

先進計算機構成論 03

東京大学大学院 情報理工学系研究科 創造情報学専攻

塩谷 亮太

shioya@ci.i.u-tokyo.ac.jp

今日の内容

- 回路の消費電力
 1. クロックの消費電力
 2. アーキテクチャの違いによる消費電力の違い
 3. FPGA による回路
- (余談) ムーアの法則と周波数

消費電力について

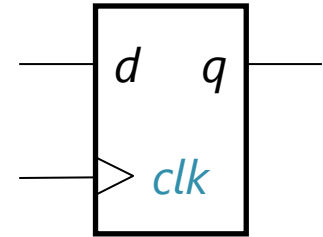
CPU やその他回路の消費電力について

- 回路の消費電力について
 1. クロックの消費電力
 2. アーキテクチャの違いによる消費電力の違い
 3. FPGA による回路

クロックによる消費電力

■ クロック信号：

- ◇ 記憶素子の更新タイミングを制御



■ 具体的な D-FF の動作：

- ◇ クロックの立ち上がりのたびに, d の値がサンプリング
- ◇ その値が次のサイクルの間 q から出力される

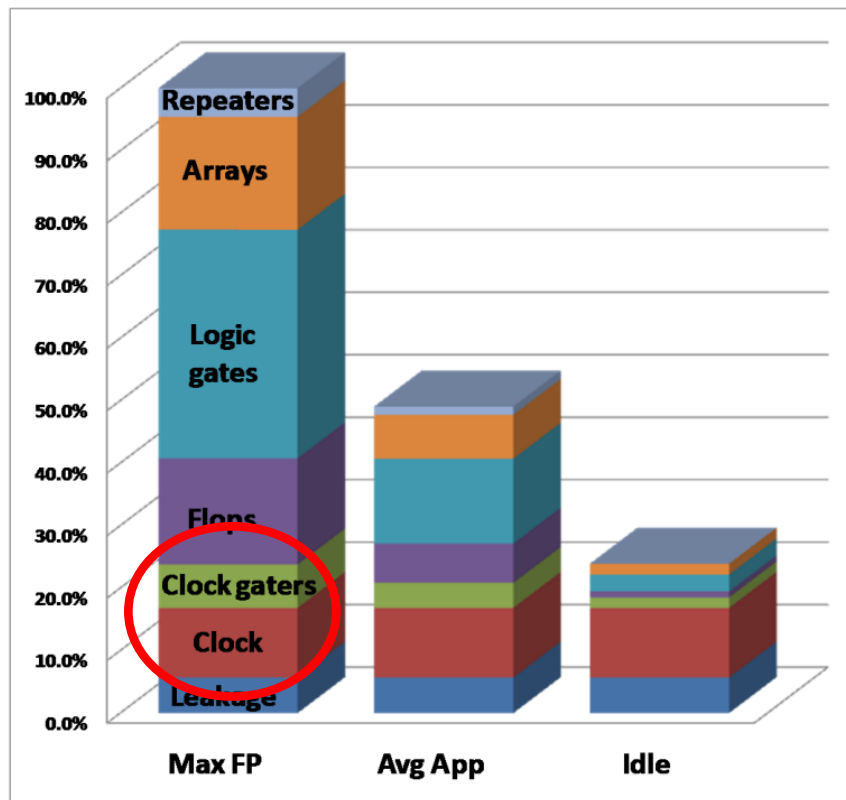
■ クロックによって消費される電力は非常に大きい

- ◇ CPU 全体で消費される電力の数割におよぶこともある
- ◇ なぜただ同期をとるためだけに, それほど電力が食われるのか？

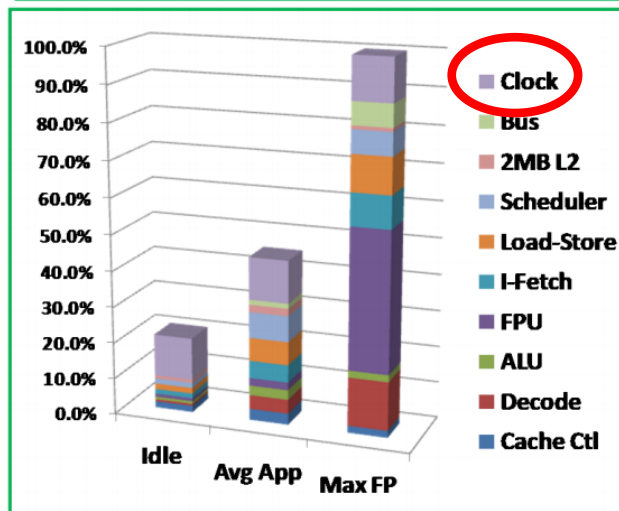
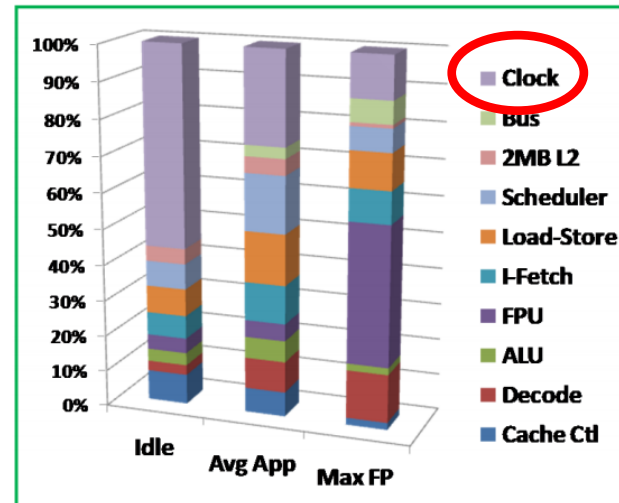
AMD Steamroller の消費電力のうちわけ

実際にクロックが大きな割合を占めることがわかる

Active Power Breakdown



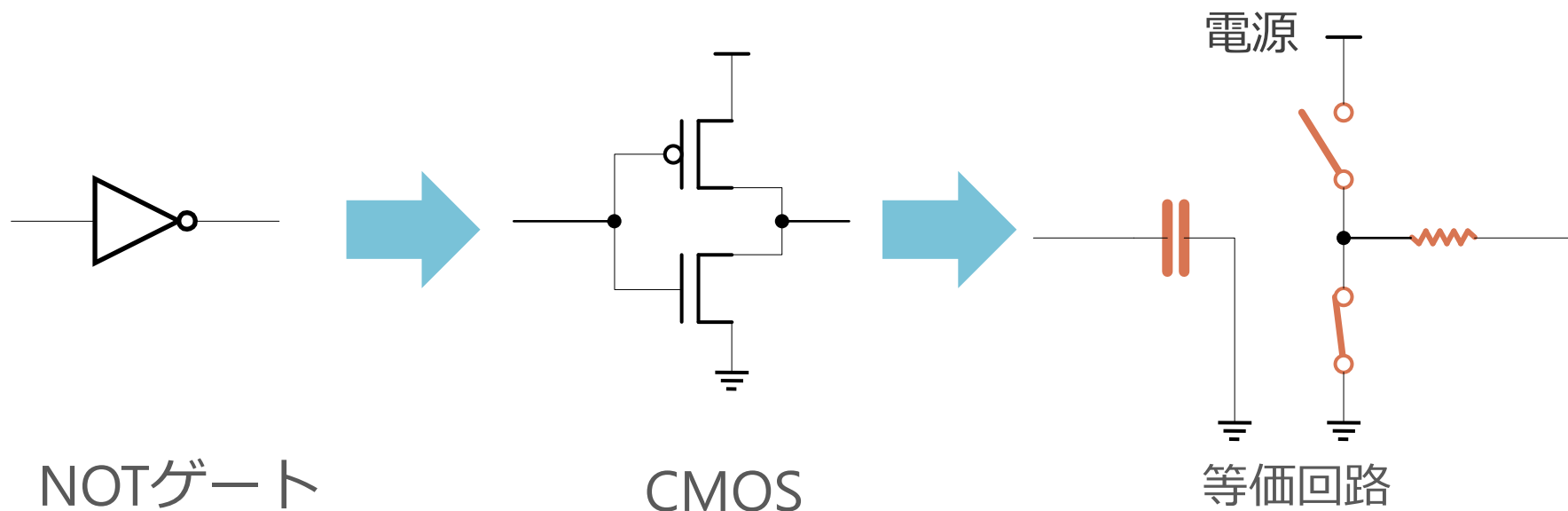
- Substantial power reduction across applications



クロックによる消費電力

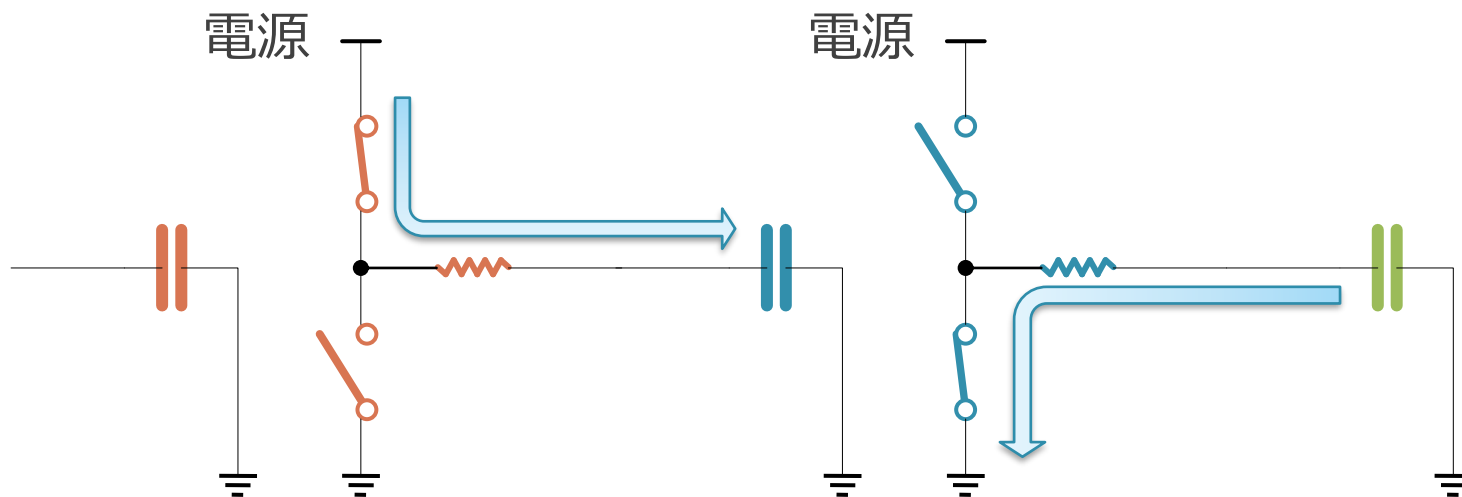
- なぜ更新タイミングの制御でそんなに電力が消費されるのか？
- CMOS 回路の消費電力により説明
 - ◇ 結局, コンデンサの充放電の話
 - ◇ 前回の復習からはじめる

CMOS ゲートの等価回路



- 抵抗 & コンデンサと，連動したスイッチによって表せる
 - ◇ コンデンサに充電：下のスイッチがON
 - ◇ コンデンサを放電：上のスイッチがON

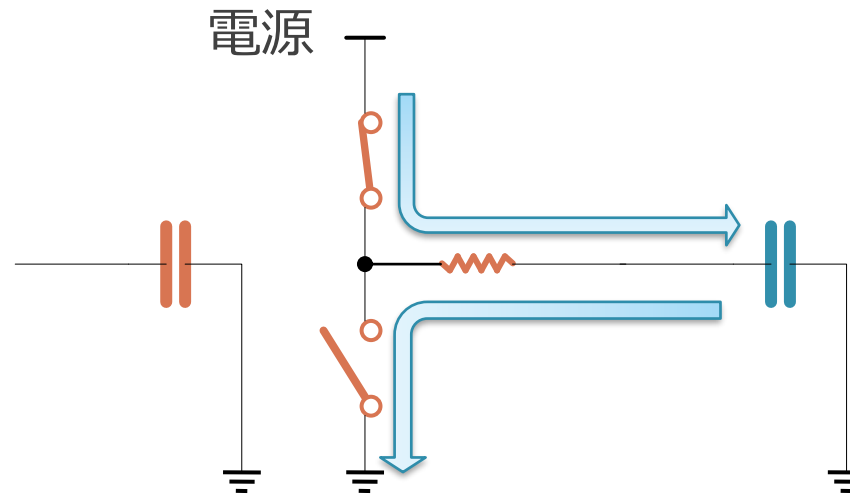
CMOS ゲートの遅延の実体



■ 遅延：コンデンサの充放電にかかる時間

1. あるゲートのスイッチが切り替わる
2. 次の段のゲートへの充放電が開始
3. 次の段のスイッチが切り替わる
4. ...

消費エネルギー

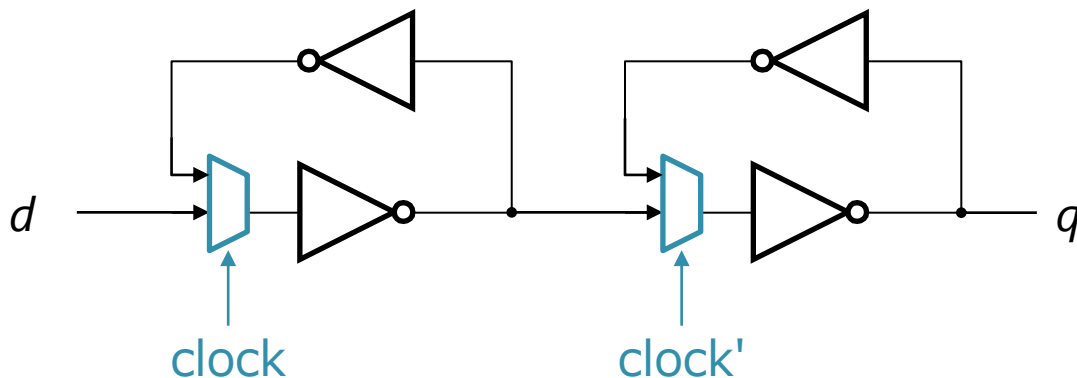


- 消費エネルギーは、主にコンデンサへの充放電で消費される
 - ◇ 消費エネルギーは電圧の二乗に比例： $E = CV^2$
 - ◇ 電荷 $Q = CV$ が、電圧 V の分だけ電源から GND へ移動するから
- 実際には、回路の性質に応じて充放電の回数は変化する
 - ◇ アクティビティ・ファクタ (α) = スイッチング発生確率 に比例

消費エネルギーの補足

- その他に，リーク電流と呼ばれるものもある
 - ◇ トランジスタを OFF にしていても，流れ続けてしまう電流
- 分類：
 - ◇ 充放電によるもの：動的（dynamic）消費電力
 - ◇ リークによるもの：静的（static）消費電力
- 通常は，静的消費電力は多くても数割で動的消費電力が主体
 - ◇ 先ほどの Steamroller では，リークは1割未満

D-FF の回路とクロックによる消費電力



■ D-FF の構造 :

- ◇ リング状に繋がっている NOT ゲート
- ◇ クロックによって切り替えられるマルチプレクサ
 - リングを閉じて情報を憶えるか, リングを開いて入力を取り入れるかをクロックで切り替えている

■ クロックによる消費エネルギー :

- ◇ マルチプレクサ (のトランジスタ) への充放電で消費
- ◇ クロック信号が反転するごとに発生

クロックによる消費電力が大きくなる理由

- 理由 1 : CPU 全体の D-FF で毎サイクル必ず充放電が行われるため
 1. クロックなので毎サイクル必ず反転する
 - = アクティビティ・ファクタは 1
 2. 充放電されるトランジスタの総数もすごく多い

クロックによる消費電力が大きくなる理由

■ 理由2：クロック供給のための配線が長大なため

- ◇ 配線も寄生コンデンサを作るので、そこで充放電が起きる

1. クロックでは、配線の総延長がすごいことになる

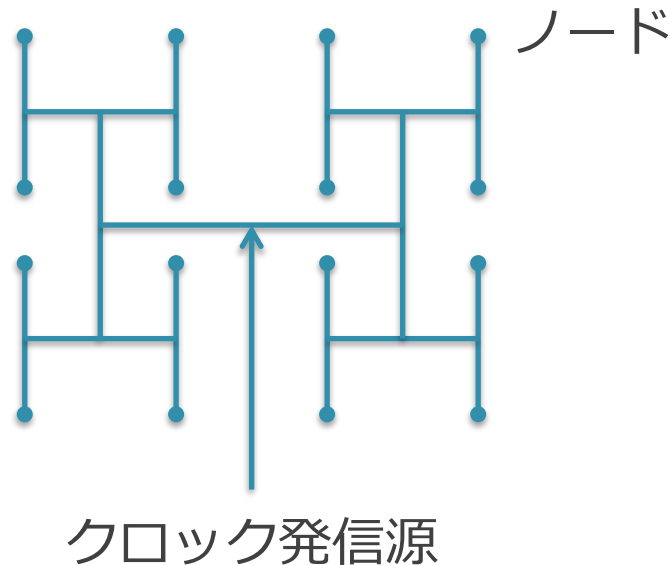
- ◇ すべての D-FF が単一のクロック発信源まで接続

2. スキュー（skew）を無くすために、配線はより長くなる

- ◇ スキュー：D-FF 間のクロックの到達のずれ

- ◇ クロック発信源から各 D-FF までの配線長が等しくなるように配置

クロックの配線方法の例：H-TREE



■ H-TREE

- ◇ クロック発信源から各ノードへの配線長が全て等しくなる
- ◇ しかしその分、物理的に近いノードであっても遠回りになる

H-TREE による配線の例 : IBM Power PC

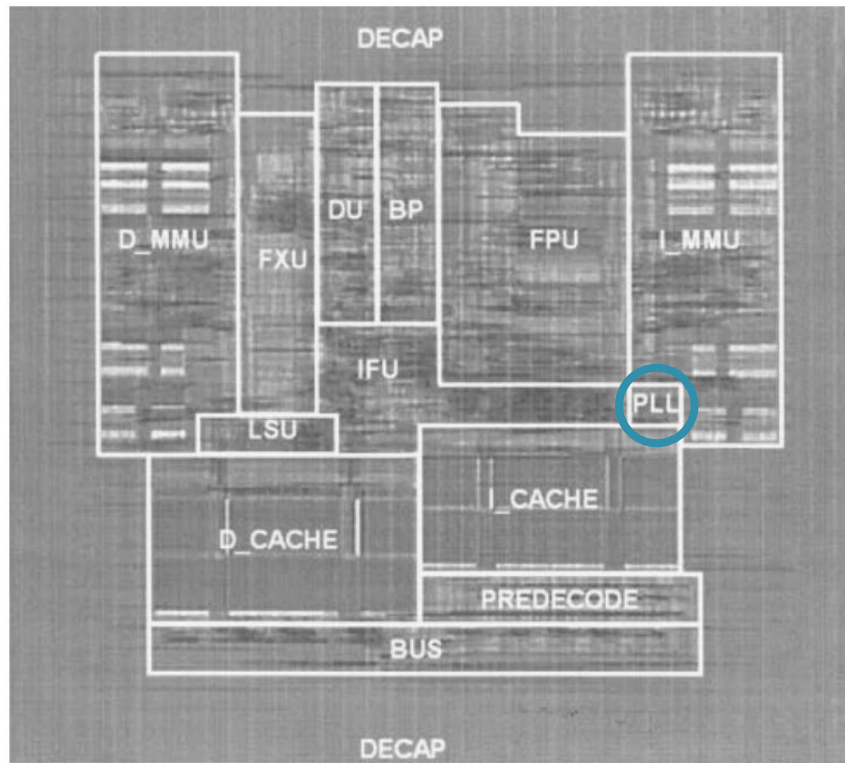


Figure 5.4.2: Die micrograph and floorplan.

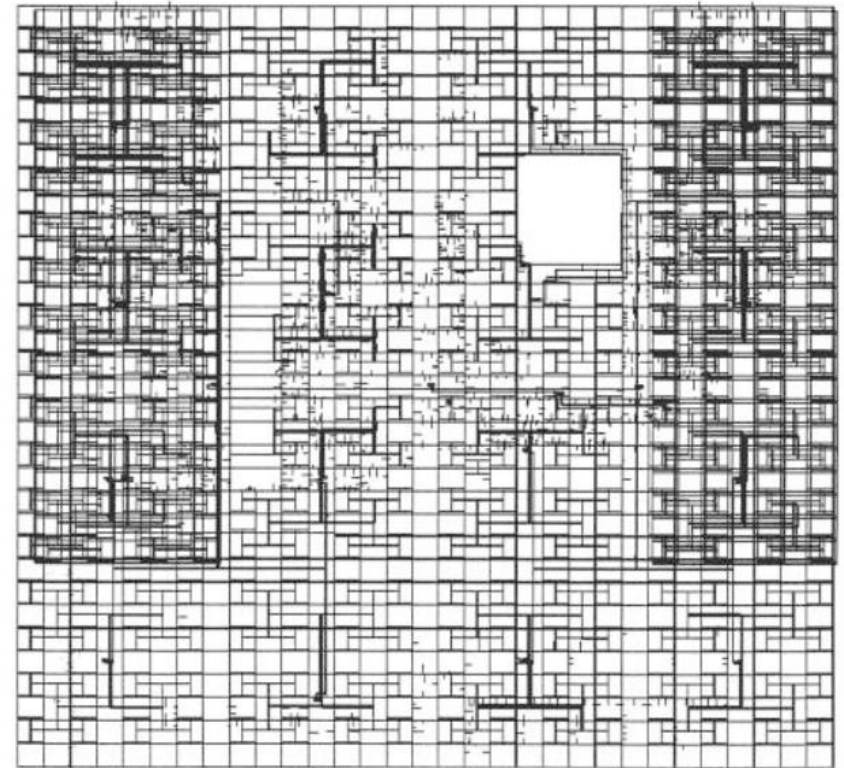


Figure 5.4.7: Clock distribution.

- P. Hofstee, N. Aoki, D. Boerstler, P. Coulman¹, S. Dhong, B. Flachs, N. Kojima, O. Kwon, K. Lee, D. Meltzer², K. Nowka, J. Park, J. Peter, S. Posluszny, M. Shapiro³, J. Silberman², O. Takahashi, B. Weinberger, MP 5.4 A 1GHz Single-Issue 64b PowerPC Processor, ISSCC 2000 より

クロックの消費電力のまとめ

- クロックによる消費エネルギー：充放電で消費
 - ◇ D-FF 内にあるトランジスタ
 - ◇ クロックを配るための配線
- 大きくなる理由：
 - ◇ クロック信号が反転するごとに充放電が毎回発生
 - ◇ トランジスタ数や配線長が膨大

CPU やその他回路の消費電力について

■ いくつか補足

1. クロックの消費電力
2. **アーキテクチャの違いによる消費電力の違い**
3. FPGA による回路

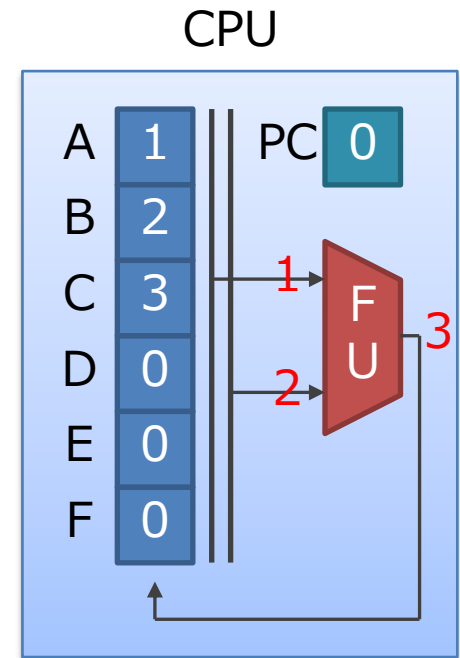
回路の遅延と消費エネルギー

- 回路の消費エネルギー：
 - ◇ 主にコンデンサへの充放電で消費
- おおざっぱには、トランジスタ数に比例すると考えて良い
 - ◇ トランジスタ数が増えると、コンデンサが増える
 - ◇ トランジスタ数 \propto 回路面積

命令を処理するのに必要な回路

■ 内訳：

- ◇ 命令の読み出し
- ◇ デコード
- ◇ レジスタ読み書き
- ◇ メモリの読み書き
- ◇ (命令のスケジューリングや投機関係など
- ◇ 演算



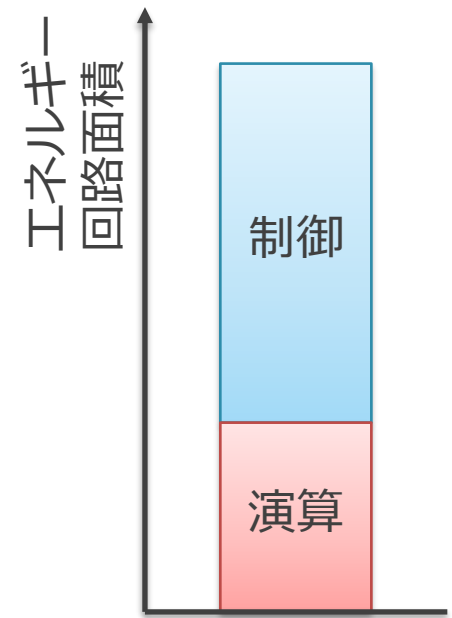
回路は命令制御と演算に大きく分けられる

■ 命令制御：アーキテクチャによって大きく異なる

- ◇ 命令の読み出し
- ◇ デコード
- ◇ レジスタ読み書き
- ◇ （命令のスケジューリングや投機関係など

■ 演算：基本的に同じ

- ◇ 論理演算，算術演算，浮動小数点演算
- ◇ これら演算単体は，だいたいどのアーキでも同じことをする



⇒ 命令制御部分をどれだけケチれるかが，アーキの効率を大きく決める

いろいろな回路の規模

- 1bit NAND 演算器 :
(小さすぎて見えない)


4 トランジスタ

- 64bit 整数加算器 :


4k トランジスタ

- MIPS R3000 プロセッサ :


115k トランジスタ

- 64bit 浮動小数点 乗算+加算器 :


68k トランジスタ

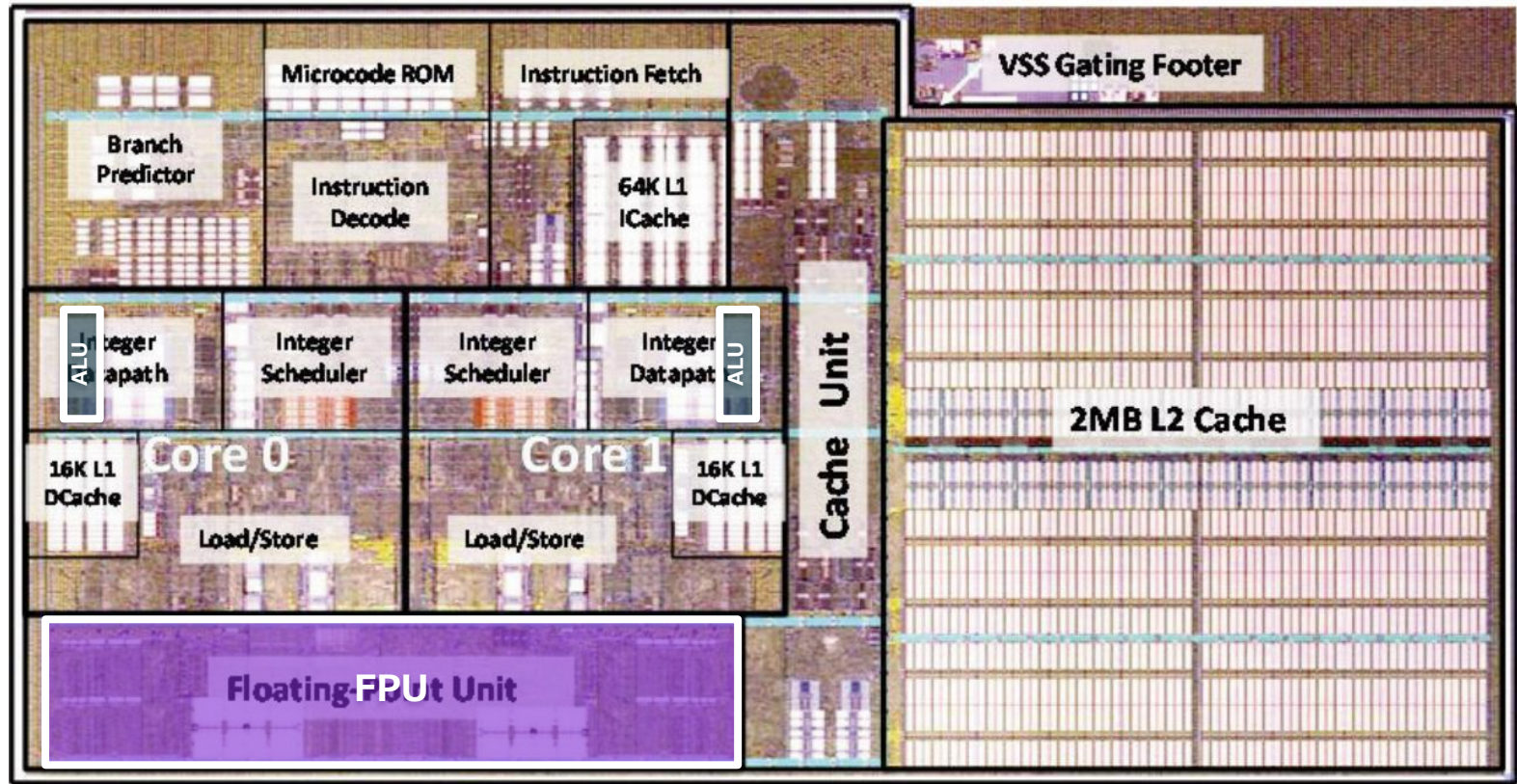
Ryota Shioya, Masahiro Goshima, and Hideki Ando: A Front-end Execution Architecture for High Energy Efficiency, MICRO 2014.

SOHN, Jongwook, et al.: Enhanced Floating-Point Multiply-Add with Full Denormal Support, ARITH 2023

回路規模の例からわかること

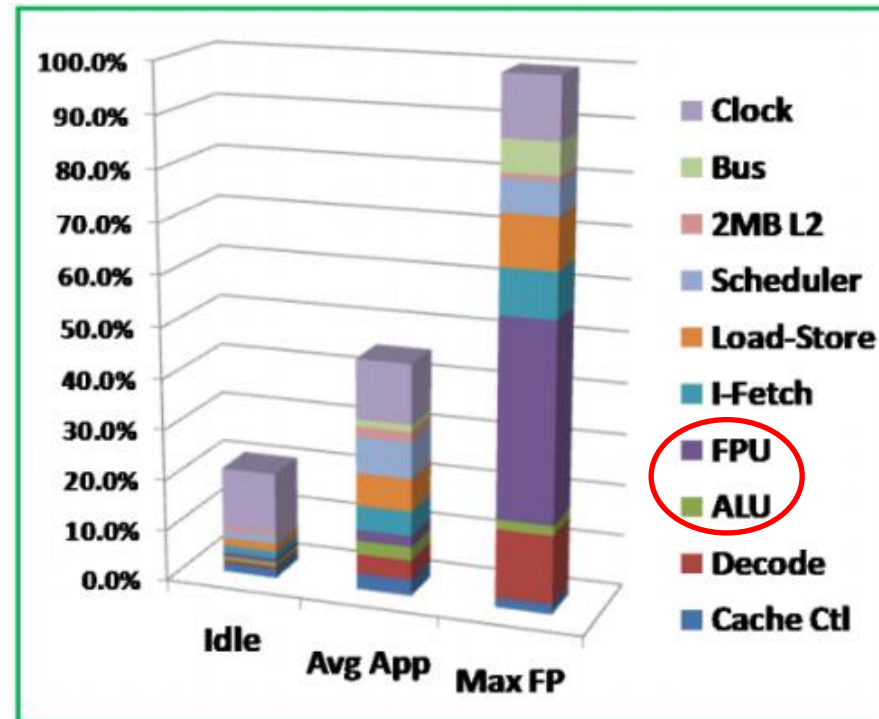
- MIPS プロセッサ全体に対する相対的な大きさで考える：
 - ◇ 1ビット論理演算：
 - 命令制御がほぼ全てを占める
 - ◇ 64 ビット加算
 - 命令制御が大半を占める
 - ⇒ 専用回路にすると一桁ぐらい効率が上がりうる
 - ◇ FP 演算
 - 演算器の方やや小さい
 - ⇒ 専用回路にすると2倍ぐらいは効率が良くなるかも
- 消費エネルギーも、これに準じた大きさとなる

AMD Bulldozer のチップ写真



- Tim Fischer¹, Srikanth Arekapudi², Eric Busta¹, Carl Dietz³, Michael Golden², Scott Hilker², Aaron Horiuchi¹, Kevin A. Hurd¹, Dave Johnson¹, Hugh McIntyre², Samuel Naffziger¹, James Vinh², Jonathan White⁴, Kathryn Wilcox, Design Solutions for the Bulldozer 32nm SOI 2-Core Processor Module in an 8-Core CPU, ISSCC 2011 より

AMD Steamroller の消費電力



■ ALU : 整数演算器

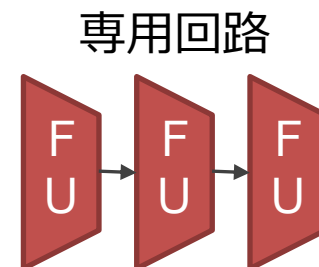
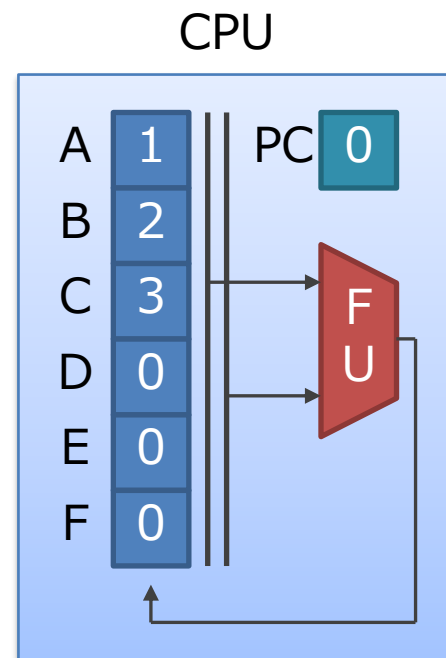
◇ 全体からみると, 大きな割合を占めていない

■ FPU : FP 演算器

◇ 稼働時 (Max FP) には, かなり大きな割合を占める

エネルギーは命令制御と演算の比率によって決まる

- CPU：演算以外の制御部分が多い
 - ◇ 基本 1 つの命令が 1 つのデータを操作
 - ◇ 高速化のため命令の実行順を入れ替える等もする
- GPU：制御部分が相対的に小さい
 - ◇ 1 つの命令で多数のデータを操作
 - ◇ 命令フェッチ/デコードなどに必要な分が減る
- 専用回路：制御部分やレジスタがない
 - ◇ そもそも命令で処理しない
 - ◇ 演算器のみが繋がったような構造に流し込む
 - ◇ (演算器の幅自体をさらに最適化することもある)



使いやすさは、おおむね制御部分の大きさに比例

■ CPU：制御部分が大きい

- ◇ ほっといても、ハードが（ある程度）勝手に並列実行してくれる
- ◇ プログラマが一番楽

■ GPU：制御部分が小さい

- ◇ 単一の命令で複数のデータを操作
- ◇ 規則正しくデータが並んでいるようにお膳立てしないと性能がでない

■ 専用回路：制御部分が（あまり）ない

- ◇ そもそもプログラムを実行できない
- ◇ 目的ごとに回路の設計からしないといけない

牧野先生のアーキテクチャの「効率」の定義

- Principles of High-Performance Processor Design
For High Performance Computing, Deep Neural Networks and Data Science
Junichiro Makino
<https://link.springer.com/book/10.1007/978-3-030-76871-3>
- あるアーキテクチャにおいて、あるアプリケーションを実行する場合に、
 - ◇ FP 演算そのものに必要な最小のエネルギー： P_{\min}
 - ◇ 実際に消費されたエネルギー： P_{actual}
 - ◇ 効率： $P_{\min}/P_{\text{actual}}$
- 演算以外の余計なことをどれだけ減らすかが効率を決めるという点で、おおよそ同じ事を言っている

CPU やその他回路の消費電力について

■ 消費電力について

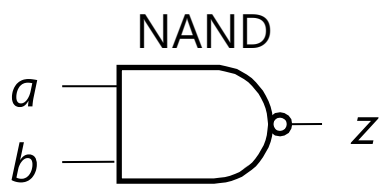
1. クロックの消費電力
2. アーキテクチャの違いによる消費電力の違い
3. **FPGA による回路**

FPGA の場合

- FPGA : Field-Programmable Gate Array
 - ◇ 中身を書き換えることのできる回路
- FPGA で専用回路を作れば, いいことばかり?
 - ◇ 設計の敷居が下がりつつ, 電力効率もよくなる?
 - ◇ CPU で実行されるプログラムの処理を専用回路に置き換える
- そんなに単純な話ではない
 1. FPGA の仕組み
 2. FPGA でうまく行く場合と行かない場合

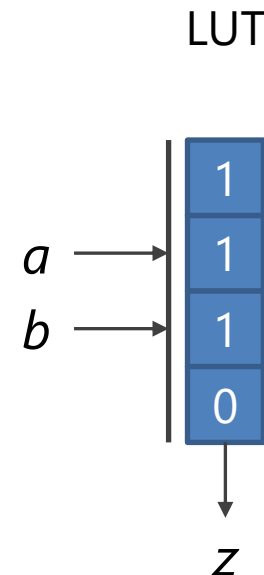
FPGA の仕組み

- 書き換え可能なテーブルにより, 回路を実現
 - ◇ LUT : Look up Table
 - 真理値表そのものを保持するテーブル
 - 事前にこれを所望の回路の真理値表に設定しておく
 - ◇ 入力をインデックスとしてテーブルにアクセスし, 出力
 - 下の NAND の場合, a と b の 2ビットのインデックス



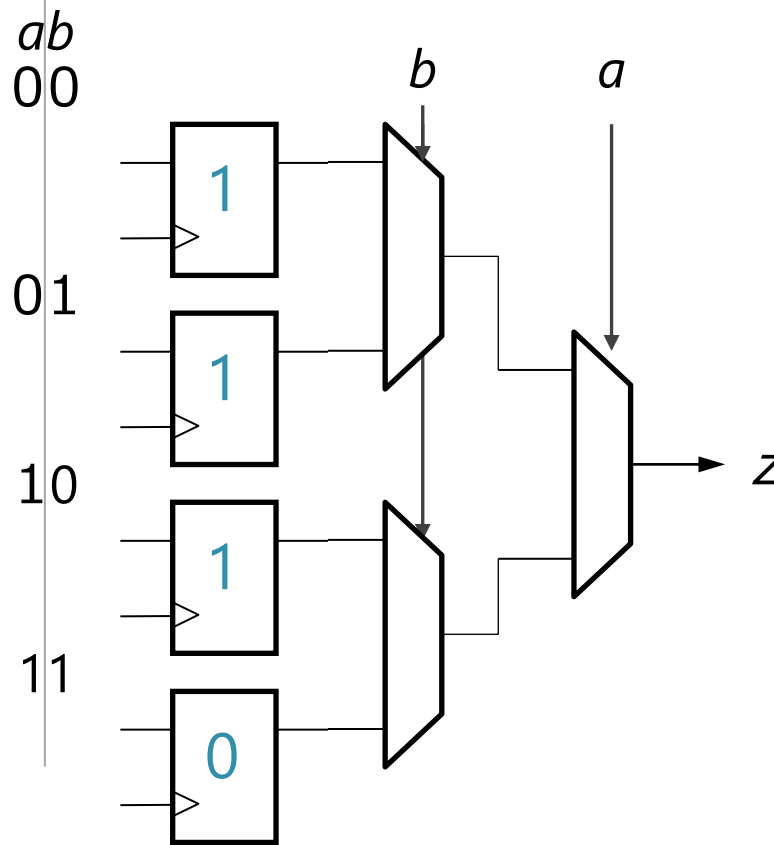
真理値表

a	b	z
0	0	1
0	1	1
1	0	1
1	1	0



LUT の回路量の見積もり

NAND		
a	b	z
0	0	1
0	1	1
1	0	1
1	1	0

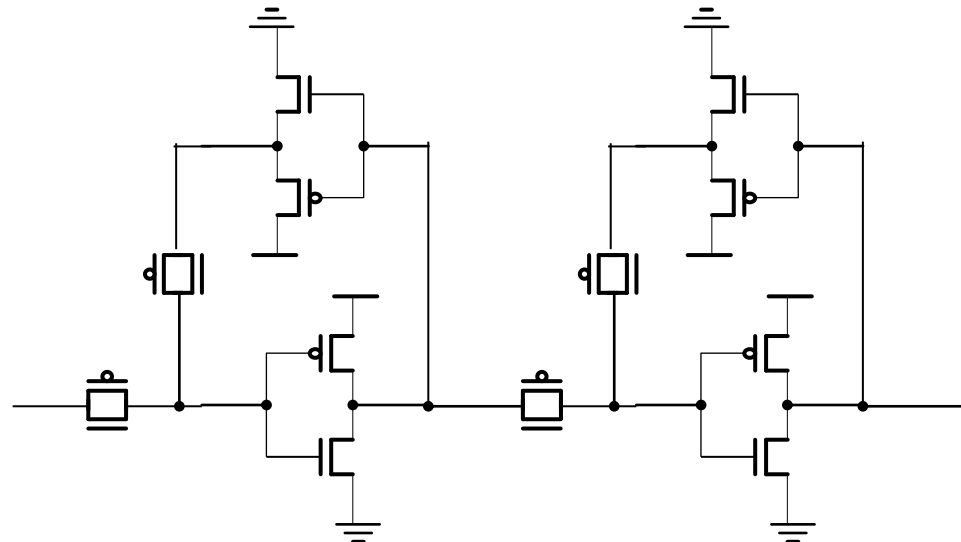
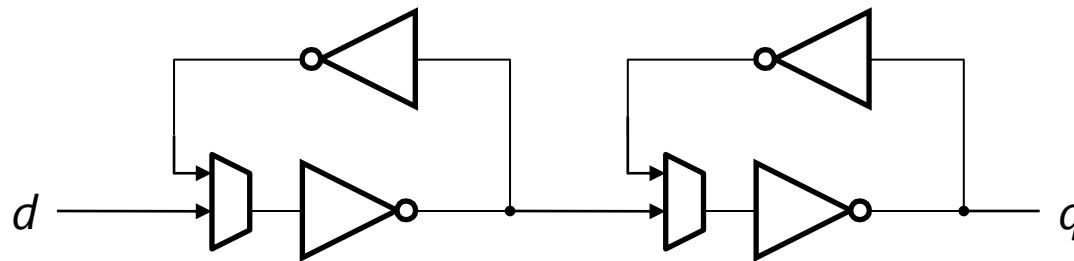
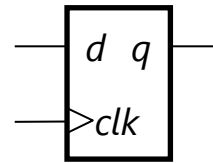


■ 4 エントリの LUT を D-FF で構成してみる

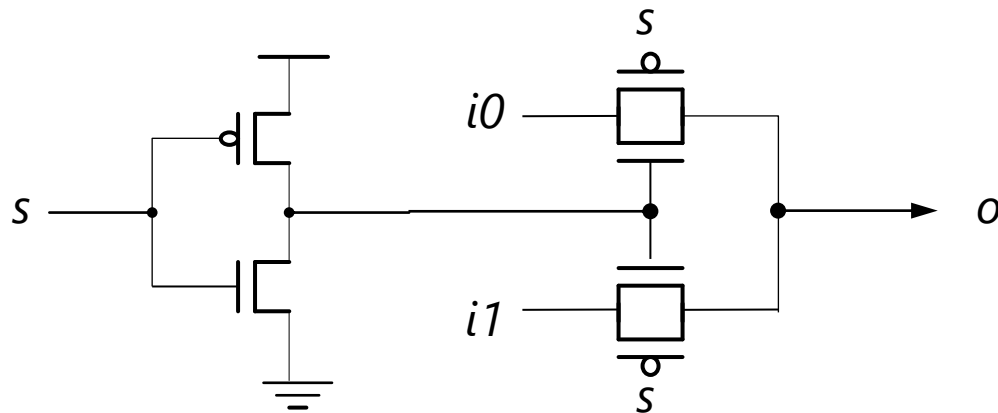
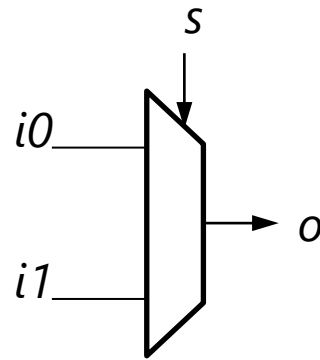
◇ 中身を憶える 4つの D-FF

◇ 場所を指定して選択する2段のマルチプレクサ

D-FF : トランジスタ 16個



マルチプレクサ：トランジスタ 6個

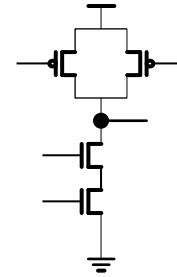
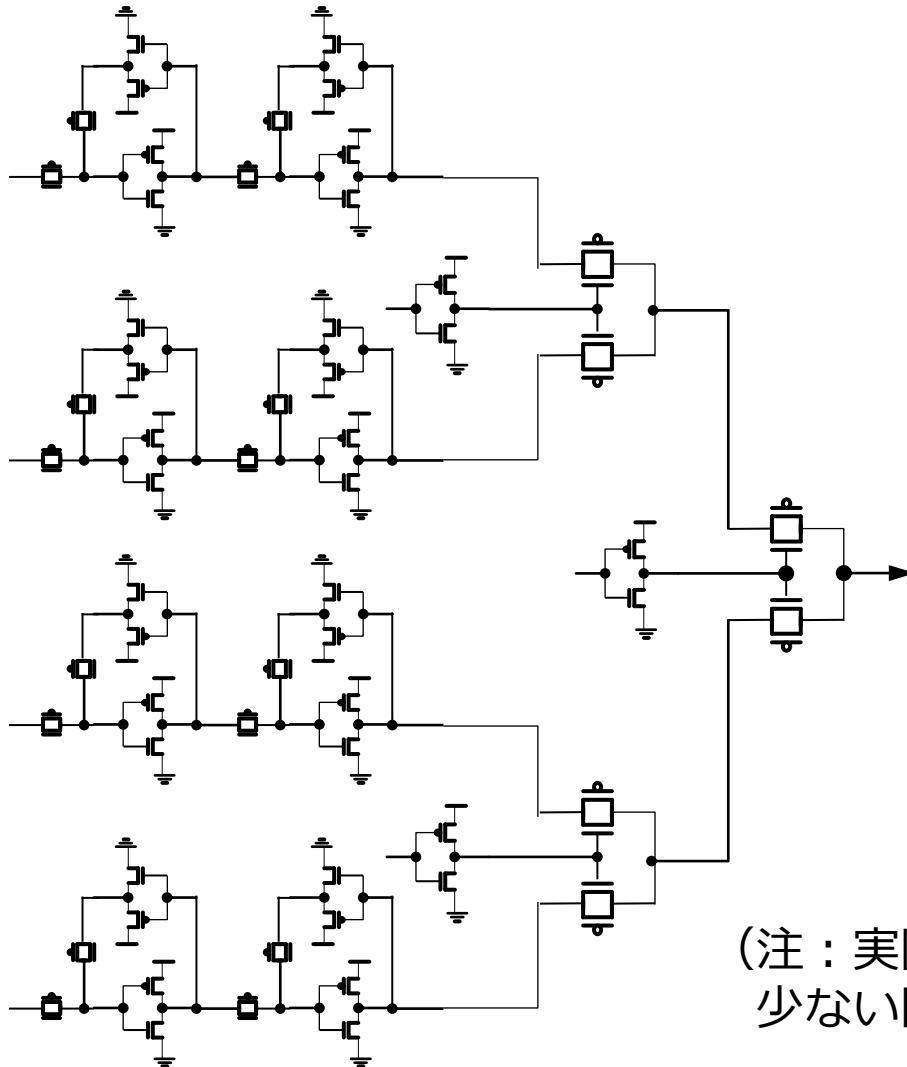


LUT vs. NAND

同じ回路を LUT で実現するのはものすごく効率が悪い

LUT : $16 \times 4 + 6 \times 3 = 82$

NAND : 4



(注 : 実際にはもう少しトランジスタ数の少ない回路でできています)

LUT で回路を構成した場合

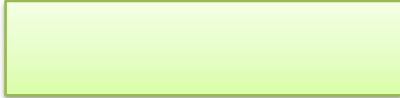
- このほかに, LUT 間を結合するためのネットワークも必要
- 結果として, 直接回路を作るより 1 ~ 2 桁は下記の特性が悪化
 - ◇ 回路面積
 - ◇ 遅延
 - ◇ エネルギー

CPU から FPGA にしたときに良い場合・悪い場合

■ MIPS R3000 プロセッサ :

115k トランジスタ

- ◇ これの上で動くプログラムを FPGA に置き換えた場合を考える



■ 1bit NAND 演算器 :

4 トランジスタ

- ◇ LUT によって 82 トランジスタになっても十分上記より小さい
(小さすぎて見えない)

■ 64bit 整数加算器 :

4k トランジスタ

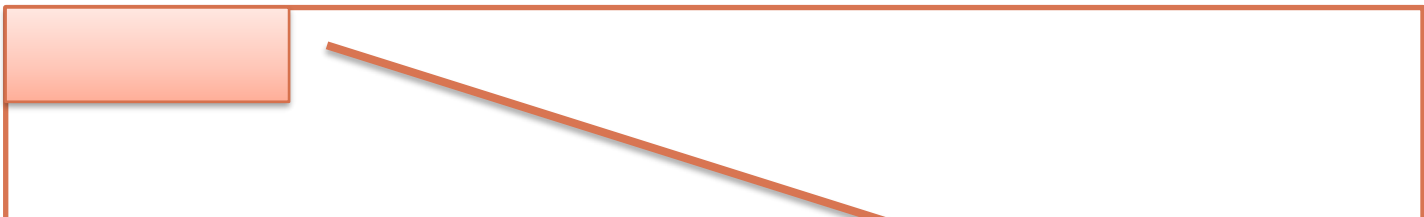
- ◇ 20倍大きくなり 80k になると, MIPS でやるのとほとんど変わらない



■ 64bit 浮動小数点 乗算+加算器 :

68k トランジスタ

- ◇ FPGA にすると巨大になりすぎるし, 何もおいしくない



FPGA の特性のまとめ

- トレードオフによって、最終的な優劣がきまる
 - ◇ FPGA 良い点：専用回路をくめば、命令制御に必要な資源が不要
 - データの受け渡のためのレジスタ・ファイルなども不要になる
 - ◇ FPGA 悪い点：回路としては一般に 1 桁から 2 桁程度性能が悪化
- 演算の種類と複雑さで FPGA 化したときにおいしいかどうかは決まる
 - ◇ 既に CPU や GPU に演算器が載っているような FP 演算などは逆効果になりかねない
- なお実際には、よく使われる回路は、FPGA では LUT ではないものが入っていることも多い
 - ◇ 加算器や乗算器などは専用の回路が入っている

ここまでのまとめ

1. クロックの消費電力

- ◇ クロックの消費電力が CPU 全体に占める割合は大きい
- ◇ チップ全体の D-FF とそれへの配線を毎サイクル充放電するから

2. アーキテクチャの違いによる消費電力の違い

- ◇ 回路は命令制御と演算に大きく分けられる
- ◇ 命令制御に回路を割くと消費電力が大きくなるが、プログラマは楽に

3. FPGA による回路

- ◇ FPGA は直接回路を作るのと比べるとかなり効率が悪い
- ◇ CPU で動いているプログラムを FPGA の専用回路にした場合、おいしいかどうかは演算の複雑さで決まる

余談：ムーアの法則と周波数

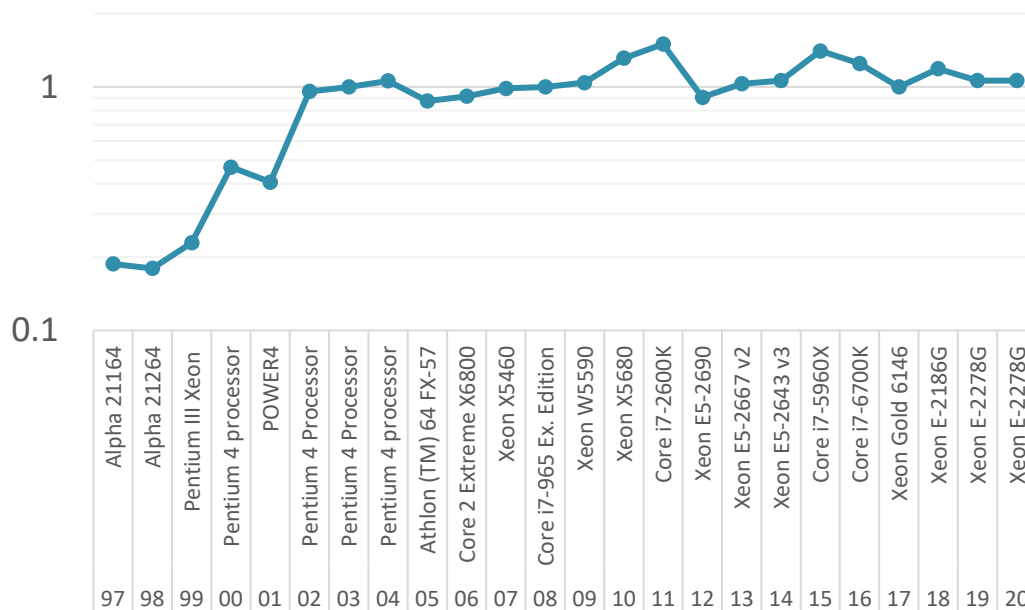
クロック周波数

■ CPU のクロック周波数：

- ◇ 1 秒間に何回処理を行えるかを表す
- ◇ 性能を大きく左右

■ 2002年頃からほぼ上がっていない

- ◇ (実はここ数年はまた少し上がってきてはいる)



周波数向上がストップ

■ なぜ？ → 電圧が下げられなくなったから

◇ エネルギーの壁にぶつかった

1. 半導体のスケーリング
2. CPU の消費エネルギーとは何なのか
3. ダークシリコン問題

ムーアの法則

- 「半導体の集積度は3年ごとに4倍になる」

◇ トランジスタのサイズを1/2に縮小（**スケーリング**）

面積： $1/2 \times 1/2 = 1/4$

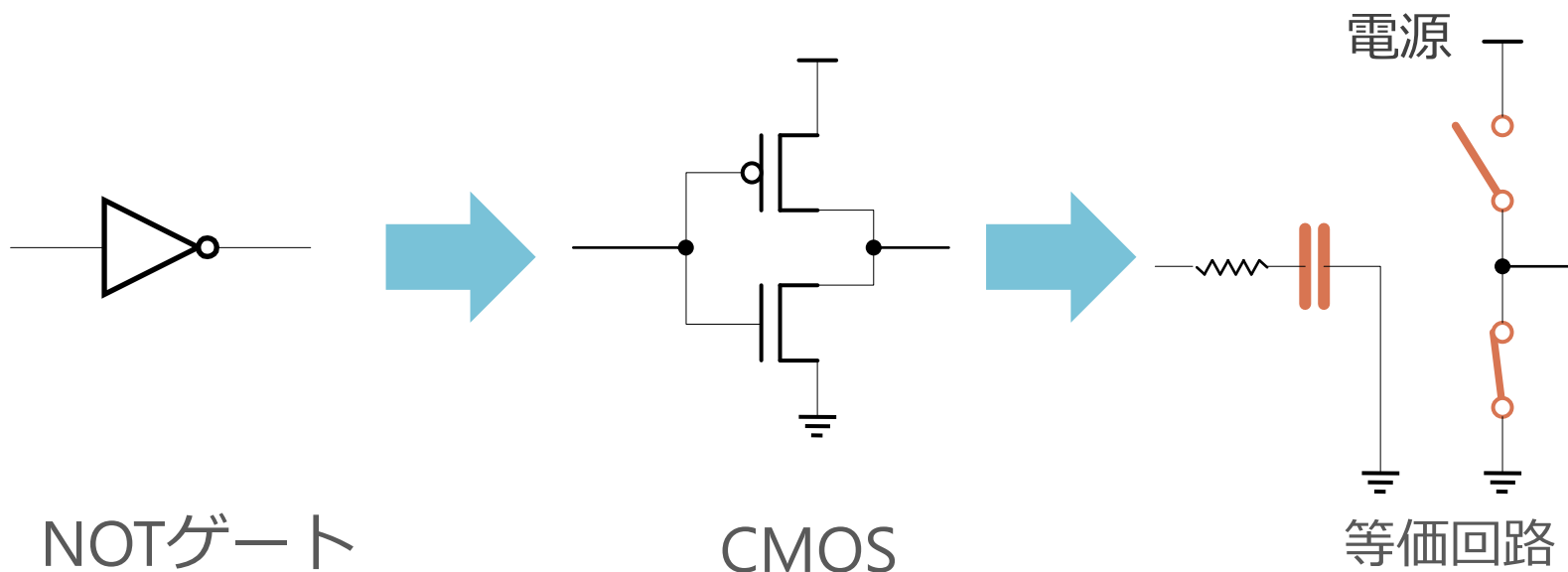


- すごいけど、数が増えるだけなの？

スケーリングの効果

- トランジスタあたりで,
 - ◇ 消費エネルギーは $1/8$ に
 - ◇ 遅延も $1/2$ に
- 等価回路を使って説明

CMOS ゲートの等価回路



- コンデンサと、連動したスイッチによって表せる
 - ◇ 充電：下のスイッチがON
 - ◇ 放電：上のスイッチがON
- CPU の計算で消費されるエネルギー：
 - ◇ スイッチ ON/OFF するためのコンデンサへの充放電

消費エネルギーと遅延

- コンデンサへの充電に必要なエネルギー

- ◇ $\frac{1}{2} CV^2$

- サイズと電圧を $1/K$ 倍にスケーリングした場合

- ◇ C: $1/K$ (面積 $1/K^2$, 厚さ $1/K$ より)

- ◇ V: $1/K$

- ◇ エネルギー: $1/K^3$

- C が小さくなり充電にかかる時間が減るので, 遅延も $1/K$ に

- ◇ 結果として, 周波数は K 倍に

- (実際は配線などにも C が存在するので, もう少し複雑

チップ全体の消費エネルギー

- $1/K$ 倍にスケーリングした場合, チップ全体では,

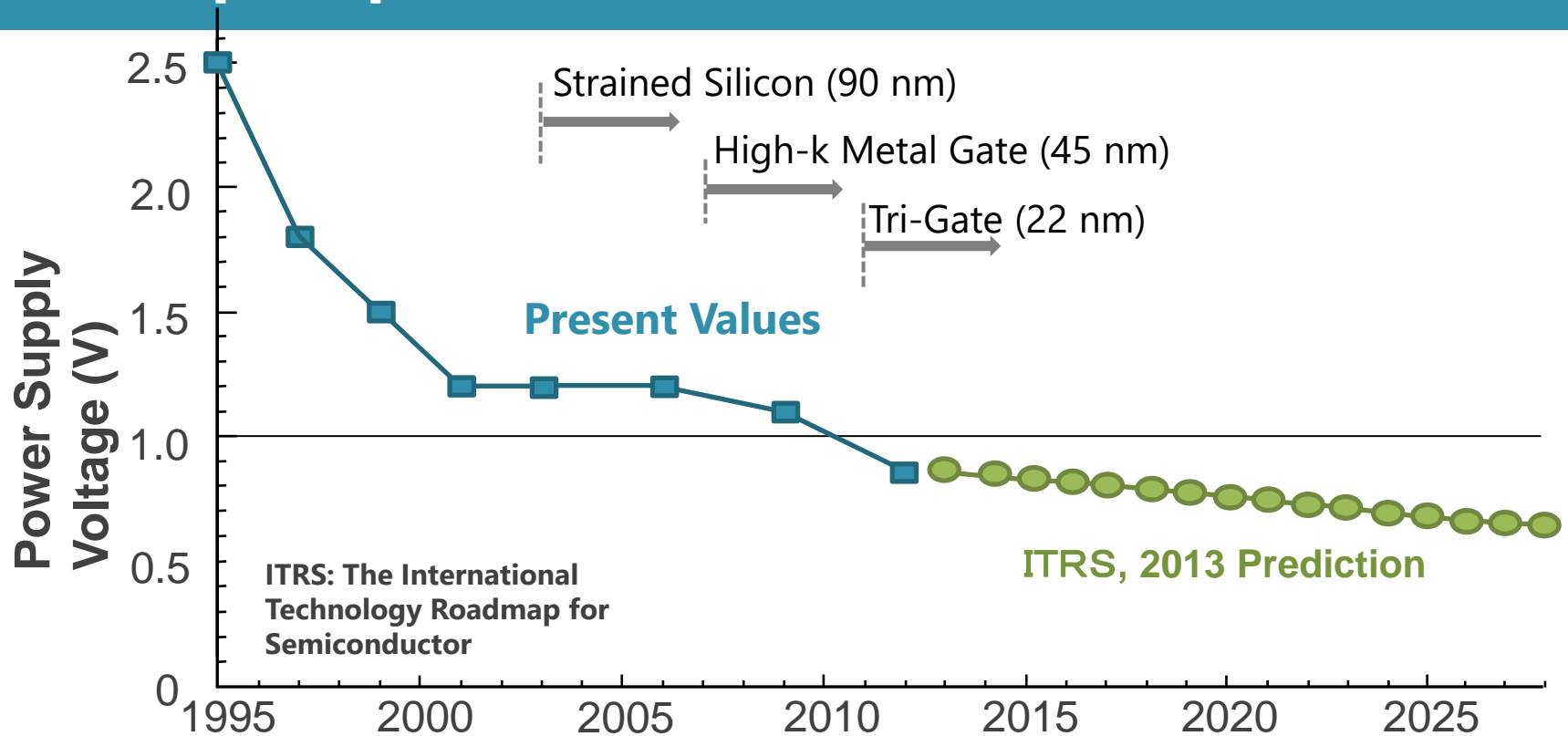
◇ 個々のトランジスタのエネルギー :	$1/K^3$
◇ トランジスタの個数 :	K^2
◇ 周波数 (動作回数) :	K
◇ 全体の消費エネルギー :	1

- まとめると, スケーリングによって

- ◇ 同じ大きさのチップに搭載できる回路量が増えるのはもちろん,
- ◇ 消費エネルギーは一定のまま,
- ◇ 周波数もすごい向上!

ところが、電圧が下げられなくなった...

グラフは [Sato17] より



- 2000 年過ぎから、電圧低下がほぼとまった
 - ◇ 電圧が下がりすぎて、ON / OFF の境界の閾値が 0V 付近に
 - ◇ OFF にしたいときも、閾値との距離がとれない
 - ◇ 微妙に ON になり電流が漏れてしまう（リーク電流という）

電圧が下がらないとどうなるのか

■ $1/K$ 倍にスケーリングした場合

◇ 回路面積: $1/K^2 \rightarrow 1/K^2$

■ トランジスタ 1 個の 1 回のスイッチにかかるエネルギー

◇ C: $1/K \rightarrow 1/K$

◇ V: $1/K \rightarrow 1$

◇ エネルギー: $1/K^3 \rightarrow 1/K$ ($E = \frac{1}{2}CV^2$ より)

■ トランジスタ 1 個の単位時間あたりのエネルギー

◇ スイッチ 1 回: $1/K^3 \rightarrow 1/K$

◇ 周波数 (動作回数): $K \rightarrow K$

◇ エネルギー: $1/K^2 \rightarrow 1$

■ トランジスタは小さくなったのに, エネルギーが減っていない!

行き着く先：ダークシリコン

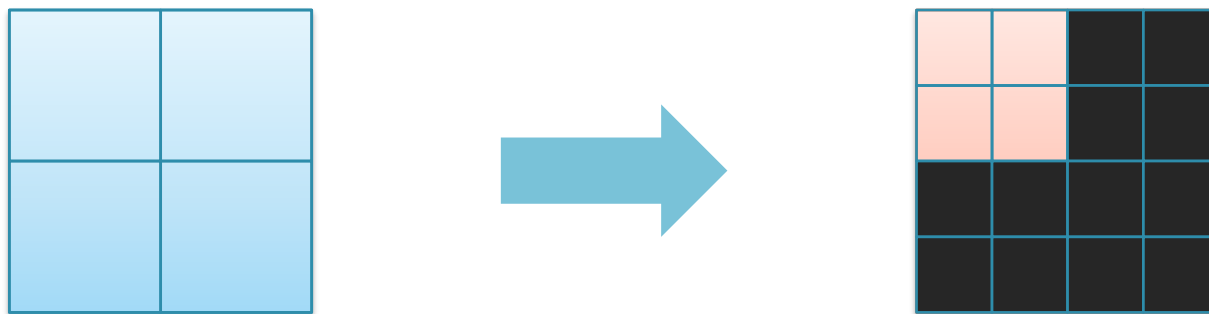
[Goulding10, Esmaeilzadeh12]

- トランジスタは小さくなったのに，エネルギーが減らない！

- ◇ サイズだけ小さくして，電圧を下げていないから

- 電力密度があがってしまう

- ◇ スケール前のチップ全体と，スケール後の赤い部分は同じエネルギーを消費



- チップへの電力供給はもう限界で，これ以上増やせない

- ◇ 電源ピンに流せる電流量や冷却能力の限界

- ◇ 使えない（**ダーク**）シリコンが...

とはいえ、いまのところ



- まだ、ダークにまでは至っていない
 - ◇ デバイス技術が進歩
 - ◇ 電圧が一応ちろちろ下がってはいる
 - ◇ 周波数を上げなくした
 - 動作回数が減るので、電力にダイレクトに効く
 - ◇ あまりスイッチ（充放電）しないキャッシュなどを多めに入れた
- まとめ：電圧が下げられなくなったので、電力密度増大をおさえるために周波数を上げるのをやめた

まとめ

- 回路の消費エネルギー

1. クロックの消費電力
2. アーキテクチャの違いによる消費電力の違い
3. FPGA による回路

- （余談）ムーアの法則と周波数

- この講義資料では、一部、五島先生の「デジタル回路」の講義資料の図を使用しています

出欠と感想

- 本日の講義でよくわかったところ, わからなかったところ, 質問, 感想などを書いてください
 - ◇ LMS の出席を設定するので, そこにお願いします
 - ◇ パスワード:
- 意見や内容へのリクエストもあったら書いてください
- LMS の出席の締め切りは来週の講義開始までに設定してあります
 - ◇ 仕様上「遅刻」表示になりますが, 特に減点等しません
 - ◇ 来週の講義開始までは感想や質問などを受け付けます

質問や感想

- 電子回路はHWの基礎として、CPUの構造と安全性に影響を与えていますね。前にメモリセル間の電氣的な相互作用による脆弱性（Rowhammer）もありました。

- RISC-Vがなかった時代のCPUの研究では、基本的に作成したCPUは公開せず、ベンチマーク結果等のデータのみで論文を執筆していたのでしょうか

- 完備な組み合わせが色々あるのはわかった一方、NORのみ、NANDのみ、ANDとORとNOTなど、どれを採用するとどのような性能や設計上の差が出るのかよく分からなかったです。

- コンデンサの充放電が遅延の原因ということでしたが、回路の中の電気信号の伝達速度は問題にならないのでしょうか

質問と回答とか

- 学部時代に博士論文全部読み切ろうとして挫折したので、リベンジで読めるところだけ読んでみます

- RISC-Vが過去のプロセッサ設計の知見を活かして作られた命令セットで、色々優れているところがあるのだと思うが、組み込みのコントローラやメニーコアの一つ一つのコアのような利用に留まっているのが気になる

- RISC-Vでx86やarmによるデスクトップやサーバー向けの高機能なCPUを置き換えるようなプロセッサを作るとは現時点ではまだ難しいのでしょうか？
それともそのようなプロセッサが普及するのも時間の問題でしょうか？

- また、即値が命令の中にバラバラに配置されている理由について興味を持ったので、補足スライドなどで触れていただけると嬉しいです。

31	30	25	24	21	20	19	15	14	12	11	8	7	6	0	
funct7				rs2		rs1	funct3		rd			opcode		R-type	
imm[11:0]						rs1	funct3		rd			opcode		I-type	
imm[11:5]				rs2		rs1	funct3		imm[4:0]			opcode		S-type	
imm[12]	imm[10:5]			rs2		rs1	funct3		imm[4:1]	imm[11]		opcode		B-type	
imm[31:12]									rd			opcode		U-type	
imm[20]	imm[10:1]				imm[11]	imm[19:12]			rd			opcode		J-type	

Figure 2.3: RISC-V base instruction formats showing immediate variants.

Figure 2.4 shows the immediates produced by each of the base instruction formats, and is labeled to show which instruction bit ($\text{inst}[y]$) produces each bit of the immediate value.

31	30			20	19			12	11			10			5	4			1			0	
— inst[31] —												inst[30:25]		inst[24:21]		inst[20]		I-immediate					
— inst[31] —												inst[30:25]		inst[11:8]		inst[7]		S-immediate					
— inst[31] —										inst[7]		inst[30:25]		inst[11:8]		0		B-immediate					
inst[31]		inst[30:20]				inst[19:12]				— 0 —										U-immediate			
— inst[31] —						inst[19:12]				inst[20]		inst[30:25]		inst[24:21]		0		J-immediate					

Figure 2.4: Types of immediate produced by RISC-V instructions. The fields are labeled with the instruction bits used to construct their value. Sign extension always uses $\text{inst}[31]$.

- cpu内に入っている論理素子を秋月とかで変えるようなもので構成すると、体育館 1 つ分ぐらいのサイズになるという話を聞いたことがあるが、そのような複雑で、規模が大きい回路を構成するプロセス、また設計するプロセスがとても気になった。

- CPUのパイプラインが深ければ深いほど周波数が上がりやすいと言われてます、これはなぜでしょうか