

電気電子工学実習 論理回路設計演習

この資料:

PandA >> 配布物・テキスト >> 6 論理回路設計演習 >> slides.pdf

本実習の概要

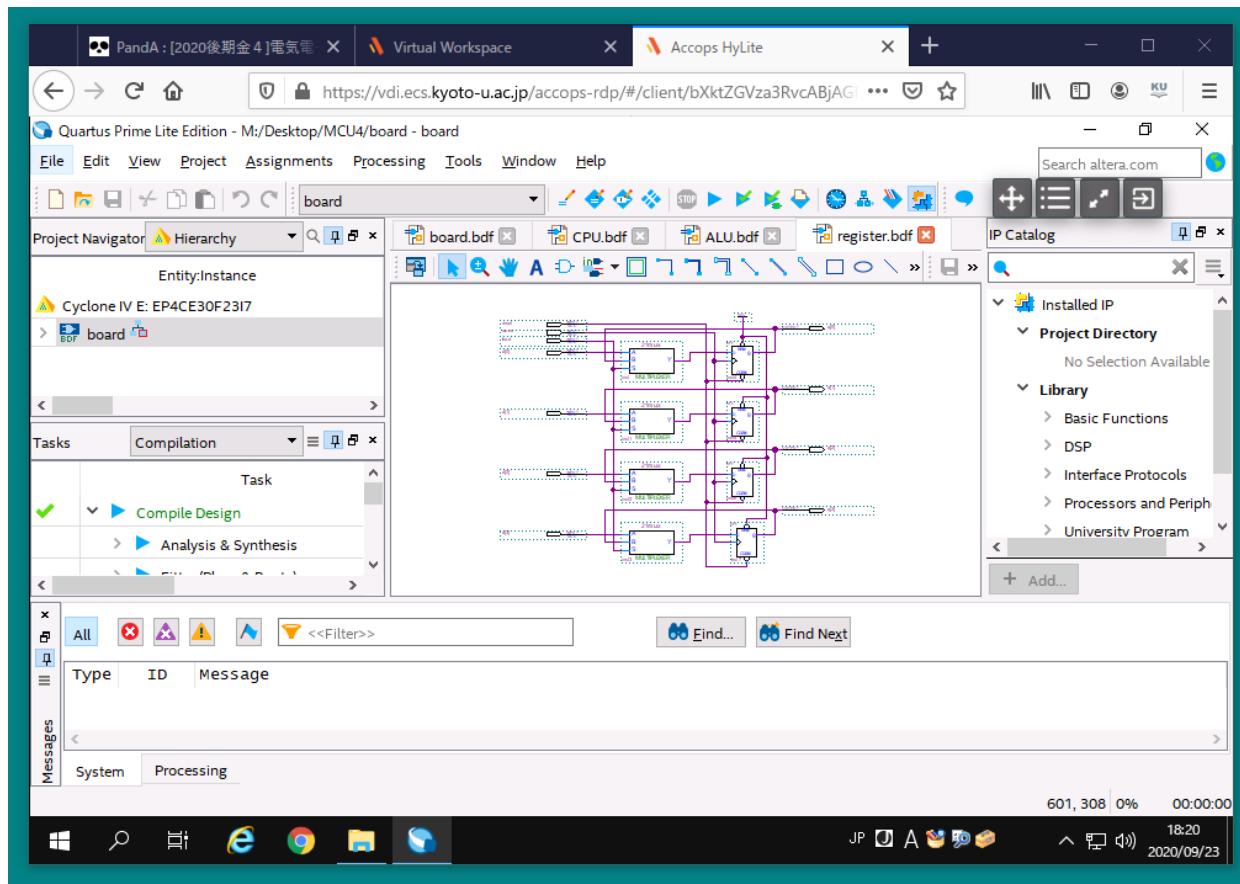
- マイクロプロセッサやマイクロコンピュータに代表されるデジタル集積回路は今日の生活のあらゆるところで利用されている
 - この演習では、マイクロコンピュータ(マイコン)の設計を通じて、マイコンの構造とその動作を理解する
 - シミュレータやFPGAボード(エミュレータ)などの設計支援ツールを使用して、設計した回路を検証する方法やデバッグする方法を学ぶ

進め方1

- 各自PCを持ち込み、実験もレポート提出も個人単位で行います(班での共同作業は無し)
- およそ下のような進度を想定しています
 - 1週目:マイコンの構造と動作の理解、ツール(Quartus Prime)とFPGAボードの使い方(実験6.1, 6.2)、マイコンのプログラムカウンタの設計(実験6.3)、時間があれば予習6.1
 - 2週目:プログラム制御ユニット(PCU)の設計(実験6.4, 6.5)および時間があれば予習6.2の実施
 - 3週目:演算ユニット(ALU)の設計とマイコン設計の完了(実験6.6)および時間があれば予習6.3の実施
 - 4週目:汎用レジスタの設計とマイコンへの組み込み(実験6.7)および時間があれば予備実験課題の実施

進め方2

- 仮想端末サービス (VDI) で実習します
- 各自のノートパソコンからブラウザ経由で接続



ブラウザから
Windows10 + ツール
を使用できる

スタッフ紹介

- 教員

島崎秀昭

松山顕之

TA

– 白

– 中村

生活に浸透するコンピュータ

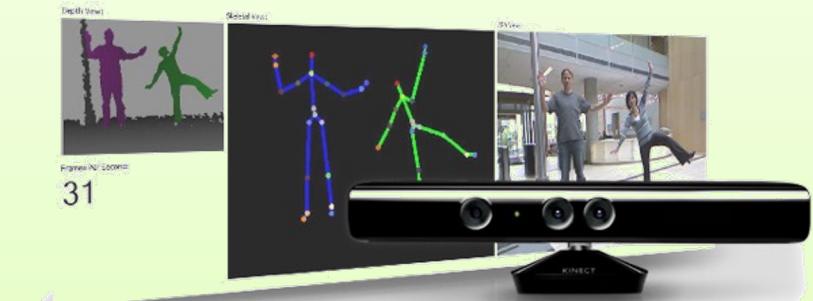
集積回路の高性能化・小型化により
「未来のデバイス」が次々と現実に

身に付ける
コンピュータ
(ウェアラブル
コンピュータ)



Google Glass

直感的な操作

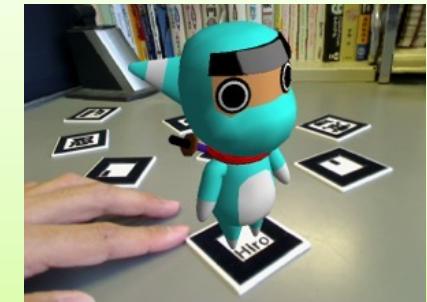


Microsoft Kinect



Leap Motion

現実世界と仮想世界の融合
(仮想現実VRと拡張現実AR)



日本のお家芸マイコン

快適

カーナビ
音楽, 映像
エアコン

安全

加速度センサー
- 姿勢制御, ブレーキ制御
レーダー¹
- 衝突防止
カメラ
- 衝突防止, 自動走行

安心

キーレスエントリー
盗難防止装置



環境(燃費)

各種センサーと連携した
最適なエンジン制御

高級車では1台当たり約
三百個のユニットを搭載
自動車のコストの約40%
が電装品

自動運転カー



マイコンで高い存在感の日本

車載半導体市場での強固なポジション

- 車載半導体トータルでWW No.1を堅持
- 車載アナログ＆パワー半導体では、WW No.5からTOP3入りを目指す

車載半導体 EU			
Company	CY09	CY10	CY11
1 Infineon	13.6%	13.8%	14.7%
2 STMicro	10.5%	10.9%	11.3%
3 Bosch	10.4%	9.1%	9.4%
4 Freescale	8.5%	9.3%	8.7%
5 NXP	8.2%	8.8%	8.2%
6 Renesas	7.6%	7.9%	8.0%

車載半導体 WW		
Company	CY11	
1 Renesas	13.8%	
2 Infineon	9.8%	
3 STMicro	8.7%	
4 Freescale	7.9%	
5 NXP	6.4%	

車載半導体 北米			
Company	CY09	CY10	CY11
1 Freescale	16.1%	15.3%	13.8%
2 Infineon	7.8%	7.7%	8.5%
3 STMicro	6.6%	7.4%	8.0%
4 Renesas	6.5%	7.0%	7.5%
5 NXP	7.1%	7.5%	6.7%
6 TI	5.6%	5.7%	6.0%

車載プロセッサ(MCU/MPU) WW			
Company	CY09	CY10	CY11
1 Renesas	41.7%	41.2%	42.7%
2 Freescale	19.5%	21.5%	19.5%
3 Infineon	6.6%	7.0%	8.3%
4 Fujitsu	6.2%	7.0%	7.0%
5 TI	6.6%	6.0%	5.9%

車載アナログASIC＆パワー半導体 WW			
Company	CY09	CY10	CY11
1 STMicro	16.6%	17.9%	18.6%
2 Infineon	16.5%	15.8%	16.6%
3 NXP	11.4%	12.5%	10.8%
4 TI	8.3%	8.0%	7.9%
5 Renesas	#7 6.6%	#7 7.0%	#5 7.3%

*) 出典: Strategy Analytics. 車載プロセッサと車載アナログ＆パワーのシェアは、上位12社の合計金額を母数に算出

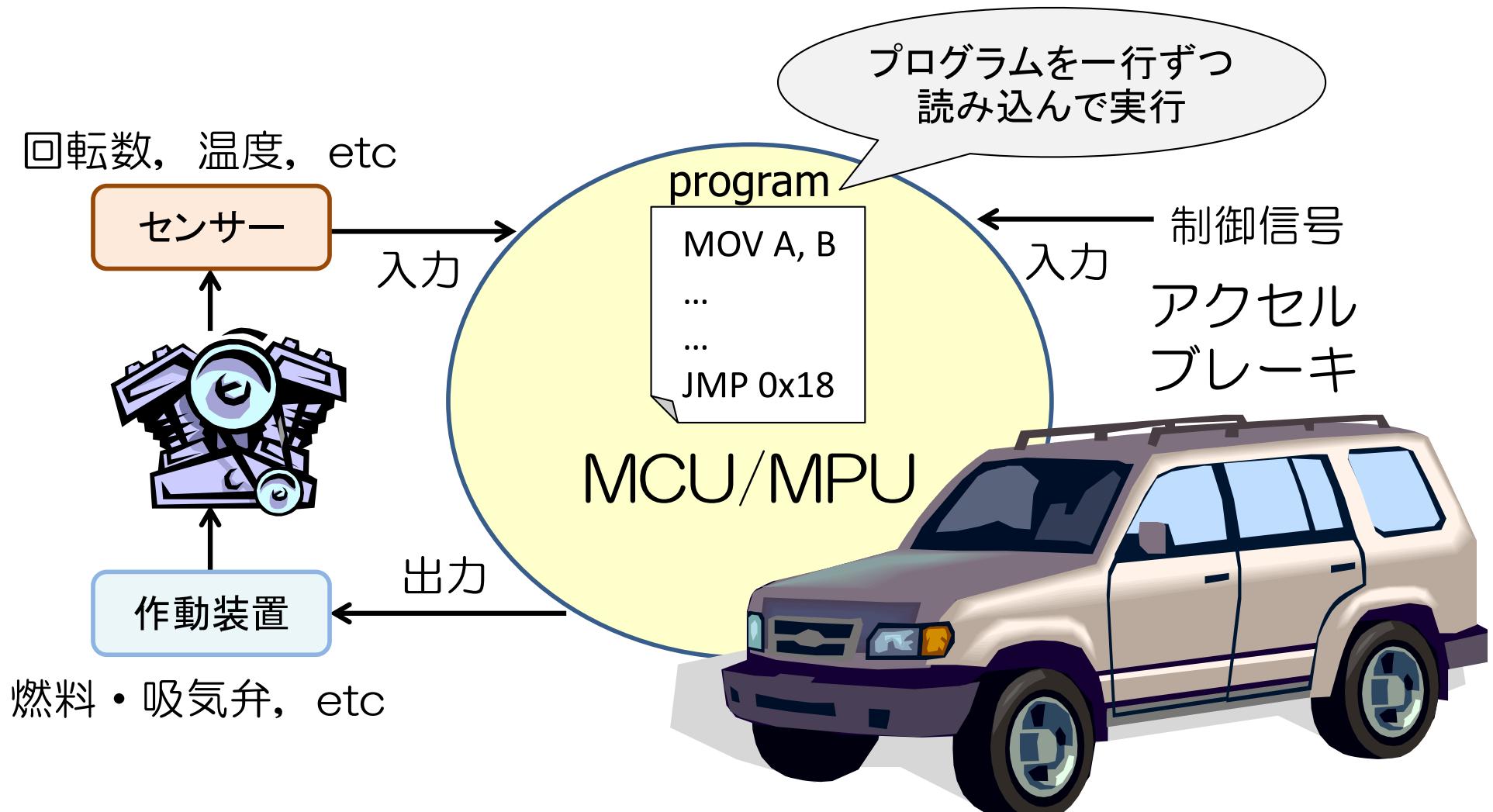
- ・マイコン世界シェア 27%
- ・車載マイコンシェア 43%

日本の貿易収入の約20%が自動車と半導体

日本の科学技術を支える技術者になつて欲しい！

マイコン制御の例

4ビットMCU(Micro Control Unit)が実習の題材

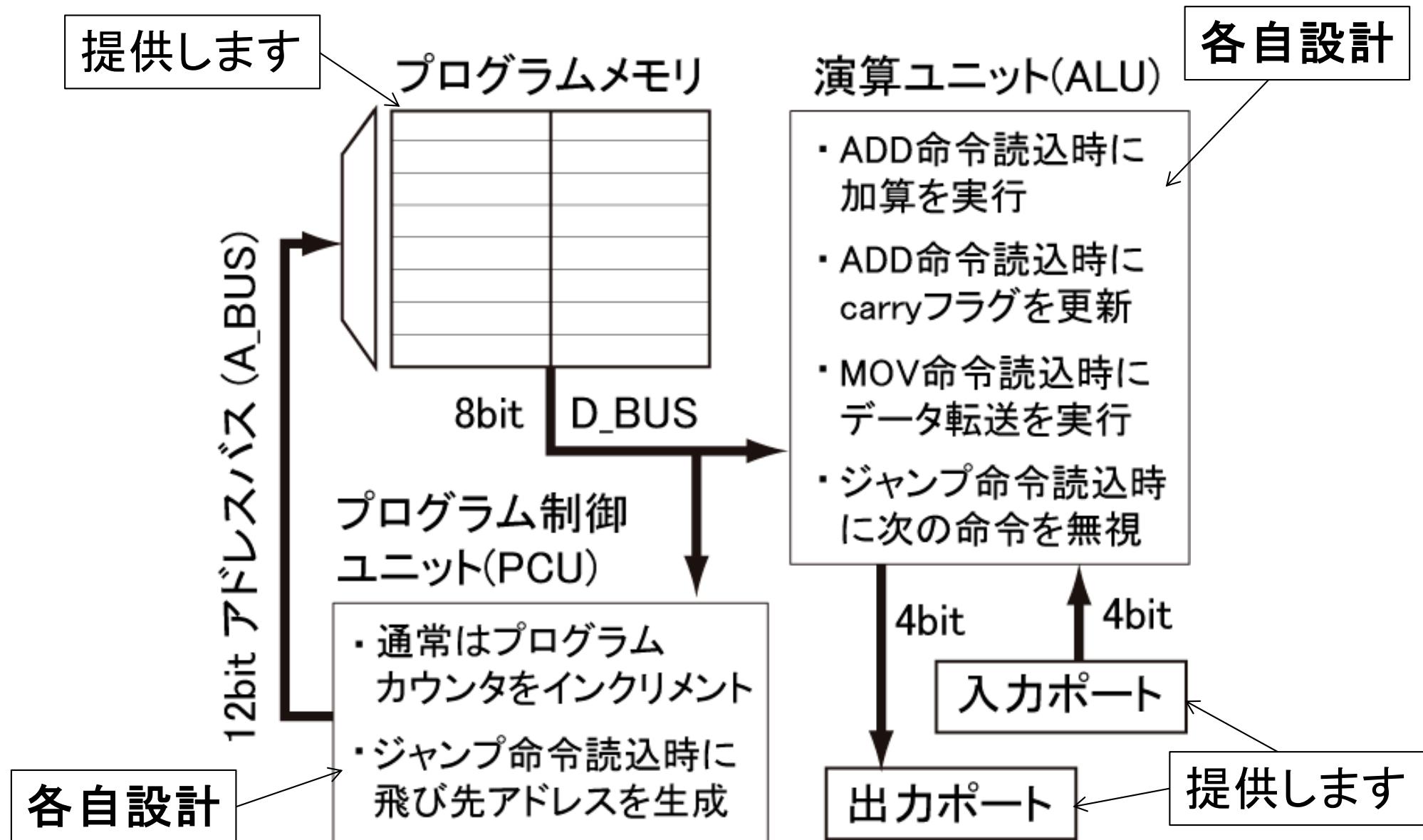


4ビットマイコンの命令セット

code	命令	機能の説明	データフロー
0000	MOV A,Imm	Aレジスタに即値(Imm)を転送	$A \leftarrow Imm$
0001	ADD A,Imm	AレジスタにImmを累積、Carryフラグを更新	$A \leftarrow A + Imm, \text{set } C$
0010	MOV A,B	AレジスタにBレジスタの値を転送	$A \leftarrow B$
0011	MOV A,IN	Aレジスタに入力ポートの値を転送	$A \leftarrow \text{入力ポート}$
0100	MOV B,Imm	Bレジスタに即値(Imm)を転送	$B \leftarrow Imm$
0101	ADD B,Imm	BレジスタにImmを累積、Carryフラグを更新	$B \leftarrow B + Imm, \text{set } C$
0110	MOV B,A	BレジスタにAレジスタの値を転送	$B \leftarrow A$
0111	MOV B,IN	Bレジスタに入力ポートの値を転送	$B \leftarrow \text{入力ポート}$
1000	MOV OUT,Imm	出力ポートに即値(Imm)を転送	$\text{出力ポート} \leftarrow Imm$
1001	MOV OUT,B	出力ポートにBレジスタの値を転送	$\text{出力ポート} \leftarrow B$
1010	MOV B,GPR	GPR[Imm]の値をBレジスタに転送	$B \leftarrow \text{GPR}[Imm]$
1011	MOV GPR,B	Bレジスタの値をGPR[Imm]に転送	$\text{GPR}[Imm] \leftarrow B$
1110	JNC Imm	Carryフラグが0のとき、即値(Imm)の示す番地にジャンプ	If $C=0$, $PC \leftarrow Imm$ If $C=1$, No operation
1111	JMP Imm	即値(Imm)の示す番地ヘジャンプ	$PC \leftarrow Imm$

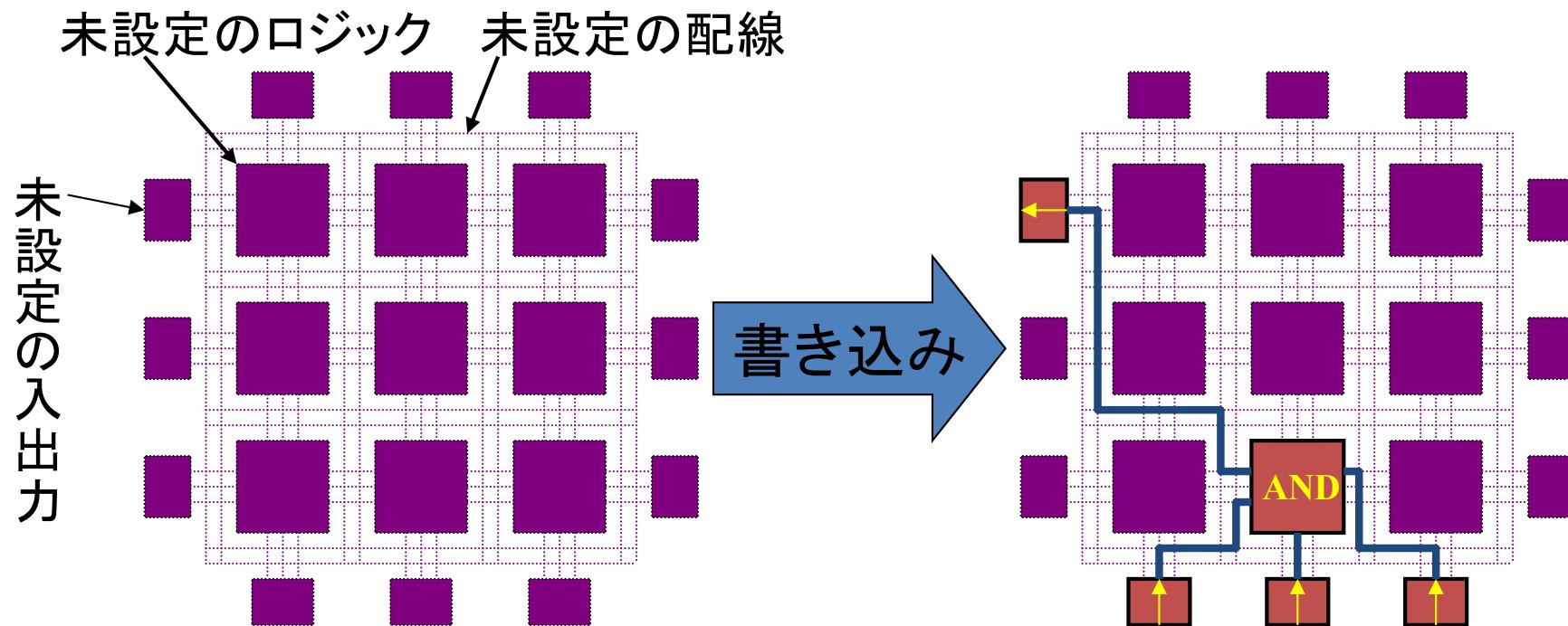
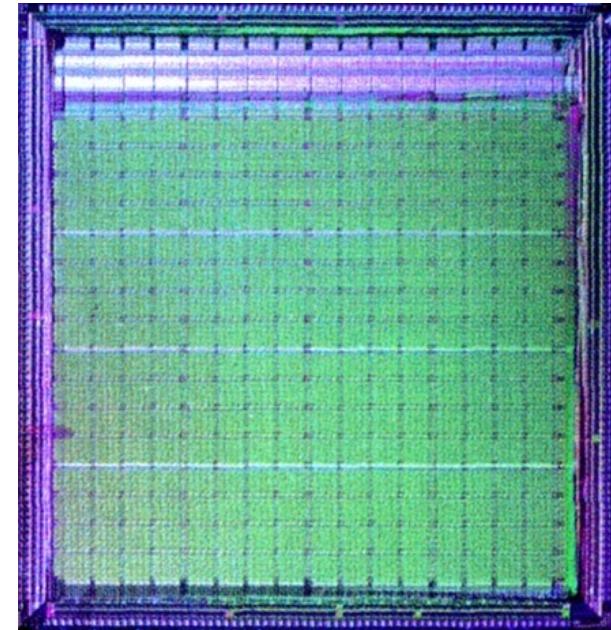
※ Imm は即値データ

4ビットマイコンの全体像



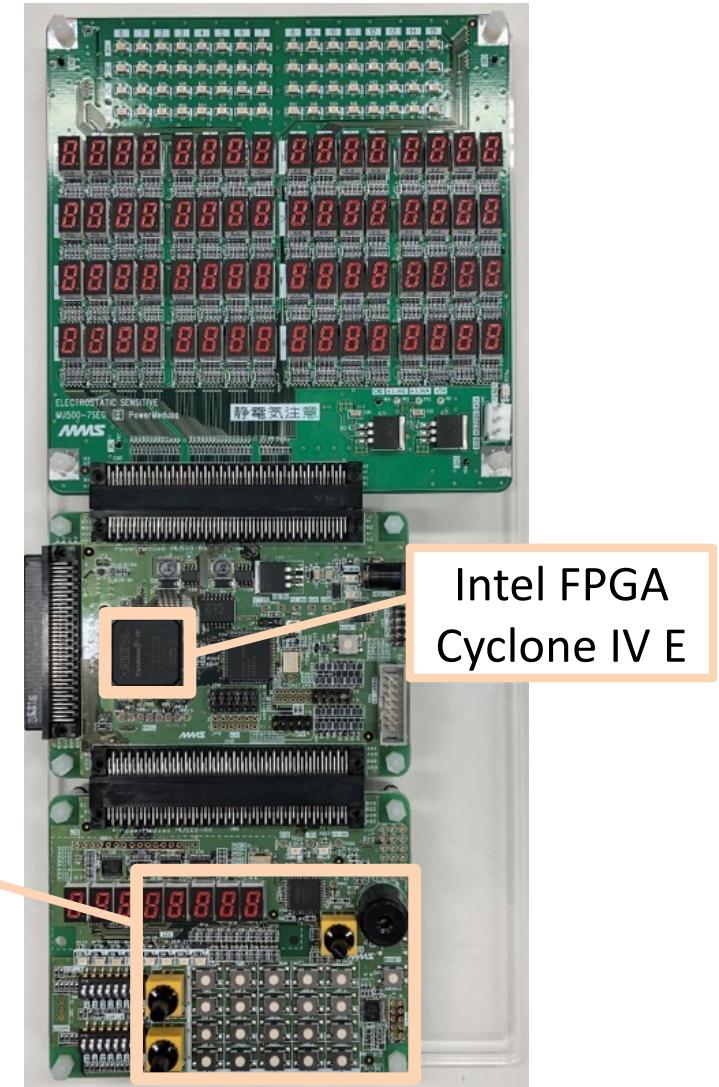
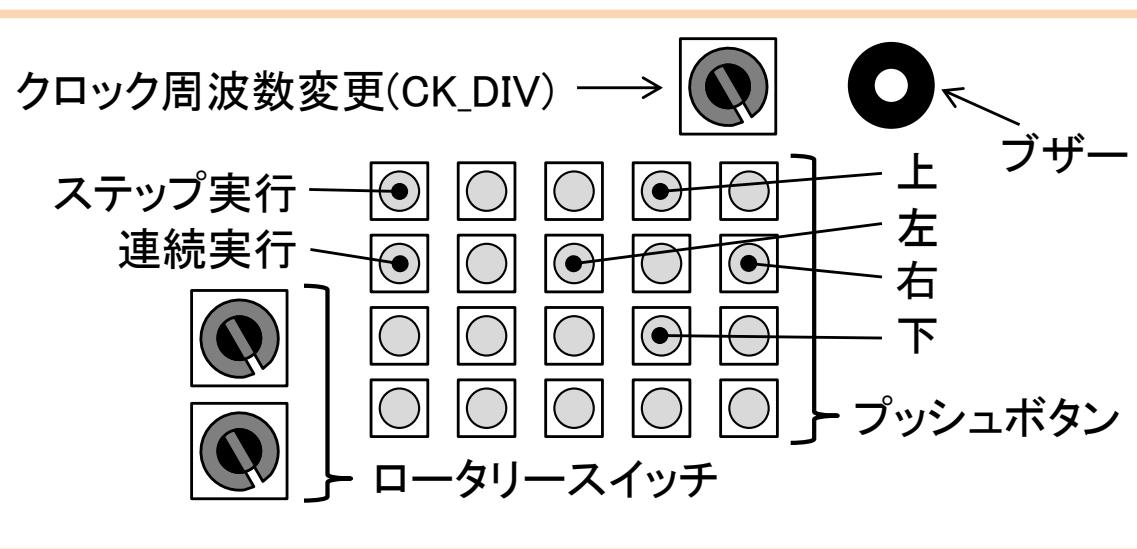
FPGA

- FPGAは、「書き込み」を行うことで任意のデジタル回路を実現できる既成のLSIです

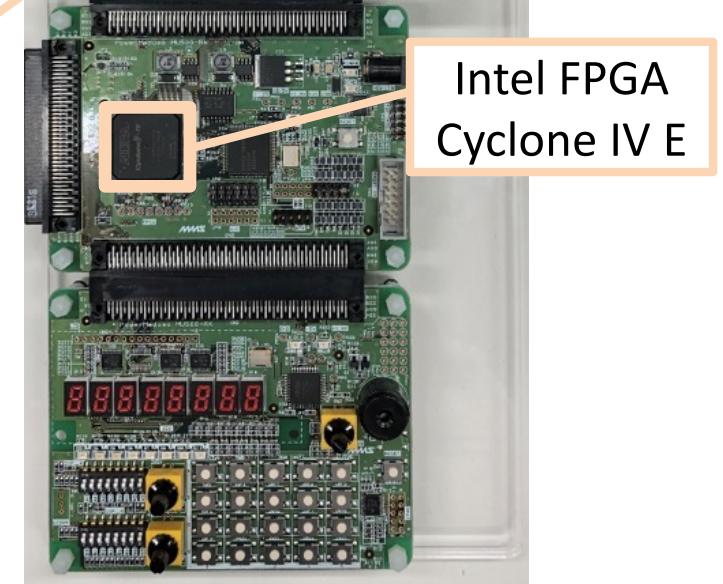
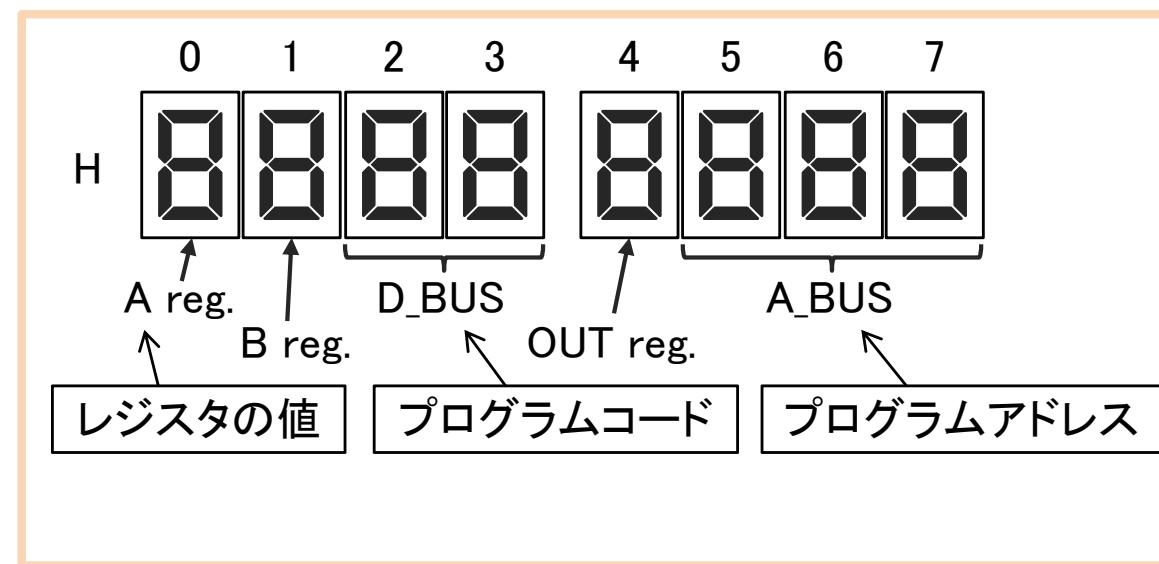
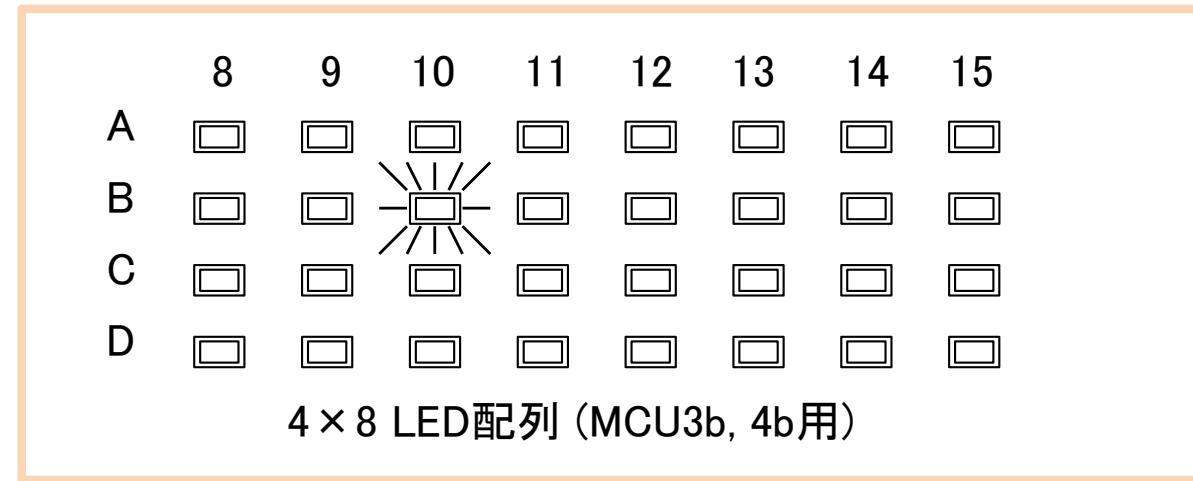


ボード上のスイッチ、ブザー

- 入力に応じてLED、ブザー出力が変化する
- 様々なプログラムを走らせて動作検証



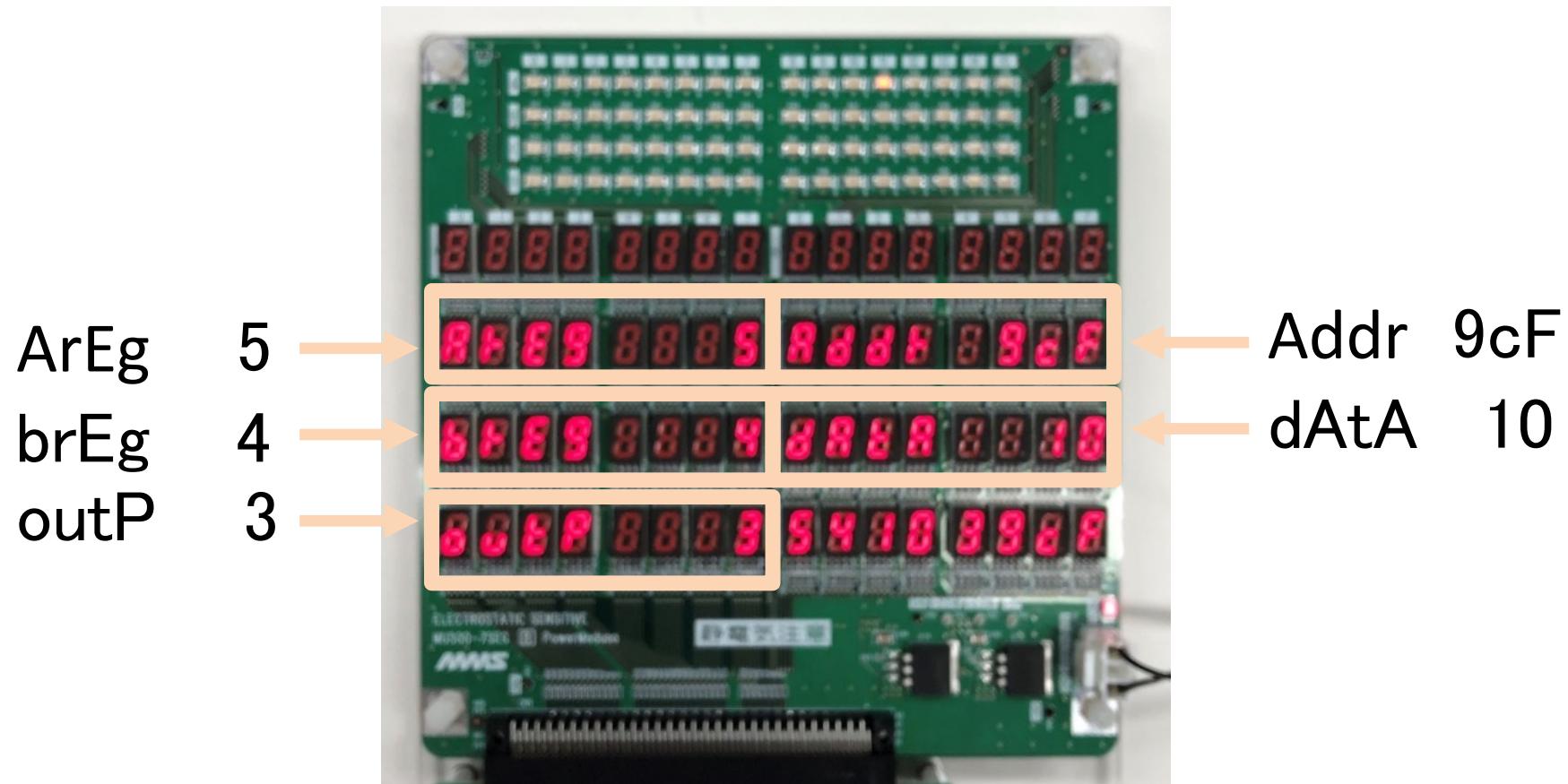
ボード上の7セグメントLEDの表示



※ ボード詳細 → 配付資料“FPGAボードマニュアル”を参照 14

7セグメントLED デバッグ用表示

- 各レジスタ、プログラムデータ・アドレスを確認できる
- デバッグ用に活用してください



ツールとボードの使い方

実験6.1～実験6.3

各種提供ファイルはPandAより取得できます。

仮想端末サービスへの接続

- PandAにログイン
<https://panda.ecs.kyoto-u.ac.jp/portal/>
- PandA → 電気電子工学実習 → VDIを選択
- 電気電子工学実習を選択し、VDIを起動
- ECS-IDを使ってログイン
- PandA → 電気電子工学実習 → リソース
 - 論理回路設計演習のフォルダを開く
- lcf.zip をダウンロードし、デスクトップに展開

シミュレーションの流れ

- 教科書の実験6.1の説明通り実施
- lcf¥register¥ で作業する
- シミュレータ(ModelSim)のパスを設定(図8)
- Quartus Primeに回路図ファイル(4.1節参照)を読み込ませ、回路を確認する(図7)
- コンパイル後の回路図を表示させ、期待した回路図と一致することを確認する(図9)
- Quartus Primeでシミュレーションし、設計が正しいことを確認する(図11)
 - load信号が“1”の時のみdの値を取り込む

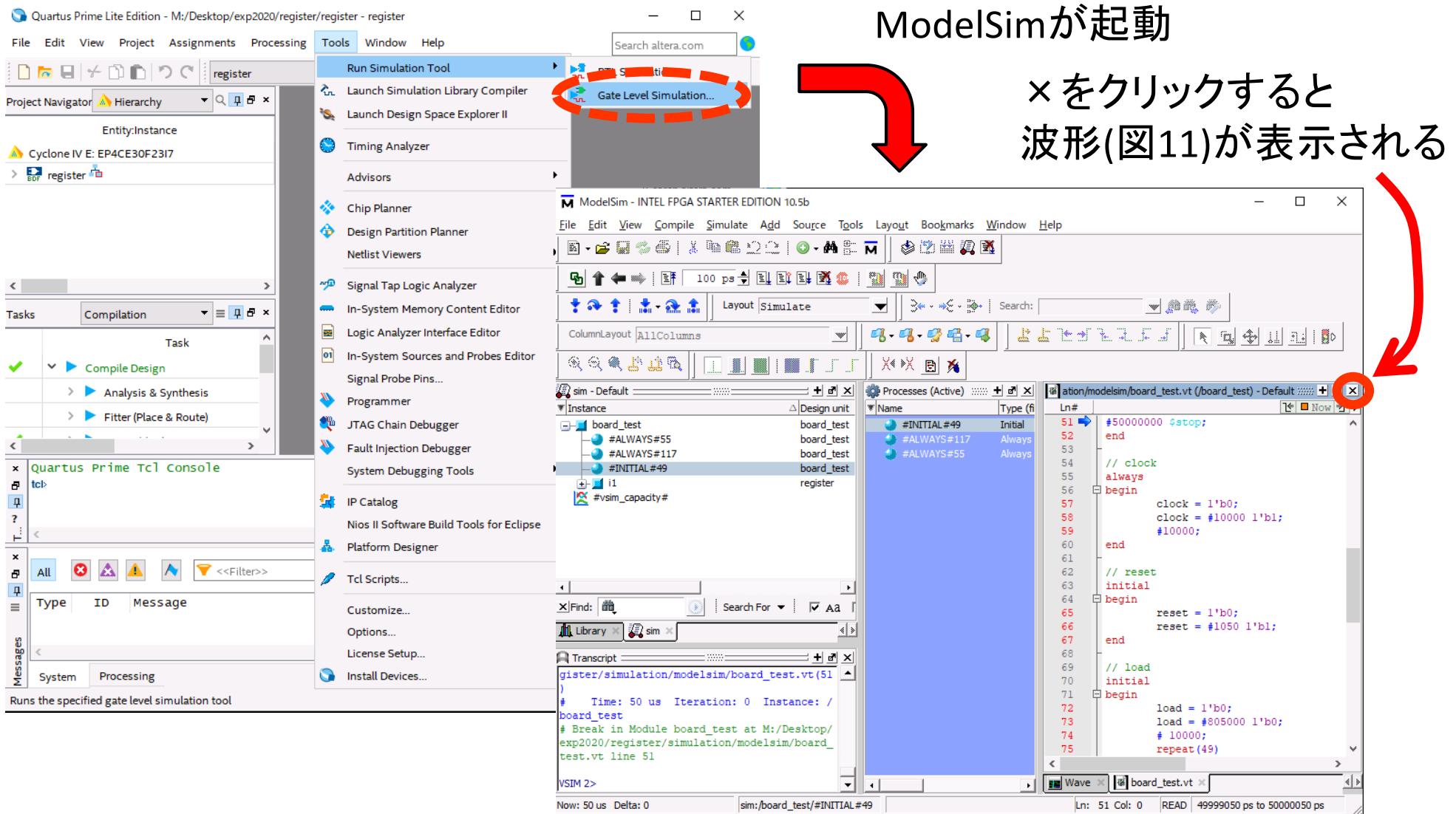
図8で画面下部の「OK」ボタンが押せない場合

- VDIのウィンドウを全画面にする
- 画面の解像度を変える。解像度変更後はVDIのウィンドウを閉じて再起動
- Windowsのツールバーを左に持っていく(ドラッグすればできる)
- quartus2.ini ファイルを Homeディレクトリにおく



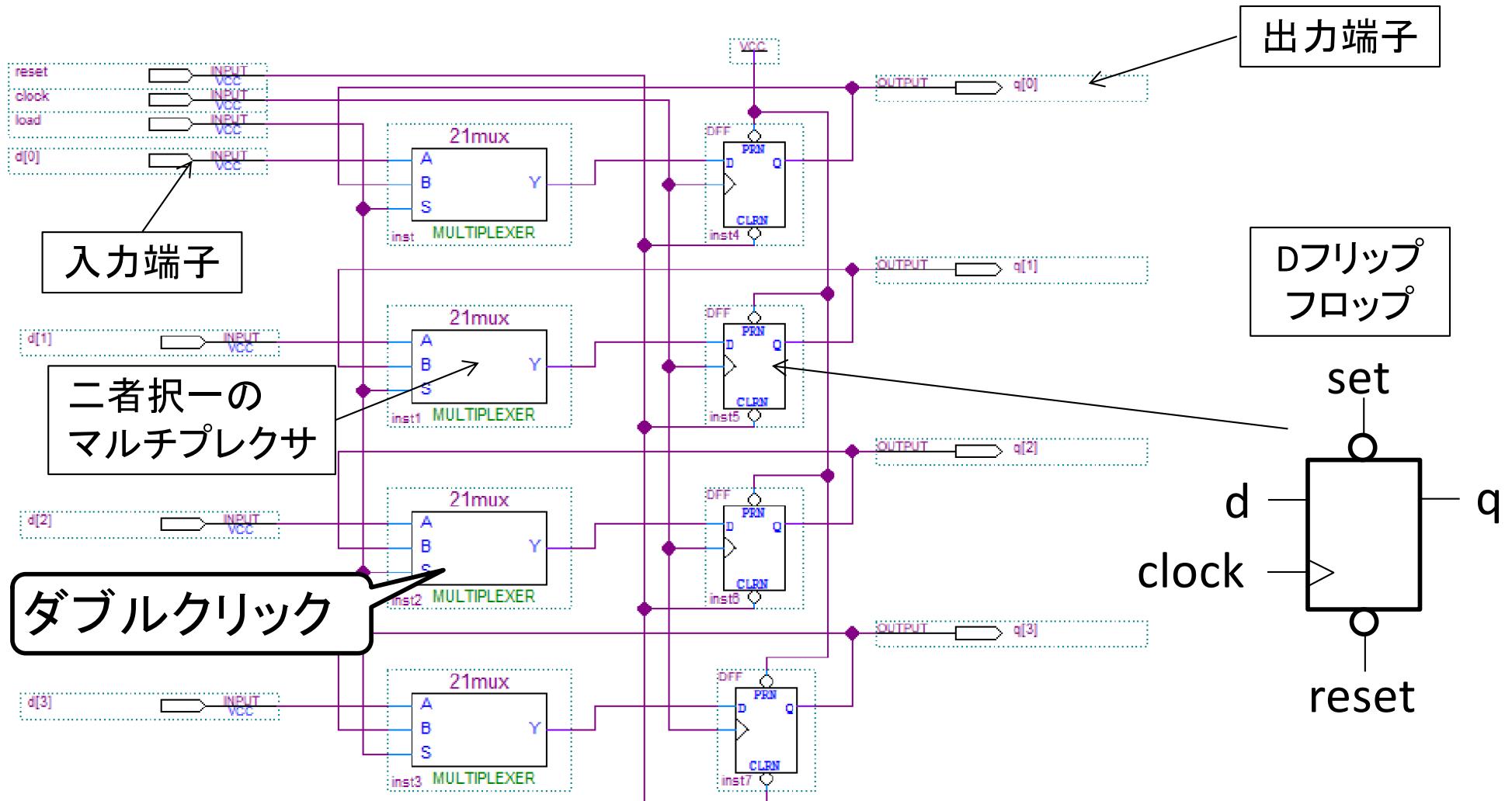
シミュレーション結果表示まで(図10)

Tools -> Run Simulation Tool -> Gate Level Simulation



ロード付き4ビットレジスタ

これより後の設計では各自下記のような回路図を描いてもらいます。



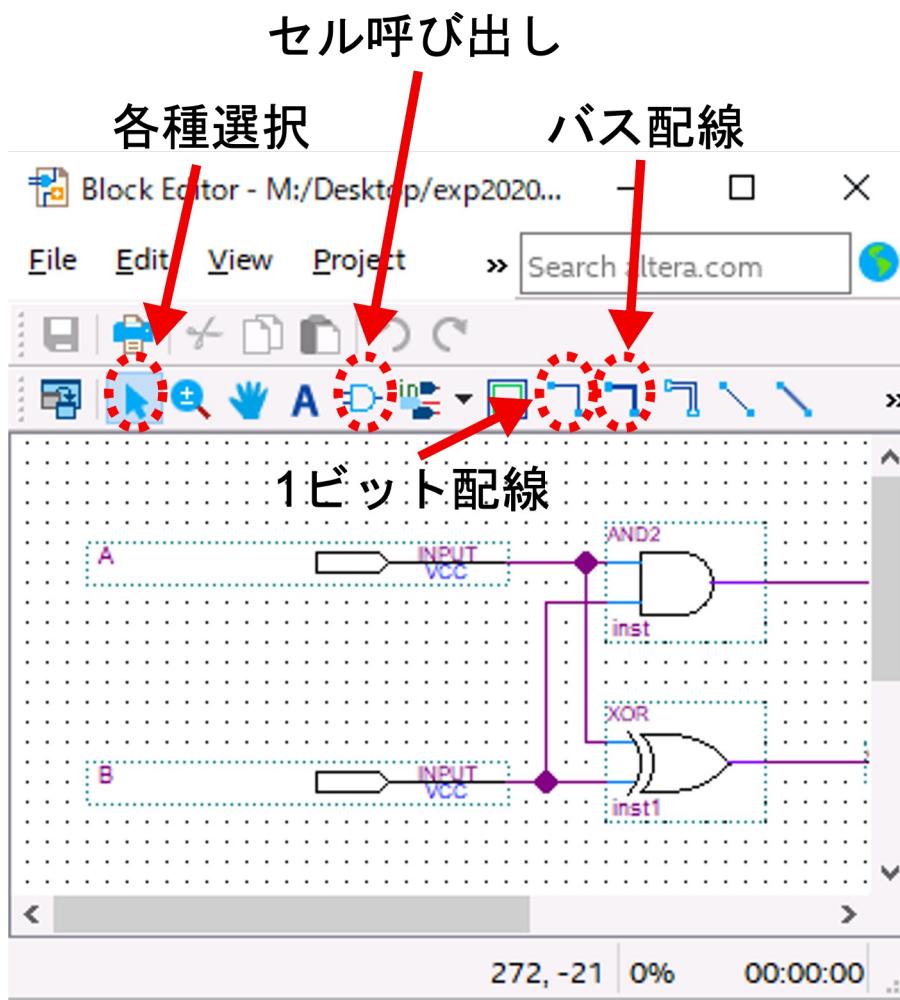
教員/TAにサポートを求める場合

- 短い質問であれば、席から教員/TAを、挙手で呼んでください。大声で呼ばない。
- サポートが長時間になると予想される場合は、教員/TAの机にくる
 - アクリル板越しに話してください。
 - 画面を見せるときは、アクリル板越しに見せるか、Zoomの画面共有を使います。
 - PandA → Zoom (KU Licence) → [2023後期金4]電気電子工学実習 6 論理回路設計演習

実験6.2： ロード付き12ビットカウンタの設計

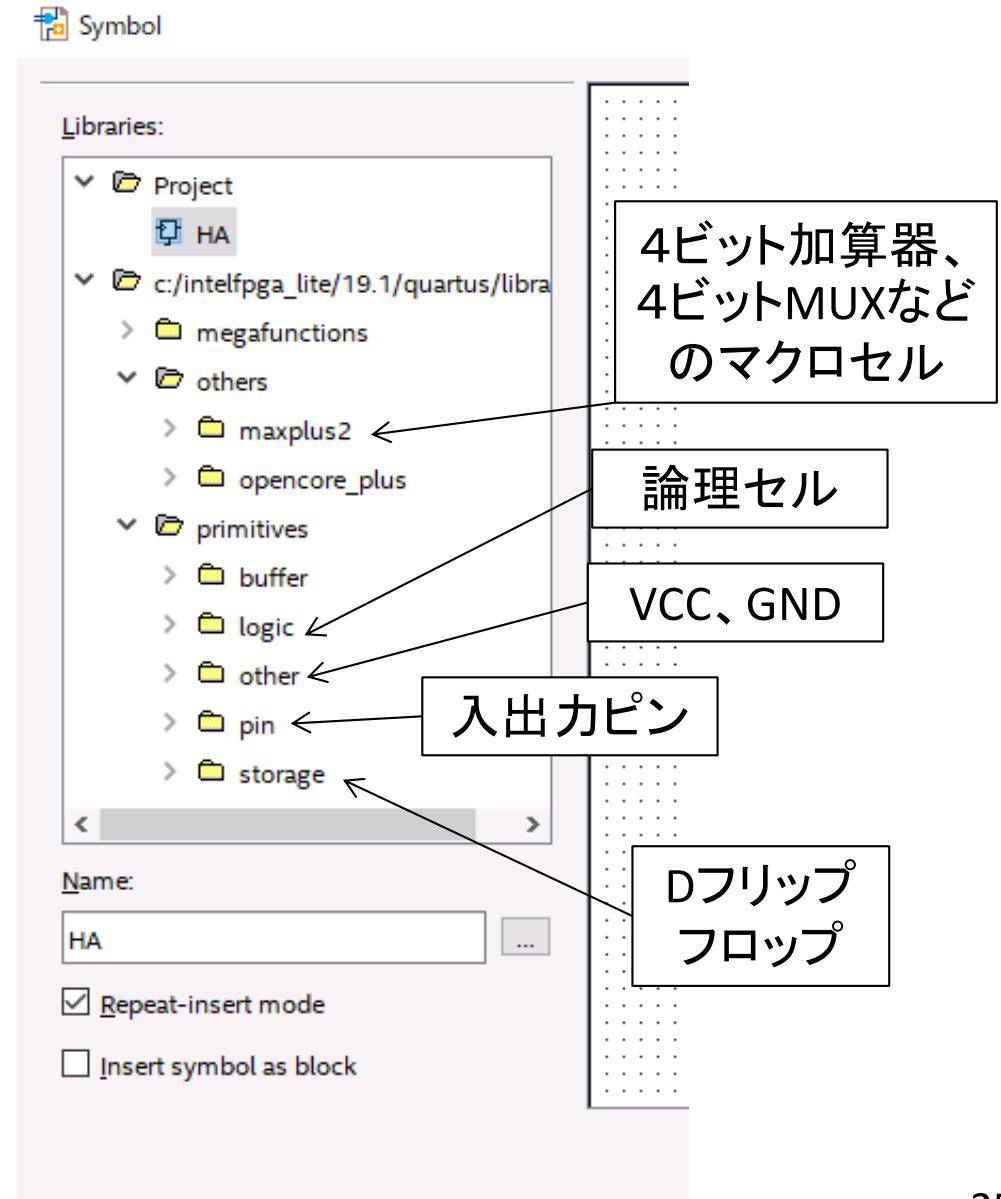
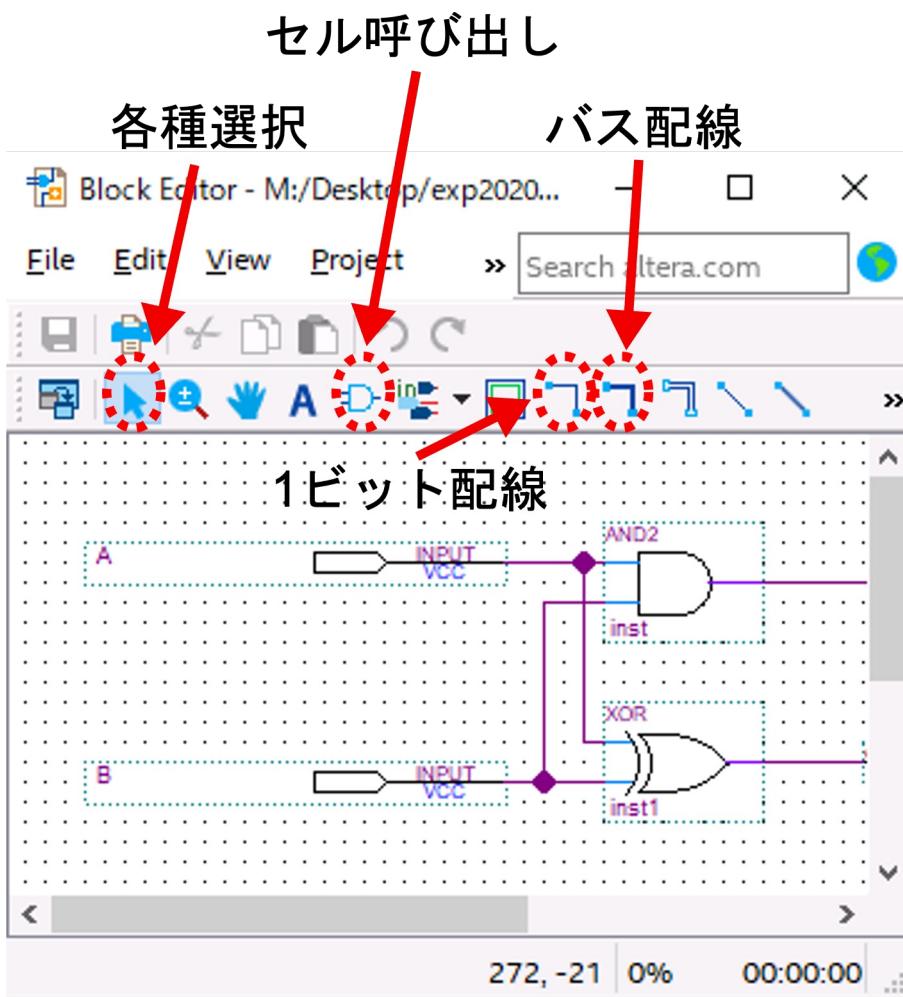
- 教科書の実験6.2の説明通り実施
- 回路図を各自で記述(図12を参考に書く)
- Quartus Primeで回路をコンパイルする
- 回路図を表示させ、期待した回路図と一致することを確認する
- Quartus Primeでシミュレーションし、設計が正しいことを確認する

回路図の書き方



- Quartus Primeの回路図エディタを使います
- 回路図の新規作成は File→Newの次にBlock Diagram/Schematic Fileを選択
- 既存の回路の編集は File→Openの次にファイル選択 ウィンドウの中から該当するファイルを選択
- 左図のコマンドアイコンを使って基本セル呼び出し、配線を行う

基本セルの呼び出し

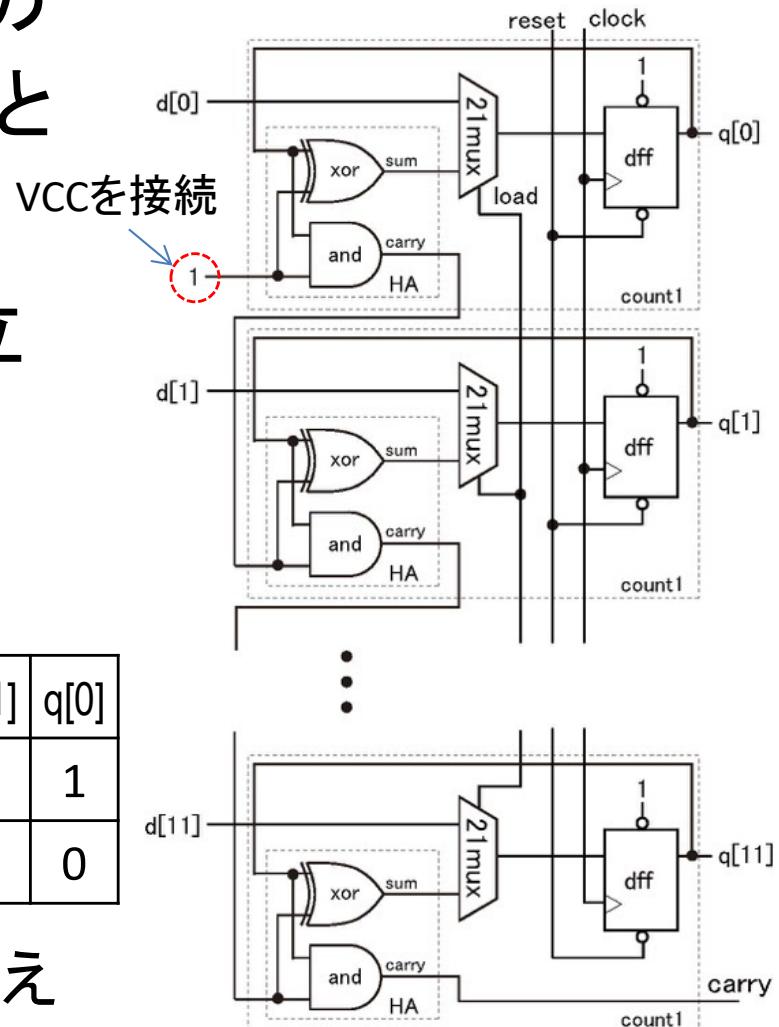


ロード付きカウンタとは(実験6.2)

- load信号が1のとき:d入力端子の値をフリップフロップに記憶すると同時にその値をq端子に出力
- load信号が0のとき:クロックが立ち上がるたびにカウントアップ
- カウントアップの例

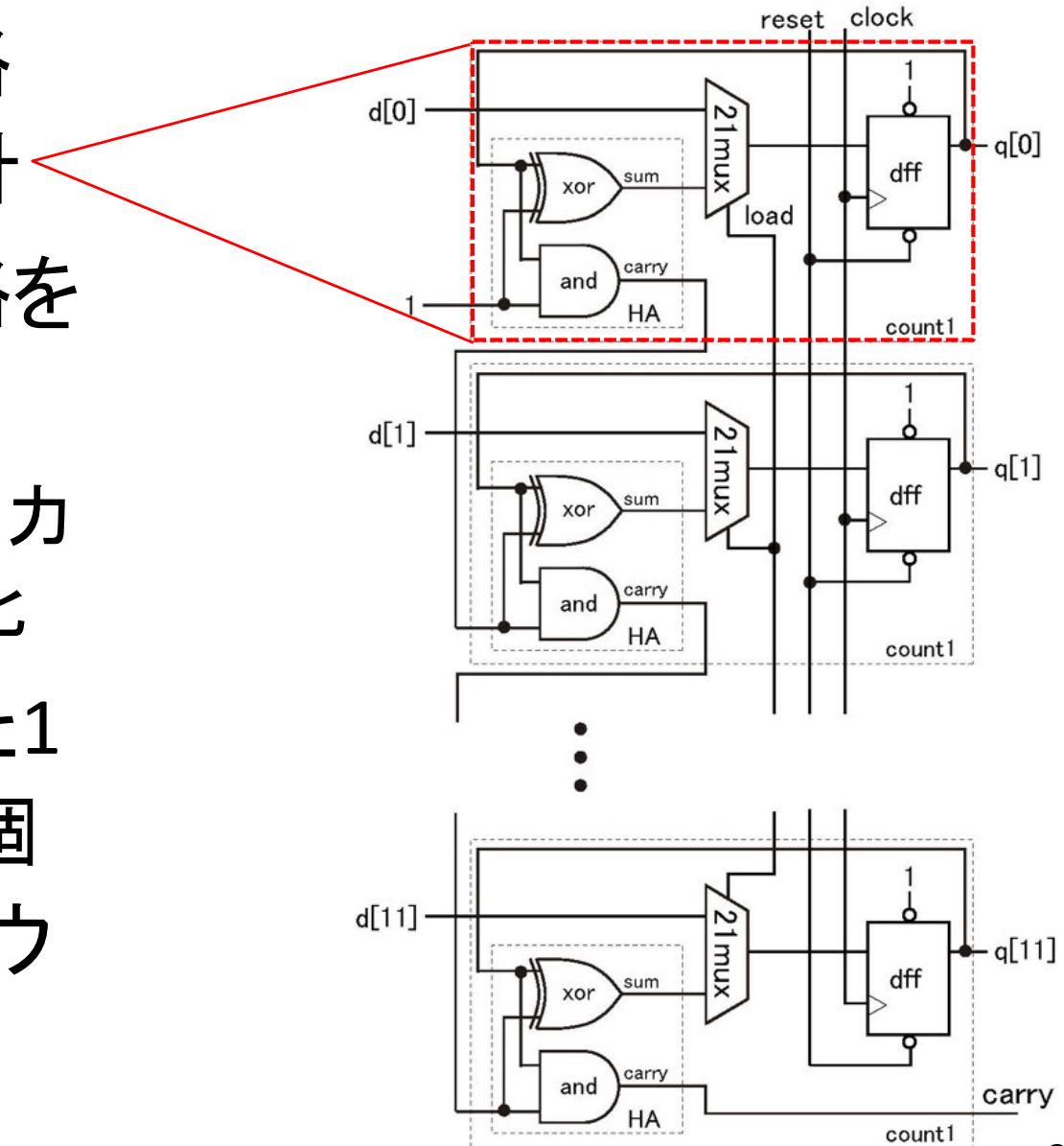
	q[11]	q[10]	q[9]	q[8]	q[7]	q[6]	q[5]	q[4]	q[3]	q[2]	q[1]	q[0]
現在の値	0	1	1	0	1	1	1	1	1	1	1	1
次の値	0	1	1	1	0	0	0	0	0	0	0	0

- 現在の値(2進数12ビット)に1を加える加算回路を設計すれば良い



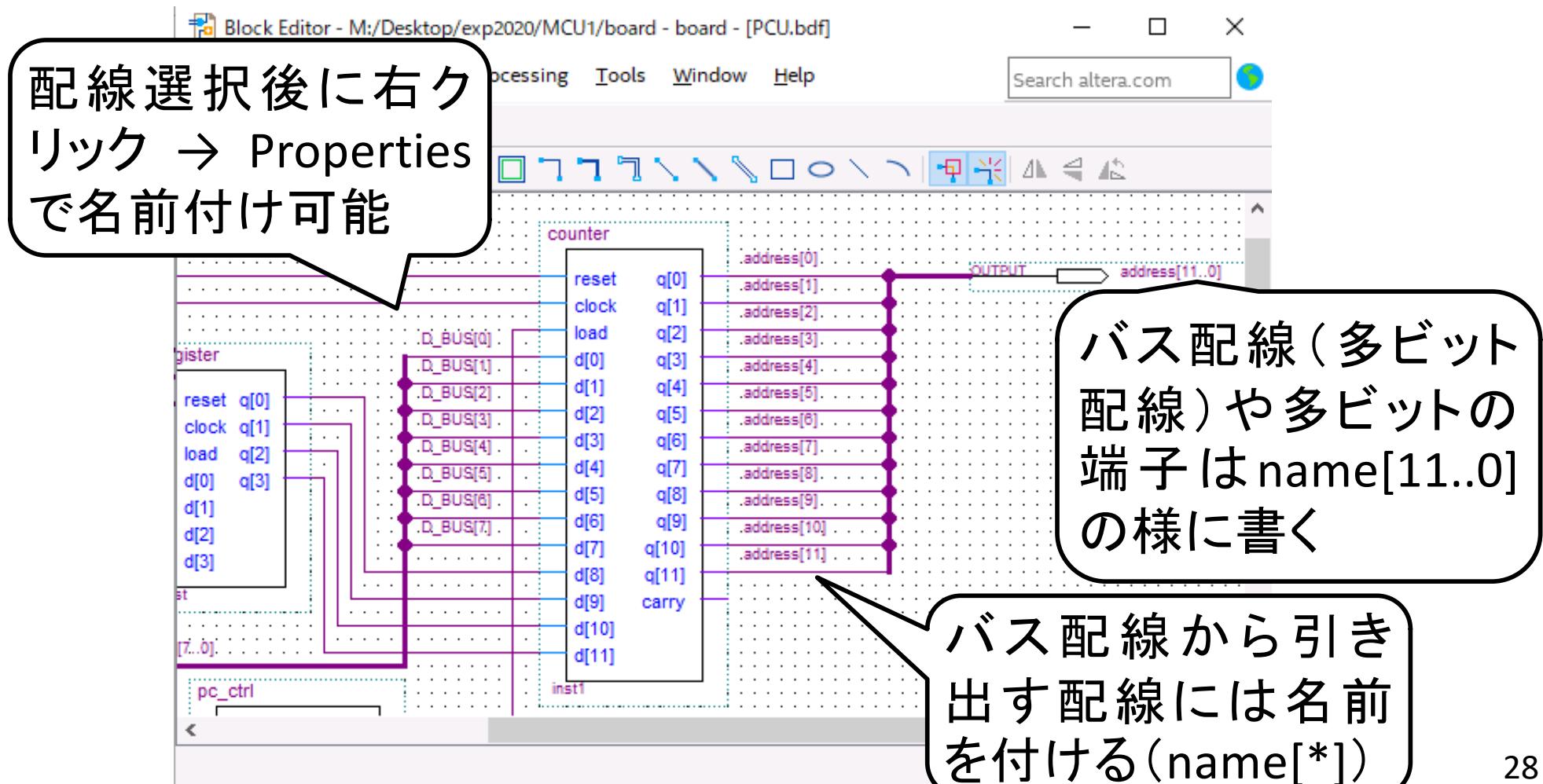
カウンタ設計手順(実験6.2)

- 1ビットカウンタ回路
(count1.bdf)を設計
- 1ビットカウンタ回路をモジュール化
 - モジュール化: 入出力ピンを付けて階層化
- モジュール化された1ビットカウンタを12個接続して12ビットカウンタを完成



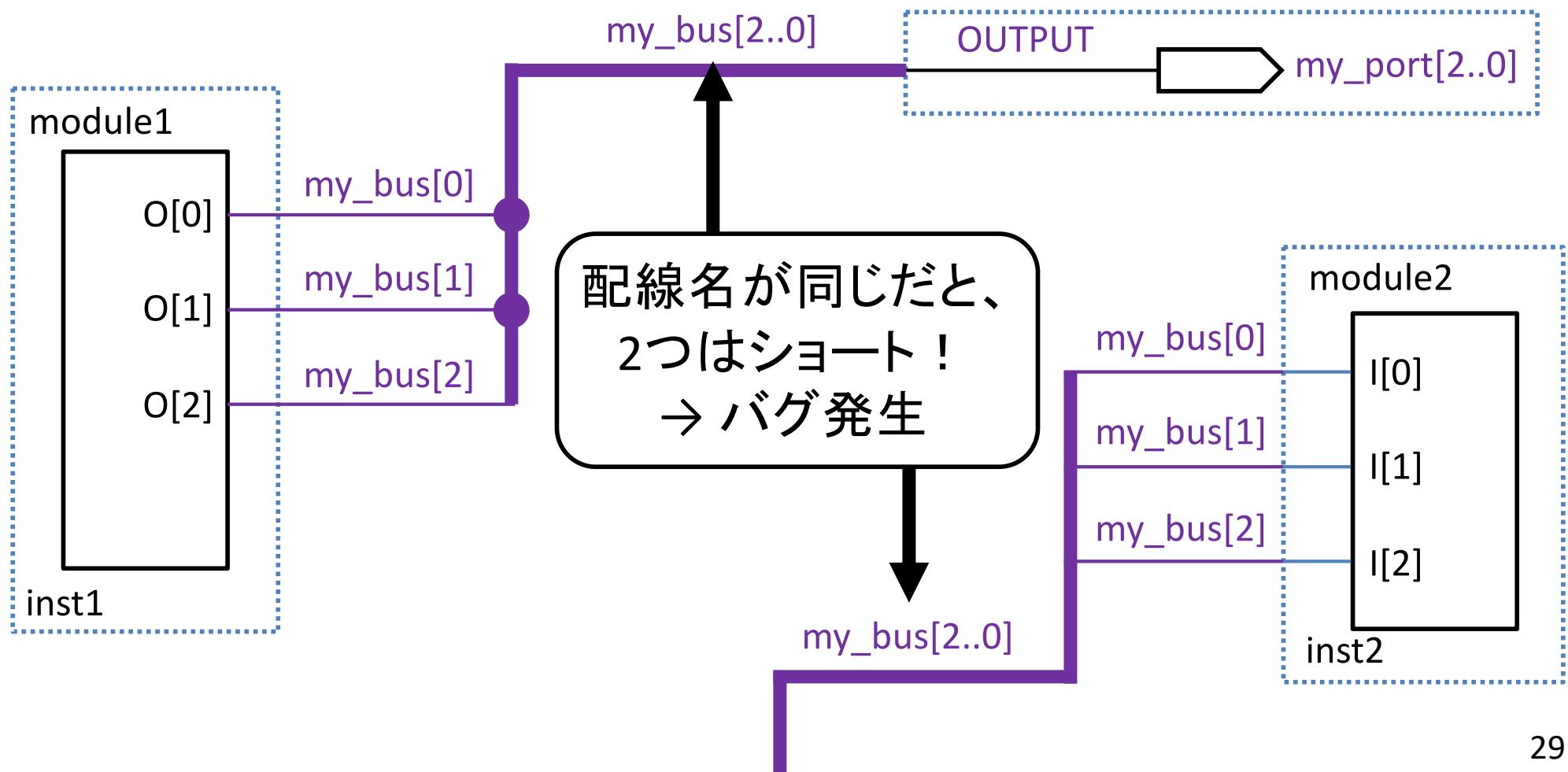
配線、バス配線の書き方

- ・ バス配線(多ビット線)から1ビットの配線を分離する場合は必ず配線に名前を付ける



バス配線の注意

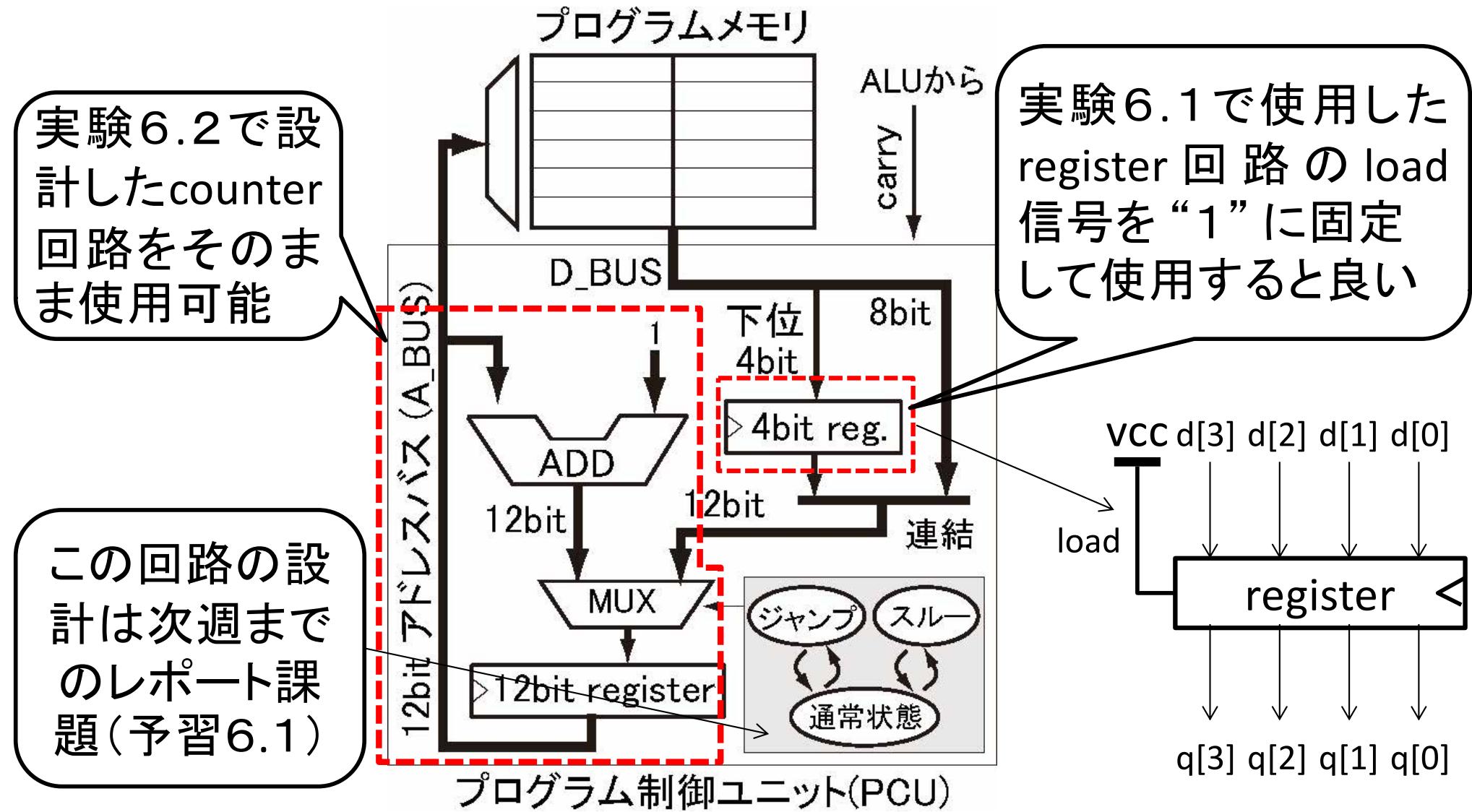
- 配線名が同じだと、ショートする！



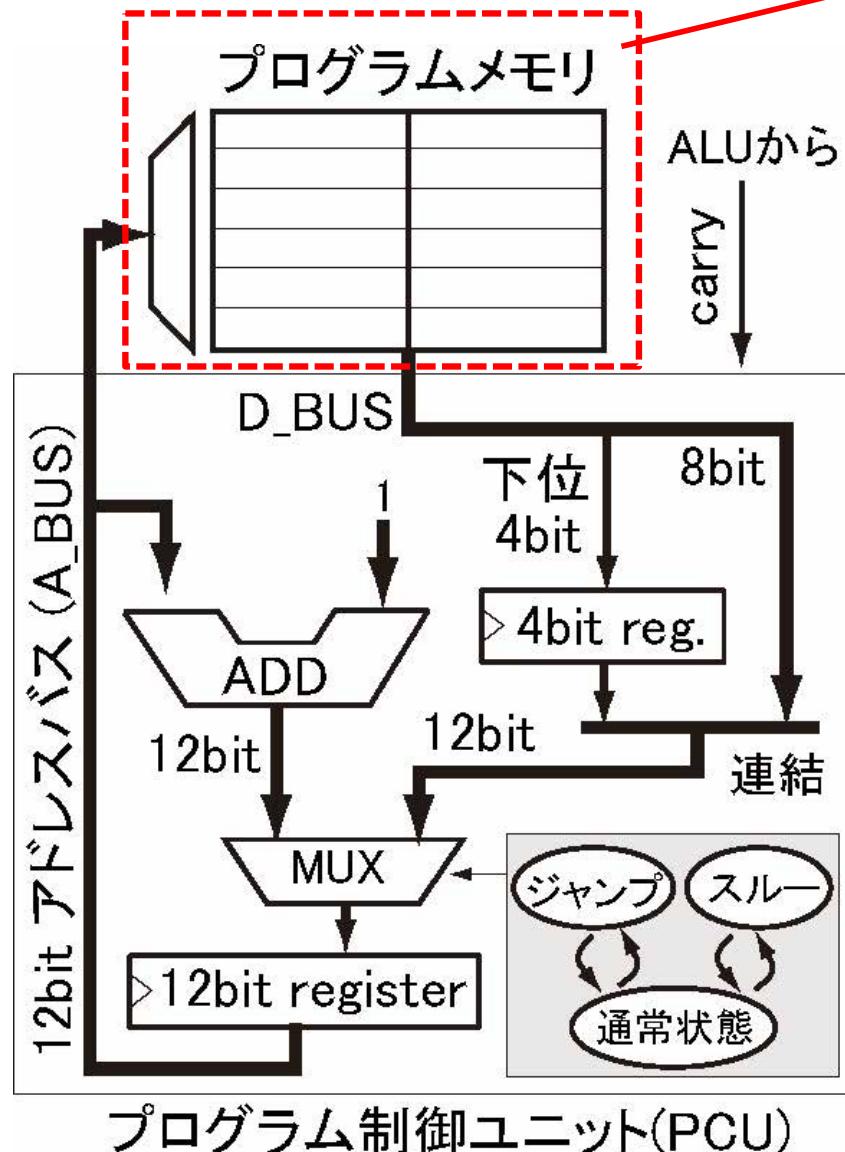
マイコンの設計(実験6.3)

- lcf\MCU1\ で作業する
- コピーしたMCU1内のファイルに実験6.1、6.2で記述した回路(register.bdf, counter.bdf, count1.bdf)を追加し、実験6.3の流れに沿ってマイコン(未完成)をFPGA用にコンパイルします
- board_sample1.tclファイルで以下の情報を与えます
 - シミュレータ(ModelSim)の環境設定
 - ターゲットのFPGAデバイスの品種、グレード
 - 入出力信号をそれぞれFPGAのどのピンに割り当てるか
(ボード上で既にLED等の外付け回路がFPGAのピンと結線されているので、FPGA内の配置配線をそれに合わせてやる必要がある)
 - ボード上の外付け回路とFPGAピンとの対応は、MU500-RXボードのマニュアルに明記されています

PCUの回路図(実験6.3)



プログラムメモリの実装(提供済み)



アドレス

000
001
002
003
004
005
006
007
008
009
00A
00B
:
0F
1E
2D
3C
4B
5A
69
78
87
96
A5
B4
C3
D2
E1

D_BUS[7..0]
(オペコード、即値)
がテキスト形式で
保存されている

MOV OUT,Imm (Imm=7)
と解釈される

JNC Imm (Imm=1)
と解釈される

テキストエディタ(メモ帳、Vim、Emacsなど)で rom_sample1.hex を見てみよう!

実験6.3用のテストプログラム

address code (16進数) ← rom_sample1.hex に記載

000 :	0F;
001 :	1E;
002 :	2D;
003 :	3C;
004 :	4B;
005 :	5A;
006 :	69;
007 :	78;
008 :	87;
009 :	96;
00A :	A5;
00B :	B4;
00C :	C3;
00D :	D2;
00E :	E1;
00F :	F0;
.....



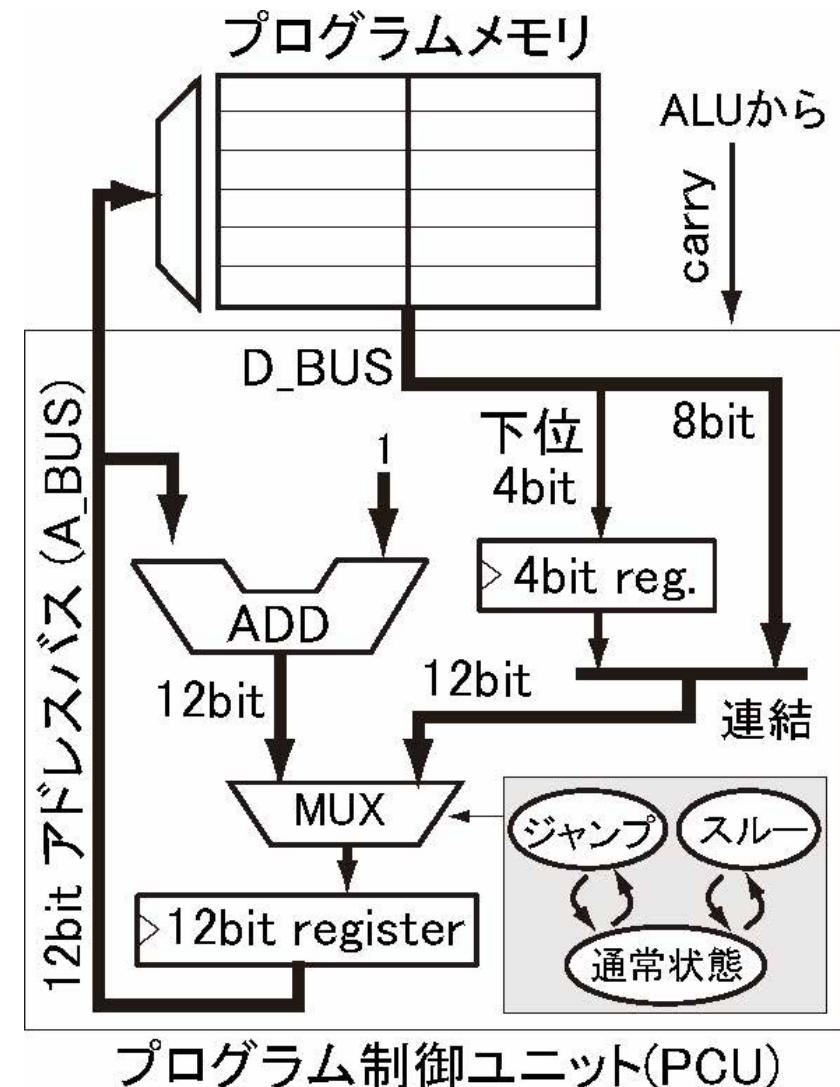
回路が正しく設計できていれば、アドレス値
が1ずつインクリメントされ左図のコードが順
にROMから読み出されるのが確認できる

ボードへのダウンロードの流れ

- 教科書の実験6.3の説明通り実施
 - 実験6.1、6.2同様、Quartus Primeでコンパイル、回路図確認、シミュレーションを実行する
 - output_files¥board.sof をGoogle Driveにコピー
 - PandA >> 配布物・テキスト >> 6 論理回路設計演習 >> FPGA転送用Google Drive(<https://drive.google.com/drive/folders/1OA6ckrdNDt8WfAJ3xLo9p82YdYSJB9VD>)
 - FPGAノートパソコンでGoogle Driveに接続し、board.sof を取得
 - FPGAボードの電源を入れ、備え付けPCとFPGAボードをUSBケーブルで接続する
 - FPGAに回路を書き込み、動作を確認 (終わったら .sof を削除 !)
- ※ 未完成のMCUなので、実験6.3の時点では プログラムカウンタのカウントアップ動作 以外の動作はしないことに注意する

予習6.1(=サブレポート課題)

- 状態遷移図の作成
 - 教科書の表2を参照
 - 通常状態とジャンプ状態を遷移する状態機械
- カルノー図の作成
- 回路図の作成
 - not, nand, and, nor, or, xnor, xor, dffなどの基本セルを使用可



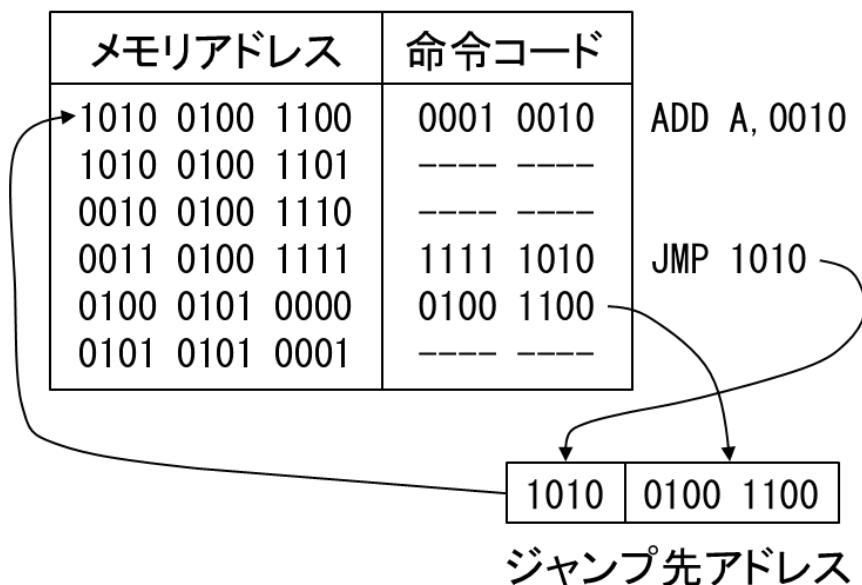
課題の詳細はテキストで確認のこと

JNC:Jump No Carry

キャリーが立っていないときのみジャンプするという意味

ジャンプ先アドレスの生成

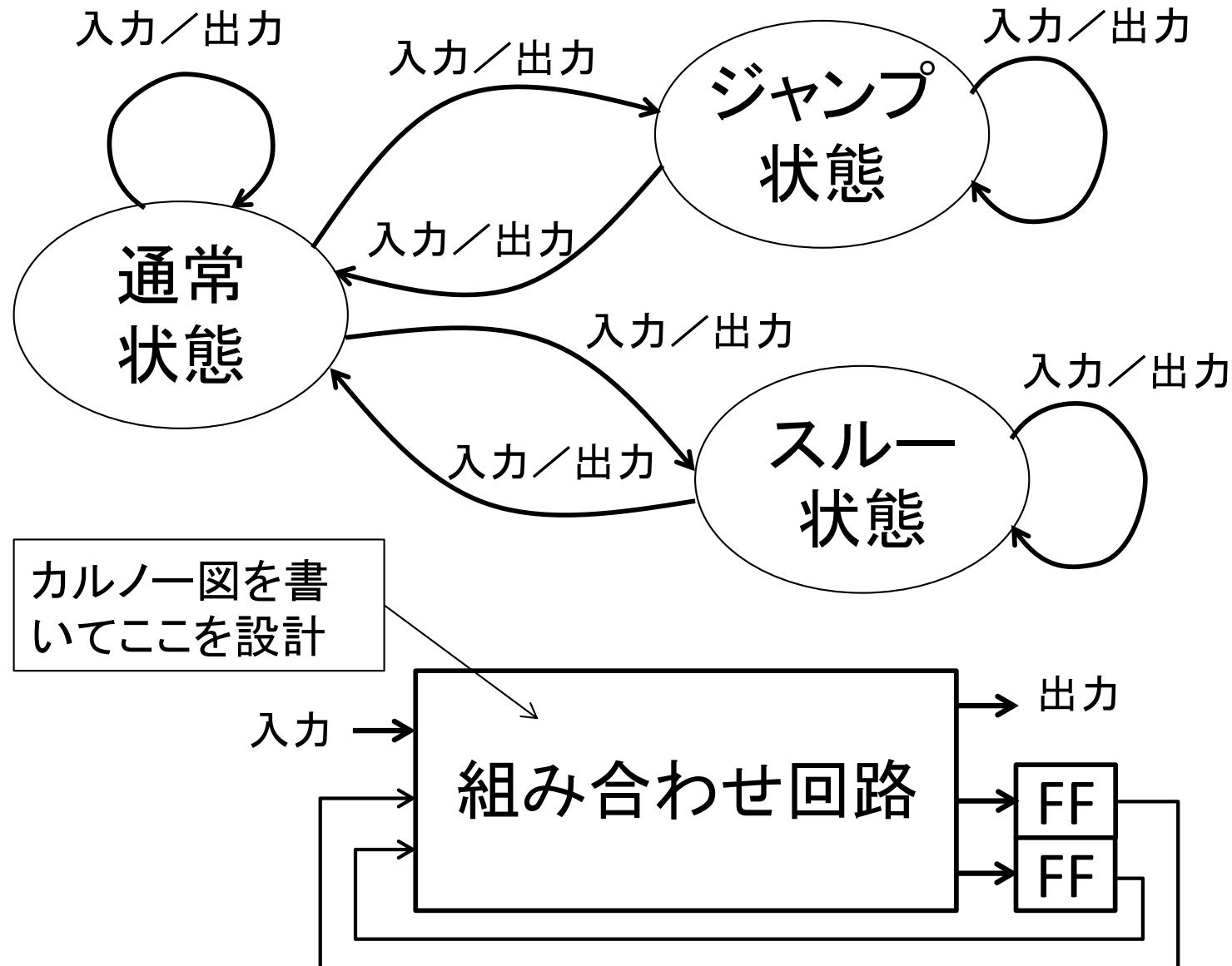
現在の状態	入力	次状態	動作
通常状態	(JMP)または(JNCかつcarry=0)	ジャンプ状態	PC ← PC + 1
	JNCかつcarry=1	スルー状態	
	上記(ジャンプ命令)以外	通常状態	
ジャンプ状態	任意	通常状態	命令は実行しない PC ← 飛び先アドレス
スルー状態	任意	通常状態	命令は実行しない PC ← PC + 1



ジャンプ命令(JMPまたはJNC)の次のアドレスの8ビットコードは必ずアドレスであることに注意する

つまり、JNCかつcarry=1となった時の次の8ビットコードは命令として扱ってはいけない(スルー)

状態遷移図と回路設計

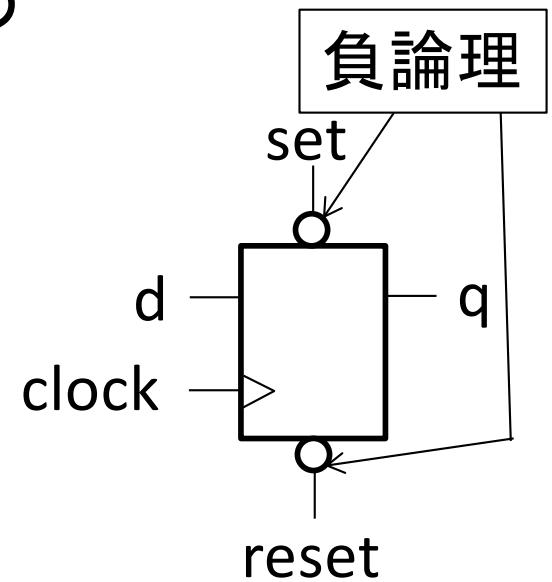


サブレポート(1週目)

- 制御部(pc_ctrl)の状態遷移図の作成
 - 各状態とその入力は何か, その時の出力は何か
- カルノー図の作成
- 基本セルを使って回路図を記述
 - not, nand, and, nor, or, xnor, xor, dff などの基本セルを使用可
 - dff (d, clock, reset, set, q)
- 考察 (回路の解説や工夫した点の説明)

サブレポートを PDF 化してPandAへ提出
lcf を zip 化してPandAへ提出

使わない場合は vcc を接続



第2週

プログラム制御ユニットの設計

実験6.4

本日の課題

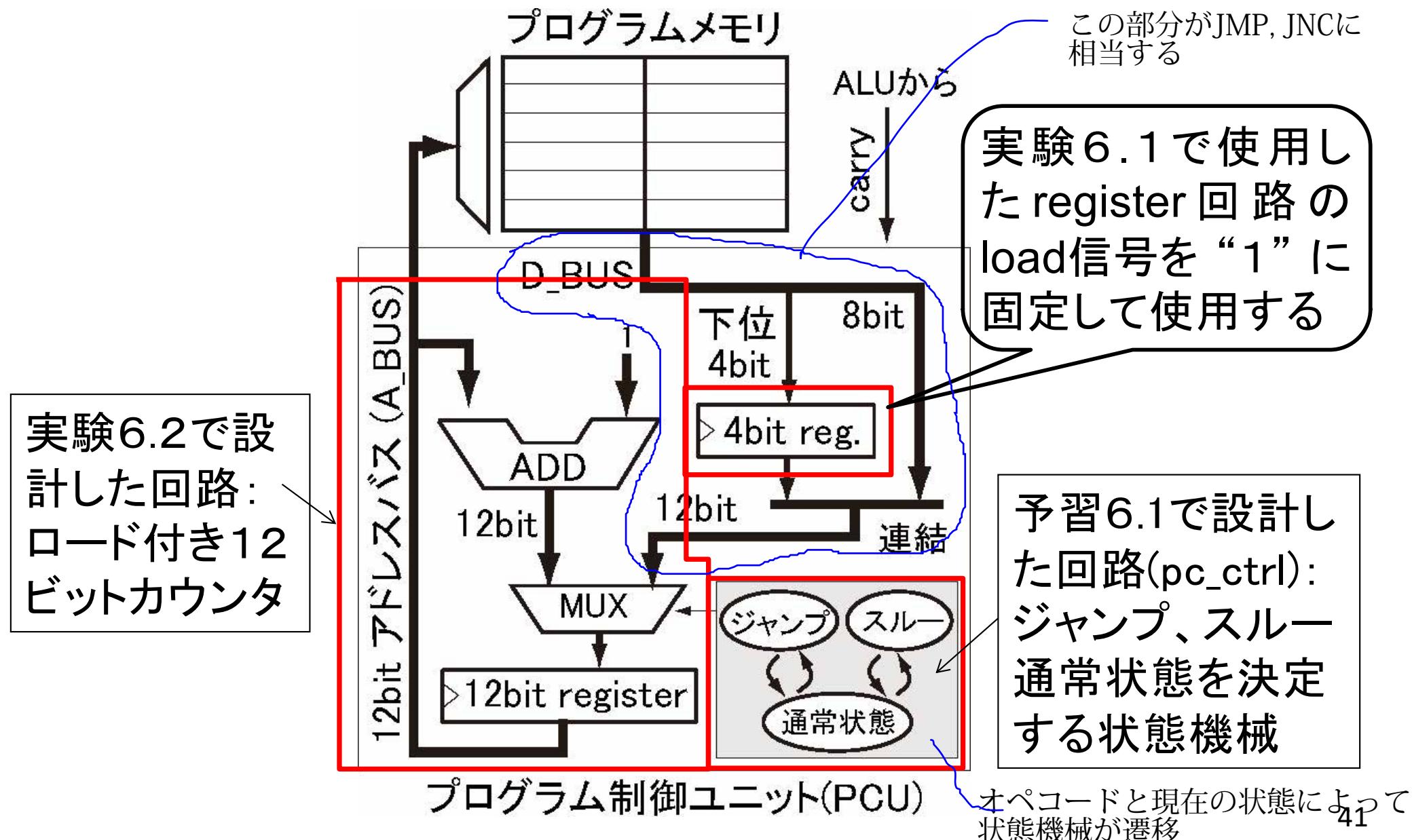
1. 実験6.4の実施

- プログラム制御ユニットを完成させ動作検証
 - ステップ実行させ、アドレスとコードを記録→レポート課題
- MCU2のフォルダを使用
 - プロジェクト名は「board」. この後の実験も.

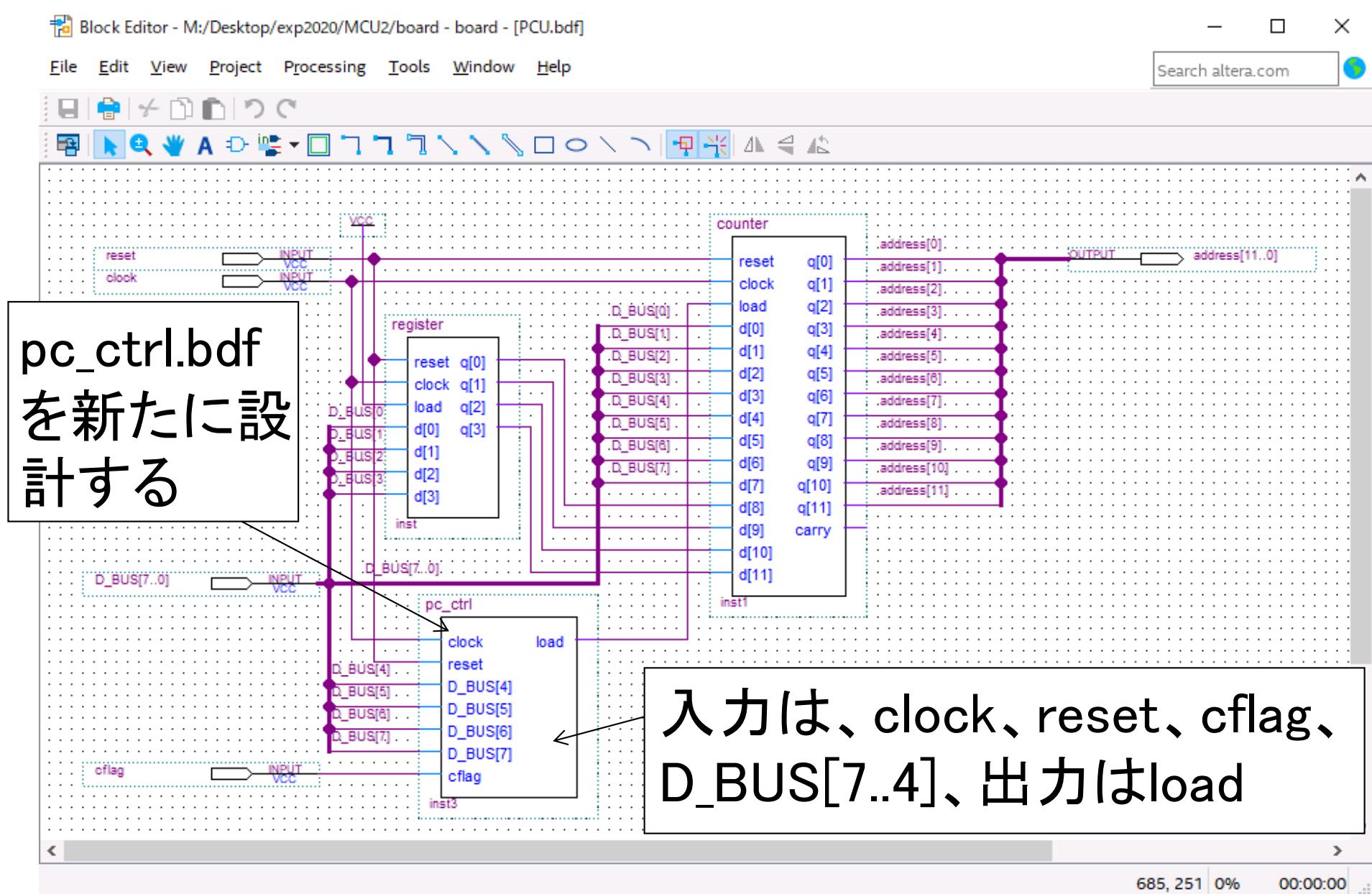
2. 予習6.2の実施(できるところまで)

- 演算ユニット(ALU)の机上設計 → レポート課題
- 制御部(alu_ctrl.bdf)とデータパス部を分けて設計
 - データパス部は本日中に完成させる
 - 制御部(alu_ctrl.bdf)はレポート課題

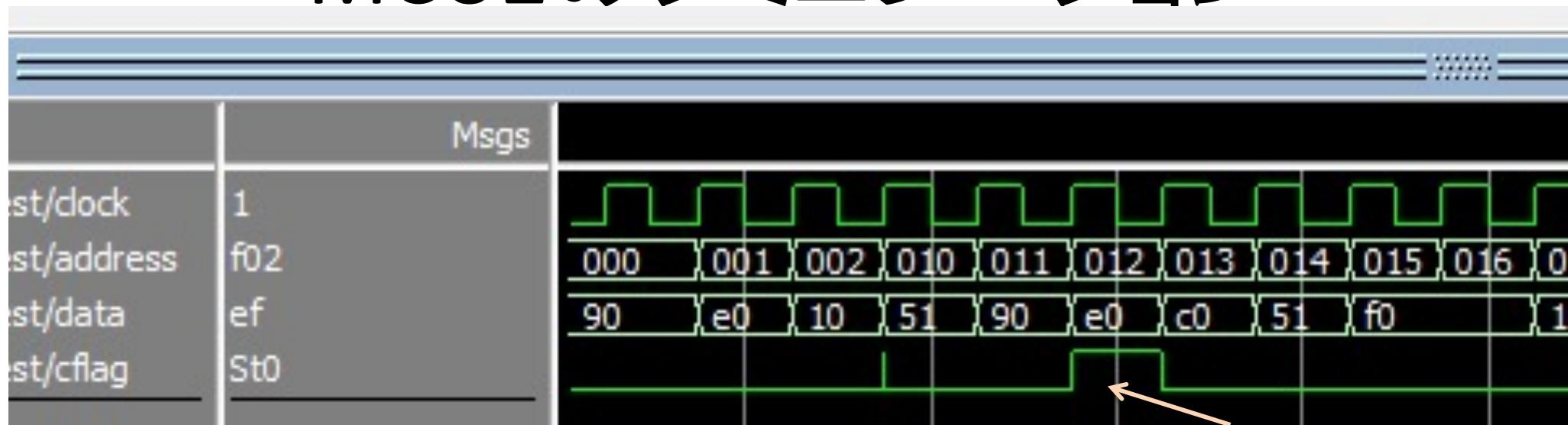
プログラム制御ユニット(PCU)



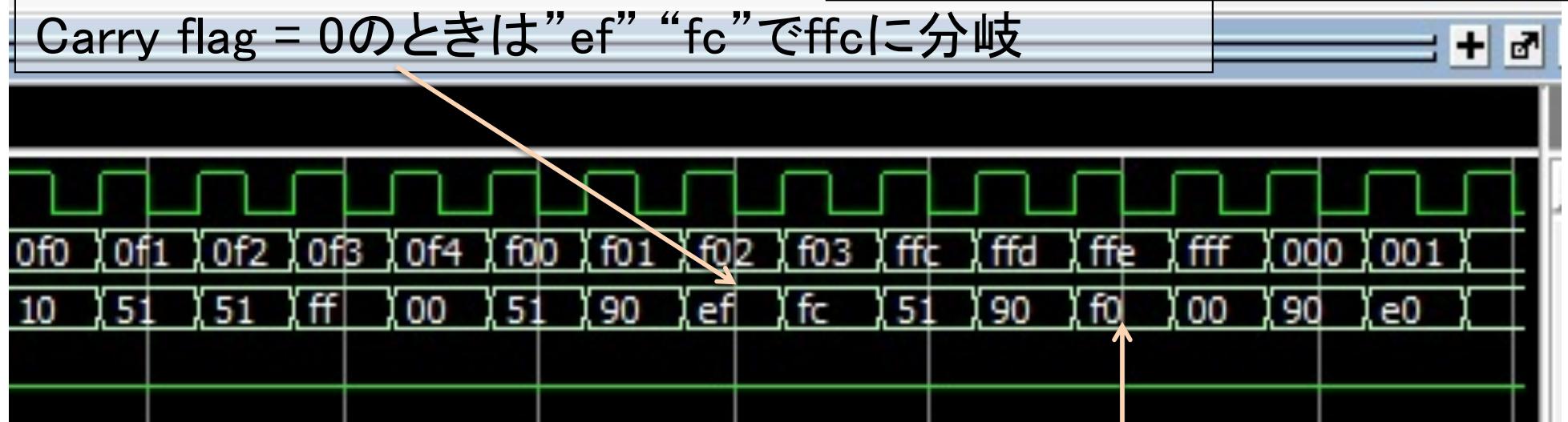
提供される PCU.bdf



MCU2のシミュレーション



Carry flag = 1のときは分岐しない



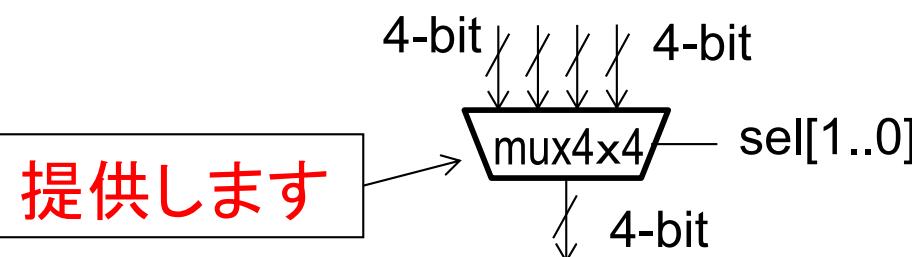
”f0” “00”で000にジャンプ

予習6.2のヒント(1)

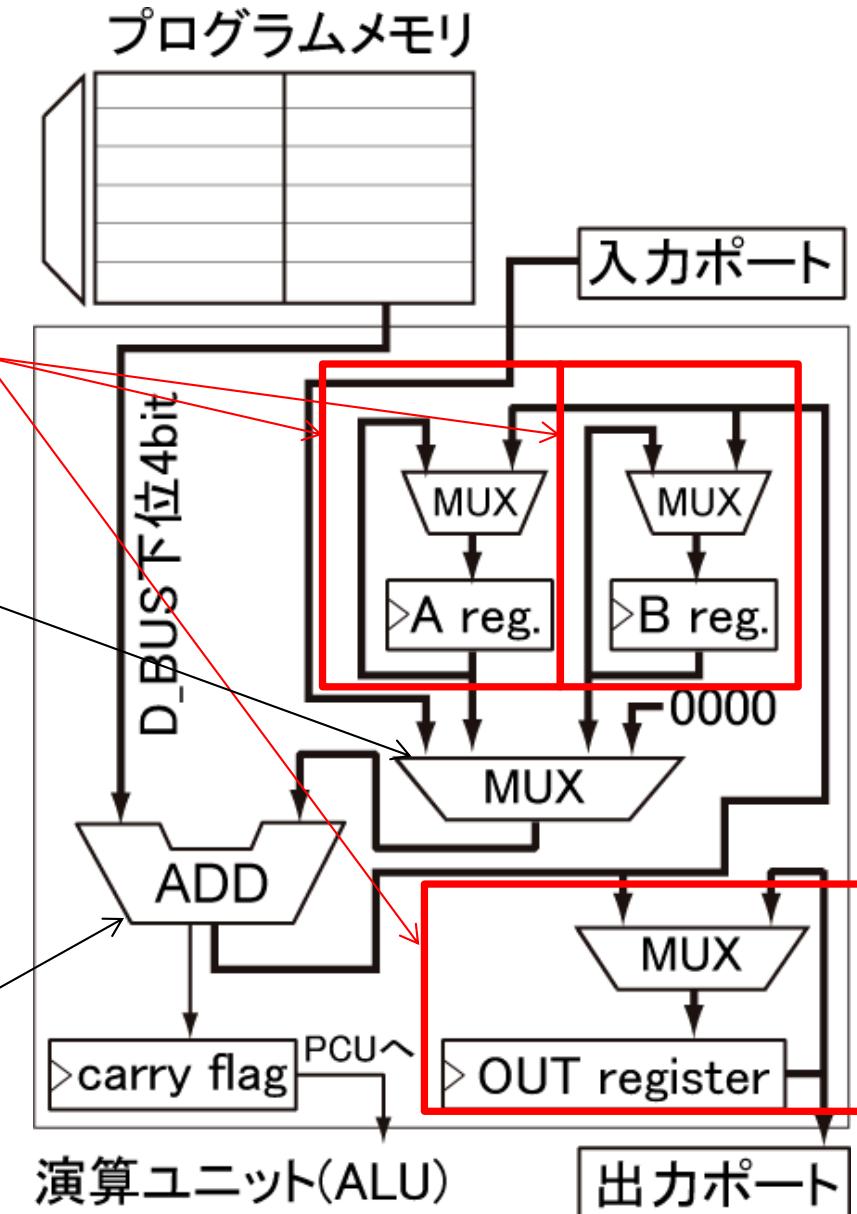
ここに見えているのは
データパス部のみ！

実験6.1で使用した回路：
ロード付き4ビットレジスタ

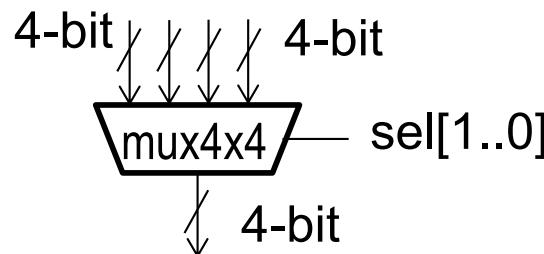
四者択一のマルチプレクサ
※ 入出力はすべて4ビット



加算器7483を
使用しても良い



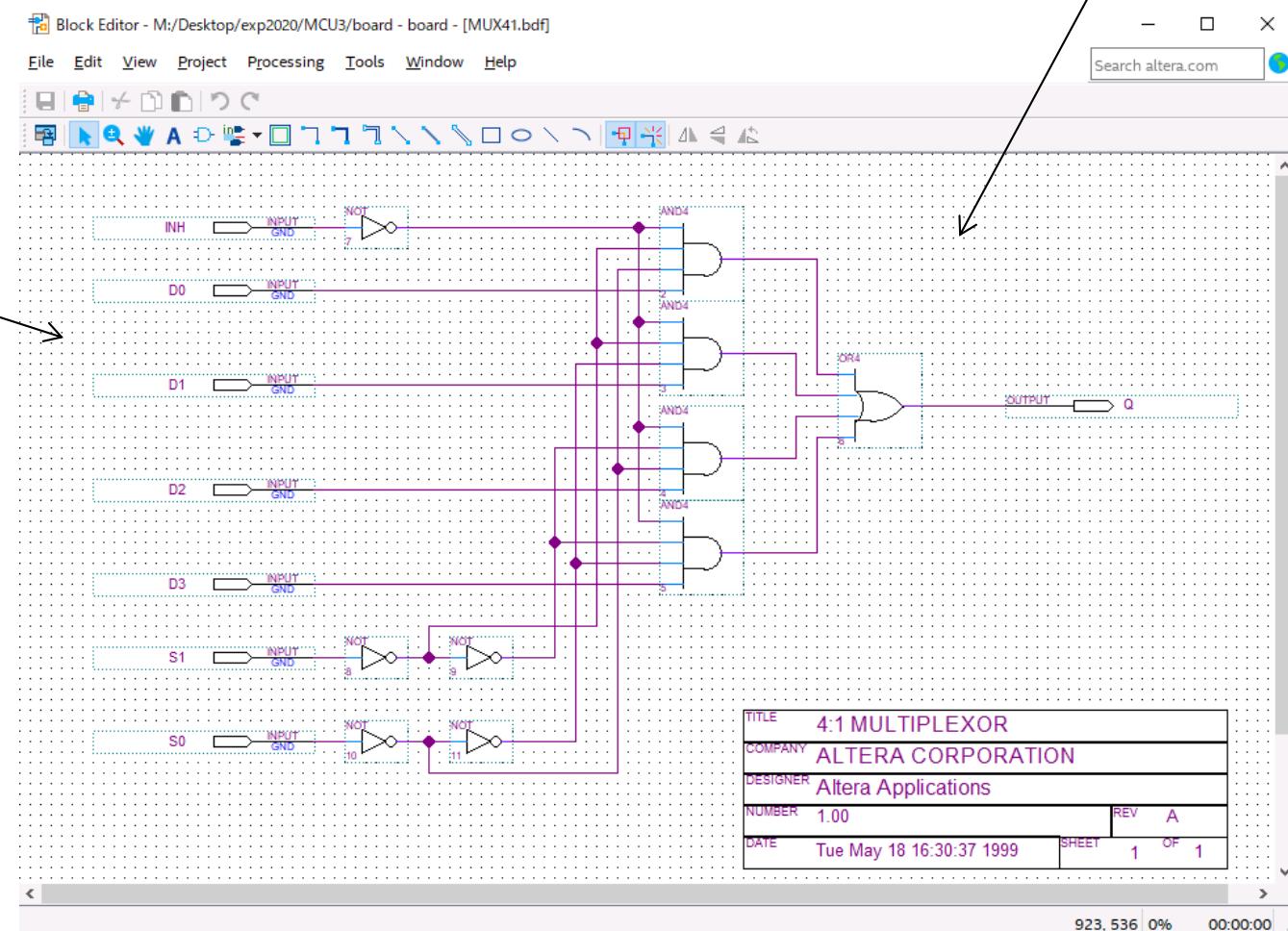
4bitの四者択一マルチプレクサ



ただし、これは
1bit の 四 者 択 一
マルチプレクサ

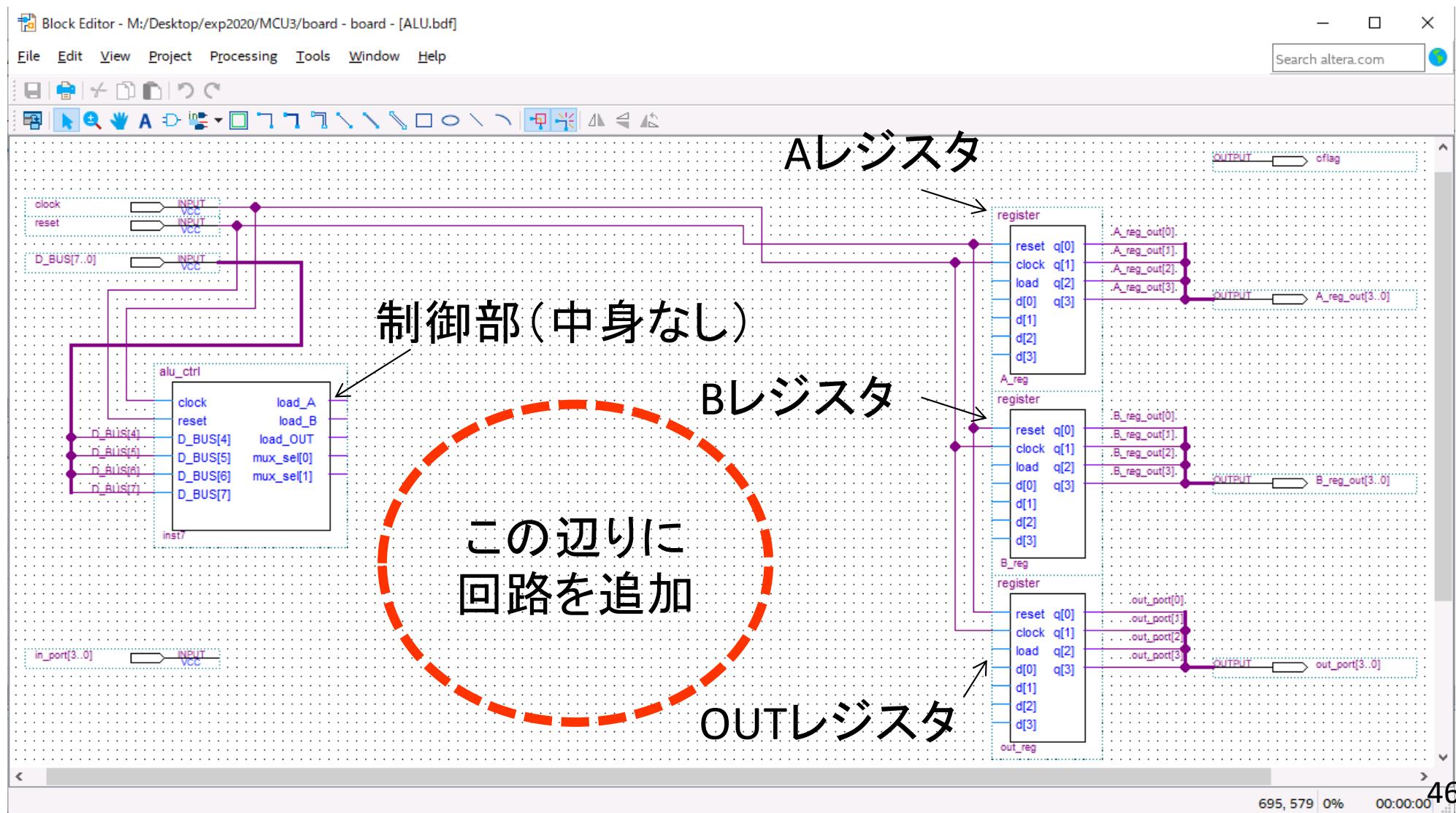
これを4つ使って
4bitマルチプレクサ
が設計されている
ことを確認する

基本セルライブラリ中のMUX41
(others/maxplus2/MUX41.bdf)
を利用していることを確認する



ALUの設計

MCU3のフォルダを使用



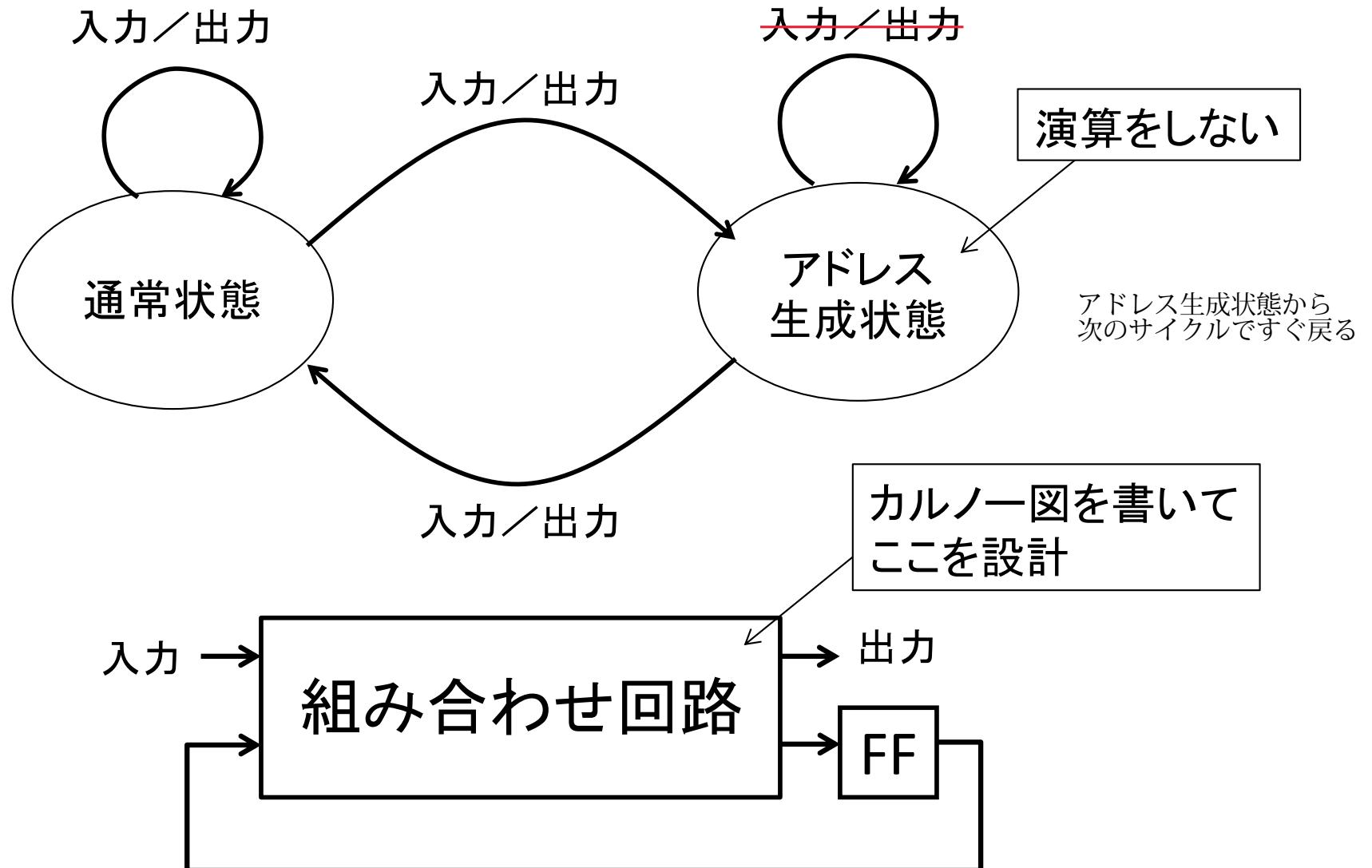
予習6.2のヒント(2)

- 演算をしない条件の生成
 - JNC命令とJMP命令の次のクロックサイクル
 - 「演算をしない」 \equiv 即値(D_BUS[3:0])と0を足す
 - ただし、「D_BUSの上位4ビットが1110かまたは1111になつた次のクロックサイクルは演算しない」という論理では仕様通りには動かない
 ← JNC、JMPの次が1110、1111になることもある
- A、B、OUTレジスタへのデータ取り込み条件の生成
 ← JNC、JMPの次のサイクルでは書き込まない
- “MOV A, B”の下位4bitは常に0000
- GPRに関する命令(1010, 1011)は無視して良い。ドントケアにしない方が、3週目の実験には都合が良い

MOV命令はImm(即値)が必ず0000になっているとみなせるということ?
→それでよい

ADD命令ではキャリーフラグを更新することとなっているけど特に何もしなくていいよね?
→キャリーフラグに関する条件を生成する必要がない

予習6.2のヒント(3)



具体的な設計内容

alu_ctrl.bdfの設計と動作検証

オペコードの上位3ビット

3ビットあればJMPとJNCを判別できる

D_BUS[7:5]

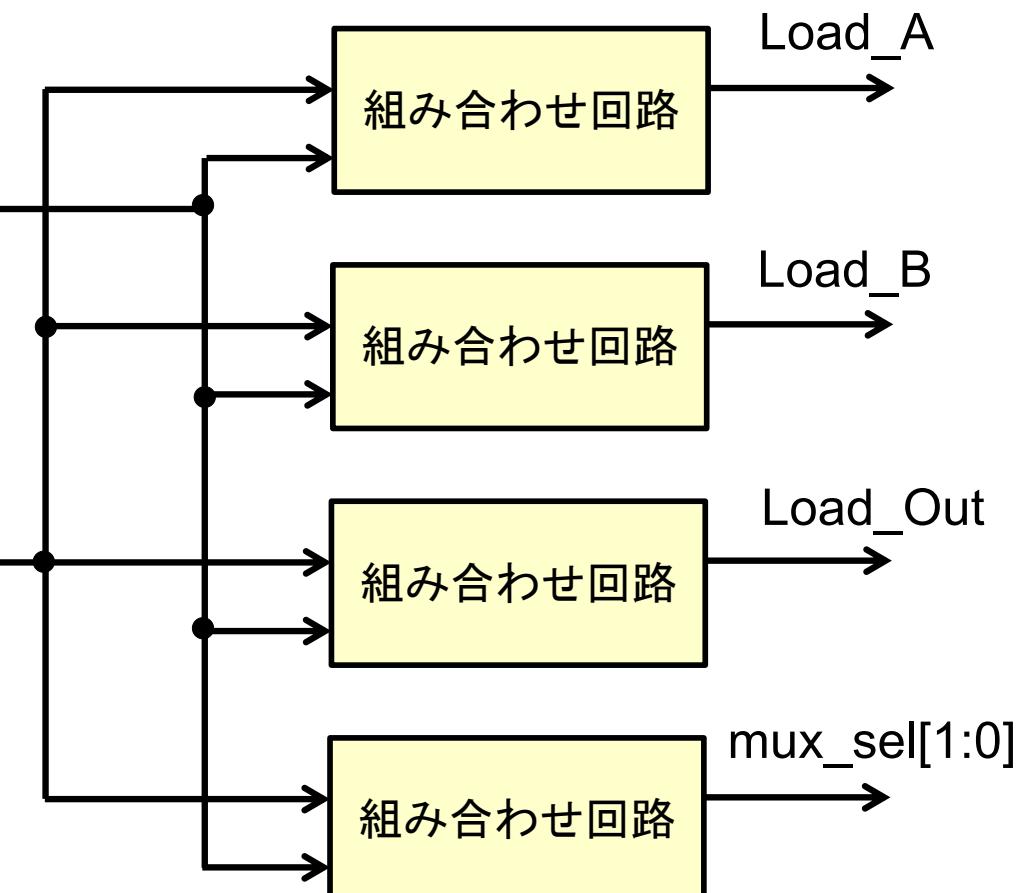
状態機械

組み合わせ回路

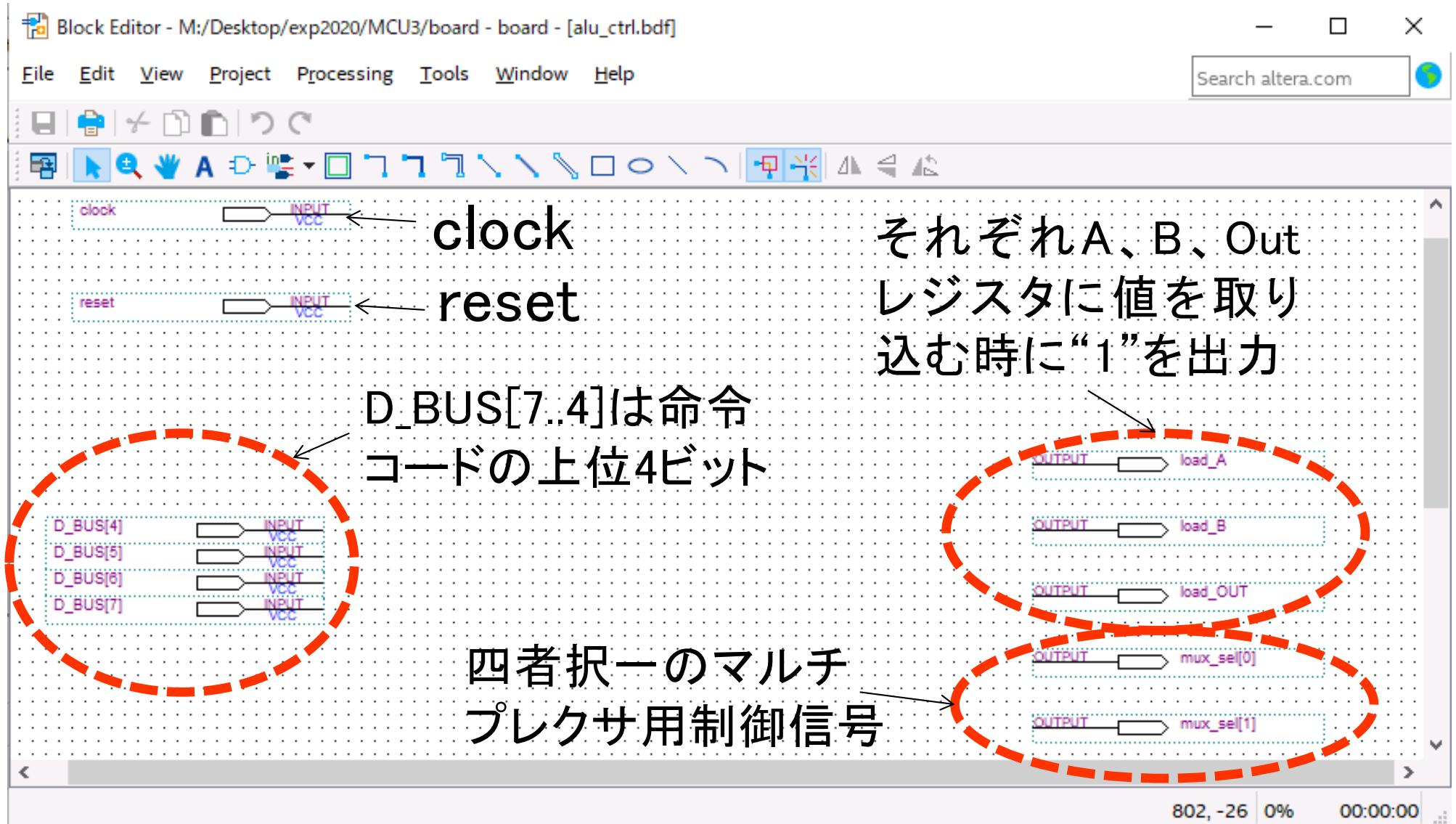
オペコード

D_BUS[7..4]

アドレス生成状態では「読み込んだコードを命令コードとして解釈しない」



alu_ctrlの設計



サブレポート(2週目)

- 実験6.4の実行結果の報告
 - 表1を参考に各ステップで実行される命令を記録
- 演算ユニット(ALU)の設計(予習6.2)
 - 制御部(alu_ctrl)の状態遷移図の作成
 - 各状態とその入力は何か, その時の出力は何か
 - カルノー図の作成
 - 基本セルを使って回路図を記述
 - マルチプレクサやレジスタなどは階層化すると良い
 - 信号線には名前も付ける
 - 考察(回路の解説や工夫した点の説明)

サブレポートを PDF 化してPandAへ提出
lcf を zip 化してPandAへ提出

第3週

演算ユニットの設計

実験6.5～6.6

3日目の課題

1. 実験6.5、6.6の実施

- ・演算ユニットを完成させ動作検証(実験6.5)
 - ✓MCU3(音が鳴る)とMCU3bの2種類のフォルダで検証
- ・回路規模と性能の調査(実験6.6)

2. 予習6.3の実施(できるところまで)

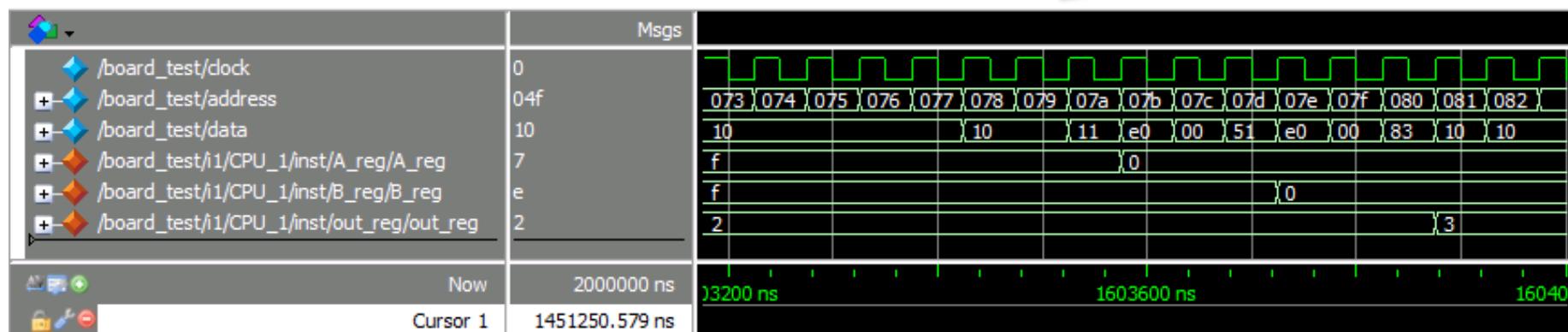
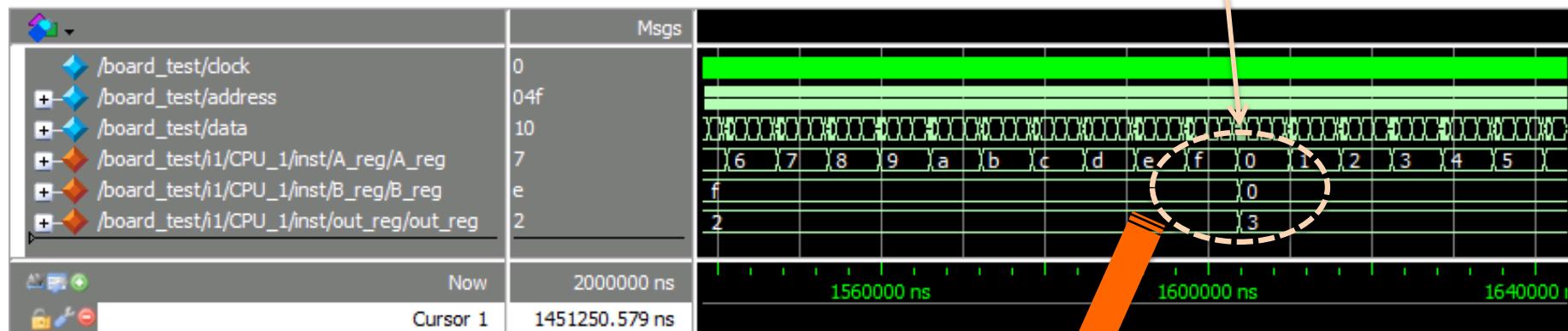
- ・汎用レジスタとMOV命令の追加 (MCU4を使用)
 - ✓データパス部と制御部を分けて設計
 - ✓まずはデータパス部の設計からはじめよう
 - ✓制御部の設計(変更)はレポート課題

※ 実験6.5、6.6が終わり次第予習6.3を実施

MCU3のシミュレーション

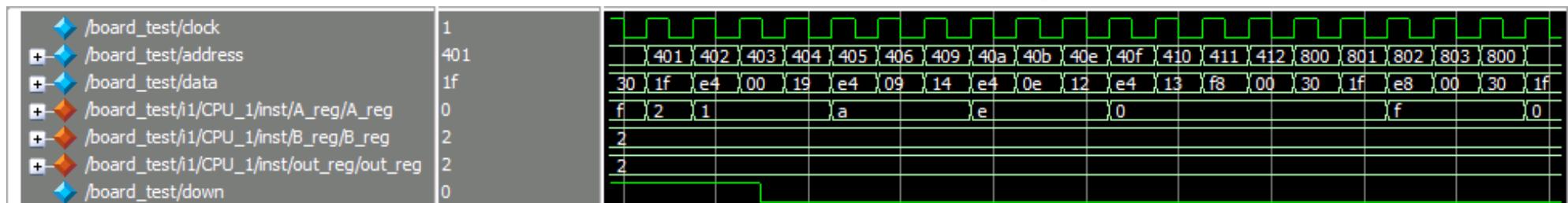
OUTレジスタにブザーが接続されていてregOutの値に応じてブザー音が変わる

regAの値が加算されて行き“F”になるとregBが加算され、regBが“F”になるとregOutの値が変わる

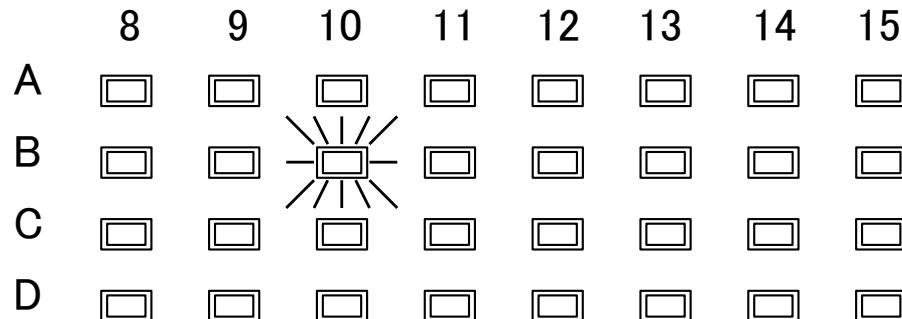


MCU3bのシミュレーション

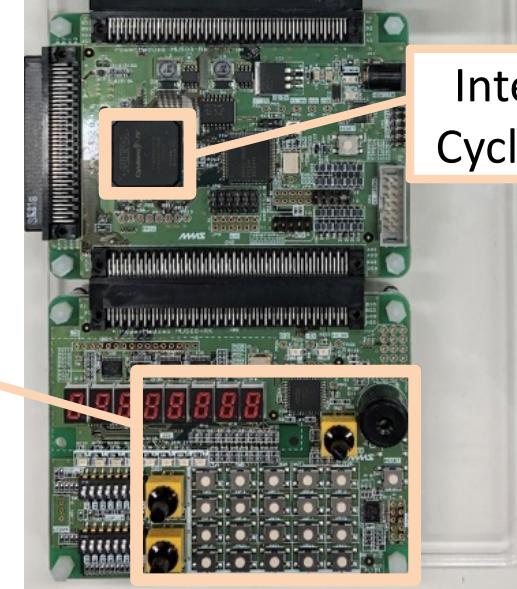
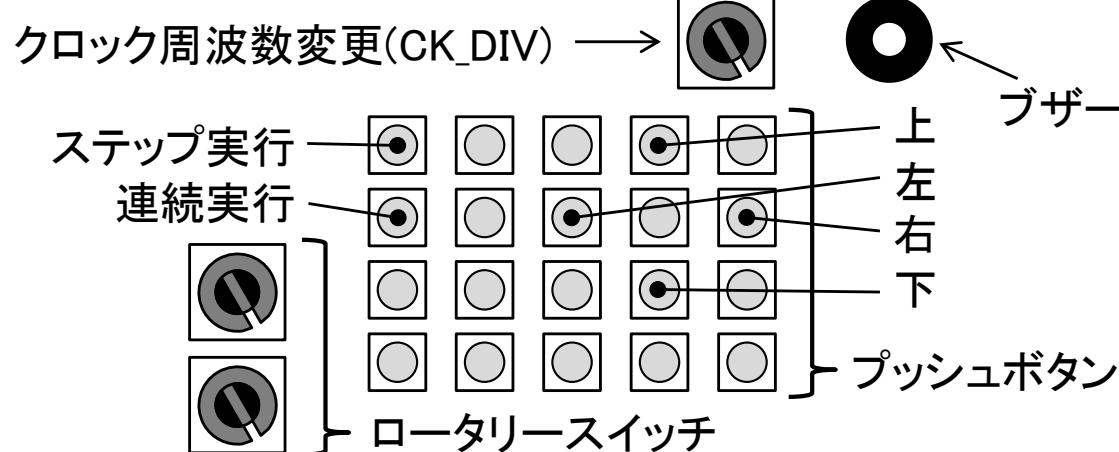
プッシュボタンがINポートに接続されていて、INポートに入力された上下左右の信号に応じてLEDの点灯箇所が動く



MCU3bのテンキー操作



テンキー操作で点灯箇所が移動



Intel FPGA
Cyclone IV E

※ ボード詳細 → 配付資料“FPGAボードマニュアル”を参照

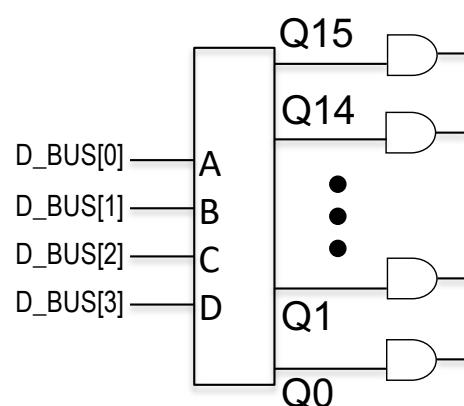
予習6.3 汎用レジスタとMOV命令の設計

- 16個の4ビットレジスタを持つ汎用レジスタ(GPR)とGPRへの書き込み命令およびGPRからの読み出し命令を設計する
 - (MOV B←GPR) 命令(コードは1010)が実行された時にGPRの値をBレジスタに書き込む
 - (MOV GPR←B) 命令(コードは1011)が実行された時にBレジスタの値をGPRに書き込む
- 16個の4ビットレジスタ回路(register16.bdf)と4ビットデータの16者択一マルチプレクサ回路(mux4x16.bdf)は既製回路を提供する

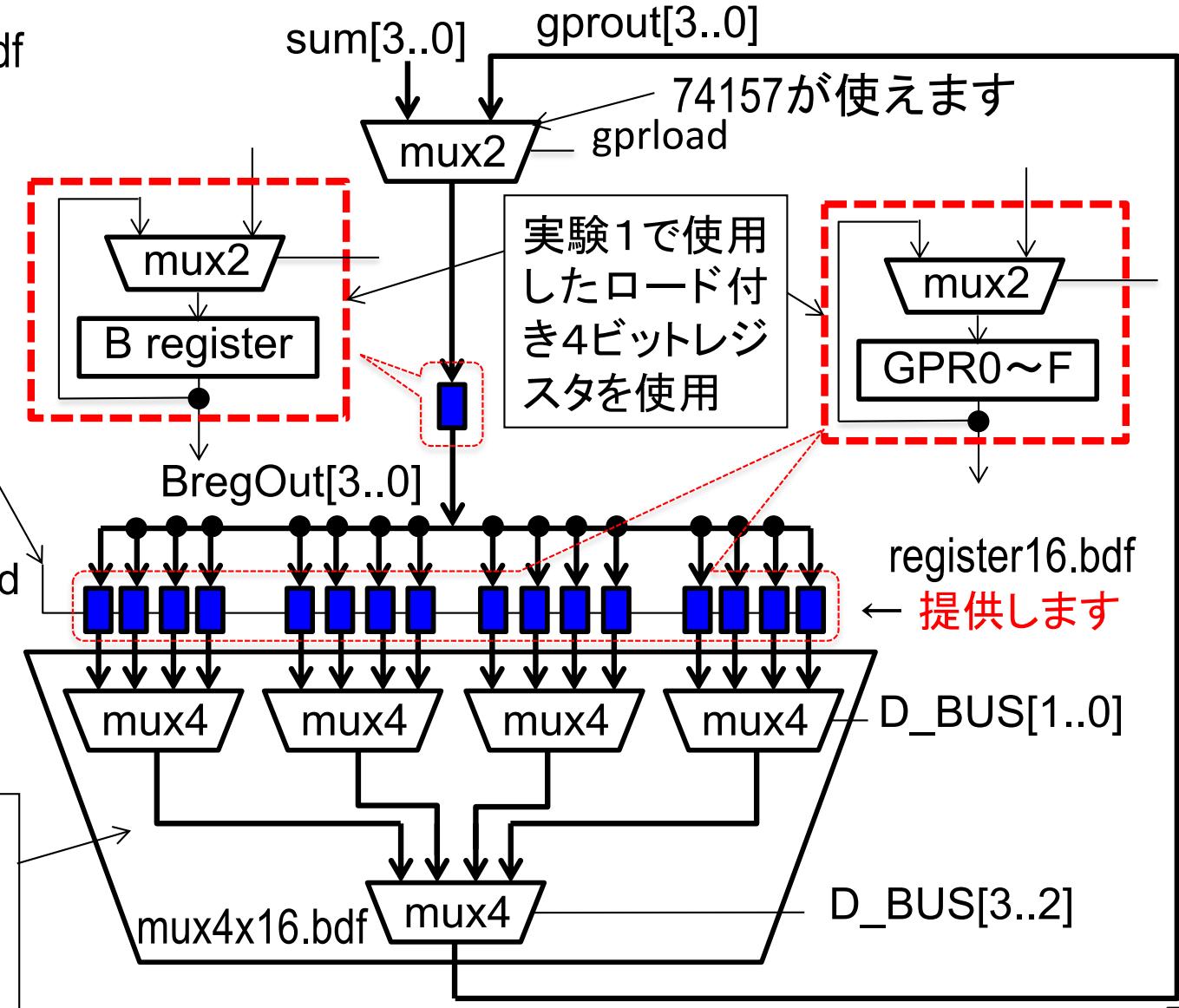
汎用レジスタの設計に必要な回路

/others/maxplus2/16dmux.bdf
を使うと便利

load_0～load_Fを生成
する回路を別に設計



4入力マルチプレク
サを使った16者択一
回路 ← 提供します



注意点

MOV_A_Imm	0000
ADD_A_Imm	0001
MOV_A_B	0010
MOV_A_IN	0011
MOV_B_Imm	0100
ADD_B_Imm	0101
MOV_B_A	0110
MOV_B_IN	0111
MOV_OUT_Imm	1000
MOV_OUT_B	1001
MOV_B_GPR	1010
MOV_GPR_B	1011
JNC_Imm	1110
JMP_Imm	1111

- MOV_GPR_Bが実行された時のみ GPRに値を取り込む
 - GPRのload信号生成にはD_BUS[7..4]を利用する必要がある
- MOV_B_GPRが実行された時のみ Bレジスタに値を取り込むもともとの取り込み条件または
 - Load_Bを生成する回路を変更する
- 他の命令と同様に通常状態の時のみ“命令”として実行する → アドレス生成状態ではレジスタ値の更新は一切行わないようにする

Load_GPR
gprload
はそれぞれ役割が違うので注意
後者はmux2でGPRを選ぶための入力信号

サブレポート(3週目)

LogicElement:実際に使った回路素子の数、実装できる回路素子の最大数

- MCUの回路規模と性能の結果報告
 - 実験6.6の結果をレポートで報告
 - クリティカルパスがどこかを明示する
- 汎用レジスタとMOV命令の追加(予習6.3)
 - ブロック図の作成
 - 必要であれば信号線名を明示する
 - 基本セルを使って回路図を記述
 - 要素回路をモジュール化して階層化すると良い
 - 考察(回路の解説や工夫した点の説明)

サブレポートを PDF 化してPandAへ提出
lcf を zip 化してPandAへ提出

第4週

汎用レジスタとMOV機能の追加

実験6.7

本日の課題

1. 実験6.7の実施

- 汎用レジスタとMOV命令の追加(実験6.7)
 - MCU4(音楽), MCU4b(LED), MCU4c(電卓) で確認

2. 予備課題の実施（時間の余った人）

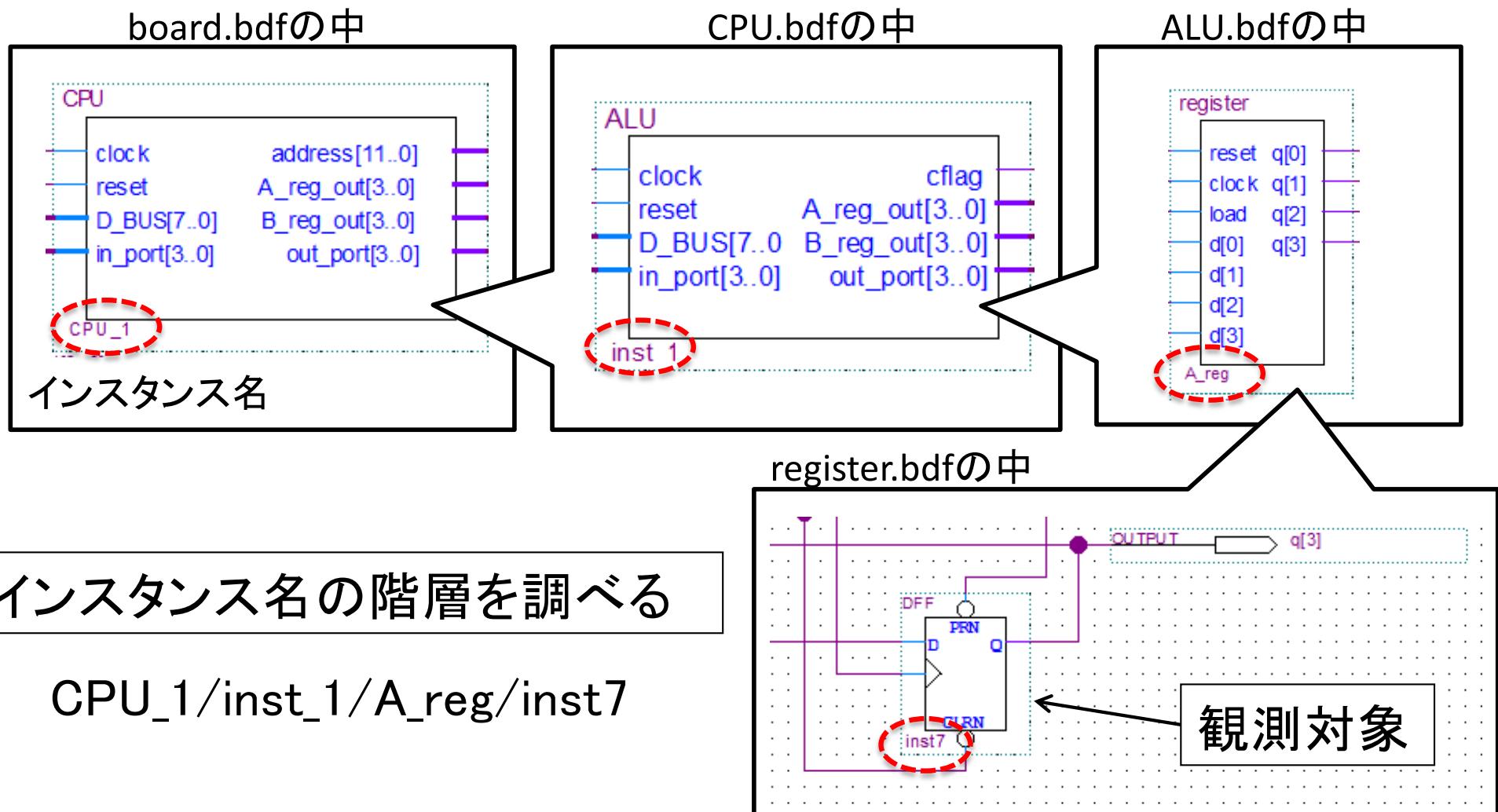
- 減算命令の追加(実験6.8)
- 乗算命令の追加(実験6.9)
- MCUのパイプライン化(実験6.10)

実験6.7 良くある間違いと対策

- ALU内の4入力MUXの間違い
 - ✓ 4つの入力に割り当てる信号が間違っている
- アドレス生成状態がflagcに依存している
 - ✓ ALU内の状態機械はflagcには依存しない
 - JNC命令の次の番地はflagcに関係なく8ビットのアドレス
- レジスタのload信号が0に固定してしまっている
- Compile後の回路図をnetlist viewerで目視確認
 - ✓ 紙に書いた回路図とnetlist viewerの回路図を比較
- シミュレーションをしましょう！！！
 - ✓ ROMコードが正しく実行されるかを順を追って確認

観測するレジスタの追加(1/4)

- A_reg の MSB 部分を観測対象に追加する場合



観測するレジスタの追加(2/4)

- simulation\modelsim\board_test.do に追記
 - テキストエディタ(メモ帳、Vim、Emacsなど)で編集

➤ レジスタ単体

```
add wave sim:/board_test/i1/CPU_1/inst_1/A_reg/inst7/q
```

全課題で共通

先ほどのインスタンス名を記載

q = DFFの出力

➤ 複数のレジスタをまとめると

16進数表記

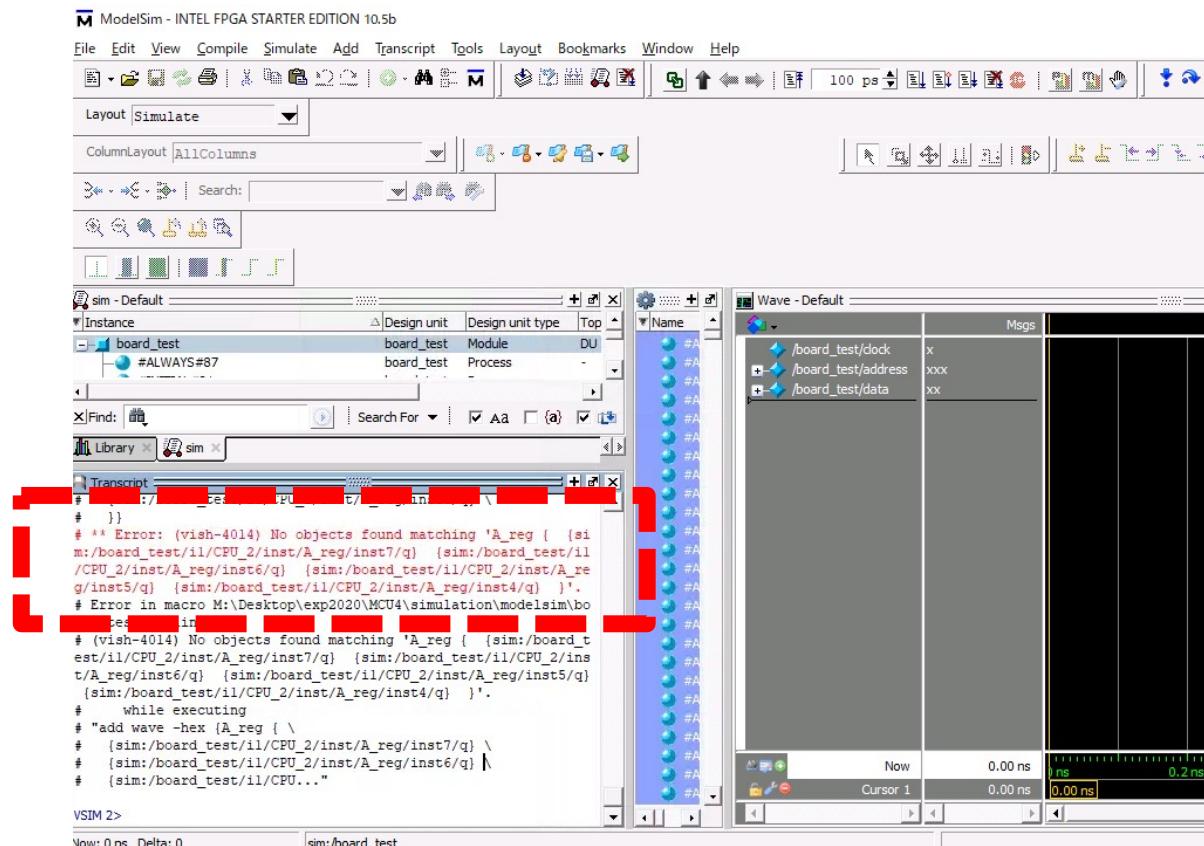
まとめた信号名を記載

複数行にわたって書く場合、
行末に \ を書く

```
add wave -hex {A_reg { \
{sim:/board_test/i1/CPU_1/inst_1/A_reg/inst7/q} \
{sim:/board_test/i1/CPU_1/inst_1/A_reg/inst6/q} \
{sim:/board_test/i1/CPU_1/inst_1/A_reg/inst5/q} \
{sim:/board_test/i1/CPU_1/inst_1/A_reg/inst4/q} \
}}
```

観測するレジスタの追加(3/4)

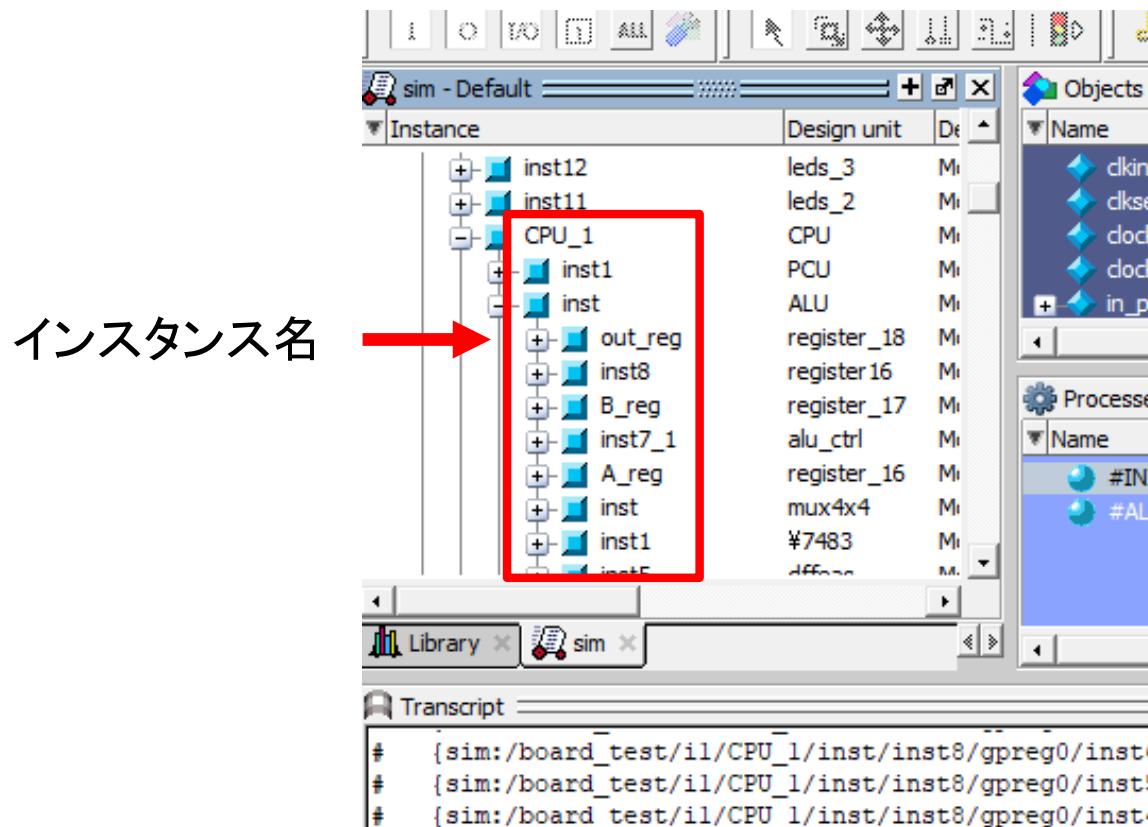
- board_test.do の記載内容を間違えると...



- 画面左下(Transcript)に赤色でエラー表示
- エラー内容を読んで対応し、シミュレータ再起動

観測するレジスタの追加(4/4)

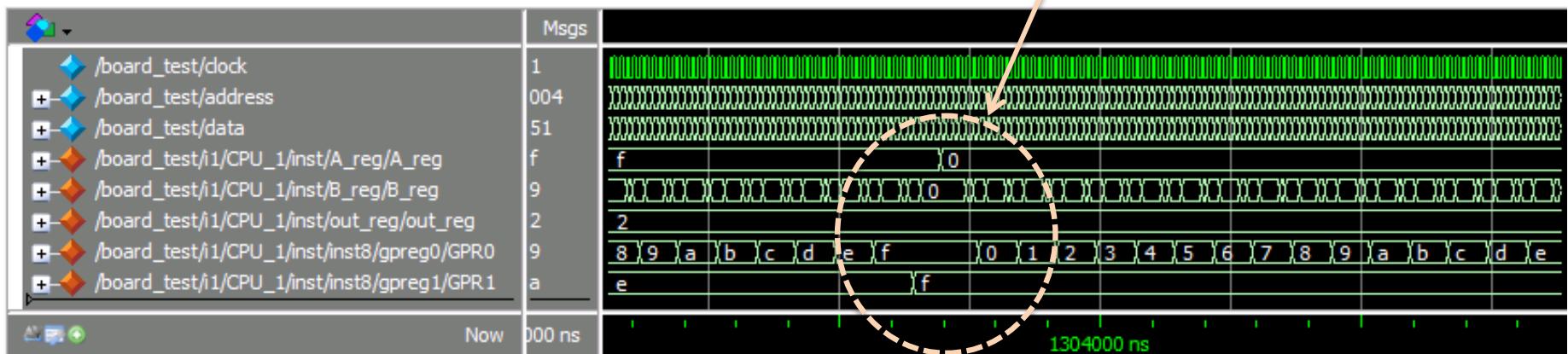
- たまにQuartusがインスタンス名の後ろに“_1”をつけることがある。
- ModelSimの画面からインスタンス名を確認



MCU4のシミュレーション

OUTレジスタにブザーが接続されていてregOutの値に応じてブザー音が変わる

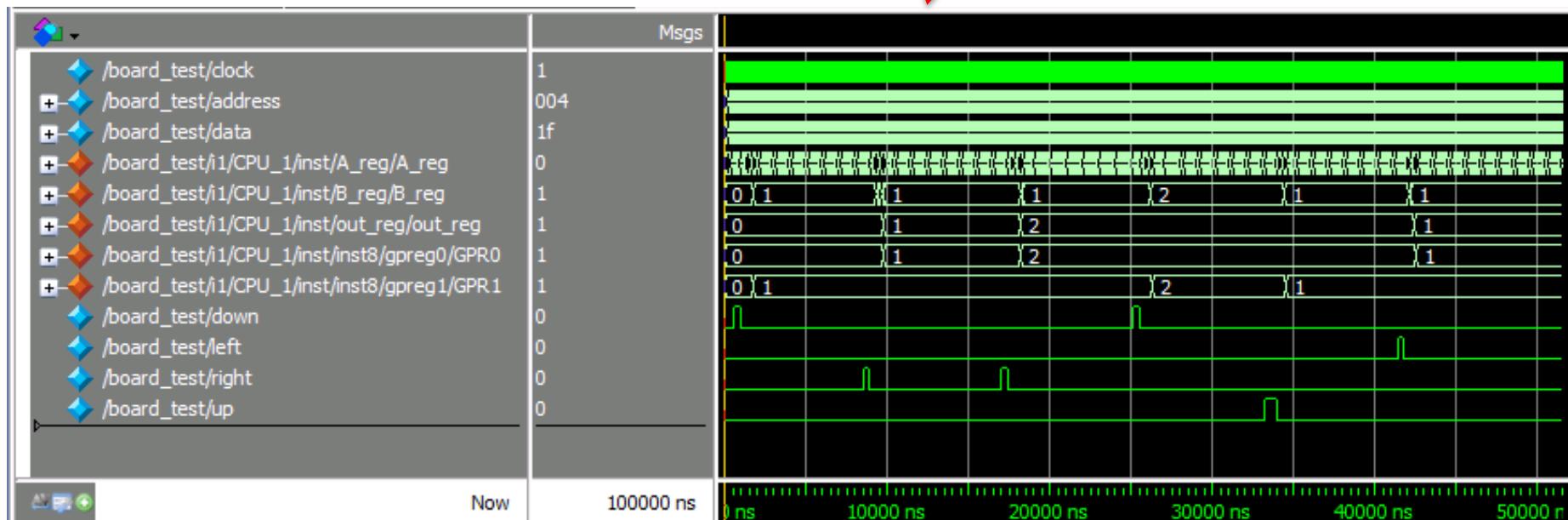
GPR0の値が加算されて行き“E”→“F”に変わるとGPR1が加算される。次に、GPR1が“E”→“F”に変わるとregAが加算され、“E”→“F”に変わるとregOutの値が変わる(仕様だが、今回の音楽サンプルではregOutは固定)



MCU4bのシミュレーション

プッシュボタンがINポートに接続されていて、INポートに入力された上下左右の信号に応じてLEDの点灯箇所が動く

上下左右のプッシュボタンでINポートに1,2,3,4を送信
regOutの下位3ビット(0,1,2,3,4,5,6,7)で左右の座標を決定し、
regBの下位2ビット(0,1,2,3)で上下の座標を決定



サンプルプログラム(電卓)

D_BUSおよびA reg.またはC reg.
とB reg.の表示位置を変更

ここに演算結果表示

MCU4c : 加算のみ可能

MCU5b : 加算・減算が可能

MCU6b : 加算・減算・乗算が可能

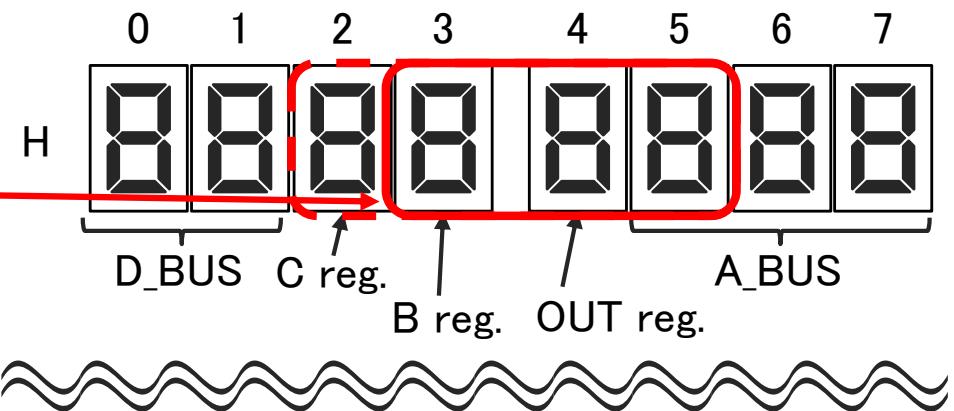
入力は10進2桁まで

出力は-99~198まで(加減算)

0~9801まで(乗算)

演算結果がマイナスの
時はB reg.にFを表示

P14とP54も参照

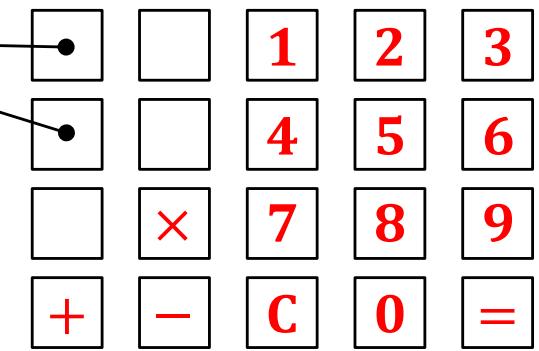


クロック周波数変更(CK_DIV) →



ステップ実行

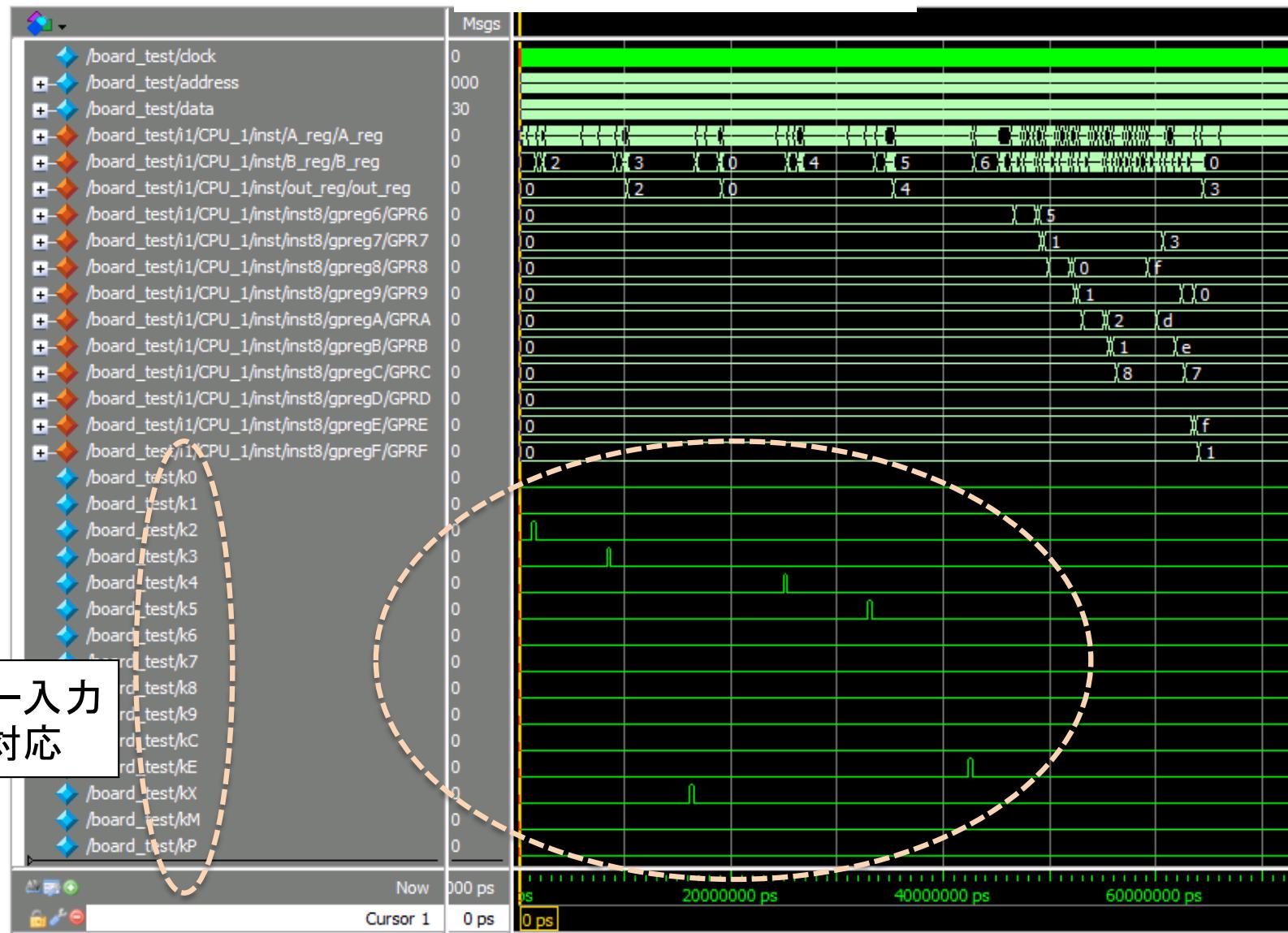
連続実行



プッシュスイッチ

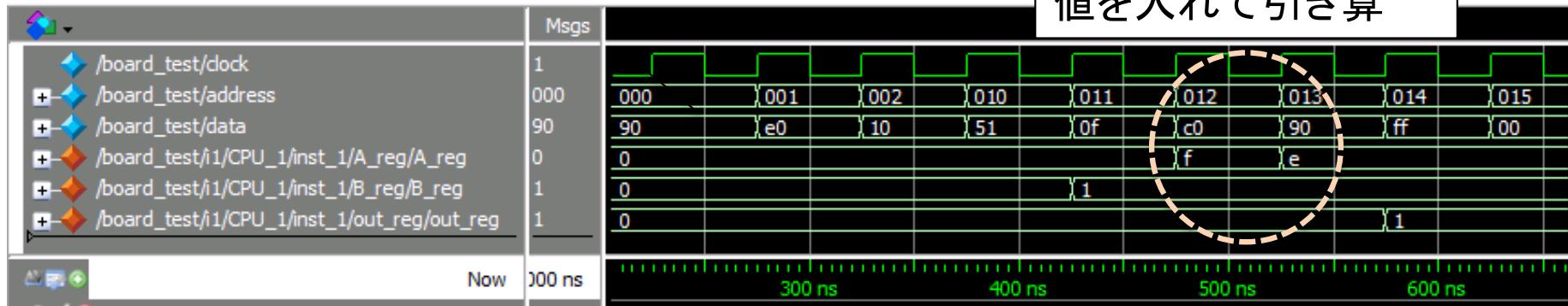
MCU4c,5b,6bのシミュレーション

MCU4c,5b,6bの波形例



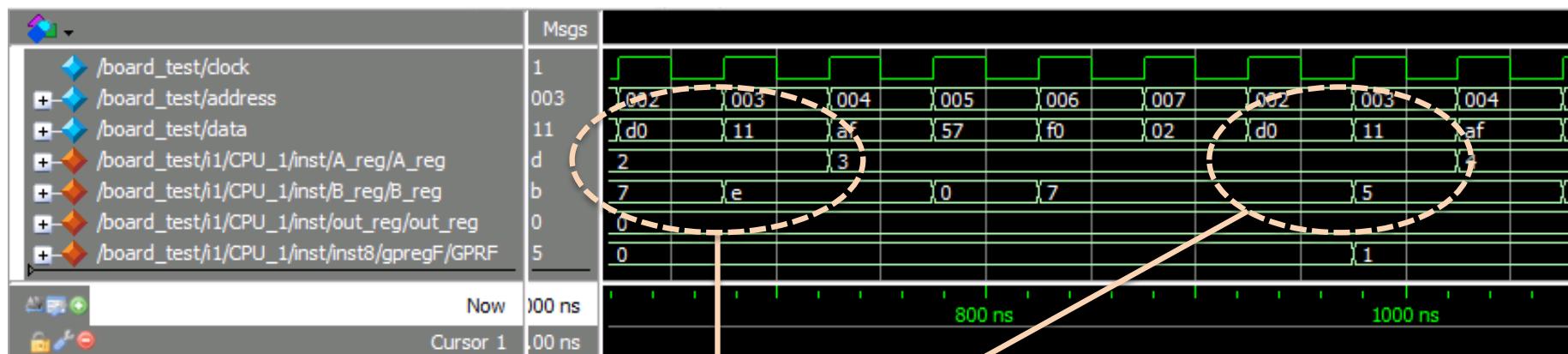
MCU5,6のシミュレーション

MCU5の波形例



regAとregBに適当な
値を入れて引き算

MCU6の波形例



regAとregBに適当な
値を入れて掛け算

レポート

全体設計図のまとめ

- 設計した命令や機能の説明(ブロック図、回路図)
 1. データバス部はブロック図の作成
 2. 制御部は論理関数と論理ゲートの回路図を作成

※予備実験課題の結果をレポートに清書する必要はない
 - クリティカルパス遅延とロジックエレメント数の報告
 1. クリティカルパス(クロック周期を決めるパス)遅延
Actual fmax (period) の値で評価
 2. 回路規模
ロジックエレメント数で評価
- ※ プログラムをより高速化すると評価が高くなる 高速化するとはどういうこと？
- ※ 予備実験課題に取り組むと評価が高くなる
- ※ レポートに掲載するブロック図や回路図として、Quartus画面のスクリーンショットを使わないこと。手書きやソフトで清書すること。

PDF化したレポートと、zip化したlcfをPandAへ提出

次回のディスカッション

- 電卓課題(MCU4c)の動作確認を一人づつ行う
- 減算・乗算まで完成した人は、MCU5b/MCU6bの動作確認
- そのほかの予備実験課題を行なった人は、その都度報告

予備実験課題

実験6.8～6.10

減算命令の追加(実験6.8)

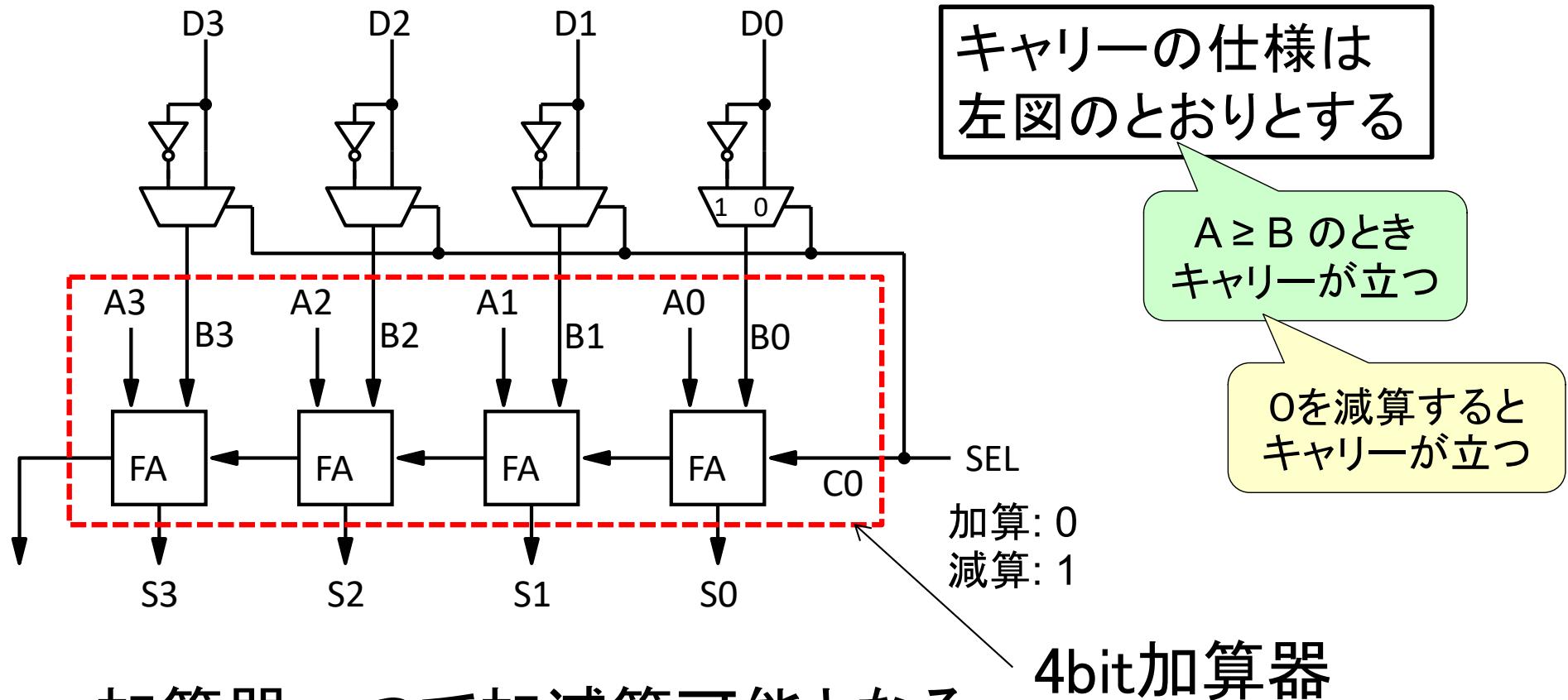
MOV_A_Imm	0000
ADD_A_Imm	0001
MOV_A_B	0010
MOV_A_IN	0011
MOV_B_Imm	0100
ADD_B_Imm	0101
MOV_B_A	0110
MOV_B_IN	0111
MOV_OUT_Imm	1000
MOV_OUT_B	1001
MOV_B_GPR	1010
MOV_GPR_B	1011
SUB_A_B	1100
?????????	1101
JNC_Imm	1110
JMP_Imm	1111

MOV_GPR_B: Bレジスタの値をGPRへ格納
MOV_B_GPR: GPRの値をBレジスタへ格納
SUB_A_B: Aレジスタの値からBレジスタの
値を減算してAレジスタに格納



追加命令に合わせて
制御回路を適宜変更

加減算器(2の補数)

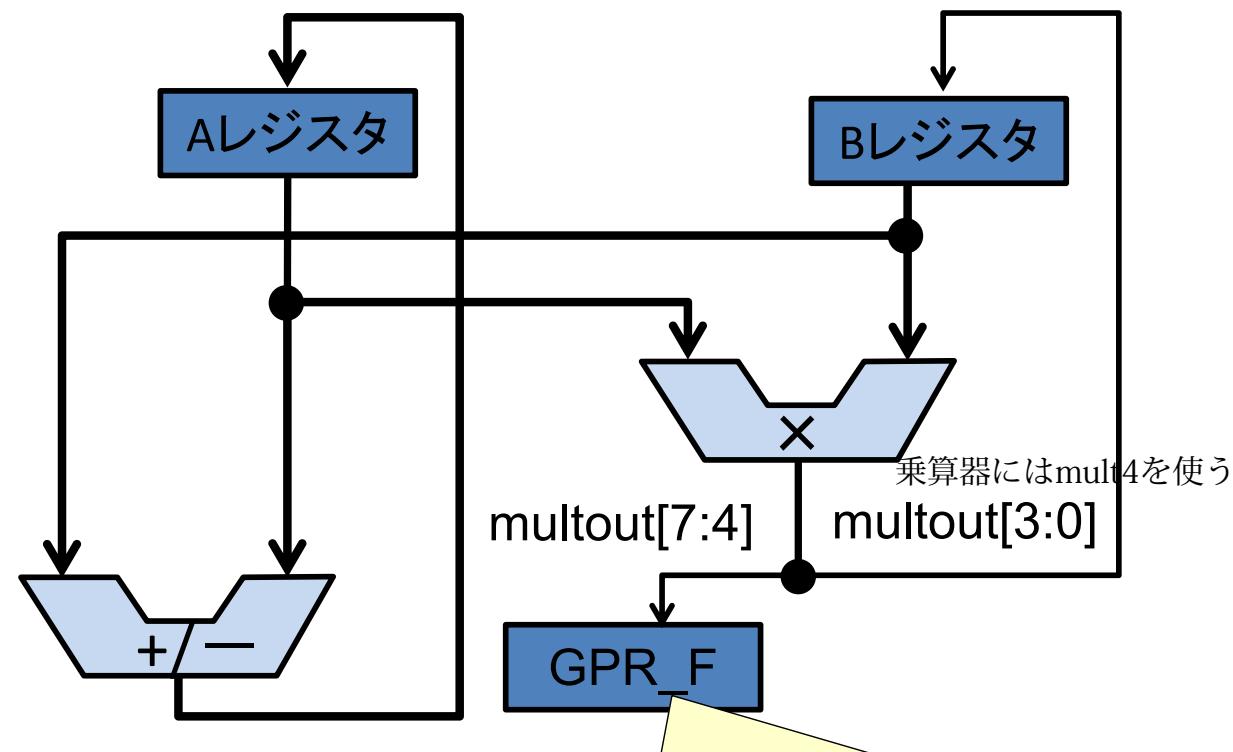


- 加算器一つで加減算可能となる
 - SELが0の時は $S = A+D$
 - SELが1の時は $S = (A+/D) + 1 = A+(D\text{の}2\text{の補数})$

乗算命令の追加(実験6.9)

MOV_A_Imm	0000
ADD_A_Imm	0001
MOV_A_B	0010
MOV_A_IN	0011
MOV_B_Imm	0100
ADD_B_Imm	0101
MOV_B_A	0110
MOV_B_IN	0111
MOV_OUT_Imm	1000
MOV_OUT_B	1001
MOV_B_GPR	1010
MOV_GPR_B	1011
SUB_A_B	1100
MULT_A_B	1101
JNC_Imm	1110
JMP_Imm	1111

MULT_A_B: Aレジスタの値とBレジスタの積をCとBレジスタに格納

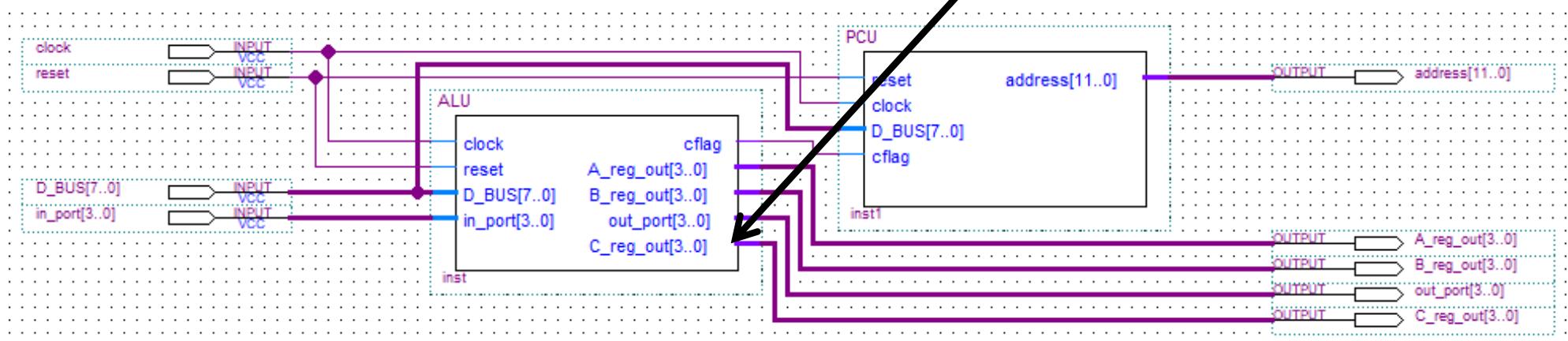


上位4ビットはGPRF(16番目)に格納！

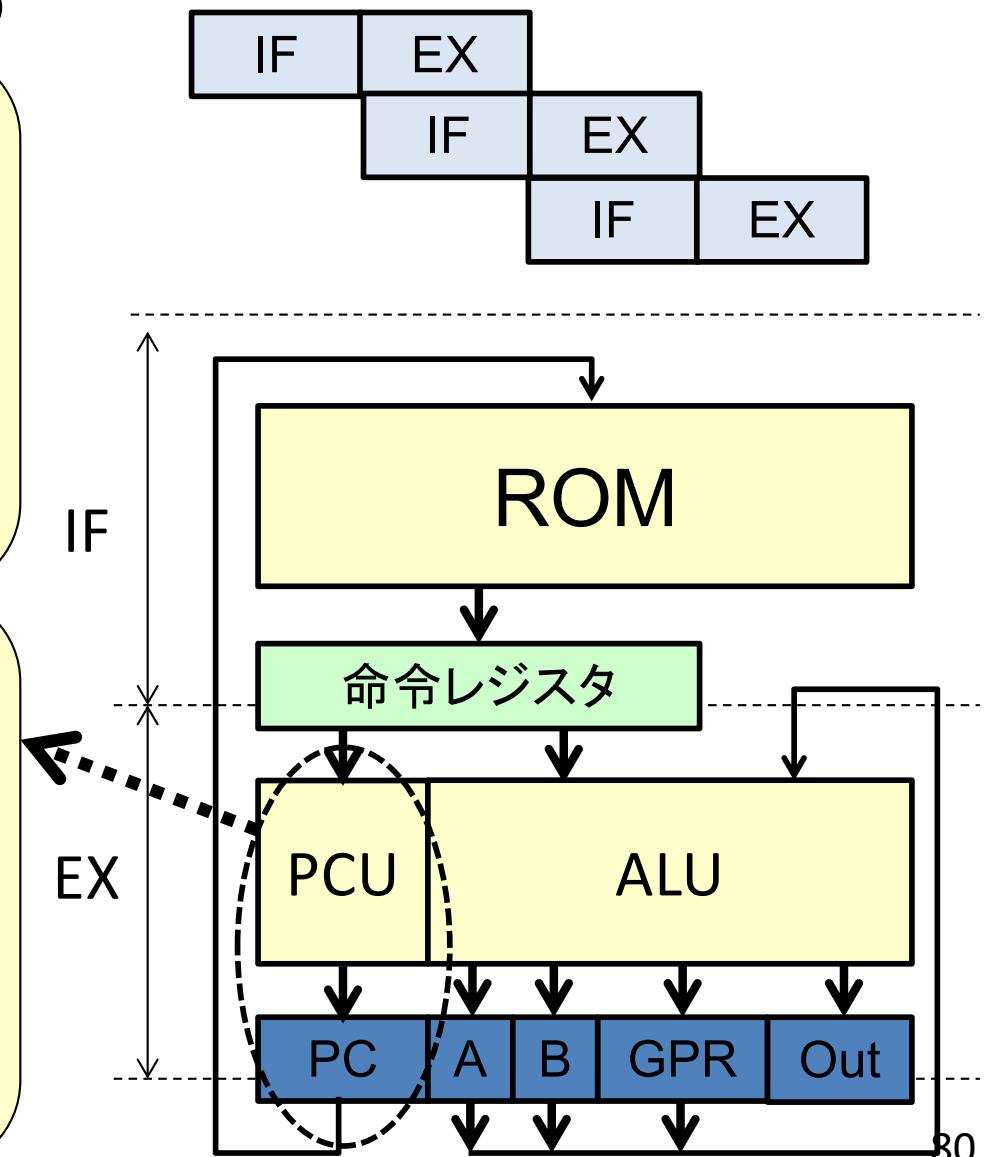
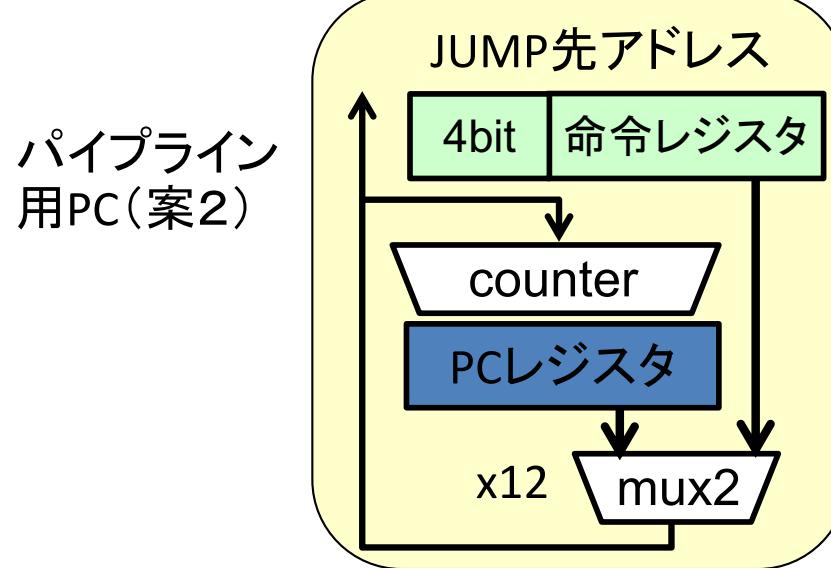
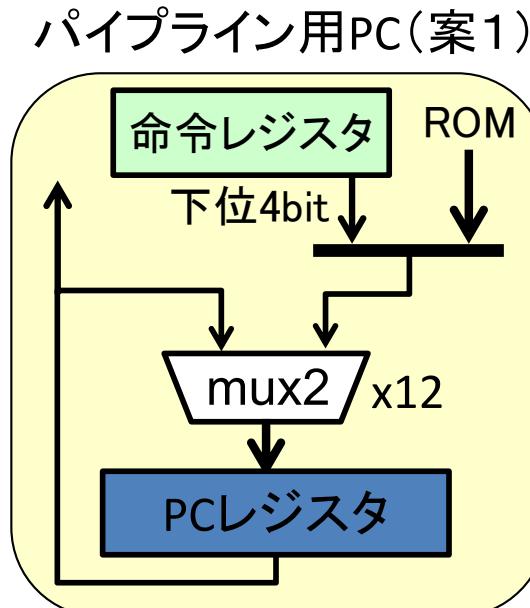
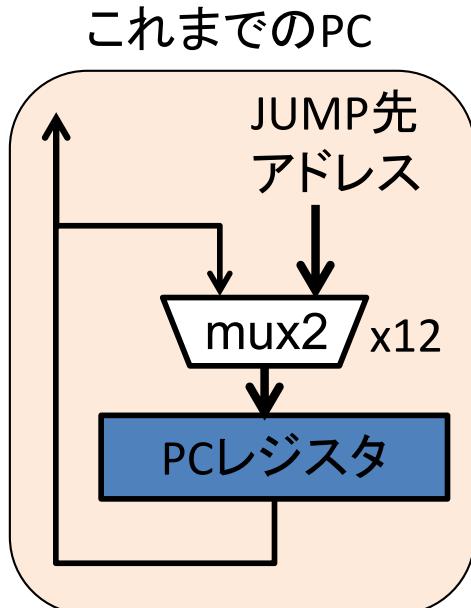
上位4ビットの値はGPR_Fから取り出す

MCU6b(減算・乗算命令追加)

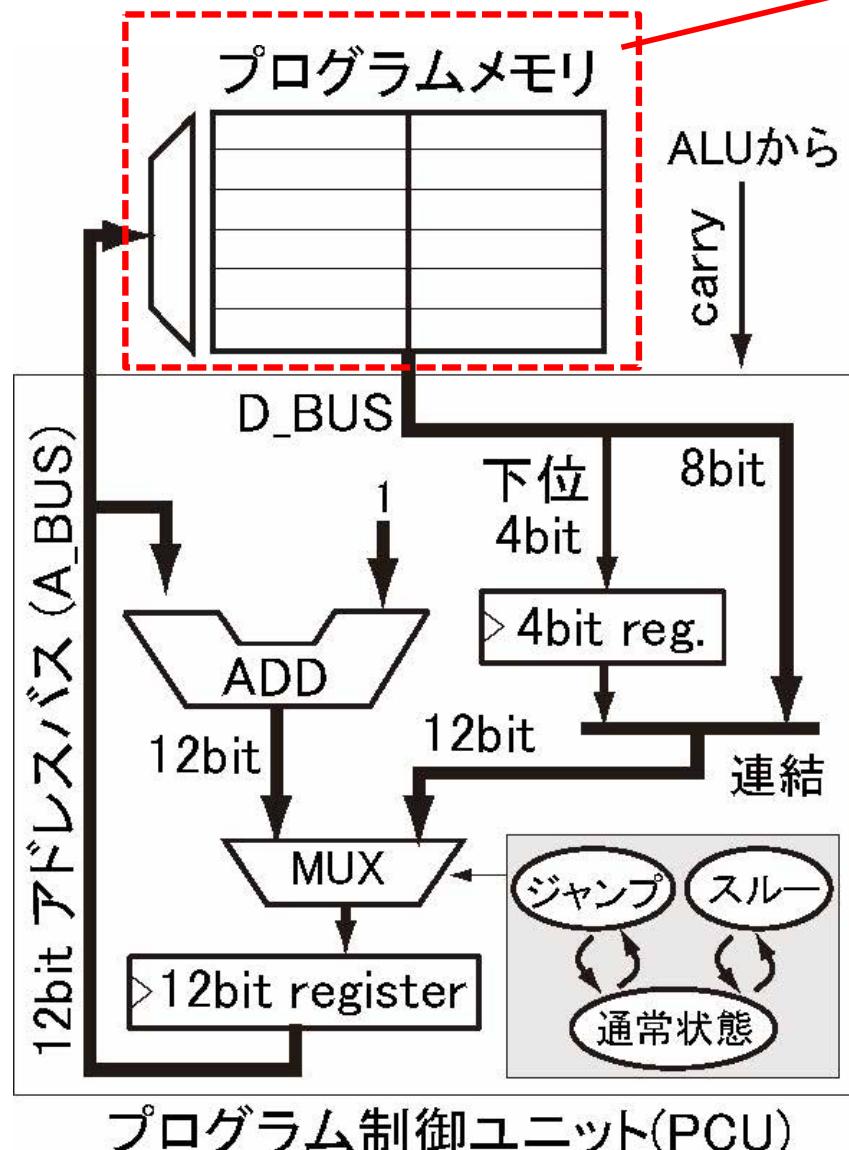
- 汎用レジスタの16番目(命令コードAFで取り出せるレジスタ)に乗算結果の上位4桁を格納
- 汎用レジスタの16番目の出力がC_reg_outに接続されるようにALUの出力ポートを変更



2段パイプライン化(実験6.10)



実験6.11: プログラムを書こう(教科書未掲載)



アドレス

000
001
002
003
004
005
006
007
008
009
00A
00B
0F
1E
2D
3C
4B
5A
69
78
87
96
A5
B4
C3
D2
E1

rom_sample*.hex
を書き換えて、
自分独自のプログ
ラムを書いてみよう

例:

素数判定し、出力ポートに0/1出力
→ フローチャートを書こう！

案1：

ROMから読み出してすぐジャンプする必要がある。

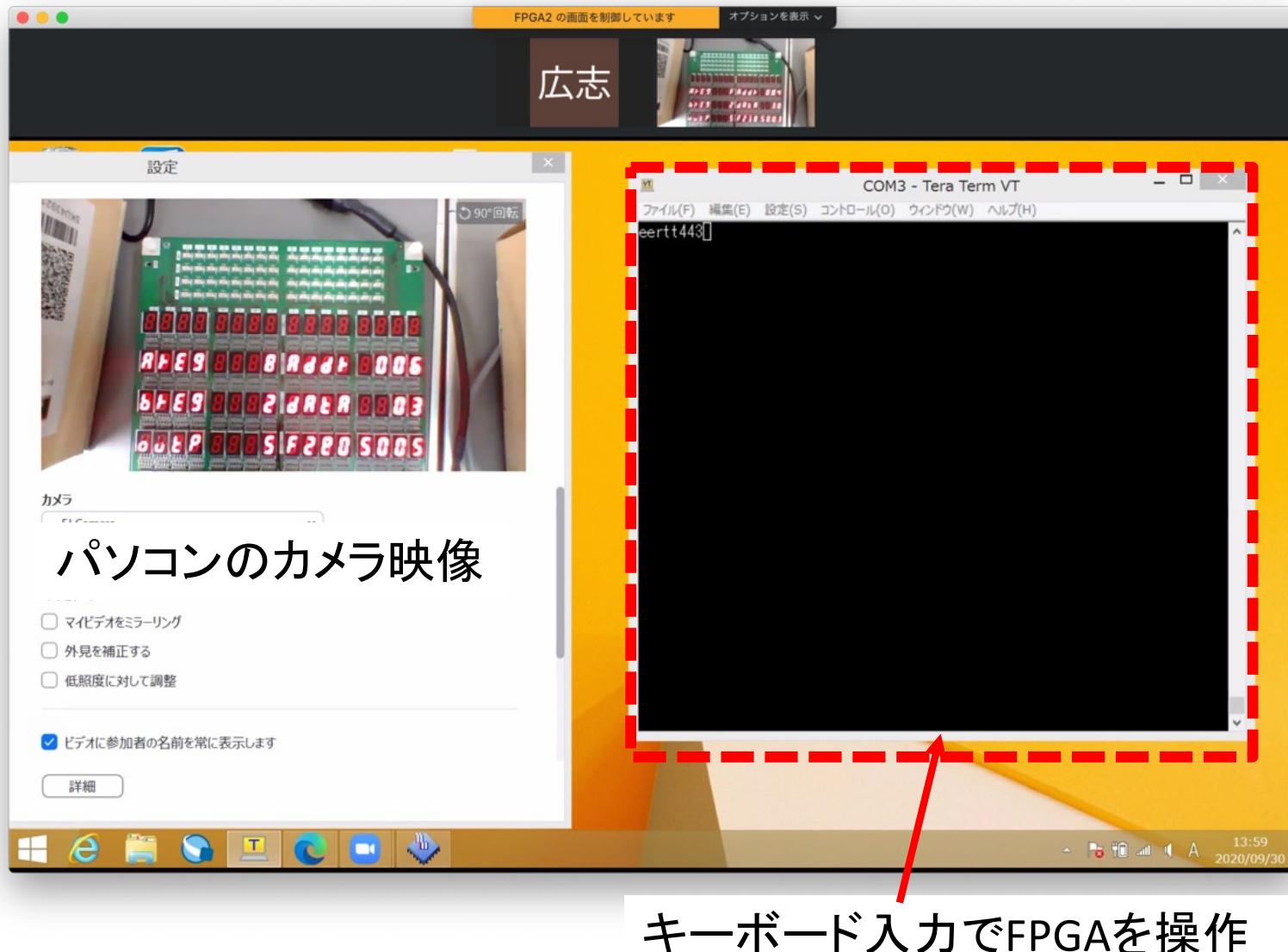
命令レジスタの下位4bitとROMから読み出した8bitを使ってアドレスを決めてジャンプしないといけない。

上手くジャンプ先アドレスを決めるように設定しないとバグになってしまう。

オンライン実験での遠隔操作

遠隔操作1 (オンライン参加用)

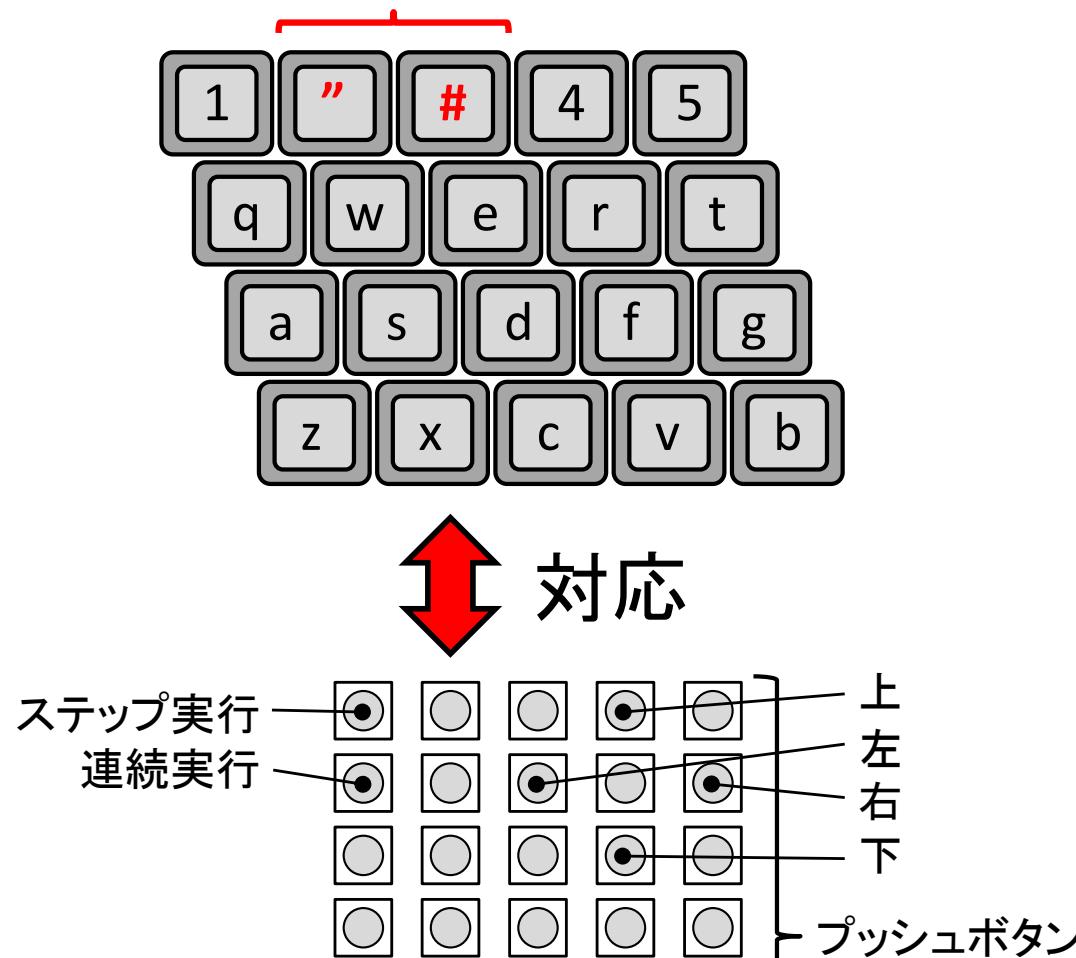
- Zoomのリモート制御機能で、FPGAを制御



遠隔操作2

- キーボード入力とプッシュボタンが対応

注意: ここだけシフトキーを押して入力



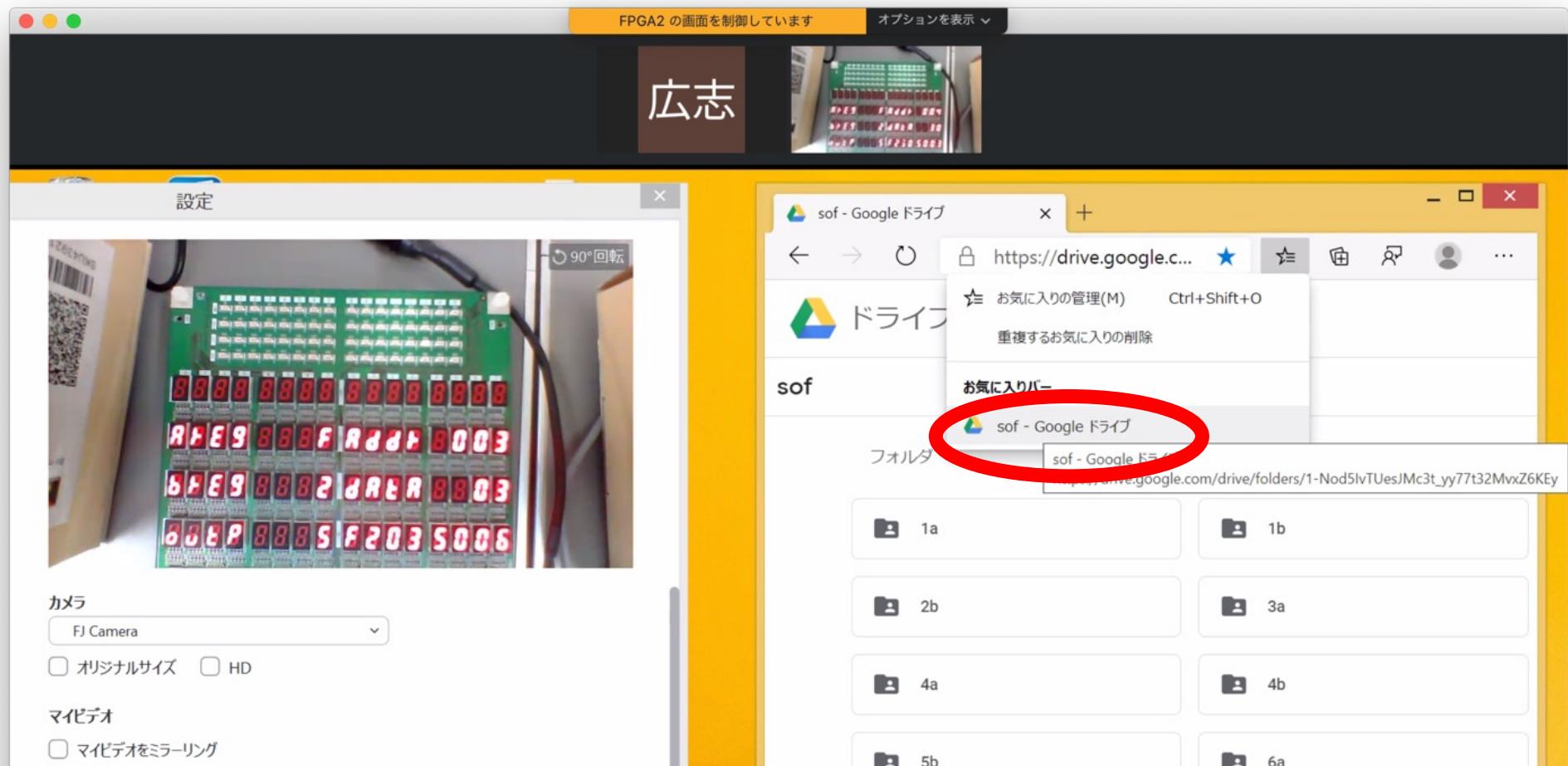
遠隔操作3

- MCU**\output_files\board.sof がコンパイル結果
- このファイルをGoogle Driveにアップロード
 - PandA >> 配布物・テキスト >> 6 論理回路設計演習 >> FPGA転送用Google Drive (<https://drive.google.com/drive/folders/1cCSWv9X33Mk6XXok2Cz7ZtLq5AbXIdCN?usp=sharing>)
- ブレークアウトセッションより、FPGA** のリモート制御を行う（オプションを表示 >> リモートコントロールを要求する）



遠隔操作4

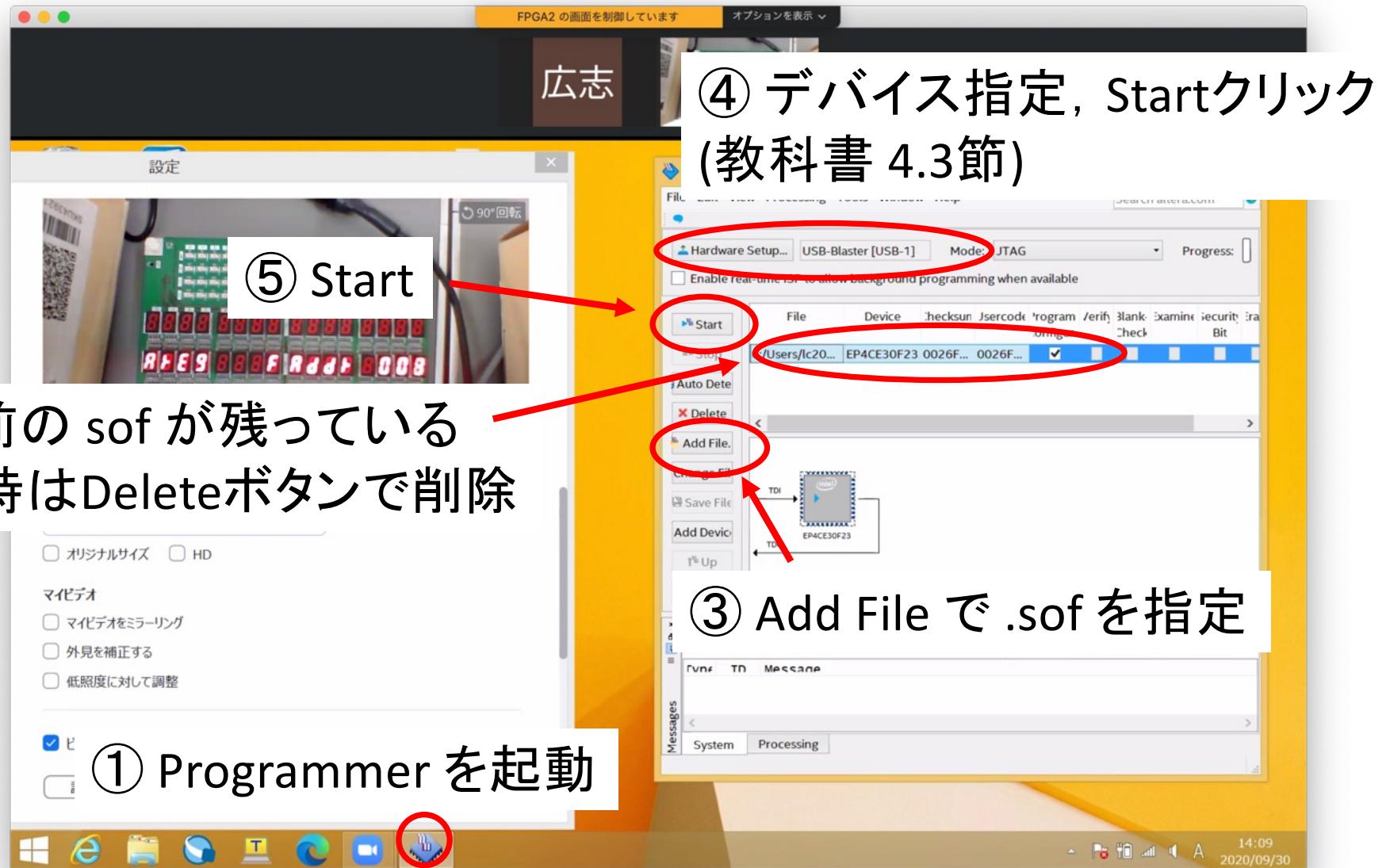
- Google Drive (Edgeの「お気に入り」からアクセス可) からsofをダウンロード



2MCU4b, 5b, 6bのシミュレーションの画面はMCU6bのもの

遠隔操作5

書き込むときはTeratermは閉じておく



遠隔操作6

① TeraTerm を起動

② この画面でキー入力。
入力した文字が出れば成功。

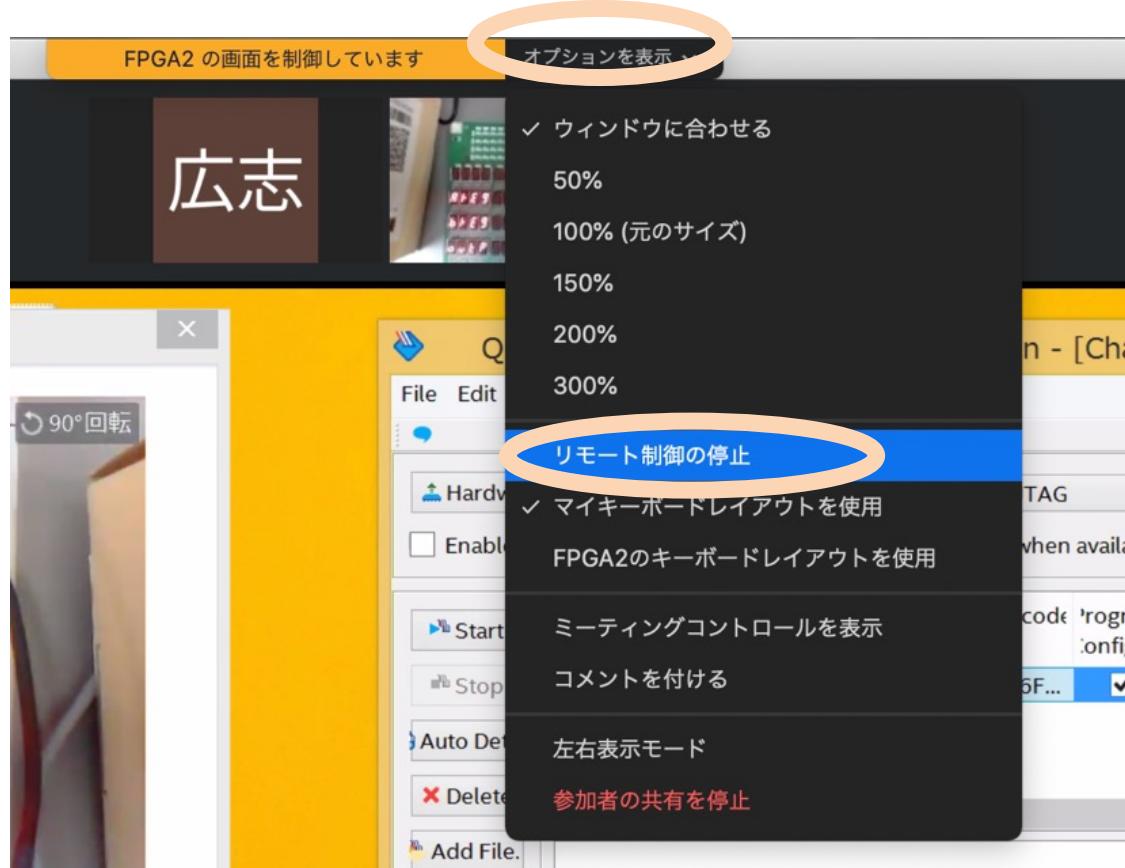
③ 入力キーに応じて
FPGA が変化

④ 実験が終わったら
TeraTermを閉じる

The image shows a composite screenshot of a computer desktop. On the left, there is a video feed from a camera labeled 'FJ Camera' with options for original size and HD. In the center, a terminal window titled 'COM3 Tera Term VT' shows the command 'eertt443'. A red dashed box highlights this window, and a red arrow points to the text input field. On the right, a larger terminal window titled 'Tera Term VT' also shows the command 'eertt443'. A red circle on the taskbar at the bottom left highlights the TeraTerm icon. The taskbar also includes icons for the Start button, Internet Explorer, File Explorer, and other system icons.

遠隔操作7

- 実験が終わったらリモート制御を終了



遠隔操作 注意点

- 実機検証が終わったら、.sof ファイルを削除してください。
- TeraTermを起動していると、Programmer が動きません。
- Programmer を使うときは、TeraTerm が閉じていることを確認すること。
- MCU3, MCU4 では音が出ます。**パソコンのスピーカー出力をONにすること。**
- キーボードを押してもFPGAが動かない場合、Programmerで再度書き込みを行うこと。