論理回路考古学のススメ

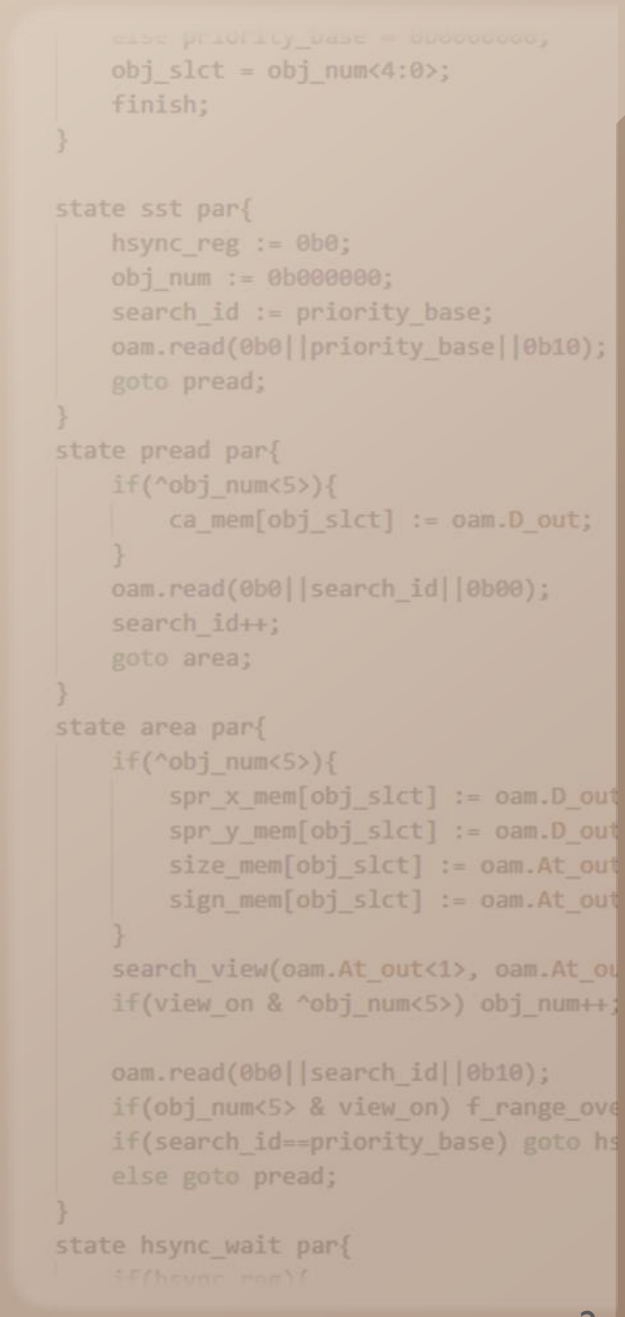
– FPGAで作る互換機の世界 –

by pgate1

初版 2018/12/08
改訂 2024/04/29

目次

- 自己紹介
- レトロハードウェア互換機とは
- 作ったモノ
- 開発環境の紹介
- 論理回路考古学としての事例
- 振り返り



こんにちは！

- かんな丸@pgate1
- 高専

  ロボコン（全国ベスト４）・プロコン・ソーラーボート等の活動
- 大学

  非同期ビットシリアル動的再構成アーキテクチャの研究
- 稀代のSFL使い
- https://pgate1.at-ninja.jp

レトロハードウェア互換機とは

- オリジナルハードウェアと差し替えてもソフトが動く
  - レトロハードウェアを維持し続けるのは難しい。
- Re:Creatorsによるクローンシステム

オリジナルと同様の
動作を実現する！

本歌と写し

本歌に相対するものは写しではない。つまり写しはそれ自身のアイデンティティを持っていて、本歌と同じ世界を共に生きている。本歌はそれ以上の成長はないが、写しは独自の成長を続けることができる。新しい道を行けるのは写しのみなわけだ。

(`･ω･´)

ただし、本歌（実機）を拡張して楽しんでいるケースも多く見かけるんです（レトロフィット）

論理回路考古学

- **Logic Circuit Architecture Archaeology**

互換機能を持つ論理回路を構築するために
見えるものから見えない部分を見出す
考古学的視点を取り入れる

- 実回路の推察により当時の設計思想を追体験できる。
- 仕様が公開されていないものを再現することに趣がある。
- 温新知故（FPGAを通じて古き回路を知る）
- アナログ部分は対象外
  - ビデオやサウンドなどに関するアナログ的完全再現は困難

ノスタルジック

ロストロジック

考古学的場面

- 長期保守製品に使用したカスタムチップやPLDの再現
- 動作レベル仕様書かネットリストしか残っていない
- コンパイル環境やデバイスに依存した記述
- 設計意図がワカラナイ
- 設計意図を掘り起こして回路を移植

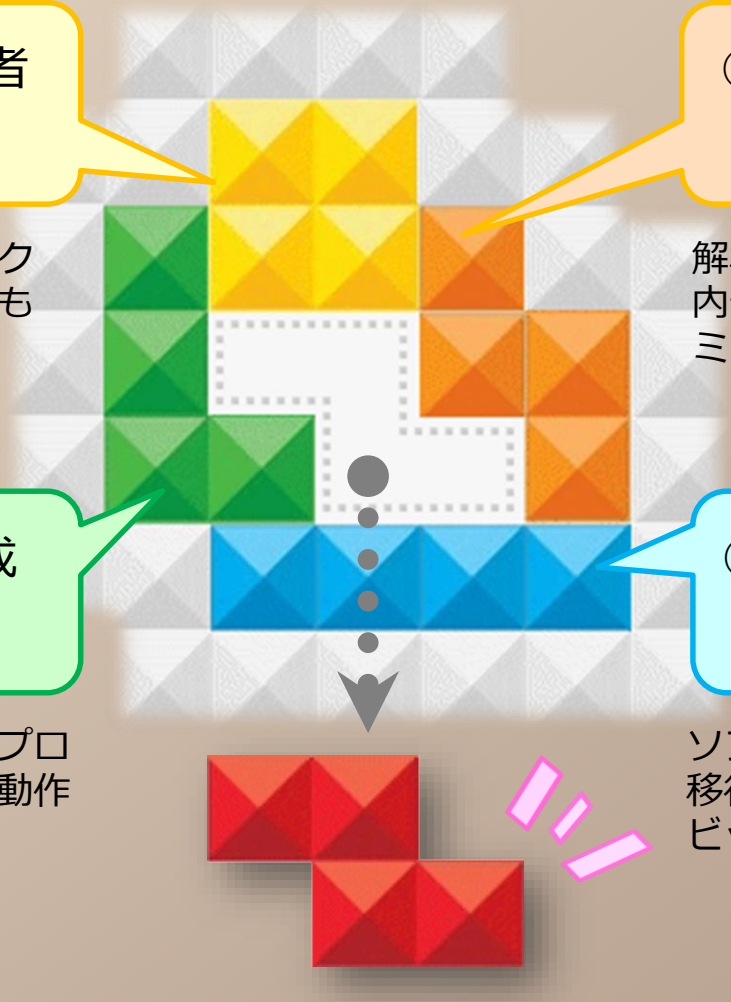
レトロハード互換機能の実装

①ソフト開発者向けの情報

よく知られている基本スペックや各レジスタの機能など、最も入手しやすい情報。

②エミュレータ開発者向けの情報

解析されたアーキテクチャや回路内部の動作などの情報。有志のコミュニティでまとめられている。



③エミュレータ作成による仕様確認

ソフトエミュを作成し実際のプログラムを実行して、各機能の動作を確認する。

④ハードウェア実装のアプローチ

ソフトエミュからハードエミュへの移行で確定する、演算や内部回路のビット幅や、動作タイミングなど。

実装すべき回路が見えてくる！

任天堂：岩田聡 氏

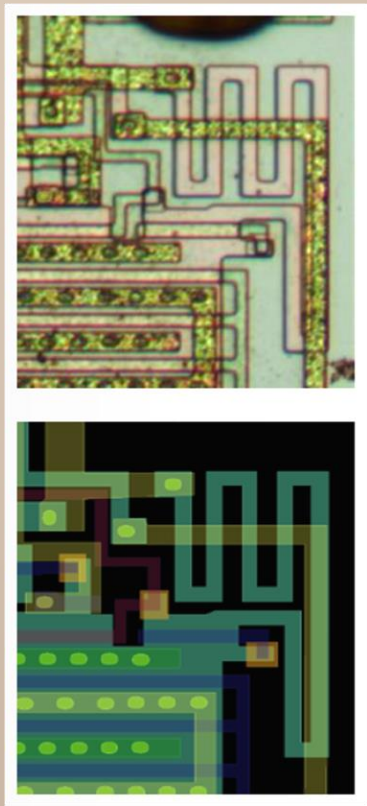
「ファミコンやスーパーファミコンの時代は、仕様書に書いていないハードの使い方を発掘して、開拓するのがプログラマーの歓びみたいなところがあって」

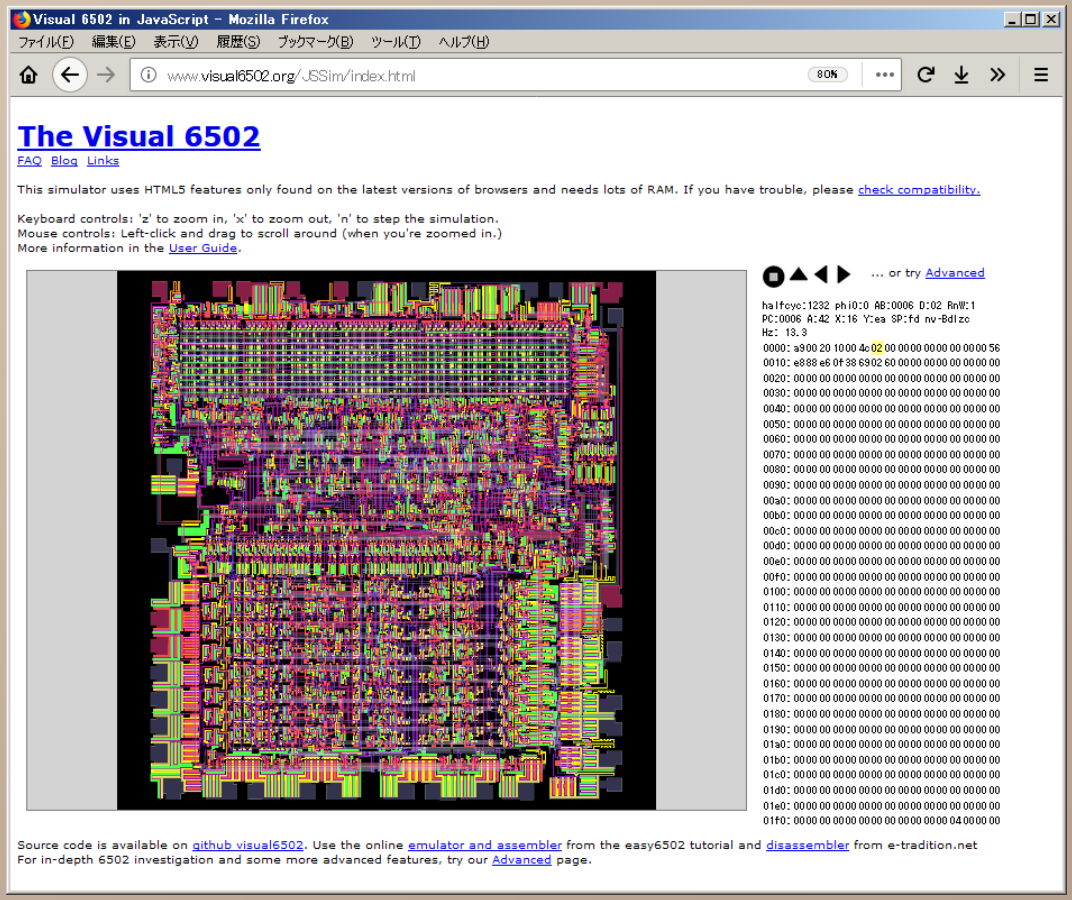
https://www.nintendo.co.jp/wii/interview/r2vj/vol1/index.html

非常に困る！
（私が）

こういう方法もあったり

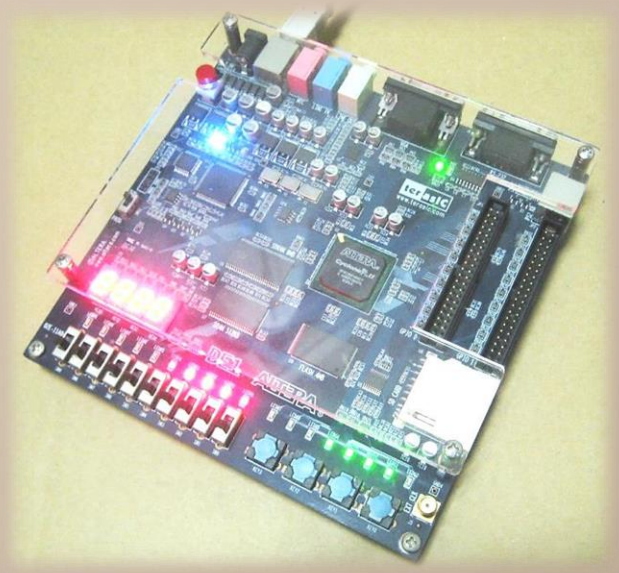
- 引用：www.visual6502.org
- ICモールド研磨 ⏵ レイアウトを取得 ⏵ グラフィカルに再現
- Transistor-level Simulation
  - ジグソーパズルにすると面白そう



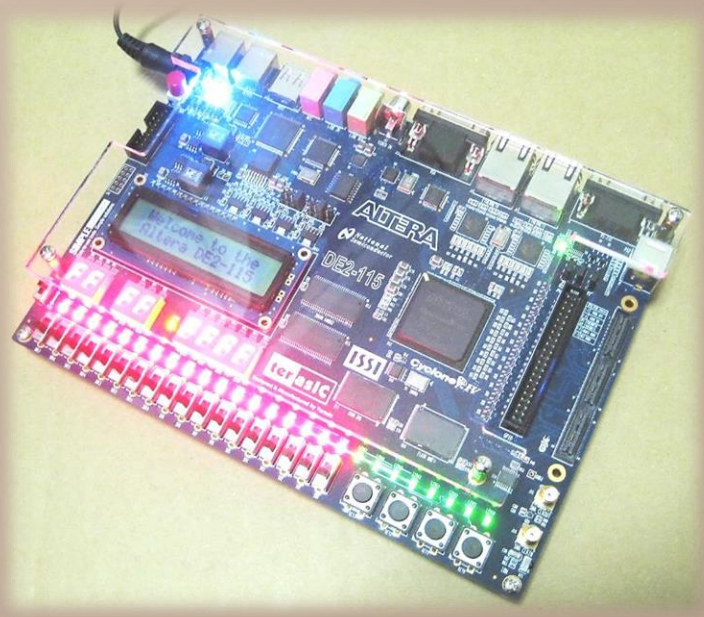


ゲーミングFPGAボード

- クローン機にちょうど良いFPGAボード
  - 映像出力、サウンド出力、SDカードスロット、外部メモリ
  - カラフルなLED、 7セグ、スイッチ、USB-JTAG
  - アクリル板で保護されている



Terasic DE1



Terasic DE2-115

作ったモノ

“Everything appears on the FPGA.”

え！FPGAでファミコンを！？

- 講義で習った論理回路の復習
- 当時まだ誰もやってなかった
- 遊んでいたものがどういう仕組みで動いているのか
- 世代問わず誰もが知っているゲーム機
- 新アーキテクチャの研究材料として
- 技術者教育用の教材として
- CPUとメモリ、グラフィック、サウンド等々が分かる
- SFLでどこまでの事ができるのか

など理由はいくらでも。

出来らぁっ！

裏コンセプト：個人でも作れる環境で

NES on FPGA（2004年）

- ファミコン互換機能をFPGAに実装
  - 当時としてはおそらく世界初。
- Spartan-IIE300評価キット ８万円 \ｶｶｲ!/
  - 約5万ゲートで実装
    - CPU(6502) 約1.2万ゲート
    - APU(サウンド) 約1.1万ゲート
    - PPU(グラフィック) 約1.4万ゲート
    - その他(SDRAM、PSパッド)
- 開発期間約２ヶ月
  - 工夫してロジック入れた。
  - 内蔵メモリもギリギリ。
  - ブロックを壊すとパッドが振動する

NES on FPGA の旅

- Electronic Design and Solution Fair(EDSF) 2005展示
- 大学での教育用FPGAボードへの活用
- Terasic社の各DEボードに移植
- sfl2vlとの出会い
  - 回路規模5分の3に削減
- 互換性の向上
  - 多種多様なゲームソフトサポート
  - NSF再生機能
    - ファミコンサウンドファンのコンペ曲を演奏できるように
- トランジスタ技術への寄稿
  - 「ファミコンサウンド回路で懐かしのゲーム音再生」
  - 韓国版トラ技「전자기술 （電子技術）」にも掲載
- 1chipMSXに移植

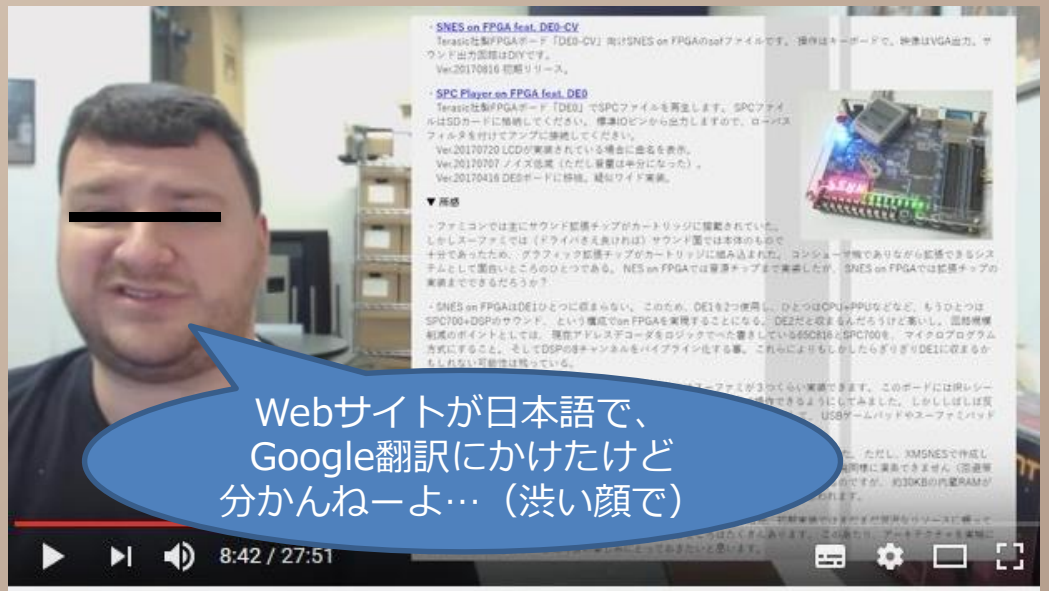
SNES on FPGA（2007年）

- スーファミ互換機能をFPGAに実装
  - FPGA「スゥ…スーパーファミコンです」
- 「ファミコンと似たようなもんでしょ」という流れで
  - そんなことはなかった…。
- 1,200本以上の検証用ソフト（当時総額約100万円）
- 初代：DE1ボード（2007年、$150）
  - 実機のサウンドモジュールを外付け。
- DE2-115ボード（2015年、$595）
  - 全実装が１チップに入った。
- DE0-CVボード（2017年、$150）
  - 全実装できる安価なボード。
  - **FPGAビットデータを公開してみた！**

海外勢の反応

- Readme_en.txtが全文海外掲示板に晒される
- 「ソースコードも公開しろよ」
- 「SFLって日本の高位言語？こいつHDL書けないの？」
- 「ヤツのSNESは本物かフェイクか賭けようぜ」
- 「互換機を販売しませんか？」
- 「ソースくれ（直球）」
- 「FPGAボード送ったからポーティングして」

みなさんオープンソースになることを期待していた模様



他のSNES FPGA互換機との比較

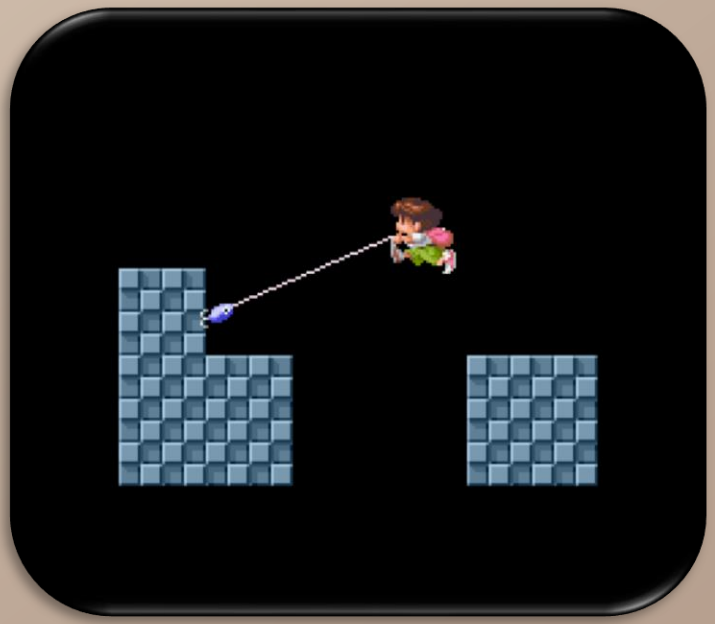
プロジェクトとして公開されているものをまとめた。いずれも複数のソフトに対して高い再現性を有している。

プロジェクト	実装者 (公開年)	行数 (記述言語)	回路規模	備考
VeriSNES	jwdonal (2016)	83k lines (VerilogHDL)	26k LEs	Closed code.
SuperNt	Kevin Horton (2018)	unknown	unknown, use CycloneV (49kLEs)	Closed code. $190, CFW exist.
FpgaSnes	Sergey Dvodnenko (2018)	17k lines (VHDL)	33k LEs	Open source. (join to the MiSTer project)
SNES on FPGA	pgate1 (2007)	13k lines (SFL+)	25k LEs	Open source. (but you can't synthesize)

おまけ（スーファミ実装NG集）

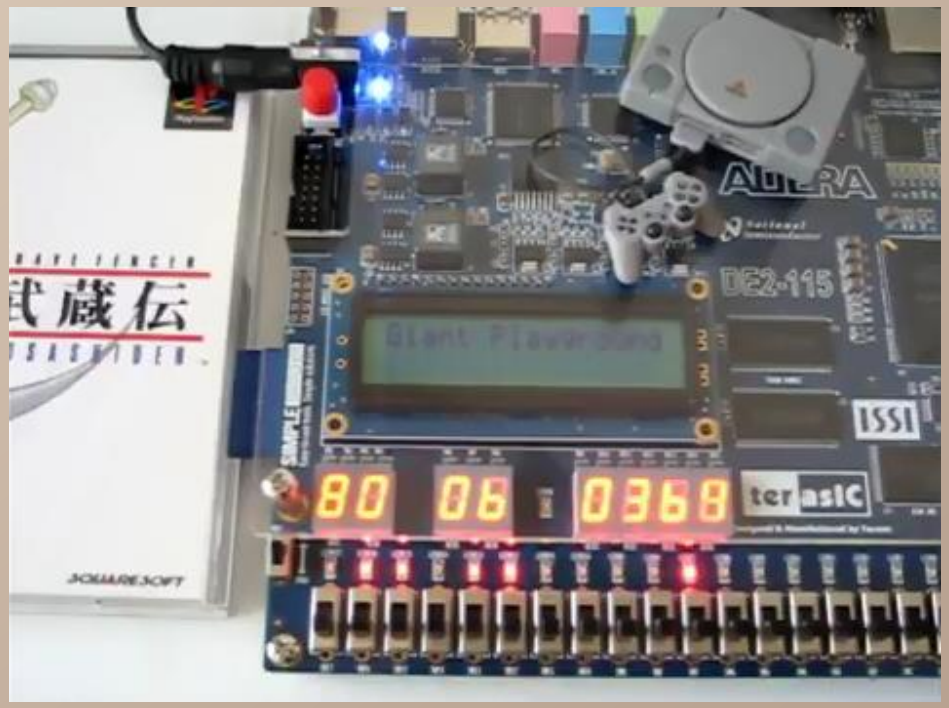
乗算器実装バグ youtube
描画プロセッサ側にある乗算器の実装ミス（符号拡張ミス）で挙動がおかしくなった。なおメインプログラムでは別の乗算器が使用されるためゲームはハングしない。

除算器実装バグ youtube
除算器の実装ミスで挙動がおかしくなった。ゲーム上ではありえない数値になるため、ゴム糸が切れるという演出が見られた。



PlayStation Sound Player on FPGA

- デモ的に作ってみた
- スーファミとの音源つながりで興味が出た
- CPU：R3000カスタム
  - 典型的なMIPSアーキテクチャ
  - 実際は5ステージだが、実装してみるとなぜか4ステージに



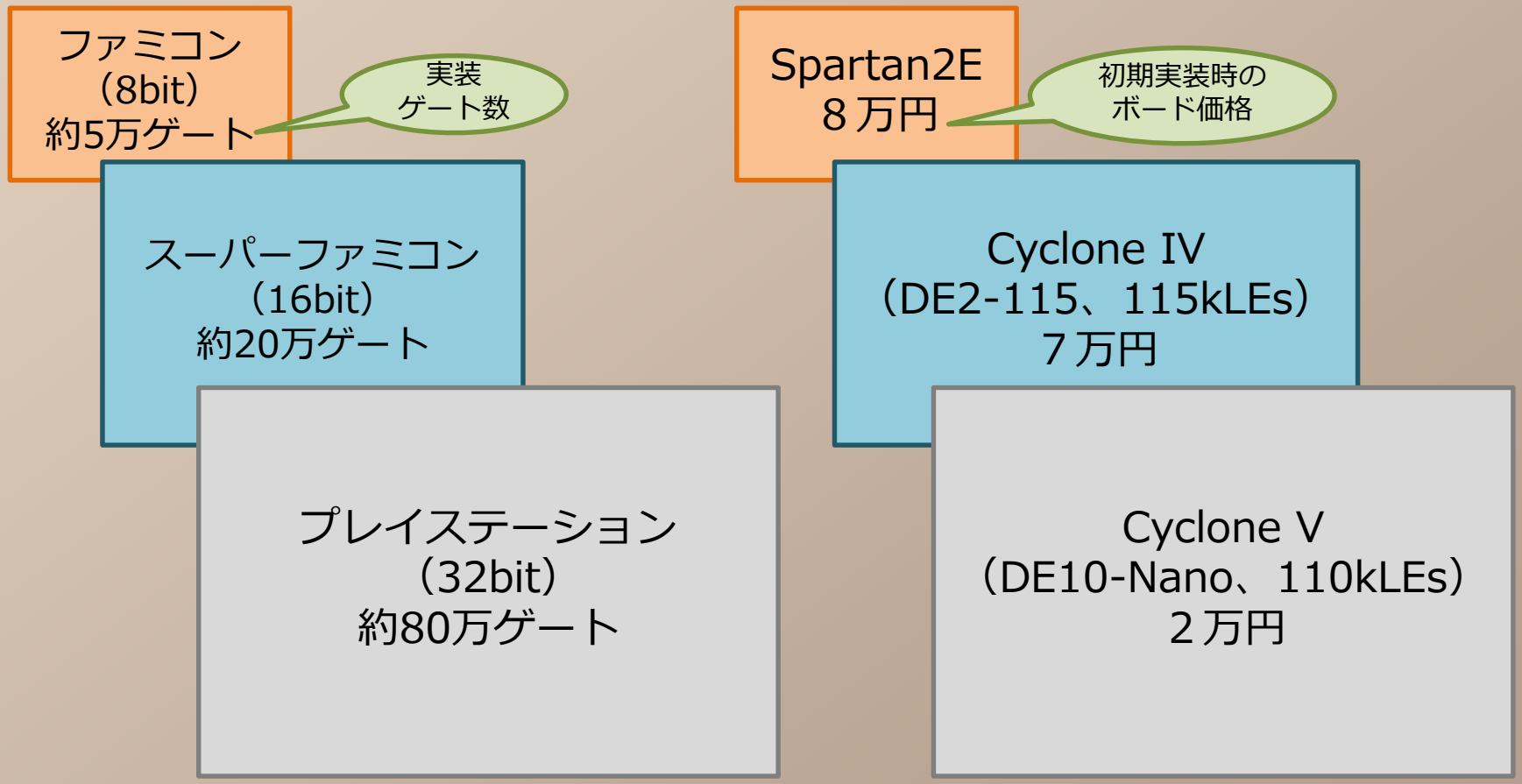
PlayStation on FPGA（WIP）

- GPUやDMAも実装してカーネルを実行できた
- いくつかのゲームが動き始めた！MotionJPEGデコーダも実装したい

Analogue Pocketに移植

回路規模的な比較

- 回路規模がbit幅の２乗に比例する感じ
- 実装に当たってはメモリの取り扱いをどうするかがキーになる



FPGA実装で苦労したこと

- NES on FPGA
  - CPUや描画プロセッサのメモリアクセスを仕様時間内に間に合わせろ！
  - フラグやレジスタのビットが共用されているので混乱するな！
- SNES on FPGA
  - CPUとサウンドプロセッサ間の通信タイミングを合わせろ！
  - 描画プロセッサの多彩な描画機能を実装しろ！
- PlayStation on FPGA
  - 描画プロセッサの描画速度を工夫しろ！（まだ遅い）
  - サウンドプロセッサの回路規模を削減しろ！（まだ大きい）
  - ムービーデコードやCD-ROMアクセスなどの仕様を理解しろ！
  - 各モジュールを適切なタイミングで動作させてソフトを動かせ！

実装対象のアーキテクチャそれぞれで
苦労する箇所が異なる

Q & A

Q．SNESの解析ドキュメントにはHWの詳細部分は書いてないのにどうやってFPGAに落とし込んだの？

A．まずSWエミュレータを作って仕様を理解します。SWなので試行錯誤を素早く繰り返しでき詳細仕様を見つけることができます。HWに落とし込むことを考慮したSWエミュ作りがコツです。

Q．一般的なプレステエミュレータってコード量が多いけど、君のPlayStation on FPGAはシンプルだね！どうして？

A．SWエミュレータは速度向上のため、早めに条件分岐しオプション毎に処理ルーチンがあるためコードが膨れ上がります。しかし実際やっていることはシンプルなので、HWに落とし込むとコード量は少なくなります。

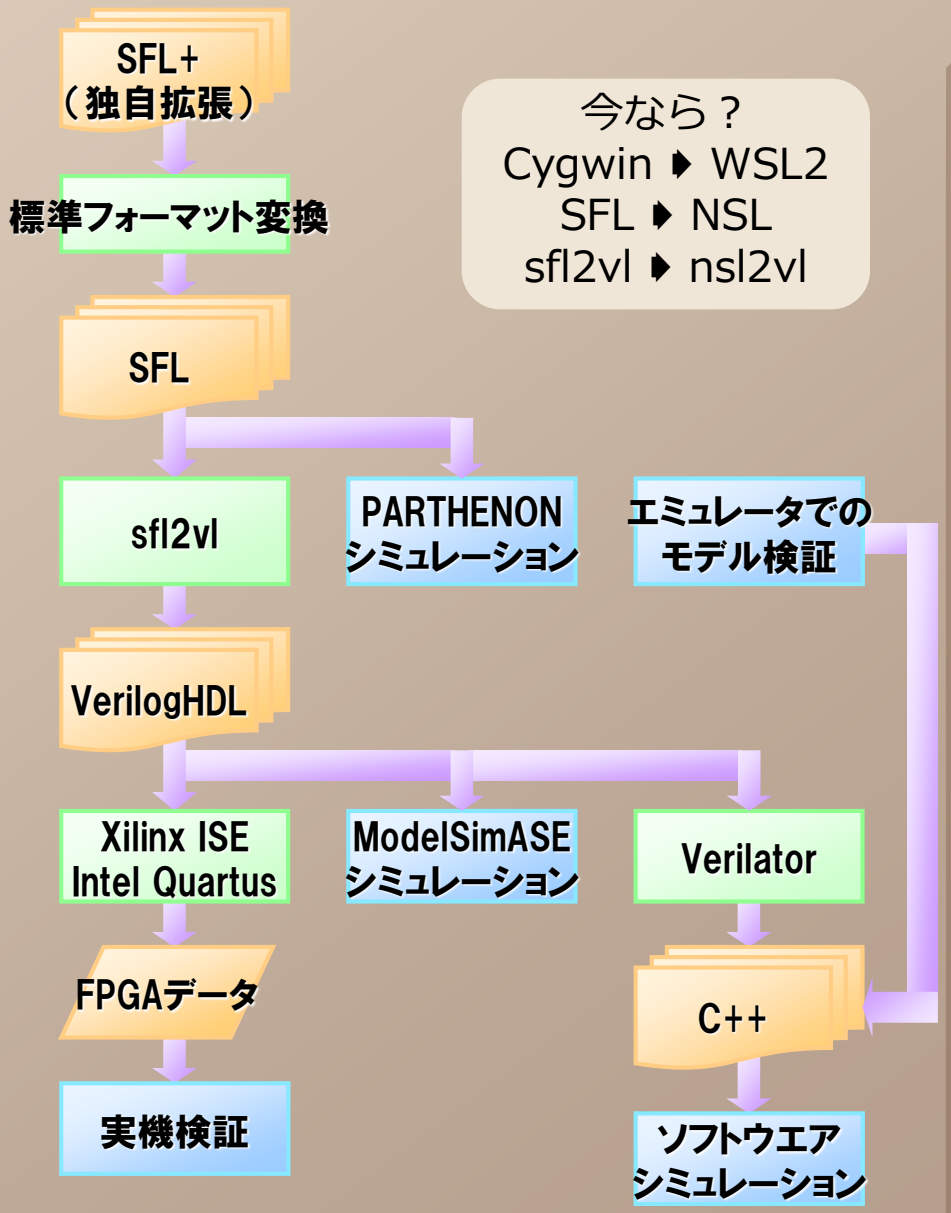
開発環境の紹介

“How they were developed.”

Myツールチェイン

- Cygwin
  - makeなどCUIベースで実施
- sfl2vl評価版
  - 2,000行制限あり
- Verilator
  - VerilogHDLからC++へ変換
- ISE無償版
  - Spartanシリーズの場合
- Quartus無償版
  - Cycloneシリーズの場合

FPGAボード以外にお金をかけない
初学者に沼と思わせない



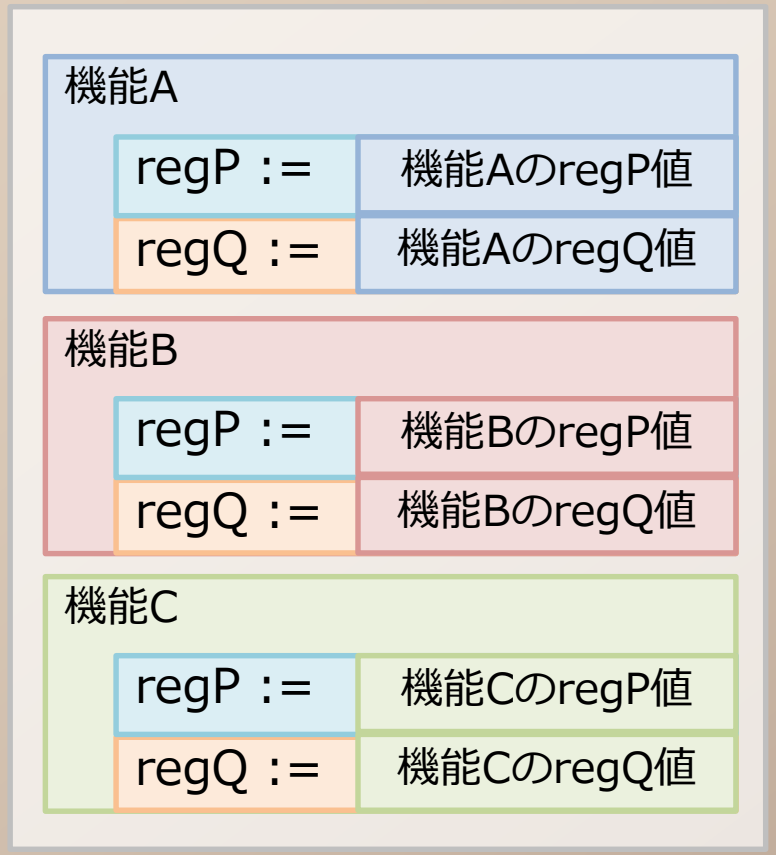
SFL

- 1980年代初頭にNTT研究所で開発
  - 日の丸HDL（DSL:Domain-Specific Language）
- PARTHENON処理系
  - SFLのシミュレーションや論理合成が可能。
- 同期式回路を前提とする
  - クロック記述が不要。
- SFL2Verilog、SFL2VHDL
  - メジャーなHDLに変換するプログラムも装備。

- 使った感想
  - メリット
    - RTLよりも仕様追加実装がしやすい。
    - 情報は少ないが構文がシンプルなので困らない。
  - デメリット
    - 処理系がオープンではないのでコード共有しづらい。
    - 乗除算は別途HDLで用意する必要がある。

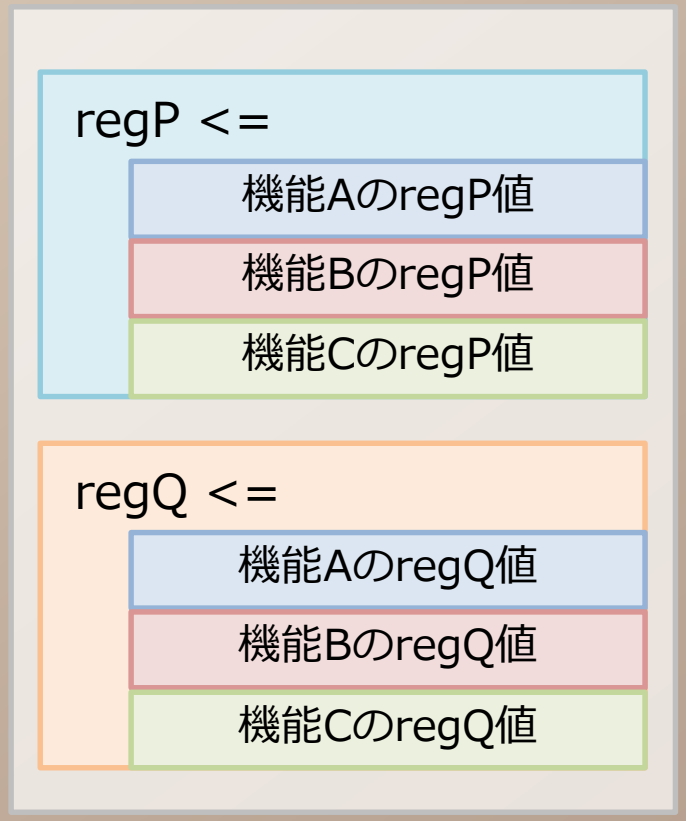
SFLの書きやすさとは

SFLの場合



機能ごとに記述でき
機能の追加削除がしやすい
⇒ 機能のデバグがしやすい

VerilogHDL/VHDL (RTL)の場合



機能ごとの記述が散らばって
しまいデバグしづらい

しかし知名度

VHDL

Verilog HDL

SFL

SFL+

SFLを個人的に使いやすくした。

- 引数宣言
  - instrin write(io_A, io_Din);
- 並列実行処理の記述（複数書かなくていいので楽）
  - par(i=0;i<32;i++){ sig[i]:=sig[i]<6:0>||0b0; }
- if-else（基本SFLにはelseが無い）
  - if(flag) ggg(); else fff();
- 文字定数、文字列定数
  - c:='A'; str="LCDﾆﾋｮｳｼﾞ";
- インクリメント、デクリメント（バリ使う）
  - a+=2; b-=2; c++; d--;
- ローカルスコープ信号
  - 局所的に使う信号をスコープ有りで宣言して使用できる。

NSL Core

- SFLの進化系
  - 今から始める人にはNSLを推奨。
- オーバートーン社によるサポート
- NSL/SFL を VerilogHDL/VHDL/SystemCに変換
  - nsl2vl、nsl2vh、nsl2sc、sfl2vl、sfl2vh
  - 変換されたVerilogHDLなどは基本構文のみの記述
    - ツールの種類やバージョンに依存しない。
- 可読性が良い
  - ステートマシンや繰り返しの記述が容易。

SFLでいろいろ実装

- SFLで記述するのが楽しくてFPGAを使っていると最近気づきました
- 楽しいことをするとIQが下がるらしい
  - 実装できない罠
- 作った機能
  - SDカードアクセス、FATファイルシステム、PSパッドアクセス
  - SDRAMコントローラ、USB HIDパッドホスト
- 一般的なHDLにありがちな「記述で惑わされる」ようなことが少ない
  - ストレスフリー

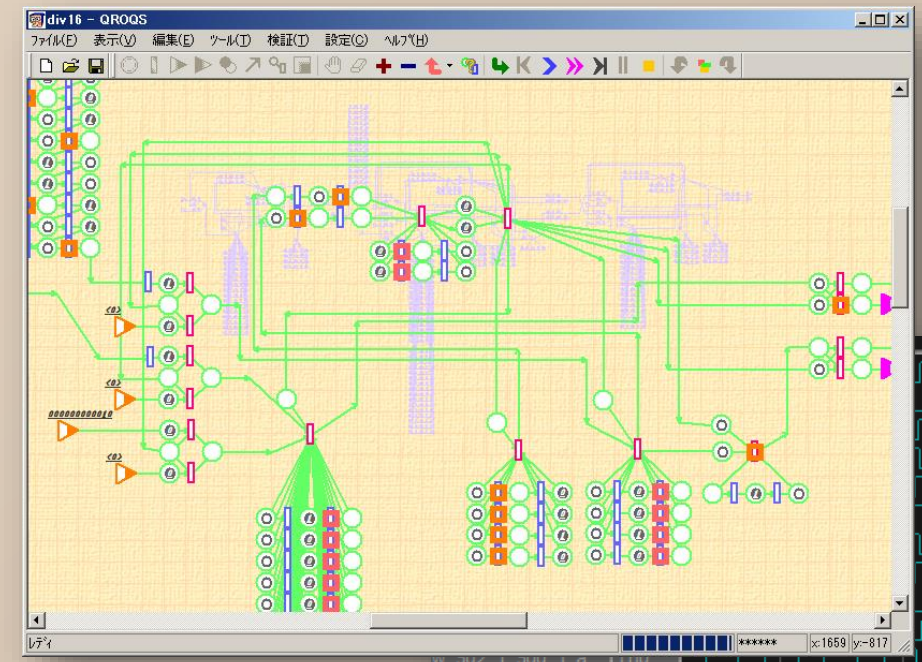
SFL/NSLを使えば…

ベンダフリー

バージョンフリー

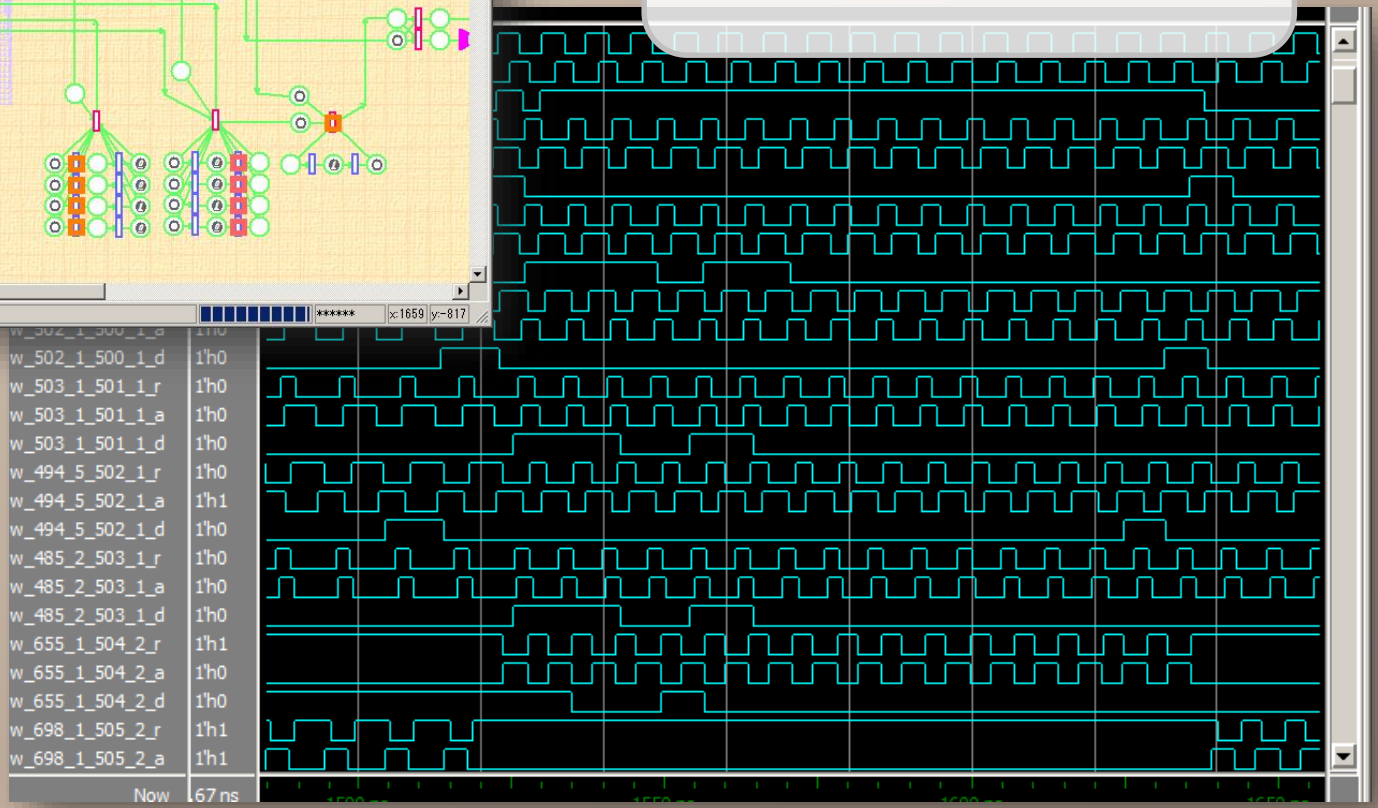
スタイルフリー

ところで、ビットシリアル非同期回路

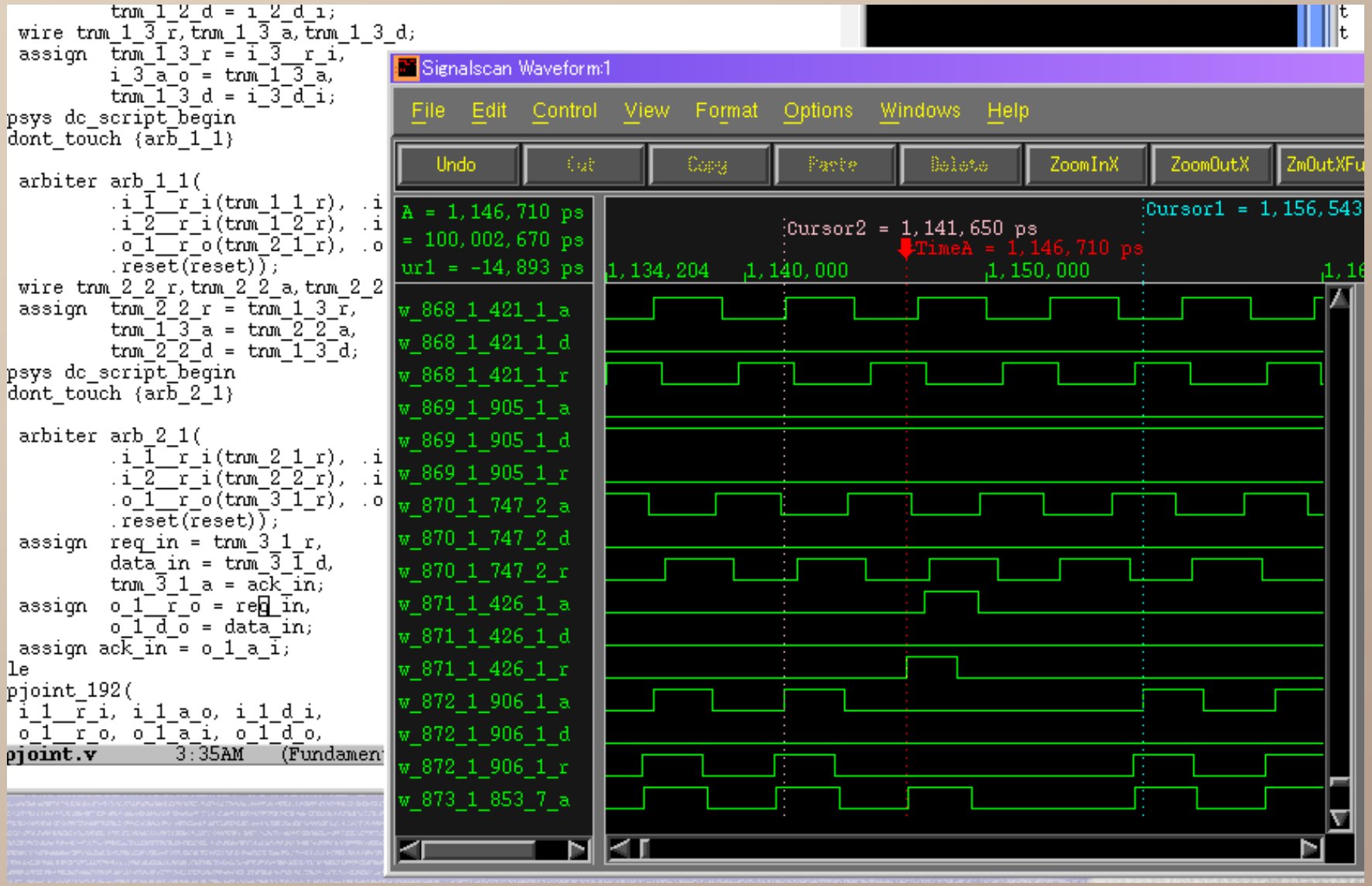


©QROQS

非同期式回路でも遅延
モデルを守って設計すれば
そんなに難しくはない

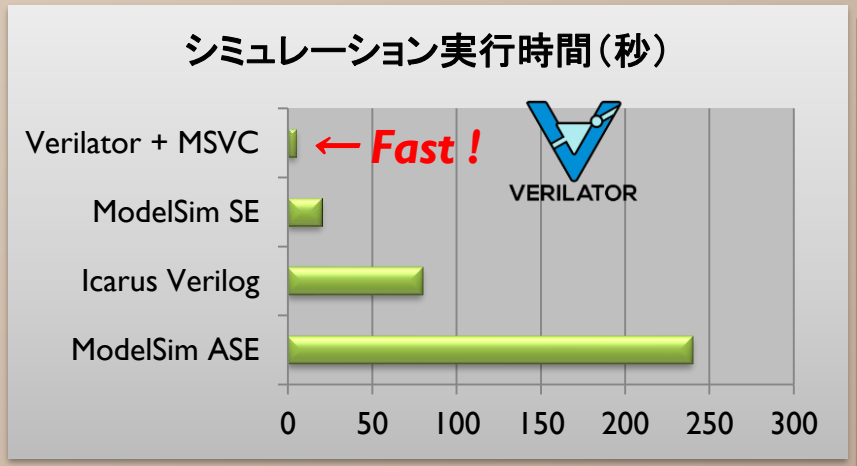


ただし波形デバグはもういやです



機能検証はさくっと

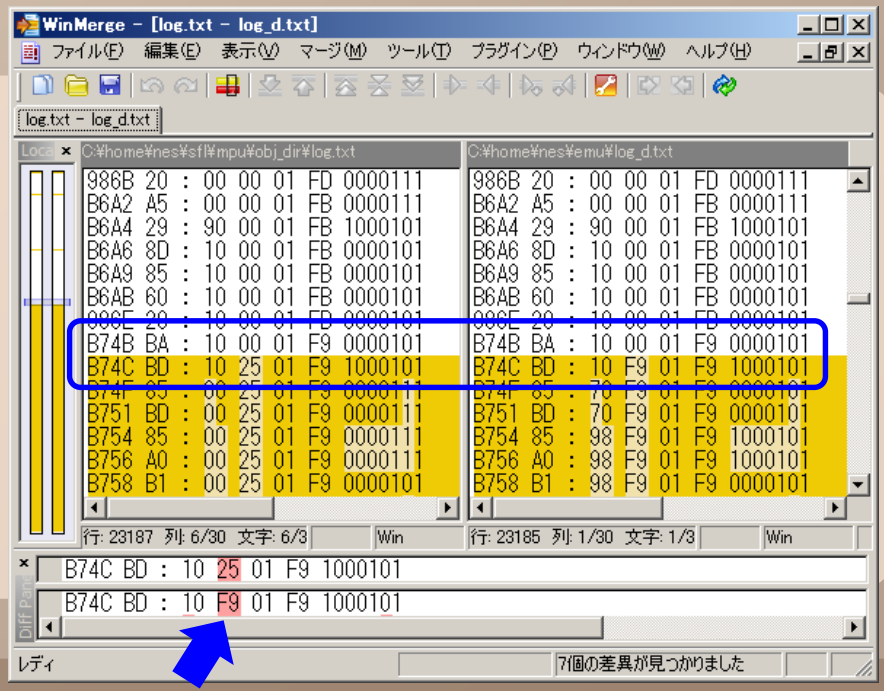
1. VerilatorでVerilogHDLをC++に変換
   - テストベンチもC++で書ける！
   - 変換時にリントチェックも！
2. exeにコンパイルして高速シミュ
   - ModelSimASEの約50倍！
3. テキストログを差分比較ソフトで確認
   - あいまい差分比較ができる！
4. さらに、変換したC++をリファレンスモデルへ組み込んで動的検証



テキストログ比較の例

現象：TSX命令(0xBA)にてXレジスタに入る値が異なっていた。

リファレンス記述　r_x = r_stack; ○
HDL記述　r_x = r_state; ×



正確な動作タイミングのために

- SFLは同期式設計が基本
  - ただしSDRAMやCODECなどのペリフェラルはPLLを使用。

- CPU、サウンド、グラフィックそれぞれ仕様に基づく動作速度がある
- それぞれでPLLのクロックを使うと非同期部分が発生
  - CDC（Clock Domain Crossing）は避けたい。

- そこで**同期カウンタで生成したサイクル**を使う！
  - 要するにカウンタで分周したイネーブル信号。

- DDFS（Direct Digital Frequency Synthesis）を活用
  - カウンタのビット幅が大きければ精度が向上し誤差が小さくなる。

ファミコン（グラフィック）
50MHzから5.369318MHzを生成
　加算値：　218125 (18bit)
　最大値：2031217 (21bit)
　誤差：0.000003Hz

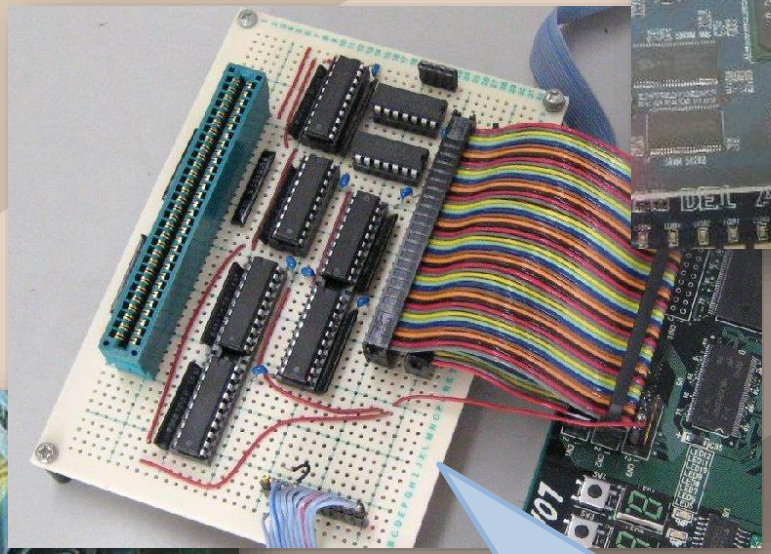
スーファミ（CPU、DMA）
50MHzから10.738635MHzを生成
　加算値：　250565 (18bit)
　最大値：1166652 (21bit)
　誤差：0.000017Hz

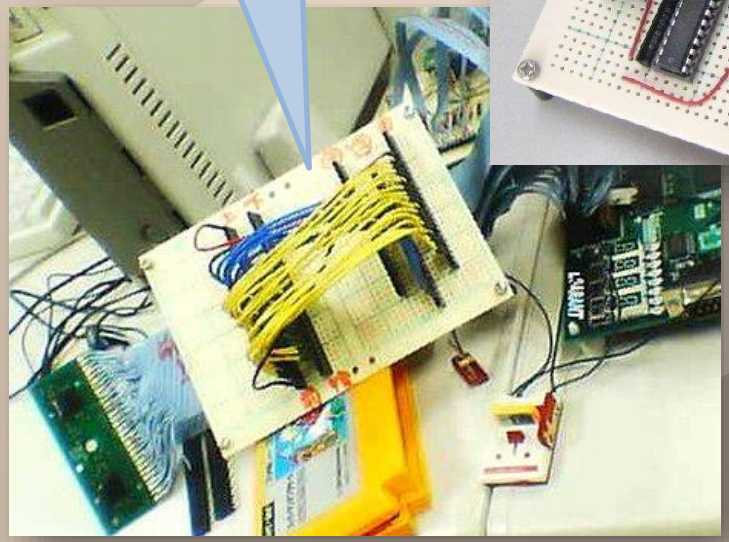
FPGAボードのベースクロック50MHzを使用。

ベース周波数とターゲット周波数の比から、
最適な加算値と最大値を見つける！
⇒ DDSカウンタ計算機公開しました！

ゲームソフトと接続

ソフトとのおしゃべりが
できるように進化！





初期
3.3V直結

後期
イサリヨクラリ氏
謹製のレベル
シフト基板

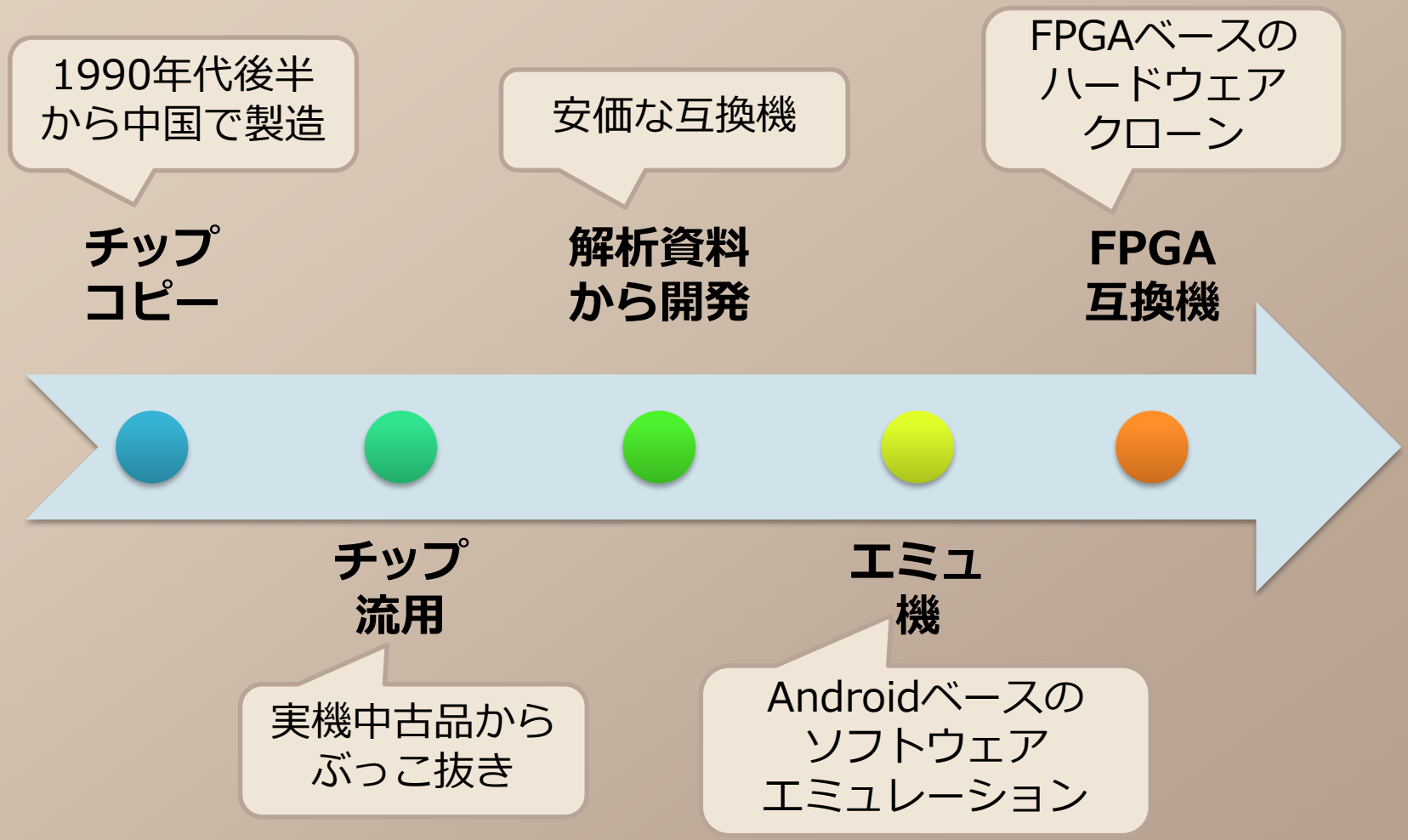
中期
3.3-5V変換で
複数ソフトをサポート

FPGAのIOは3.3V。ソフトはCMOSのものは3.3Vでも動作するが、セーブメモリからの出力は5Vのため、サポートするにはレベルシフトが必要。

論理回路考古学としての事例

“Approach in the Logic Circuit Architecture Archaeology.”

家庭用ゲーム互換機の系譜



FPGA互換機とソフトエミュレータと

- FPGA互換機とソフトエミュレータで、どちらが精度が高いとは言い切れず、結局は実装者スキルによる。
- FPGA互換機はプレイ時の遅延が少ないといったメリットはある。
- 「FPGAだから不具合あったらアップデートすればいいよね」は難しい。
  - 安易に修正すると、他のゲームで影響が出る場合がある。
- FPGA互換機なら実カートリッジが使えるので特殊チップをエミュレートする必要が無い。

それぞれの活動が
相互に利益をもたらす

コード公開は将来の可能性のため

それぞれの活動が
相互に利益をもたらす

- 「公開する意味あるの？」
  - コンピュータ資産をいかにして後世に残すか。
  - せっかく作っても死んだらなかったことと同じ。
- 「他の人がもう公開してるから…」
  - 複数の実装例を比較することができる。
- 「公開するとメンテが大変そう」
  - 有志が可能性を創出する。

互換機を作るタイミング

- 大まかなスペックが解析される
  ↓
- 有志によるエミュレータが公開される
  ↓
- 解析情報がまとめられる
  ↓
- エミュレータがオープンソース化される
  ↓
- 解析情報や解析ツールがサイトから消える

このあたりだと情報が不十分で苦労する

作りやすいタイミング

情報がサイトごと消えてしまうケースがある

互換機の実装指針

解析仕様からいきなり実装すると、その機能がどこで使われているか分からない。

↓

不具合があった時に、不具合要因を特定しづらい。

そこで…

1. ソフトを動かして**未実装の機能を見つける**。
2. 仕様を調べて**追加実装**。
3. 動作確認。

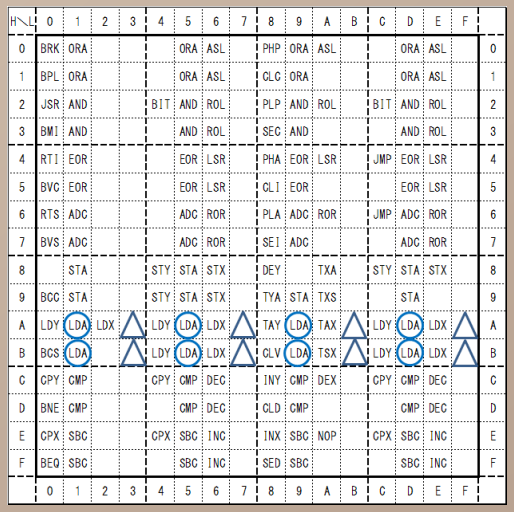
「その回路が何のためにあるのか」ということを明確にしながら実装することで不具合要因の切り分けがしやすい！

様々なデバグ手法

- リファレンスモデル（自作エミュレータ）で内部データの流れを確認。
- リファレンスモデルとハードウェア記述の**トレースログ比較**。
- リファレンスモデルの実行状態を再現。
- VerilatorでC++に変換してリファレンスモデルに組み込み検証。
- 「縮退実装」動いている状態から機能を削除していき動かなくなったところがデバグポイント。
- DMAデータストリームやメモリの内容の確認には**チェックサム**が役に立つ。

再現方法はどうするか

- CPU(6502)の多様なアドレッシングモード
  - 典型的なCISCプロセッサで、マイクロプログラム方式。
- マイクロプログラムをどう実装するか
  - テーブル or デコード回路
  - 内蔵メモリ or LUT(ワイヤードロジック)



H＼L	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0	BRK	ORA				ORA	ASL		PHP	ORA	ASL			ORA	ASL		0
1	BPL	ORA				ORA	ASL		CLC	ORA				ORA	ASL		1
2	JSR	AND			BIT	AND	ROL		PLP	AND	ROL		BIT	AND	ROL		2
3	BMI	AND				AND	ROL		SEC	AND				AND	ROL		3
4	RTI	EOR				EOR	LSR		PHA	EOR	LSR		JMP	EOR	LSR		4
5	BVC	EOR				EOR	LSR		CLI	EOR				EOR	LSR		5
6	RTS	ADC				ADC	ROR		PLA	ADC	ROR		JMP	ADC	ROR		6
7	BVS	ADC				ADC	ROR		SEI	ADC				ADC	ROR		7
8		STA			STY	STA	STX		DEY		TXA		STY	STA	STX		8
9	BCC	STA			STY	STA	STX		TYA	STA	TXS			STA			9
A	LDY	LDA	LDX	△	LDY	LDA	LDX	△	TAY	LDA	TAX	△	LDY	LDA	LDX	△	A
B	BCS	LDA		△	LDY	LDA	LDX	△	CLV	LDA	TSX	△	LDY	LDA	LDX	△	B
C	CPY	CMP			CPY	CMP	DEC		INY	CMP	DEX		CPY	CMP	DEC		C
D	BNE	CMP				CMP	DEC		CLD	CMP				CMP	DEC		D
E	CPX	SBC			CPX	SBC	INC		INX	SBC	NOP		CPX	SBC	INC		E
F	BEQ	SBC				SBC	INC		SED	SBC				SBC	INC		F
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	

- 初期FPGAで内蔵メモリが足りなかった
  - デコード回路をLUTで実装。
- 巨大なデコード回路
  - 手作業で最適化して回路規模を可能な限り削減↑
- 今のFPGAならどんな実装でもOK

段階によって実装方法を変える

可読性か回路規模か

例：スーファミのサウンド波形デコード回路をDE1に実装

- 乗算器を使用（可読性重視）
  - 乗算器使用率100%になり他の機能が実装できない！
- シフトと加減算を駆使して実装
  - 乗算器を使わずデコード回路が実装できた。

→ 乗算器よりも低コストで実装できるのでは？

Microchip ProASIC3(乗算器無し)、SynplifyProMEで計測

- 乗算器記述：780 cells
- シフトと加減算：650 cells ←こっちの方が回路規模小さい

実機はシフトと加減算でデコード
回路を実装しているのか？

再現度を上げるとうれしいこと

- 例：スーファミのサウンド用プロセッサ(SPC700)
  - CMPW(2バイト比較)は4クロック
    - これは16ビット演算１回でよさそう
  - ADDW(2バイト加算)とSUBW(2バイト減算)は5クロック
    - もしかして8ビット演算を**２回行っている**？

演算器の数まで再現
vs.
影響なければ動けばいい

実は、再現度を上げれば
設計意図が見えてくる可能性がある

機能モジュール単位での再現

- 例：スーファミのチップ構成
  - 初期型
    - CPUチップ、グラフィックチップ、サウンドチップが別。
  - 後期型
    - CPU、グラフィックが１チップに統合。

DMA機能は
どのチップに
あるんだろう？

- 後期型でも内部モジュール構成は機能ごとと思われる。
- わざわざ初期型の構成まで再現する必要性はないが、機能別になっていると作りやすい。
- ただし内部データバスは入出力をまとめるより、入力と出力で分けた方が作りやすい。

実チップのピン機能まで再現するかどうかによる

たまには実装と仕様の照合を

- プレイ途中に強制リセットされる不具合
  - 途中の不具合再現は難しい
- 原因は割り込み時のDフラグクリア実装ミス
  - CPU(65C816)のDフラグ(BCD演算有効フラグ)

聖剣伝説３で
不具合が起きた

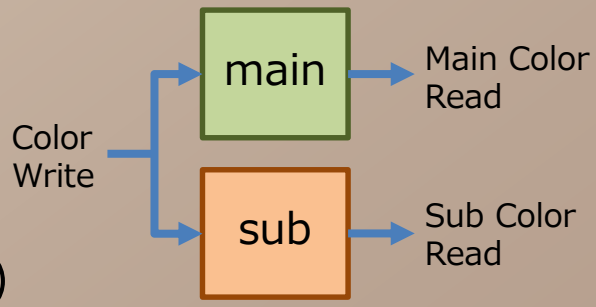
- なぜ実装ミス発見が遅れたか
  - 前世代CPUをベースにして作ったため。
    - 6502ではDフラグクリアしていない。
  - 多数のソフトで問題なかった。
  - タイミング関連の再現性不足と思い込んでいた。
    - FPGAのタイミングをエミュで再現するのに時間がかかった。
  - 解析資料の確認不足。
    - これが一番大きい。
  - Dフラグがクリアされる理由が分からなかった。

描画性能を上げるには

- スーファミの描画プロセッサ特殊モードの実装
  - 横ピクセル倍モード
  - メインカラーとサブカラー合成モード

- カラーメモリを倍速で動かす必要がある？
  - 描画プロセッサクロック：約5.3MHz
  - 出力ピクセルクロック：約12.1MHz

- 倍速で動かすにはどうすればいいか
  - 描画プロセッサクロックを倍にする
  - メモリデータ幅を増やす（デュアルポート）
  - カラーメモリ多重化（ダブルメモリ）　← 今の所これを採用

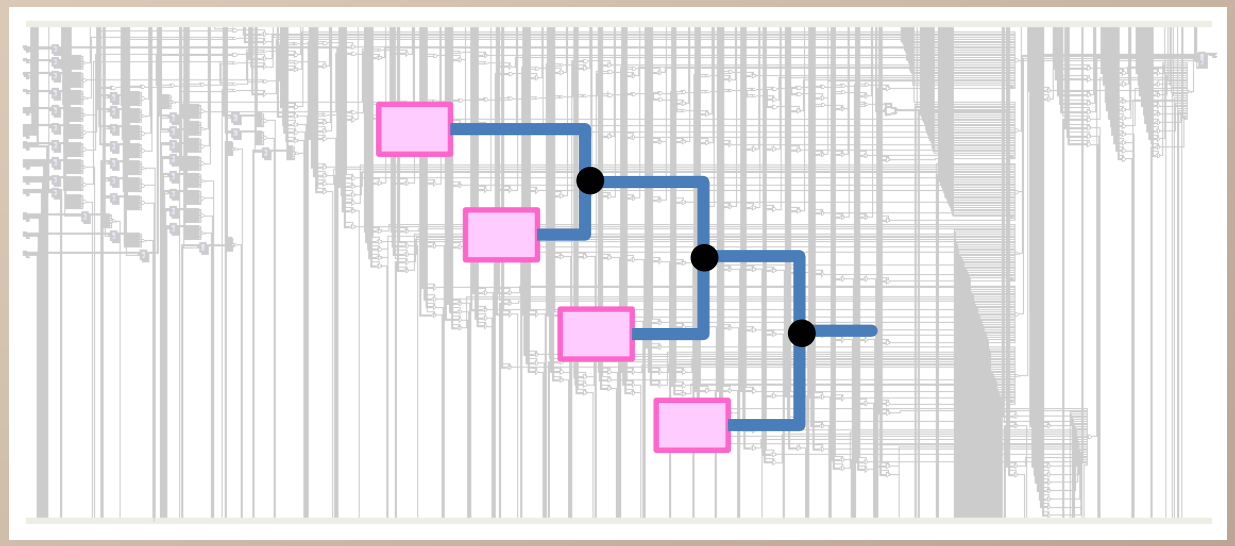


- 実際はどうなんだろう

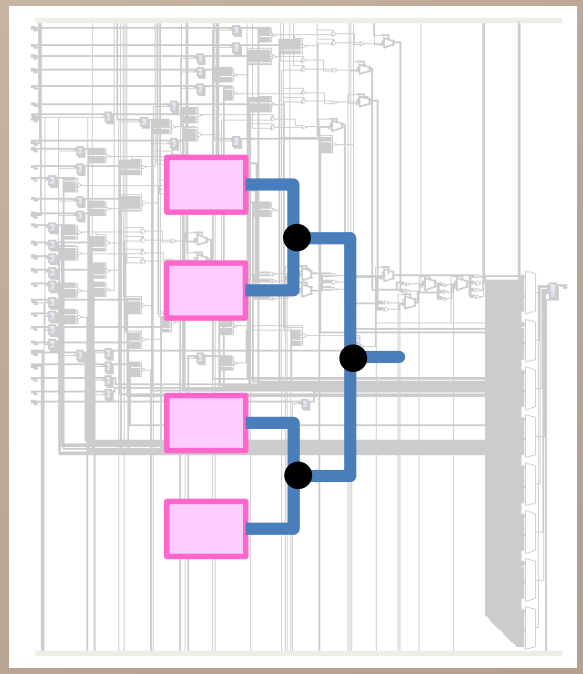
優先順位回路は深さ最小に

例：スーファミのキャラクタ描画で、34個のうちインデックスが小さい方を優先的に描画する回路を作る。

数珠つなぎカスケード接続だと**遅延が大きい**。（合成ツールによってはある程度遅延最小になるよう頑張ってくれるが）

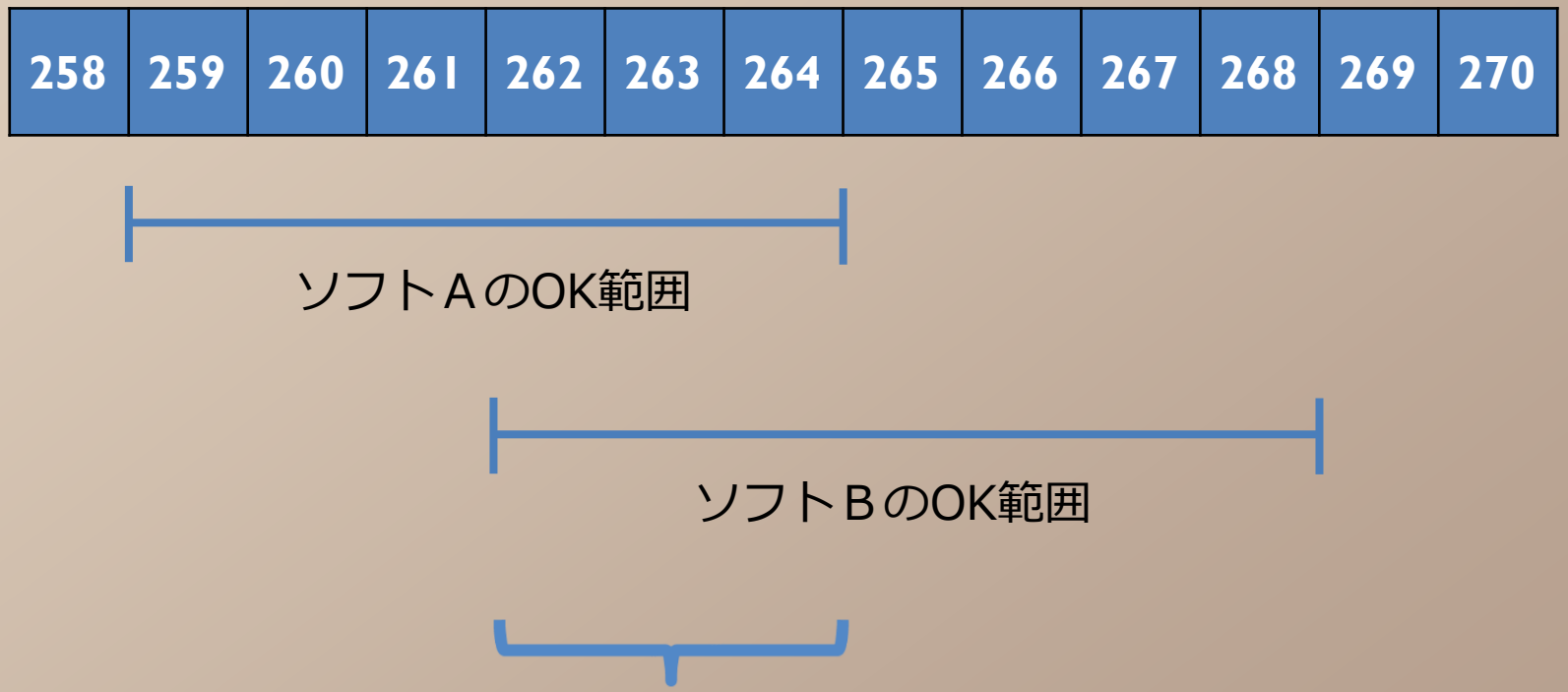


深さ最小カスケード接続で作れば、**確実に遅延最小**になる。



不明なパラメータの絞り込み

パラメータが仕様から判別できない場合、複数のソフトの動作範囲から絞りこむ。



世界のFPGA互換機

- リクリエイトされたもの
  - ファミコン互換：Nt mini（Analogue）とか8bitGoogleMapの人とか
  - スーファミ互換：SuperNt（Analogue）
  - メガドライブ互換： MegaSg（Analogue）
  - ゲームボーイ・ゲームボーイカラー互換：えだるふ 氏
  - ゲームボーイ・ゲームボーイアドバンス互換：Pocket（Analogue）
  - 初代PlayStation互換、NINTENDO64互換：FPGAzumSpass 氏
  - PCエンジン互換（K.i 氏）、Duo（Analogue）
  - 1 chip MSX：（似非職人工房）、（D4Enterprise）
  - 偽X1プロジェクト（すーぱーたーぼ 氏）
  - MZ-700をつくる（Oh！石 氏）
  - FPGAでPC-8001を作る計画（Yasuo.Kuwahara 氏）
  - FPGA版X68k（プー 氏）
  - MiSTerプロジェクト：様々なレトロ機互換機能を提供
  - openFPGAプロジェクト：Analogue OSをベースにしたコンソール互換

- あなたも論理回路考古学の世界へ！

振り返り

“Retrospective of the nostalgic.”

FPGAでファミコンを作ると

- CPUとかグラフィックとかサウンドとかﾁｮｯﾄ分かる
- 面接のときにアピールできて受かる
- エンベデッドスペシャリストが取れる
- SFLだけでなく他のHDLにも詳しくなれる
- 技術専門誌に寄稿したり論文を出せたりする
- なにかと説明や発表する機会が増える
- FPGAボードを寄贈してもらえる
- ソースコードを売ってほしいとオファーが来る
- いろんな人が挑戦するのを観測出来て楽しい
- 色々な周辺デバイスを使いたくなる
- 互換性向上しがいがあって飽きない
- 他の互換機も作りたくなる

プロジェクトの勝因

- 飽きずに試行錯誤できた
- 自作エミュレータをリファレンスモデルとして活用した
- コアを完全同期設計し複雑に考えずに済んだ
- SFLが仕様変更（追加実装手法）に有効だった
- 移植性が高くなるように実装できた
- FPGAボードが安価になった

VerilogHDLやVHDLだと使いこなすのに手間取っていたかも

反省点

- 作ったらとっとと動いているところを公開する
  - 進捗報告したら次の実装に取り掛かる。
- オレオレFrameworkに凝り出さない
- 互換性向上に凝り過ぎるのもどうなのか
  - 20%→70%は楽だが、70%→90%は時間がかかる。
- 効率的な検証方法を考えないとキツい
- 可読性重視で記述したら回路規模が少し大きめに
  - 合成系システムがあまり進化しなかった。
- オープンソースにした世界線も見てみたかったかも

SNESのFPGA実装について、のちにSFL+版のコードを公開してみたものの、他の人は環境が無く合成することができず直接的な発展性は得られなかった。
対してsrg320氏のVHDL版はMiSTerプロジェクトにマージされ、多くの協力のもと互換性向上や特殊チップ対応など図られている。

今後の展開

- 回路規模削減や動作タイミング合わせ
  - ちゃんと設計を見直す
    - 並列実装している機能をパイプライン化
    - メモリアクセスの最適化
- 作りかけPlayStation on FPGA
- ローグライクゲームを自動プレイするAI
- ゲームカートリッジを使うガジェット的な何か
- なにか共有できるものを共有する
- 論理回路考古学として寄与する

まとめ

- FPGAで伝説の回路を再現することは考古学に通じていて、パズルが解けた時のような楽しさがある！
- ただし、設計意図を読み解くのは難しいことを実感したのでコメントや判断材料などは積極的に残そう。
- SFLでいろいろ作れるものですね！
  - 清水先生謹製のツールがすごい。
  - 他に記述しやすいDSLでも試したい。

謝辞

- 数多の解析情報を公開されている有識者の皆様へ
- FPGAとSFLを教えてくださった小栗清先生へ
- sfl2vlを開発された清水尚彦先生へ
- 各プロジェクトを応援して頂いた皆様へ

ありがとうございました

“Have a nice life of FPGA.”