        move    $2, $0
        move    $3, $0
        move    $4, $0
        move    $5, $0
        move    $6, $0
        move    $7, $0
        move    $8, $0
        move    $9, $0
        move    $10, $0
        move    $11, $0
        move    $12, $0
        move    $13, $0
        move    $14, $0
        move    $15, $0
        move    $16, $0
        move    $17, $0
        move    $18, $0
        move    $19, $0
        move    $20, $0
        move    $21, $0
        move    $22, $0
        move    $23, $0
        move    $24, $0
        move    $25, $0
        move    $26, $0
        move    $27, $0
        move    $28, $0
        move    $29, $0
        move    $30, $0
        move    $31, $0
        li $sp, 0x7ff00     # stack pointer 512KB
        j       main        # jump to the main
        nop

        .end _start
```

すべてのレジスタを
値0で初期化する.

スタックポインタの値
を適切に設定し, ジャンプ

start.S

001_dot: ドットを描くプログラム

kterm

```
ENTRY(_start)

SECTIONS
{
  .startup 0x0000   : { startup.o(.text) }
  . = 0x0200;

  .init             : { KEEP (*(.init)) } = 0
  .plt              : { *(.plt) }
  .text             : { *(.text .stub .text.* .gnu.linkonce.t.*)
                        KEEP (*(.text)) } = 0
  .fini             : { KEEP (*(.fini)) } = 0
  .rodata           : { *(.rodata .rodata.* .gnu.linkonce.r.*) }
  .tdata                 : { *(.tdata .tdata.* .gnu.linkonce.td.*) }
  .tbss              : { *(.tbss .tbss.* .gnu.linkonce.tb.*) *(.tcommon) }
  .ctors            : { start_ctors = .;
                        KEEP (*(SORT(.ctors.*)))
                        KEEP (*(.ctors))
                           end_ctors = .; }
  .dtors            : { start_dtors = .;
                        KEEP (*(SORT(.dtors.*)))
                        KEEP (*(.dtors))
                        end_dtors = .; }
  .data             : { *(.data .data.* .gnu.linkonce.d.*)
                        SORT(CONSTRUCTORS) }
  .got.plt         : { *(.got.plt) }
  . = .;
  _gp = ALIGN(16) + 0x7ff0;
  .got              : { *(.got) }
  .bss              : { *(.dynbss)
                        *(.bss .bss.* .gnu.linkonce.b.*)
                        *(COMMON) }
}
```

stdld.script

リンカスクリプトとディスアセンブル出力

```
init:     file format elf32-tradlittlemips

Disassembly of section .startup:

00000000 <_start>:
   0:   00000000        nop
   4:   00000821        move    at,zero
   8:   00001021        move    v0,zero
   c:   00001821        move    v1,zero
  10:   00002021        move    a0,zero
  14:   00002821        move    a1,zero
  18:   00003021        move    a2,zero
  1c:   00003821        move    a3,zero
  20:   00004021        move    t0,zero
  24:   00004821        move    t1,zero
  28:   00005021        move    t2,zero
  2c:   00005821        move    t3,zero
  30:   00006021        move    t4,zero
  34:   00006821        move    t5,zero
  38:   00007021        move    t6,zero
  3c:   00007821        move    t7,zero
  40:   00008021        move    s0,zero
  44:   00008821        move    s1,zero
  48:   00009021        move    s2,zero
  4c:   00009821        move    s3,zero
  50:   0000a021        move    s4,zero
  54:   0000a821        move    s5,zero
  58:   0000b021        move    s6,zero
  5c:   0000b821        move    s7,zero
  60:   0000c021        move    t8,zero
  64:   0000c821        move    t9,zero
  68:   0000d021        move    k0,zero
  6c:   0000d821        move    k1,zero
  70:   0000e021        move    gp,zero
  74:   0000e821        move    sp,zero
  78:   0000f021        move    s8,zero
  7c:   0000f821        move    ra,zero
  80:   3c1d0007        lui     sp,0x7
  84:   37bdff00        ori     sp,sp,0xff00
  88:   1000005d        b       200 <main>
  8c:   00000000        nop

Disassembly of section .text:

00000200 <main>:
 200:   3c030090        lui     v1,0x90
 204:   24020007        li      v0,7
 208:   ac6210c0        sw      v0,4288(v1)
 20c:   1000ffff        b       20c <main+0xc>
 210:   00000000        nop
        ...
```

make dump
コマンドによりディスアセンブル

ナイトライダー(アセンブリ言語版)

- Z:¥Emb¥SDK¥asm¥003_night¥main.S
  - 002_bar と同じ内容. これをベースに編集する.

- アセンブリ言語にて, 画面上に, ドットが左右に反射しながら移動する回路を作成する.
  - 美しくみえる様に修正する.

- ナイトライダーの動作確認 (CP10)

103_aba: 文字を表示するプログラム（C言語版）

Z:¥Emb¥SDK¥app¥103_aba¥main.c

```
/*******************************************************
static const unsigned char fonts[2][16][8] = {
    {{0,0,0,0,0,0,0,0},//A
     {0,0,0,0,0,0,0,0},
     {0,0,0,0,0,0,0,0},
     {0,0,0,0,1,0,0,0},
     {0,0,0,0,1,0,0,0},
     {0,0,0,1,0,1,0,0},
     {0,0,0,1,0,1,0,0},
     {0,0,0,1,0,1,0,0},
     {0,0,0,1,0,1,0,0},
     {0,0,1,0,0,0,1,0},
     {0,0,1,1,1,1,1,0},
     {0,0,1,0,0,0,1,0},
     {0,0,1,0,0,0,1,0},
     {0,0,0,0,0,0,0,0},
     {0,0,0,0,0,0,0,0},
     {0,0,0,0,0,0,0,0}},

    {{0,0,0,0,0,0,0,0},//B
     {0,0,0,0,0,0,0,0},
     {0,0,0,0,0,0,0,0},
     {0,0,1,1,1,1,0,0},
     {0,0,1,0,0,0,1,0},
     {0,0,1,0,0,0,1,0},
     {0,0,1,0,0,0,1,0},
     {0,0,1,1,1,1,0,0},
     {0,0,1,0,0,0,1,0},
     {0,0,1,0,0,0,1,0},
     {0,0,1,0,0,0,1,0},
     {0,0,1,0,0,0,1,0},
     {0,0,1,1,1,1,0,0},
     {0,0,0,0,0,0,0,0},
     {0,0,0,0,0,0,0,0},
     {0,0,0,0,0,0,0,0}}
};
```

文字 A のフォントデータ.

文字 B のフォントデータ.

```
/*******************************************************************************/
void mylib_putc(int x, int y, char c, int color){
    int i, j;

    for(i=0; i<16; i++){
        for(j=0; j<8; j++){
            if(fonts[(int)(c-'A')][i][j]) e_vram[(x+j)+(y+i)*128] = color;
        }
    }
}

/*******************************************************************************/
int main(void){
    mylib_putc(0,  0, 'A', 7);
    mylib_putc(8,  0, 'B', 7);
    mylib_putc(16, 0, 'A', 2);

    while(1);
}
/*******************************************************************************/
```

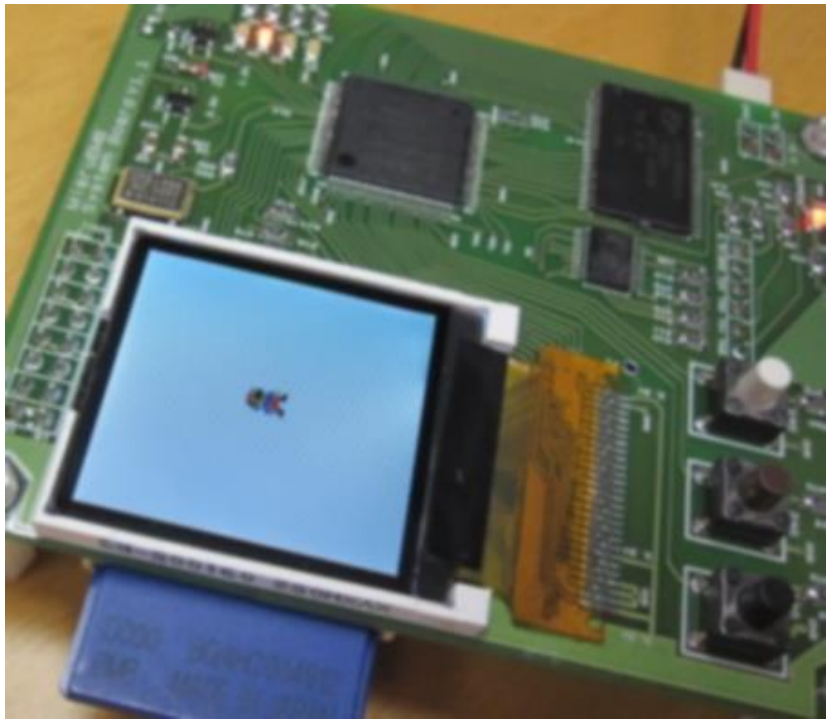
x,y で指定した場所に色 7 にて，A という文字を表示.

main.c

104_pic：画像表示とスイッチ入力のサンプル（C言語版）

Z:¥Emb¥SDK¥app¥104_pic¥

- SW1 左移動，SW2 上移動，SW3 右移動，SW1 & SW3 下移動



ナイトライダー（C言語版）

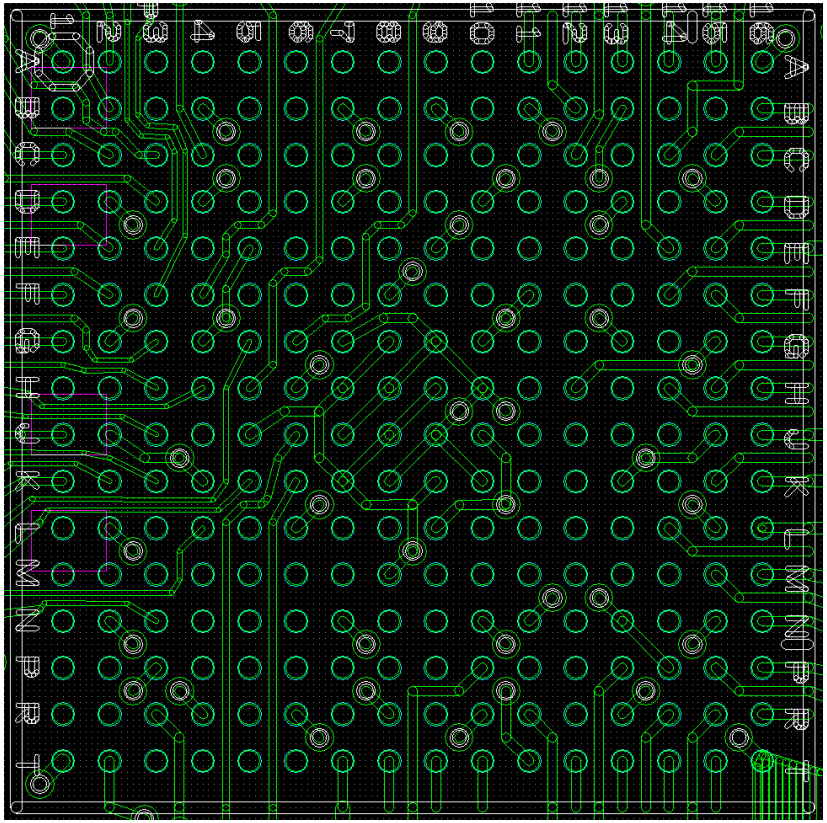
- Emb¥SDK¥app¥200_night¥
  - 104_pic と同じプログラム，これをベースに修正．

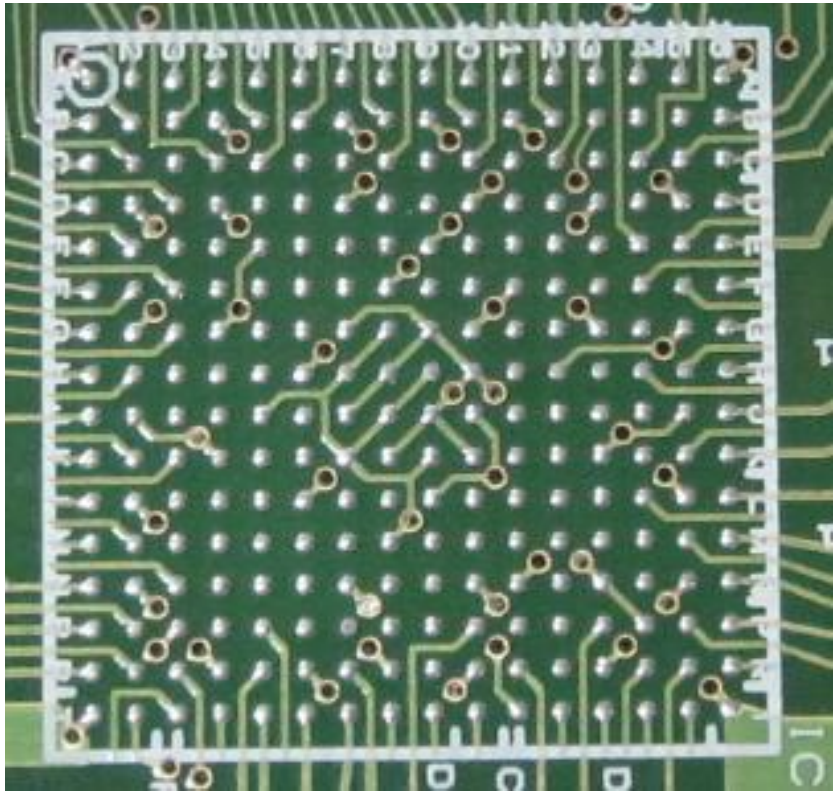
- C言語にて，画面上に，ドットが左右に反射しながら移動する回路を作成する．
  - 美しくみえる様に修正する．

- ナイトライダーの動作確認 (CP11)

- ナイトライダーを，（1）Verilog HDLによるハードウェア実装，（2）アセンブリ言語によるソフトウェア実装，（3）C言語によるソフトウェア実装という3種類で記述した．
  それぞれの利点，欠点を考えてみる．
- app 以下にたくさんのサンプルプログラムがあるのでこれらを動かしてみる．
  - 106_fig, 107_gpio, 125_space など

BGAのはんだ付けは？





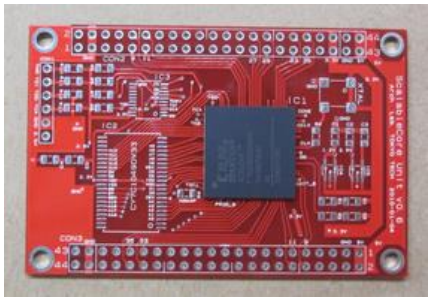
17mm

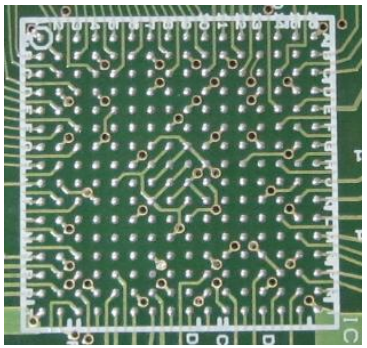
17mm

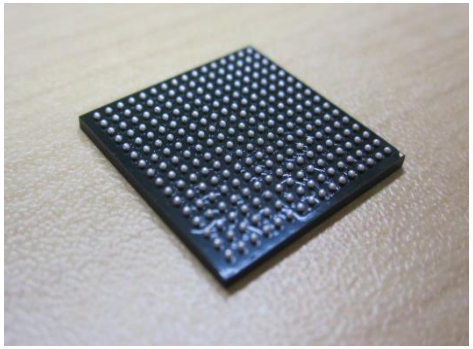
BGAのはんだ付け

- HAKKO 882 ヒートガン（チップから12cm, 3分30秒）

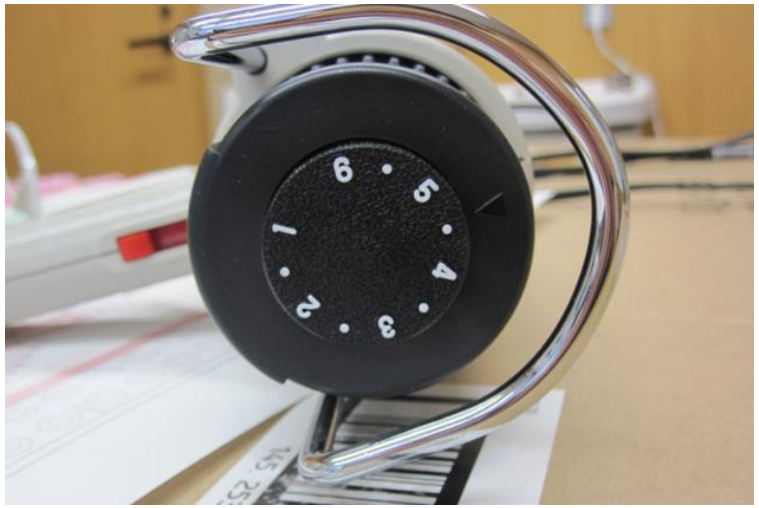
Quad Flat Package



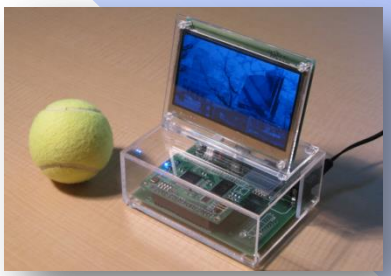


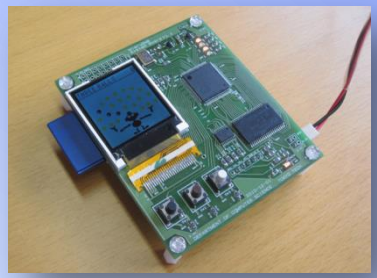


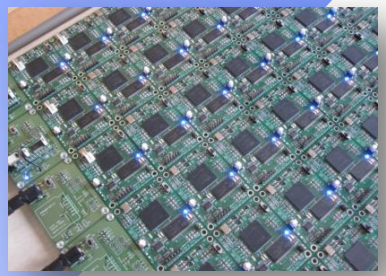
Ball Grid Array



議論：そこから見えてきた今時のハードとソフトの学び方







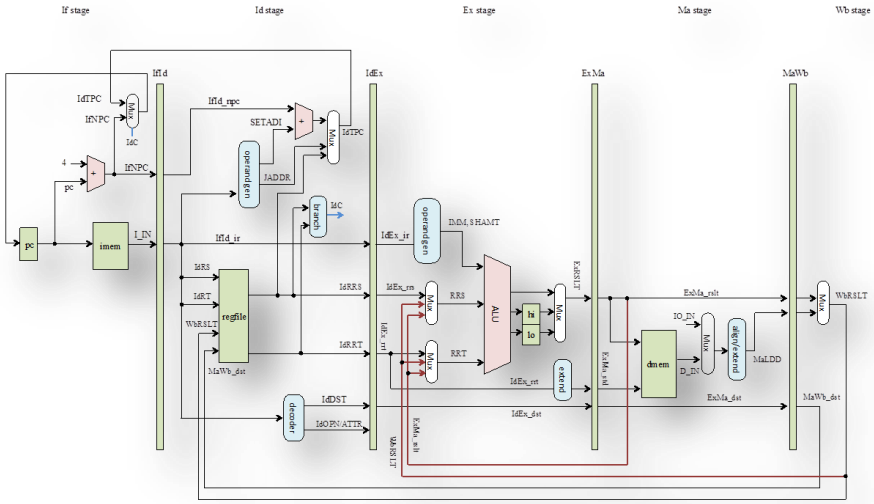
作ったハードウェアは本当に動くんですか？

- MieruPC は，MieruPC株式会社を通じて，100台以上が出荷されています．
- 組み込みシステムMieruEMBは，東工大 情報工学科の情報実験で，80人の学生が80台を作成しています．
  全て動作しています．実験中，深刻な故障なく動いています．
  - MieruEMBは，「東京都立産業技術高専」でも採用されています．

FPGAに搭載できるハードウェア規模と性能は十分？

- MieruEMB は，Xilinx Spartan3E XC3S500E を搭載
  - マルチサイクルの2ビットRISCプロセッサ（MIPS準拠ただし浮動小数点演算は非対応）が2個実装できるサイズです．

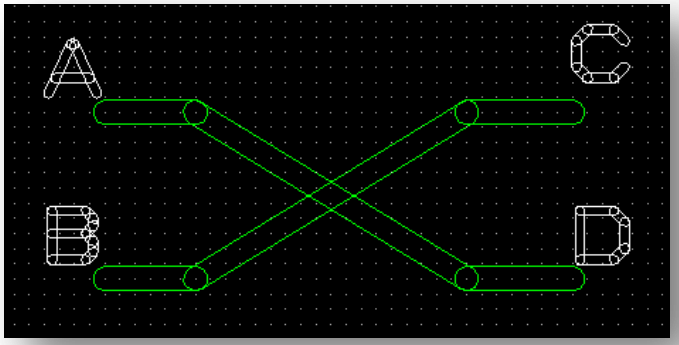
- パイプライン化してキャッシュを搭載（計算機アーキテクチャの学習に最適）すると，性能は 20 MIPS程度，1980年代の計算機の性能に相当



NEC PC-8801mkII FR カタログ

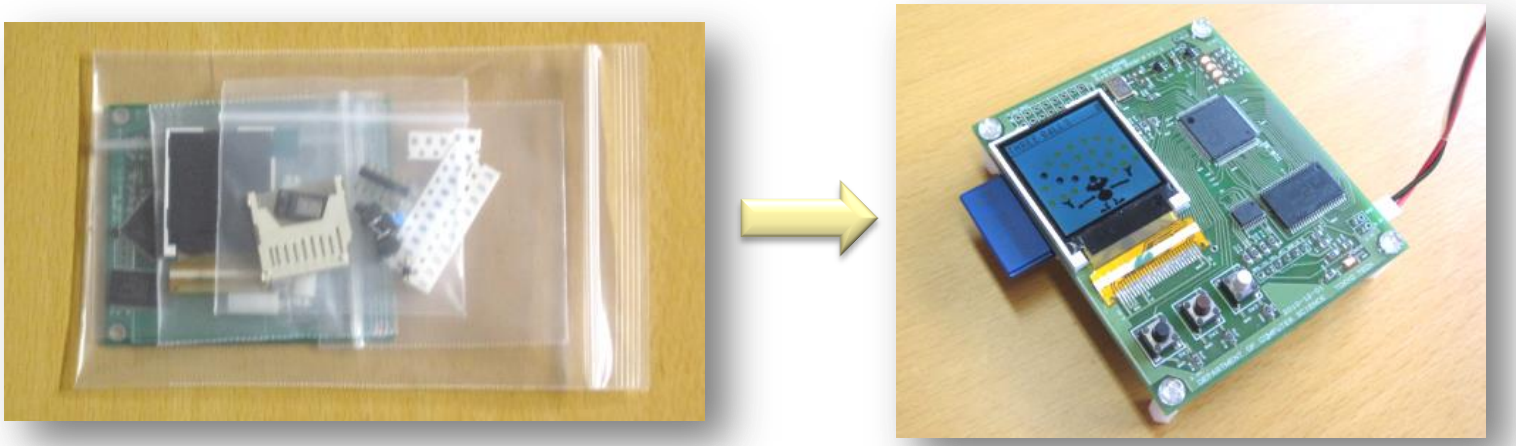
どうしてFPGAを使うの？

- やわらかいハードウェアなので使いやすい.
- 2層基板との相性が良い.
  - 大部分は接続すれば良い.
- 実現したいハードウェア(プロセッサなど)を変更できる.



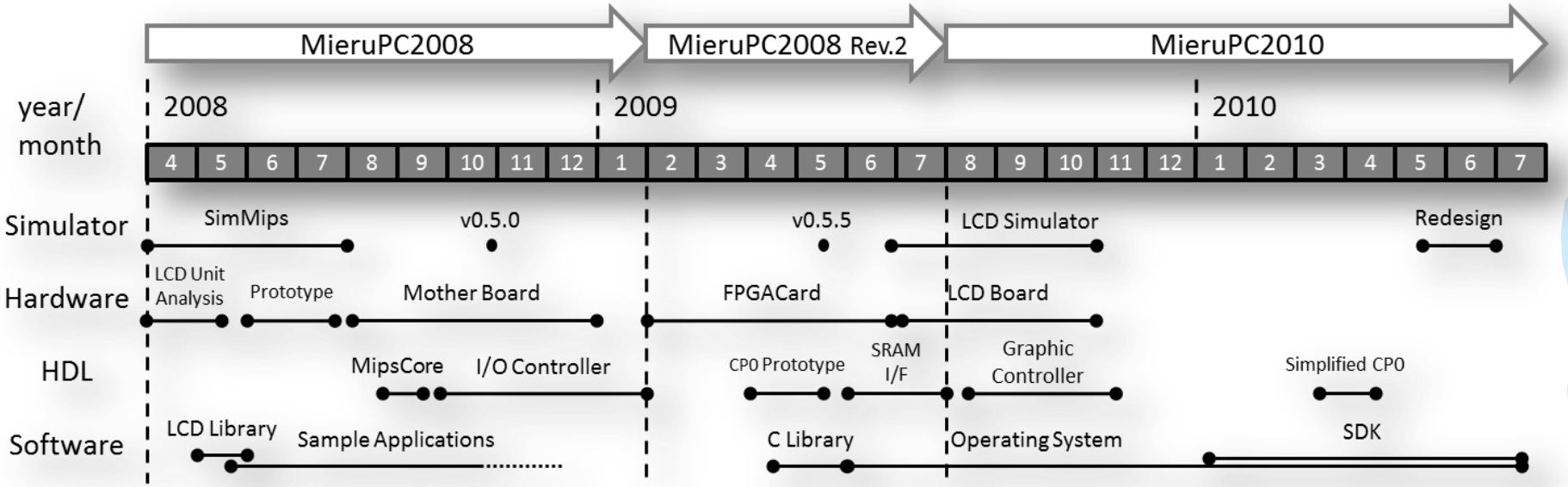
部品等が高価ではないですか？

- MieruEMBシステム20セットの原価は10万円程度（100セット導入時）.
- MieruEMBシステムを設計，修正するためのソフトウェアは全て無料で利用できます.
- MieruEMBシステムの設計データや，実験資料などは無償で利用可能です.
  - http://www.arch.cs.titech.ac.jp/lecture/emb/
  - 100枚を超える実験スライドが利用可能です.
  - はんだ付けのビデオもあります.



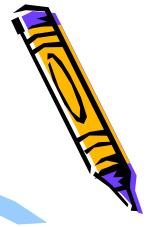
システム開発に長い期間が必要？

- MieruPCの開発には3年程度を要しています．
- MieruEMBの開発には1．5年程度を要しています．
  - これらのデータを活用すれば，短期間でシステム開発が可能です．



吉瀬謙二: シンプルな計算機システムの開発に向けた挑戦, 情報処理学会論文誌, Vol.54, No.7, pp1902-1912 (July 2013).

ハードウェア開発は敷居が高くないですか？

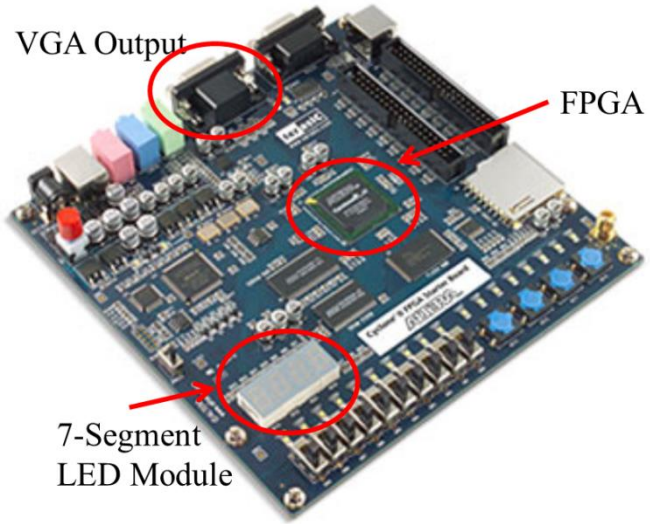


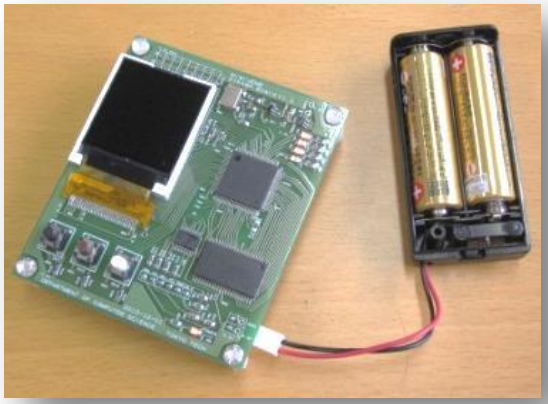
- MieruEMBのFPGAに実装する回路は約2,000行の Verilog HDL で記述されています.
  - そこに含まれるプロセッサは，約650行です.
- Verilog HDLとハードウェアの基礎知識があれば，１週間程度で習得できます.

- Verilog HDL はちょっと大変
  - 大丈夫，**JavaRock** があります.
  - **JavaRock**ではじめる高位合成による気楽なFPGA開発：三好健文（イーツリーズ・ジャパン）

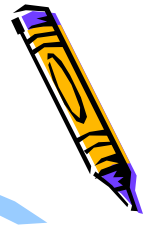
魅力的な計算機システムを作りたい！

- 誰にとって魅力的か？
  - 教員
  - 学生
  - ？？





教育目的のハードウェア(計算機システム)設計・開発のヒント



- 安定して動作するハードウェアの開発を目指す.
- 自分で扱える複雑さのハードウェアを開発する.
  - SRAMにするか DRAMにするか?
  - シリアルにするか, USBにするか?
- 配線の本数をできるだけ少なくする.
  - 8bitメモリ, 16bitメモリ
  - FPGAのピンを全てを引きだそうとしない. 必要な分だけ使う.
- インクリメンタルな開発
  - 実績のある部分は変更しない. 挑戦的な要素はひとつずつ.
- 自慢できるハードウェアを開発する.
  - デザイン, 機能
- 計算機は成熟した分野なので教科書を良く読む.
- 設計・開発を楽しむ.

どうしてFPGAのはんだ付けをするところ（大学）が少ないの？

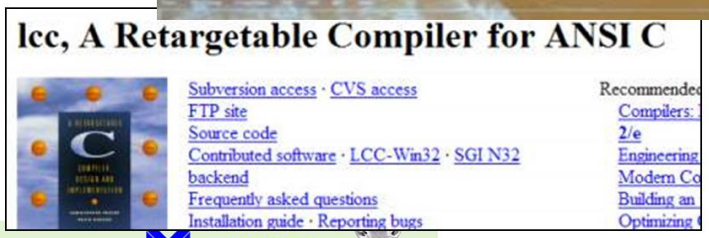
- それは．．．

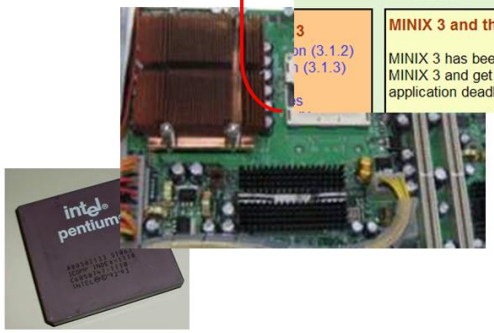
今時のハードとソフトの学び方

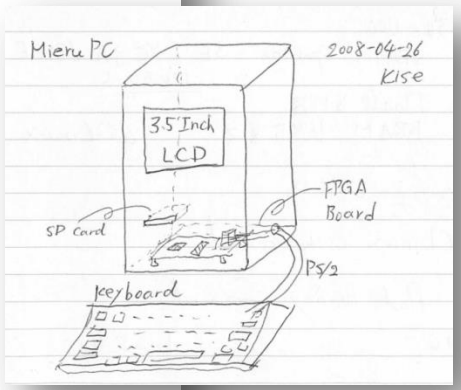
2008年4月11日／12日 東工大5類新入生セミナースライド 吉瀬謙二

高性能マイクロプロセッサと本当のマイコンピュータのすすめ

Happy Hacking!





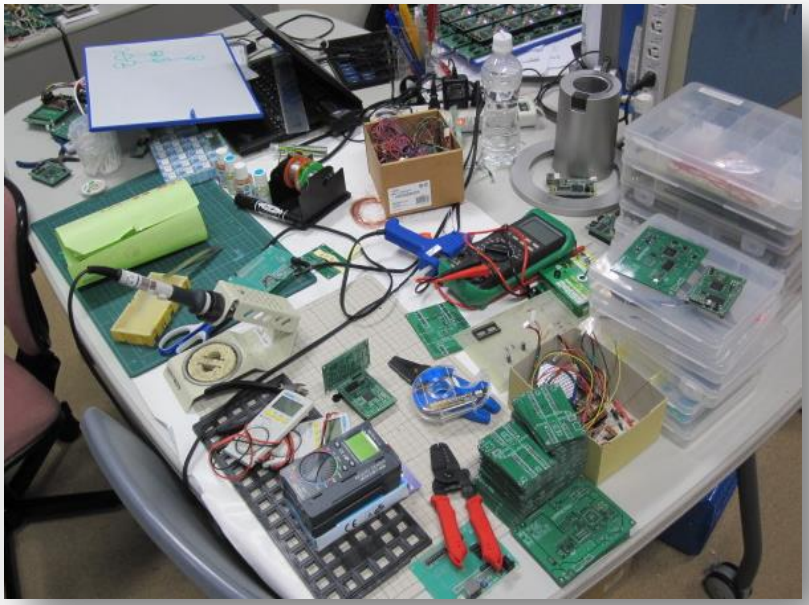


my designed computer

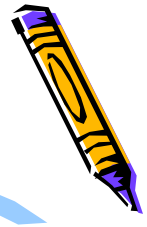
できます！ バックアップします!

おわりに

- 計算機システム（組み込みシステムを含む）がお手軽に開発できる環境が整っています．さて，どんなシステムを作りましょうか？
- 本チュートリアルが，「今時のハードとソフトの学び方」を考えるヒントになれば幸いです．
- 本チュートリアルの機会をご提供いただきましたESS2013関係者のみなさまに感謝いたします．

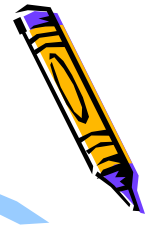


参考資料



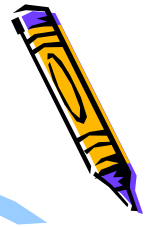
- 吉瀬研究室のホームページ
  - http:// www.arch.cs.titech.ac.jp
- MieruPC株式会社のホームページ
  - http://www.mierupc.com/
- 東工大 情報工学科 情報実験第四 組み込みシステムのホームページ
  - http:// www.arch.cs.titech.ac.jp/lecture/emb/
- プリント基板製造 P板.com
  - http://www.p-ban.com/
- 秋月電子通商
  - http://akizukidenshi.com/
- 千石電商
  - http://www.sengoku.co.jp/
- 東京エレクトロンデバイス株式会社
  - http://www.teldevice.co.jp/
- ザイリンクス株式会社
  - http://japan.xilinx.com/
- Digilent Inc.
  - http://www.digilentinc.com/
- Digi-Key Corporation
  - http://jp.digikey.com/

参考資料



- Xilinx FPGA XC3S500E-4VQG100C
  - ザイリンクス DS312 Spartan-3E FPGA ファミリ データシート
- SRAM CY7C1049DV33-10ZSXI
  - CY7C1049DV33データシート
- Mini-LCD ZY-FGD1442701V1
  - http://www.aitendo.co.jp/product/1621
  - コントローラIC ST7735
- フォトトランジスタ NJL7502L
  - http://akizukidenshi.com/catalog/g/gI-02325/
- クロックオシレータ 40MHz
  - http://akizukidenshi.com/catalog/g/gP-03617/
- SDカード 8MB
  - http://akizukidenshi.com/catalog/g/gS-02549/
- スペーサ，ネジ
  - http://akizukidenshi.com/catalog/g/gP-01861/
- タクトスイッチ
  - http://akizukidenshi.com/catalog/g/gP-01282/

参考資料



- 電池ボックス
  - http://akizukidenshi.com/catalog/g/gP-00310/
- チップLED 赤
  - http://www.sengoku.co.jp/mod/sgk_cart/search.php?multi=TLRE1002A&cond8
- チップ抵抗
  - http://www.sengoku.co.jp/mod/sgk_cart/search.php?multi=ERJ6GEYJ
- コンフィギュレーションケーブル
  - http://www.digilentinc.com/Products/Detail.cfm?NavPath=2,395,523&Prod=JTAG-USB
- MIPS32のクロス開発環境の構築
  - http://www.arch.cs.titech.ac.jp/mcore/buildroot.html
- dd for windows
  - http://www.chrysocome.net/dd