オープンソースEDAを用いたカスタムRISC-Vの設計・製作事例 :Lチカを題材として

Design and Fabrication of Custom RISC-V Chip using Open Source EDA; A Case of LED Blink

秋田純一（金沢大学）

Junichi Akita (Kanazawa University)

akita11

Interface Device Laboratory, Kanazawa University http://ifdl.jp/

自己紹介

- ☑本業：金沢大の教員
  - ☑専門：集積回路、イメージセンサ、インタラクション
  - ☑好きなプロセスはCMOS 0.35um
- ☑副業：Maker（電子工作など）

私とRISC-Vの出会い

Longan Nano（フラッシュ 64KB、RAM 20KB）

本製品のメモリは、フラッシュが64 KB、SRAMが20 KBです。メーカーのページに記載されている当初のメモリ数（フラッシュ128 KB、SRAM 32 KB）より小さいため、さらにお買い求めやすい価格に設定しました。今後、こちらの製品でメモリが大きいものを販売する際には、価格が異なる可能性があるので予めご了承ください。

RISC-V 32 bitコアを搭載したGD32VF103CBT6 MCUベースの開発ボードです。 0.96インチのIPSディスプレイ（160 × 80ピクセル）と透明なアクリルケースが付属しています。

ブレッドボード上で使用することが可能で、8 MHzパッシブ高周波水晶発振器、32.768 KHz RTC低周波水晶発振器、小型TFスロット、Type-C USBインタフェースも備えています。

名前	Longan Nano
コード番号	

4.8 ★★★★★ Google カスタマー レ

Sipeed Maixduino

Arduino IDEとライブラリを利用可能な、K210搭載のMaixシリーズのAI開発ボードです。豊富なオープンソースのライブラリを使うことで、高速なプロトタイピングや開発が可能です。2.4インチLCDとカメラモジュールがセットになっています。

Arduino環境のインストール等はSipeedのインストールガイドを参照ください。

またArduinoの他、K210用のMicroPython環境であるMaixPyでの開発も可能です。

仕様

- CPU：RISC-V デュアルコア 64 bit
- FPU：400 MHz ニューラルネットワークプロセッサ
- QVGA@60 FPS/VGA@30 FPS 画像識別
- ESP32モジュール搭載：2.4 GHz 802.11.b/g/n & Bluetooth 4.2（技適マークあり）
- Arduino Uno R3フォームファクタ
- I²Sデジタル出力 無指向性MEMSマイク搭載
- DVPカメラ用24 P/0.5 mm FPCコネクタ
- 8 bit MCU LCD用24 P/0.5 mm FPCコネクタ
- microSDカードスロット搭載
- リセット/ブートボタン搭載

名前	Sipeed Maixduino
コード番号	SIPEED-MAIXDUINO
SKU#	5707
送料区分	650
税込単価	3,564 円 (10 %off) 通常価格：3,960 円
数量	
在庫	

4.8 ★★★★★ Google カスタマー レビュー

中国でアツいらしい？

→HiFiveを買う　→積む

→LonganNano買う→ほぼ積む

→SipeedのK210に出会う

→Sipeedに行く→K210開発者の妻に会う

→Interfaceとかでお勉強

ifDL Kanazawa Univ.

Interface Device Laboratory, Ka

こんな動画をつくってみました

https://www.youtube.com/watch?v=A188CYfuKQ0
http://www.nicovideo.jp/watch/sm23660093

ニコ動でいただいたコメント

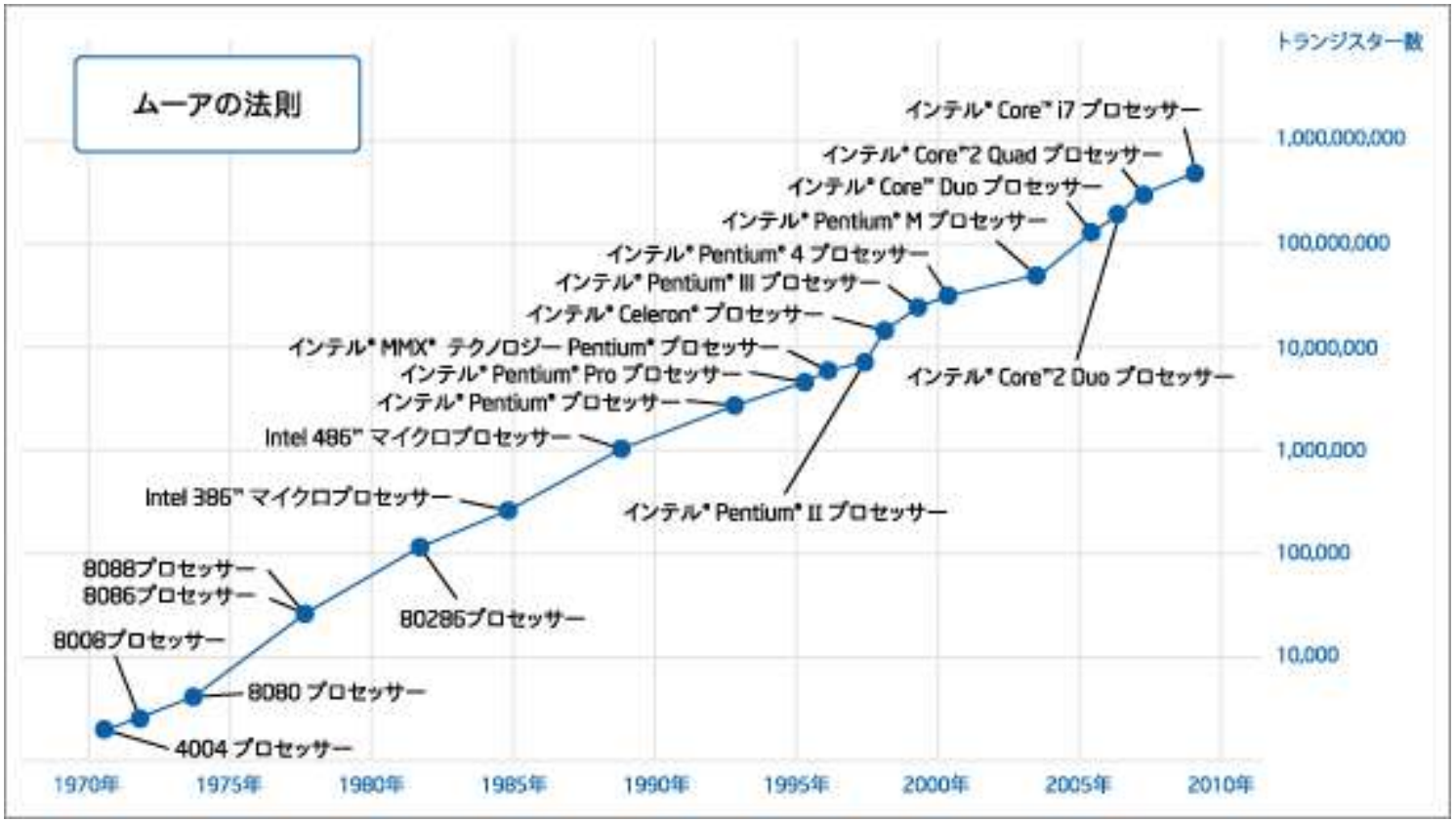
- ☑ こっから？
- ☑ ニコ技界のTOKIO
- ☑ ゲートの無駄遣い
- ☑ ここから！！？
- ☑ ひでえ、勿体ない使い方wwwww
- ☑ マジかよ。レジストレベルの設計とかガチすぎる。
- ☑ 無駄遣い過ぎるだろw
- ☑ 贅沢というかなんというか
- ☑ え？まじでここからかよ」wwww」」

- ☑ IC版FusionPCB的なところが現れれば･･･

- ☑ (FPGAでは)いかんのか？
- ☑ 俺はFPGAで我慢することにする
- ☑ いや、そこまでは必要ないです
- ☑ 量産品すらFPGA使う時代に専用LSI･･･
- ☑ アマチュアはFPGAで良いんだよなぁ･･･w

チップをつくる＝
「すごいことをするため」
という意識

LSIの歴史：Moore（ムーア）の法則



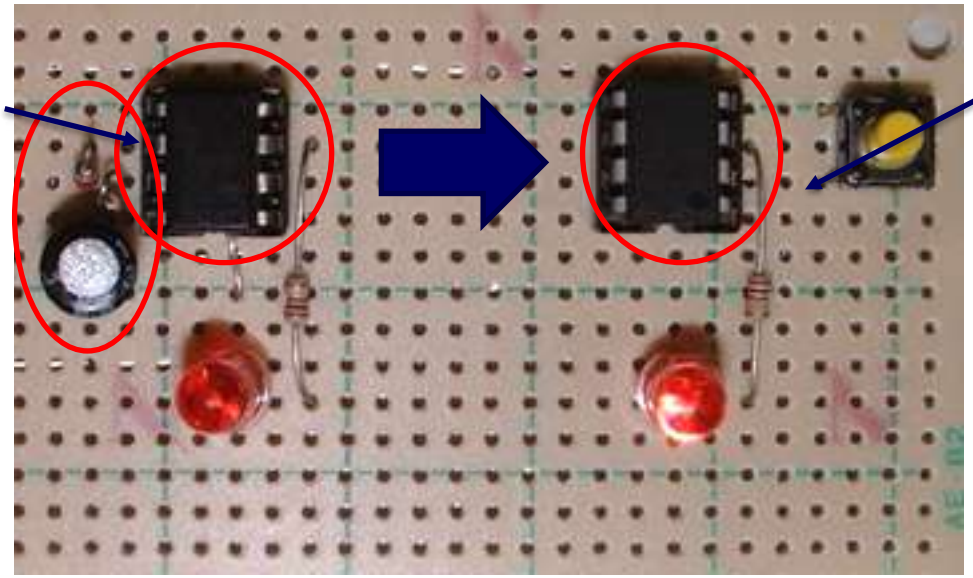
ref: http://www.intel.com/jp/intel/museum/processor/index.htm

G.Mooreが1965年に論文[1]で述べる→C.Meadが「法則」と命名→「予測」→「指針（目標）」へ
素子を微細化する＝いいことがたくさんある（性能↑、消費電力↓などなど）

[1] G.E.Moore, "Cramming more components onto integrated circuits," IEEE Solid-State Circuit Newsletter, Vol.11, No.5, pp.33-35, 1965.

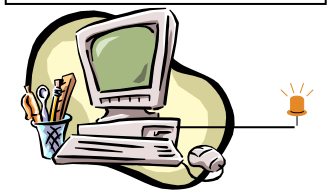
「Lチカ」のパラダイムシフト

Oscillator(555)
#Component=4
$1.5



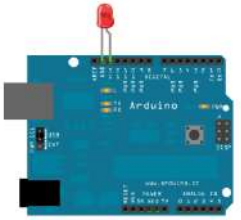
MCU
#Component=1
$1

```
while(1){
  a = 1;
  sleep(1);
  a = 0;
  sleep(1);
}
```



可能だが非現実的なLチカ

- ☑マイコン(MCU)を使って「Lチカ」しない理由がない
  - ☑コスト、機能性、・・・



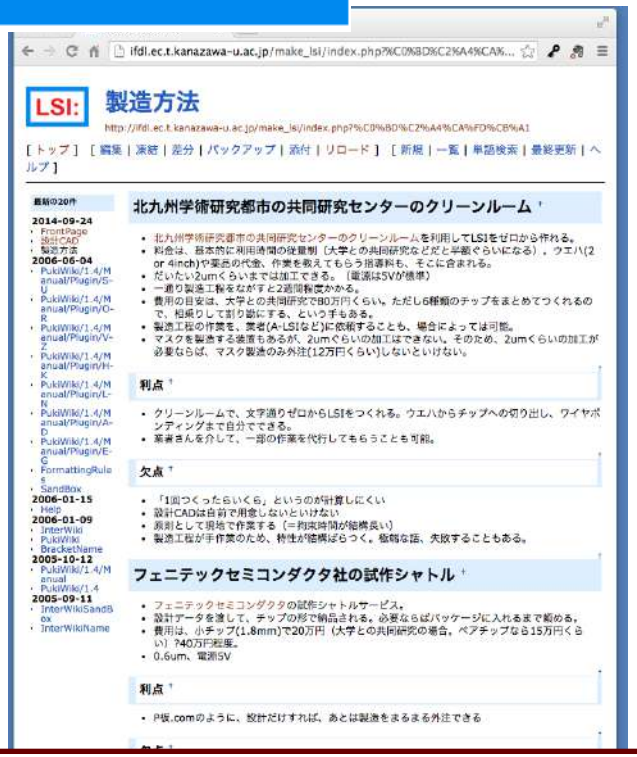
合理的なLチカ

- ☑コンピュータの使い方がもったいなくなくなった
- ☑チップ設計・製造も、Makeの「道具」になるべき

MakerでオレLSIをつくってみたい

- ☑情報収集・整理
- ☑仲間さがし
- ☑有志でつくってみる？
- ☑製造方法もいくつか
  - ☑フェニテック0.6umなど

LSI:



http://j.mp/make_lsi

・・・と思っていたら

Google, SkyWater Release Production Dev Kit, Offer Open Hardware Projects Free 130nm Manufacturing

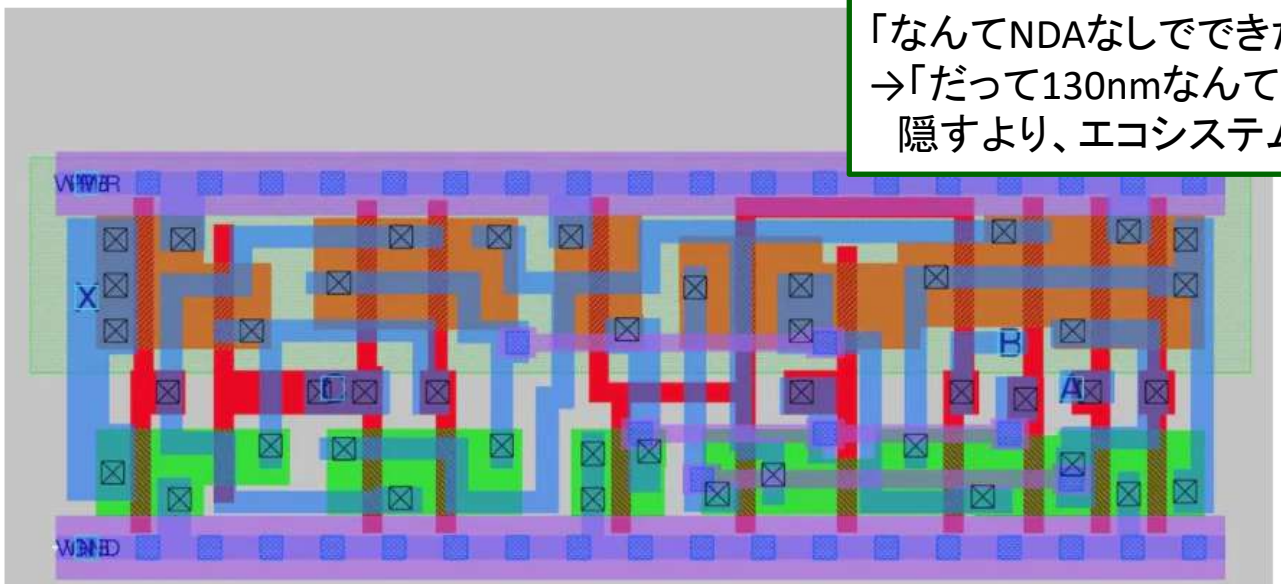
Now available in experimental form, open hardware designers can submit via the PDK for chip production this November.

Gareth Halfacree Follow

a month ago • FPGAs

NDA不要・オープンソースなチップ設計環境
やや議論が発散気味な印象だが、猛スピードで整備中
予定では11月に相乗り試作

「なんてNDAなしでできたんですかね？」
→「だって130nmなんて枯れた技術だし、
隠すより、エコシステム作ったほうがいいじゃん？」



tp://ifdl.jp/

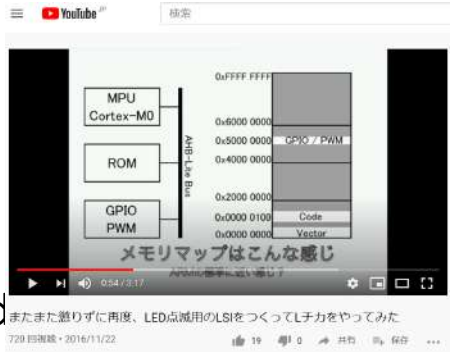
Non-commercial EDA Tools

- ☑結構ある（オープンソースも多い）
  - ☑Layout Tool: Glade, K-Layout
  - ☑Schematic Entry: KiCAD, Xscheme, …
  - ☑Circuit Simulator: LTspice / Spice3 / ngspice, …
  - ☑Synthesize/P&R: Alliance, Qflow, OpenLANE

*Open Source Softwares

これまでに・・・

- ☑「懲りずに再度、LED点滅用のLSIをつくってLチカをやってみた」
  - ☑Inkscape設計、クリーンルームで製造、センサ
- ☑「また懲りずに再度、LED点滅用のLSIをつくってLチカをやってみた 」
  - ☑555互換（デジアナ混載）
- ☑「またまた懲りずに再度、LED点滅用のLSIをつくってLチカをやってみた 」
  - ☑Cortex-M0（HDLから設計）



今回のお題：RISC-Vでしチカ

- ☑Open SourceなRISC命令セット
- ☑用途に応じて、実装する命令が階層化
  - ☑乗算器、浮動小数点、暗号化、・・・
- ☑回路の小規模下、省電力化に適している
- ☑ARMと並んで組み込み用途、特に中国で採用例が急増中

Base	Version	Frozen?
RV32I	2.0	Y
RV32E	1.9	N
RV64I	2.0	Y
RV128I	1.7	N
Extension	**Version**	**Frozen?**
M	2.0	Y
A	2.0	Y
F	2.0	Y
D	2.0	Y
Q	2.0	Y
L	0.0	N
C	2.0	Y
B	0.0	N
J	0.0	N
T	0.0	N
P	0.1	N
V	0.2	N
N	1.1	N

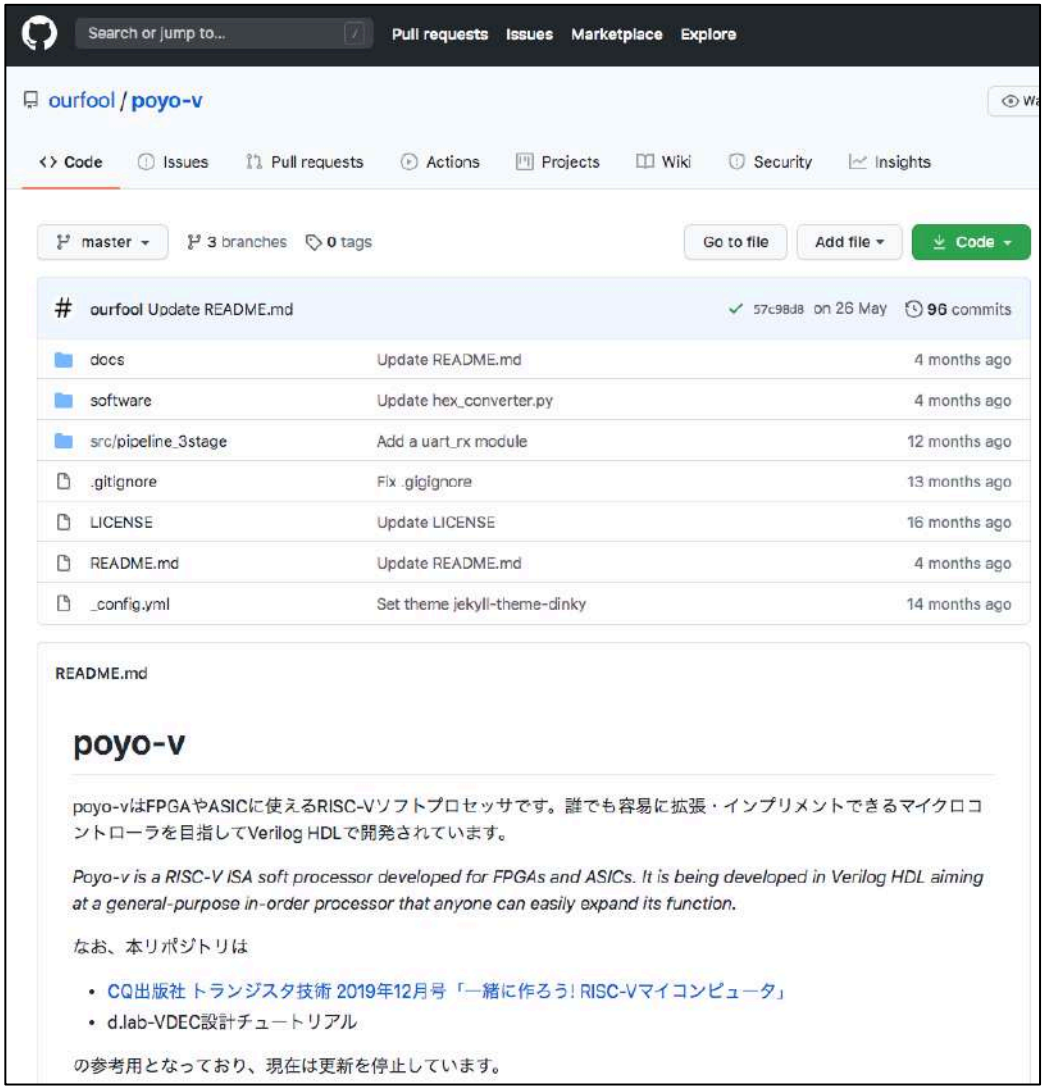
31 25	24 20	19 15	14 12	11 7	6 0	
funct7	rs2	rs1	funct3	rd	opcode	R形式
Imm[11:0]		rs1	funct3	rd	opcode	I 形式
Imm[11:5]	rs2	rs1	funct3	Imm[4:0]	opcode	S形式
Imm[12|10:5]	rs2	rs1	funct3	Imm[4:1|11]	opcode	B形式
Imm[31:12]				rd	opcode	U 形式
Imm[20|10:1|11|19:12]				rd	opcode	J 形式

The RISC-V Instruction Set Manual
Volume I: User-Level ISA
Document Version 2.2

Editors: Andrew Waterman[1], Krste Asanović[1,2]
[1]SiFive Inc.,
[2]CS Division, EECS Department, University of California, Berkeley
andrew@sifive.com, krste@berkeley.edu
May 7, 2017

https://riscv.org/specifications/

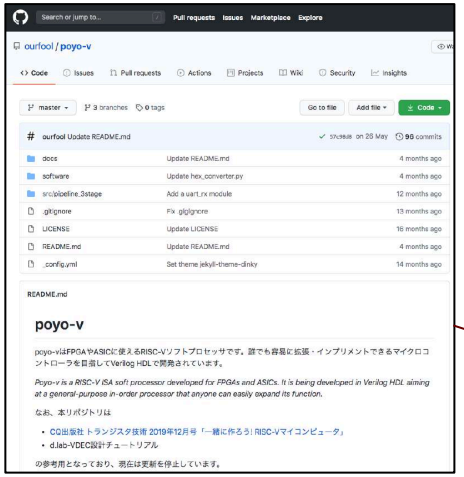
RISC-Vコアの入手



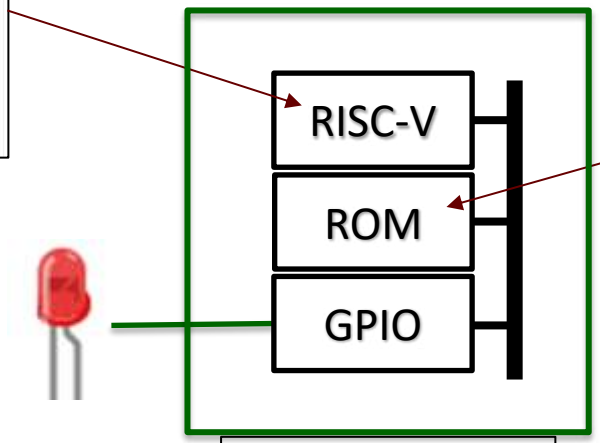
poyo-v
命令セット：RV32I
3段パイプライン
VerilogHDLソース
MITライセンス

https://github.com/ourfool/poyo-v

ペリフェラルの準備



poyo-v



チップ
（CMOS0.6um）

Lチカのプログラムを
入れておく

メモリマップ
0x00000000：プログラム
0x00001000：GPIO

プログラムの記述→命令ROM

```
0000 lui  x1,0x0
0004 addi x1,x1,0x0ff ; x1=255
0008 lui  x3,0x00001 ; GPIO base address
000c li   x4,1 / ori x0,x4,1
0010 sw   x4,0(x3) ; GPIO=1
0014 mv   x2,x0 ;  = addi x2,x0,0
0018 addi x2,x2,1 ; = x2++
001c bgeu x1,x2,00018
0020 sw   x0,0(x3) ; GPIO=0
0024 mv   x2,x0 ; = addi x2,x0,0
0028 addi x2,x2,1 ; x2++
002c bgeu x1,x2,00028
0030 j  00010
```

アセンブラをハードコーディング
→機械語へ
（確認用にgcc/gasも使用）

```
module imem(
   input wire clk,
   input wire [5:0] addr,
   output reg [31:0] rd_data
);
  wire [3:0]  iaddr = addr[5:2];
  always @(posedge clk) begin
    case (iaddr)
     // blink program
      16'h0000 : rd_data <= 32'h000000b7; // lui  x1,0x0
      16'h0001 : rd_data <= 32'h0ff08093; // addi x1,x1,0x0ff / x1=255
      16'h0002 : rd_data <= 32'h000011b7; // lui  x3,0x00001 / GPIO base address
      16'h0003 : rd_data <= 32'h00106213; // li   x4,1 / ori x0,x4,1
      16'h0004 : rd_data <= 32'h0041a023; // sw   x4,0(x3) ; GPIO=1
      16'h0005 : rd_data <= 32'h00000113; // mv   x2,x0 = addi x2,x0,0
      16'h0006 : rd_data <= 32'h00110113; // addi x2,x2,1 = x2++
      16'h0007 : rd_data <= 32'hfe20fee3; // bgeu x1,x2,00018
      16'h0008 : rd_data <= 32'h0001a023; // sw   x0,0(x3) ; GPIO=0
      16'h0009 : rd_data <= 32'h00000113; // mv   x2,x0 = addi x2,x0,0
      16'h000a : rd_data <= 32'h00110113; // addi x2,x2,1     ; x2++
      16'h000b : rd_data <= 32'hfe20fee3; // bgeu x1,x2,00028
      16'h000c : rd_data <= 32'hfe1ff06f; // j  00010
    endcase
  end
endmodule
```

VerilogHDLで命令ROMを記述

RISC-Vコア(poyo-v)の修正

- ☑そのままでは予定していたチップに入らない
  （CMOS 0.6um 3M / 1.8mm角）
- ☑poyo-vでコアで使わない機能を削減
  - ☑レジスタ数：３２個→５個
  - ☑レジスタ幅：３２bit→2bit
  - ☑命令デコーダ：使用する命令のみに

チップ製造への流れ

poyo-v + GPIO

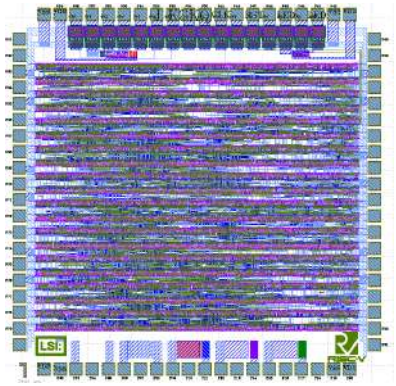
Qflow（論理合成・配置配線）

```
module imem(
  input wire clk,
  input wire [5:0] addr,
  output reg [31:0] rd_data
);
  wire [3:0]  iaddr = addr[5:2];
  always @(posedge clk) begin
    case (iaddr)
      // blink program
      16'h0000 : rd_data <= 32'h000000b7; // lui  x1,0x0
      16'h0001 : rd_data <= 32'h0ff08093; // addi x1,x1,0x0ff / x1=255
      16'h0002 : rd_data <= 32'h000011b7; // lui  x3,0x00001 / GPIO base address
      16'h0003 : rd_data <= 32'h00106213; // li   x4,1 / ori x0,x4,1
      16'h0004 : rd_data <= 32'h0041a023; // sw   x4,0(x3) ; GPIO=1
      16'h0005 : rd_data <= 32'h00000113; // mv   x2,x0 = addi x2,x0,0
      16'h0006 : rd_data <= 32'h00110113; // addi x2,x2,1 = x2++
      16'h0007 : rd_data <= 32'hfe20fee3; // bgeu x1,x2,00018
      16'h0008 : rd_data <= 32'h0001a023; // sw   x0,0(x3) ; GPIO=0
      16'h0009 : rd_data <= 32'h00000113; // mv   x2,x0 = addi x2,x0,0
      16'h000a : rd_data <= 32'h00110113; // addi x2,x2,1     ; x2++
      16'h000b : rd_data <= 32'hfe20fee3; // bgeu x1,x2,00028
      16'h000c : rd_data <= 32'hfe1ff06f; // j  00010
    endcase
  end
endmodule
```

```
0000 lui  x1,0x0
0004 addi x1,x1,0x0ff ; x1=255
0008 lui  x3,0x00001 ; GPIO base address
000c li   x4,1 / ori x0,x4,1
0010 sw   x4,0(x3) ; GPIO=1
0014 mv   x2,x0 ;  = addi x2,x0,0
0018 addi x2,x2,1 ; = x2++
001c bgeu x1,x2,00018
0020 sw   x0,0(x3) ; GPIO=0
0024 mv   x2,x0 ; = addi x2,x0,0
0028 addi x2,x2,1 ; x2++
002c bgeu x1,x2,00028
0030 j  00010
```

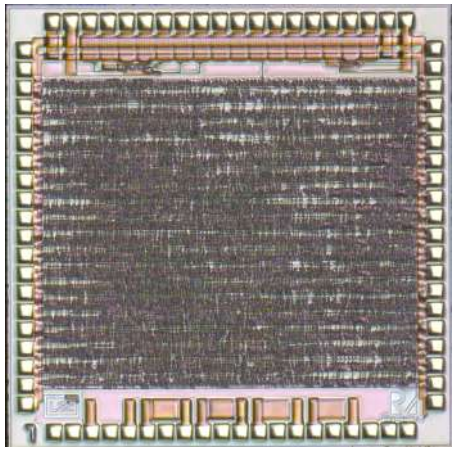
VerilogHDL

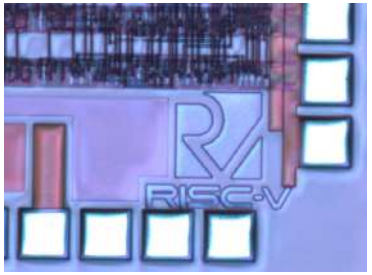
osu050
（NDAフリースタセル：一部修正）



レイアウトデータ（GDS）

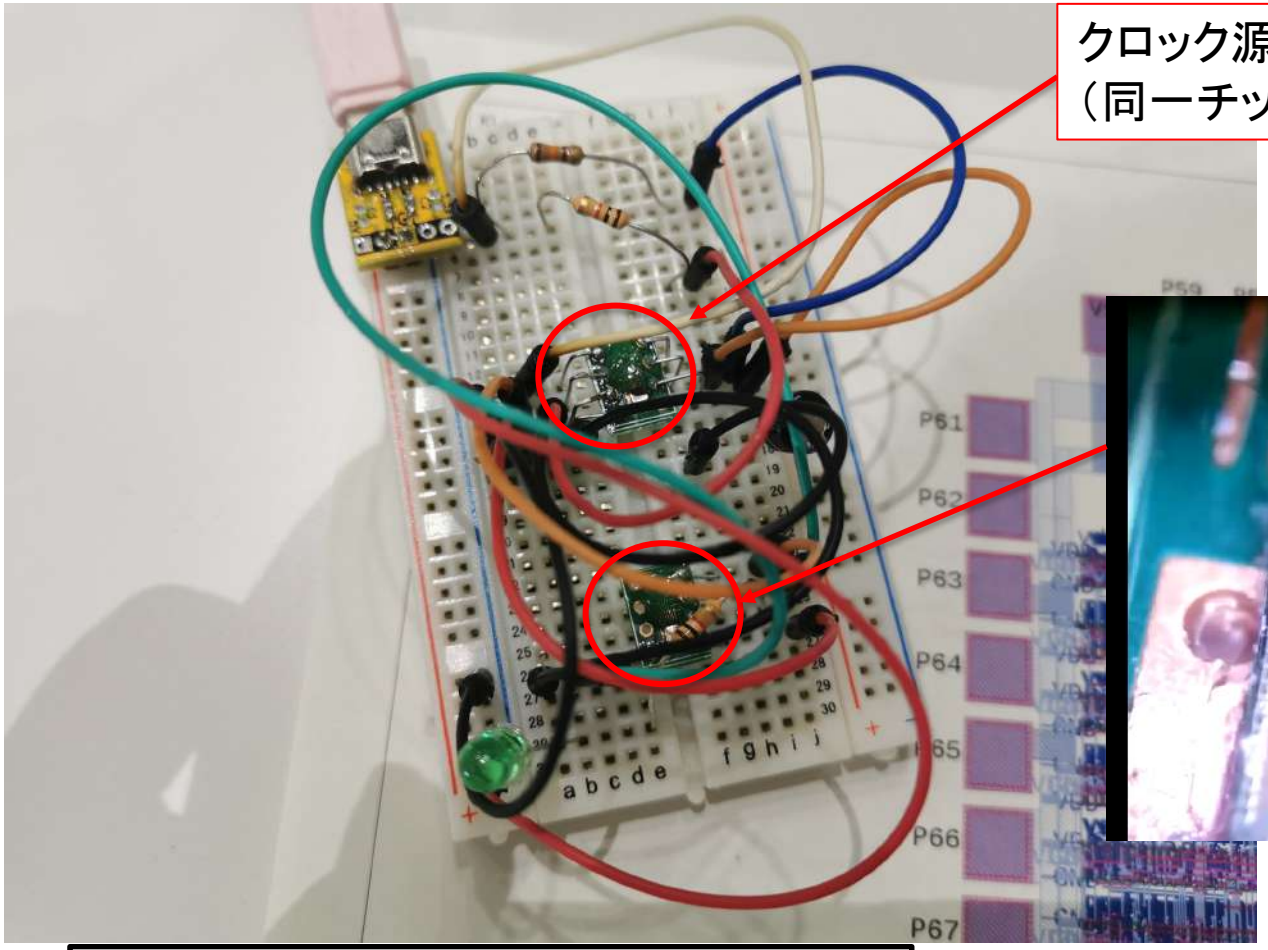
フェニテック社の
シャトル製造
（約20万円/10chip）



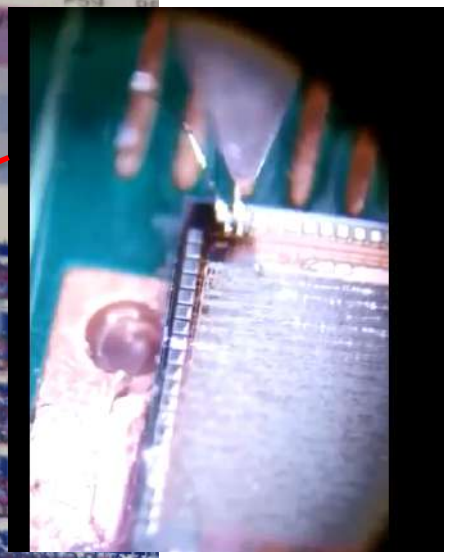


CMOS 0.6um 3Al 1.8mm(sq)
Core=1490um x 1240um

ボンディング→配線



クロック源の”555”(約10Hz)
（同一チップの余白に集積）

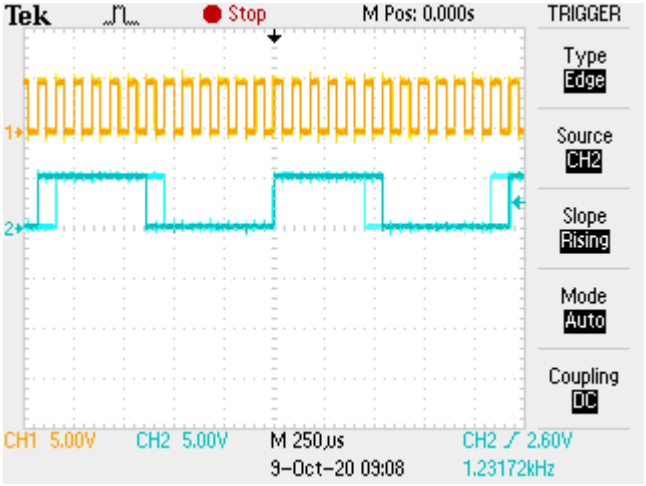


マニュアルボンダーで
ボンディング（Al線）

MakerFaireTokyo2020の１週間前に到着
→ボンディングして、ブレッドボードで

動作結果

- ☑一応、LEDは点滅している（約2Hz）
- ☑ただ、ちょっと動作があやしい
  - ☑周期が不規則にばらつく
  - ☑D-FFのセットアップ／ホールドタイムが足りない？
    ←製造プロセスとは別の論理セルライブラリを使用
    ＝遅延モデルが設計と実際で異なる？



まとめ

- ☑一応、RISC-Vをオープンソースな環境のみでシリコン化し、Lチカはできた
- ☑ただしちょっと動作が怪しい
  - ☑原因は論理セルラリブラリの不整合
    →これを調整して再挑戦したい
- ☑RISC-VのRV32Iでも、0.6umだとちょっとキツい
  →180nm程度のお手軽な製造プロセスがほしい