ISECON2017

アルゴリズムのハードウェア化から学ぶ問題解決教育プログラム

2018/3/31

東北大学工学部・工学研究科　阿部茂樹、横山梨香
東北大学医工学研究科　松浦祐司

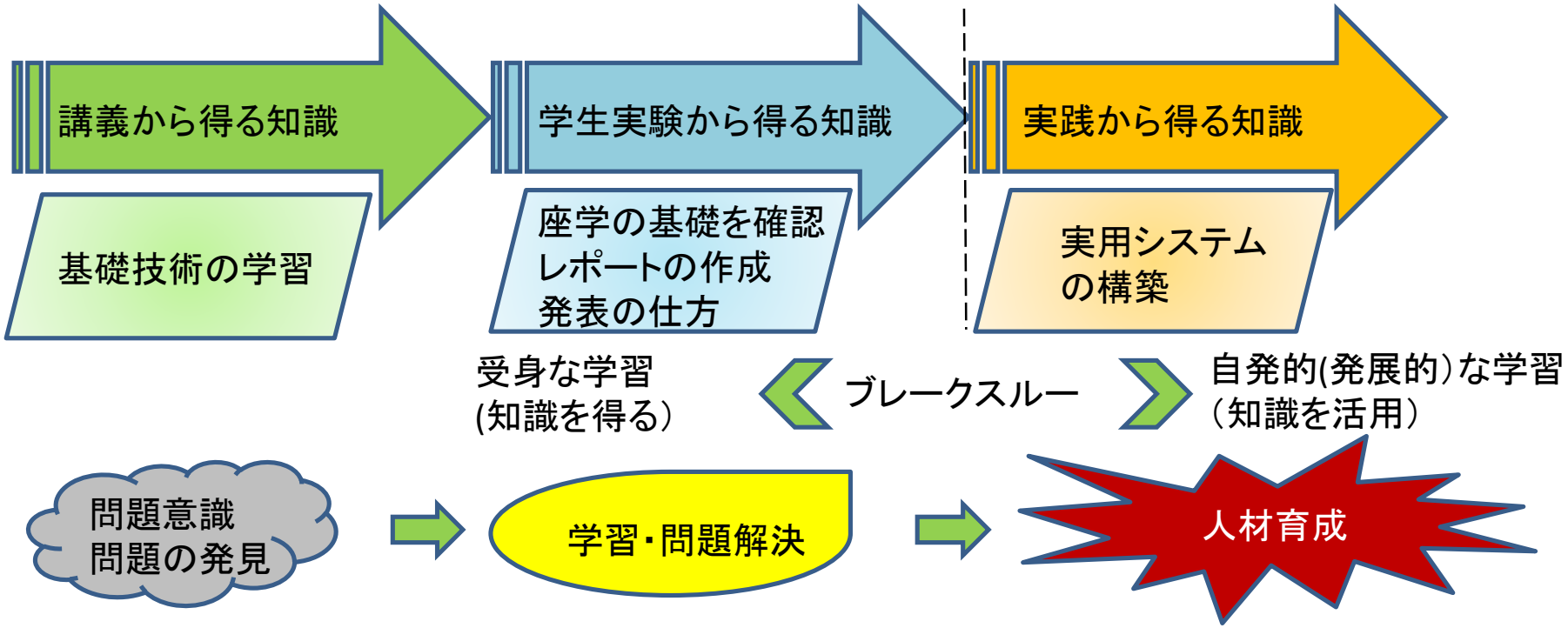
本資料について

- 本資料は、一般社団法人 情報処理学会 情報処理教育委員会 情報システム教育委員会主催による第10回情報システム教育コンテスト（ISECON2017）の本審査用資料を元に再編集されたものです。

- 本資料（阿部茂樹、横山梨香、松浦祐司,「アルゴリズムのハードウェア化から学ぶ問題解決教育プログラム」, ISECON2017, 2018.3.17）は、クリエイティブ・コモンズ 表示 4.0 国際 ライセンスの下に提供されています。

背景

社会のニーズに対応する実践的な教育プログラム
カリキュラムではないプログラムへの積極的参加



教育プログラムの対象者

- 学習意欲のある学生（予備知識が無くても可能）
- 予備知識に合わせた教育プログラム

実施している対象者

- 論理回路
- アルゴリズム
- プログラミング

に関して基礎的な知識のある大学生（学生実験の自由テーマ、創造工学実習等で実施）

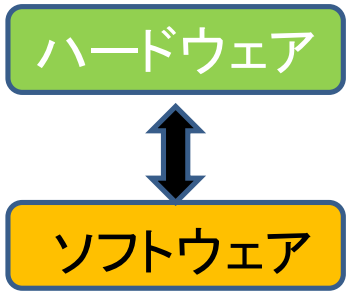
- 研究で必要となる回路技設計術の基礎を学びたい大学院生

検討している対象者

- 社会人の初任者研修プログラム
- 中高生を対象としたイベント　地域貢献

教育目標

- 論理回路の基礎から応用（計算機学の基礎を含む）
- アルゴリズムに関する知識習得、新たなアルゴリズム開発の意義
- 論理回路設計およびプログラミングを含む設計手法の習得
- C言語とハードウェア記述言語（HDL:Hardware Description Language)とのプログラム開発の違い
- 同機能のアルゴリズムの性能評価および性能向上のための設計学習

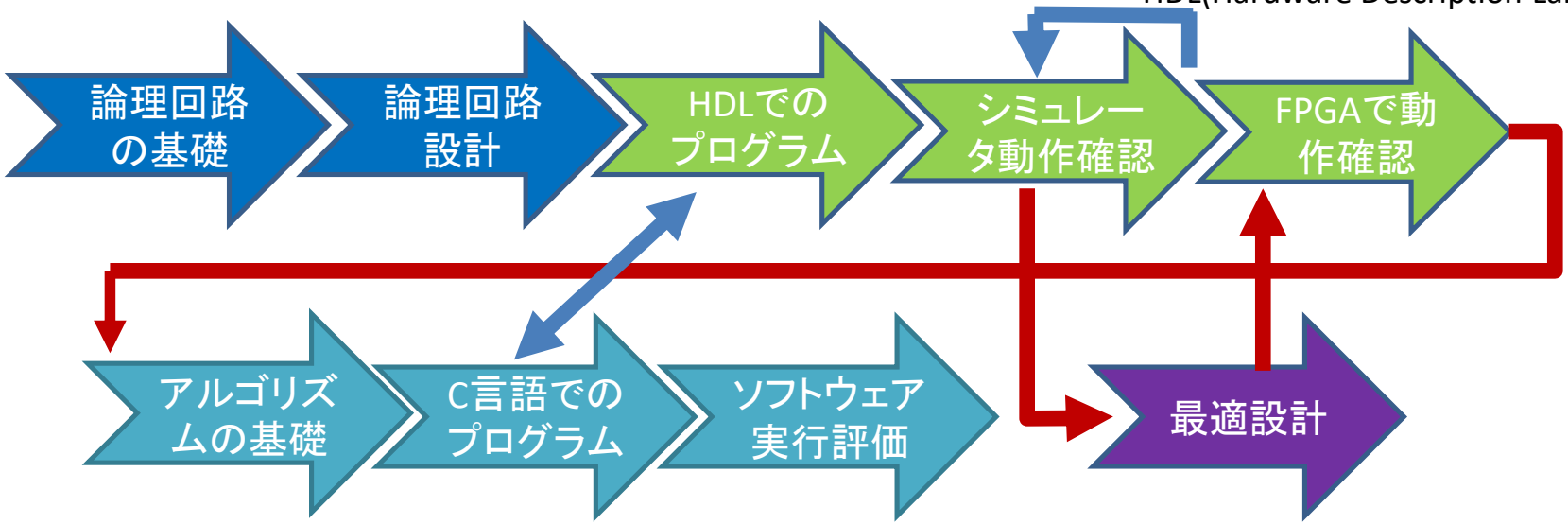


✓ 両面からのアプローチ
✓ 効率的な設計・開発
✓ 特長を活かしたシステム開発

本教育プログラムの概要

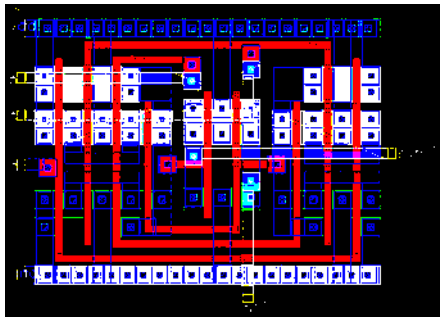
- 講義で得た知識を実システム開発に活かす能力向上
- 授業ではなく自主的な参加によるプログラム
- 予備知識に応じたレベルから実施

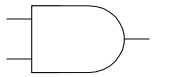
FPGA（Field Programmable Gate Array)
HDL(Hardware Description Language)



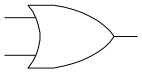
教育プログラムの内容

回路実現のための基礎知識

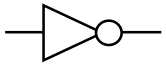




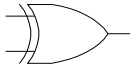
論理積



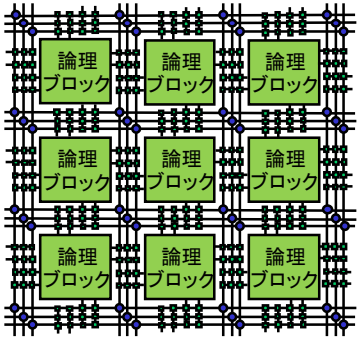
論理和



反転



排他的論理和



レイアウト設計

- 集積回路の大規模化・複雑化に伴い設計が困難
- 設計に関する専門の知識が要求
  - 複雑な設計ルール
- 長い設計期間と高いコスト
  - ルールチェックによる検証
  - 製造費用が高い
- 製造後の設計ミスは修正不可
  - 設計のやり直し不可

論理回路設計

- 回路の大規模化・複雑化に伴い設計が困難
- 回路の設計、製作が困難
  - 製作には時間が要
  - 製造費用は少ない
- 製造後の設計ミスは修正可能
  - 回路の変更は難しい
  - デバックが困難
- 専用の回路

FPGA設計

- 回路の大規模・複雑化にも対応可能
- 回路の設計はソフトウェア
  - 設計・製作は時間が短
  - 回路の再利用可能
  - 回路の修正・デバックが容易
- 回路の書き換えが可能

- 専用のソフトウェアが要
- プログラム技術が要

教育プログラム内容1（論理回路設計）

論理回路の基礎

NAND

In1	In2	Out
0	0	1
0	1	1
1	0	1
1	1	0

EXOR

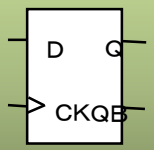
In1	In2	Out
0	0	0
0	1	1
1	0	1
1	1	0

A	B	Cin	SUM	Cout
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1

1bit フルアダーの真理値表

学習すること

種々の論理素子(回路図)
- 数系、補数、論理式の簡単化
- 組合せ回路（エンコーダ、デコーダ、マルチプレクサ等）
- 順序回路(フリップフロップ、シフトレジスタ、カウンタ)
- 状態遷移図



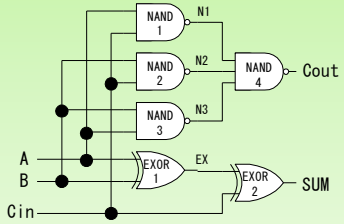
D-FF

D	CK	Q	QB
0	↑	0	1
1	↑	1	0

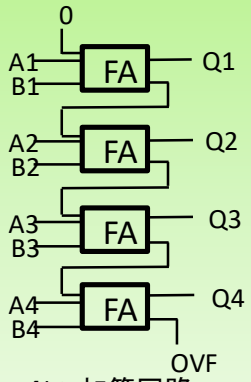
D-FFの動作

論理回路設計

組み合せ回路（加算器）

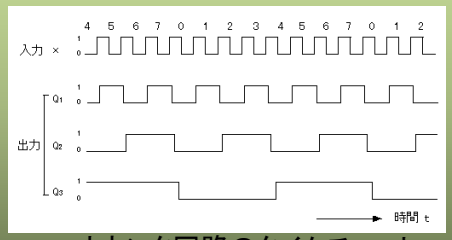


1bit フルアダー回路

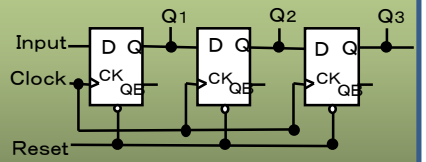


4bit 加算回路

順序回路（カウンタ）



カウンタ回路のタイムチャート



カウンタ回路

教育プログラム内容2(ハードウェア記述言語)

組み合せ回路

```
module adder4(a_in, b_in, SUM, OVF);
input [3:0] a_in, b_in;
output [3:0] SUM;
wire [2:0] CY;
output OVF;
      adder AD0 (a_in[0], b_in[0],   1'b0, SUM[0], CY[0]);
      adder AD1 (a_in[1], b_in[1], CY[0], SUM[1], CY[1]);
      adder AD2 (a_in[2], b_in[2], CY[1], SUM[2], CY[2]);
      adder AD3 (a_in[3], b_in[3], CY[2], SUM[3],  OVF);
endmodule
```

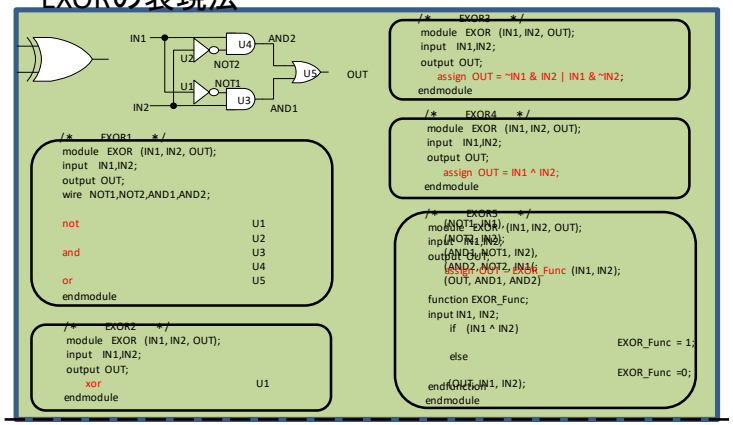
4ビット加算器（メインプログラム）

```
module adder(A, B, Cin, SUM, Cout);
input A, B, Cin;
output SUM, Cout;
wire N1,N2,N3,EX;
          nand nand1(N1, A, Cin),
               nand2(N2, B, Cin),
               nand3(N3, A, B),
               nand4(Cout, N1, N2, N3);
        xor exor1(EX, A, B),
            exor2(SUM, EX, Cin);
```

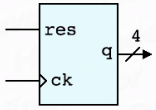
1ビット加算器

学習すること
- HDLの構文
- 同期回路のノンブロッキング代入法
- 並列計算が可能

EXORの表現法

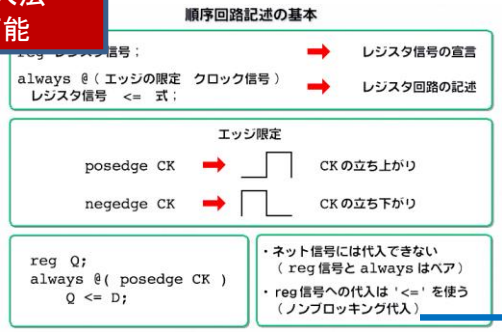


順序回路




```
module counter( ck,  res,  q );
input           ck,  res;
output  [3:0]   q;
reg     [3:0]   q;

always @( posedge ck or posedge res  )
begin
    if (  res  )
        q <= 4'h0;
    else
        q <= q + 4'h1;
end

endmodule
```

4ビットカウンタの基本プログラム



FPGAボード上での動作確認

- LEDでの確認→クロック周波数が速くて目視での動作確認ができない
- クロックを分周して目視できる速さで動作

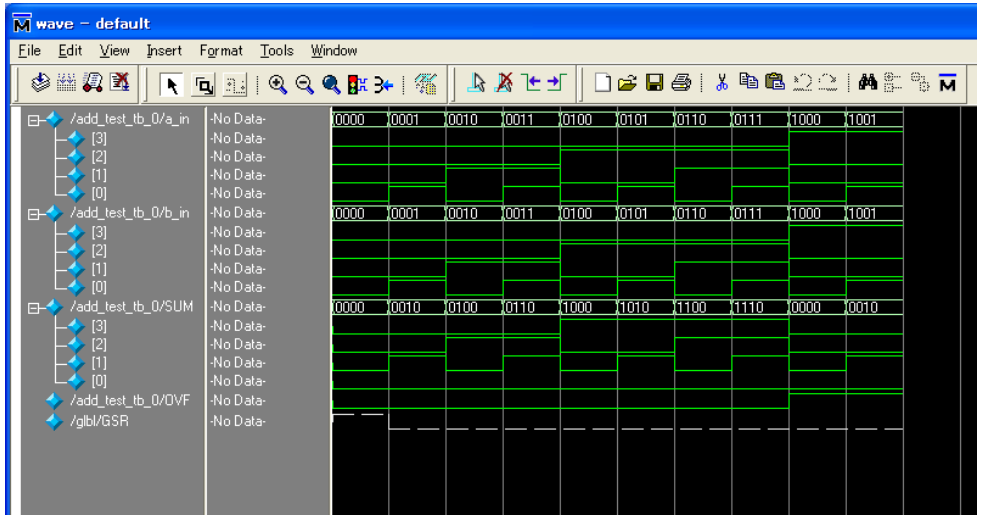
初期値 a=1, b=2, c=3	C言語	HDL
a=b	a=2	a=2
b=c	b=3	b=3
c=a	c=2	c=1

同期回路が可能

教育プログラム内容（シミュレーション）

検証用プログラム

```
`timescale 1ns/1ps

module add_test_tb_0;
   reg [3:0] a_in = 4'b0000;
   reg [3:0] b_in = 4'b0000;
   wire [3:0] SUM;
   wire OVF;

   adder4 UUT (
      .a_in(a_in),
      .b_in(b_in),
      .SUM(SUM),
      .OVF(OVF));

   initial begin
      // ------------- Current Time:
100ns
      #100;
      a_in = 4'b0001;
      b_in = 4'b0001;
      // ------------- Current Time:
200ns
      #100;
      a_in = 4'b0010;
      b_in = 4'b0010;
      // ------------- Current Time:
300ns
      #100;
      a_in = 4'b0011;
      b_in = 4'b0011;
      // ------------- Current
Time:  400ns
      #100;
      a_in = 4'b0100;
      b_in = 4'b0100;
      // ------------- Current
Time:  500ns
      #100;
      a_in = 4'b0101;
      b_in = 4'b0101;
      // ------------- Current
Time:  600ns
      #100;
      a_in = 4'b0110;
      b_in = 4'b0110;
      // ------------- Current
Time:700ns
      #100;
      a_in = 4'b0111;
      b_in = 4'b0111;
      // ------------- Current
Time:  800ns
      #100;
      a_in = 4'b1000;
      b_in = 4'b1000;
      // ------------- Current
Time:  900ns
      #100;
      a_in = 4'b1001;
      b_in = 4'b1001;
   end
endmodule
```



シミュレーションによる動作の確認

学習すること

- 検証用プログラム(テストベンチの作成法(プログラムあるいは入力波形))
- シミュレーション波形からの動作確認（デバッグへの活用）

教育プログラム内容（FPGA)

■FPGAの基礎
- ■論理回路、ディジタル回路設計
- ■FPGAとは何か？　FPGAの構造や特徴

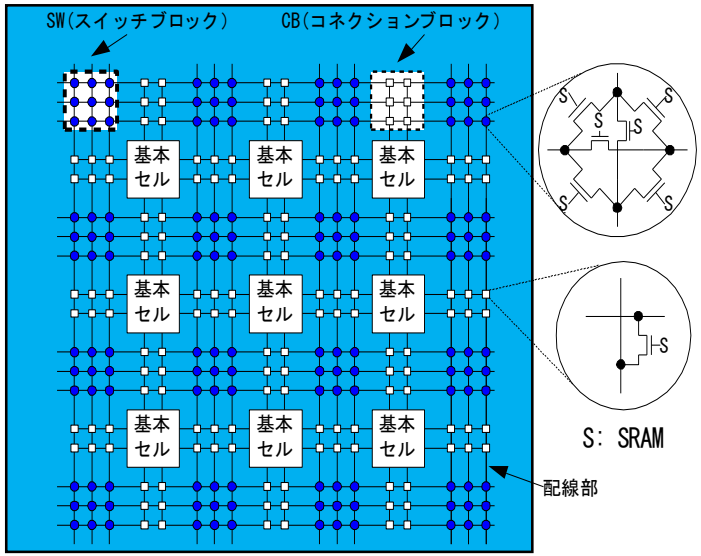
■ISE Design Suiteの使い方
- ■統合環境およびシミュレータの使い方、手順
- ■デバック機能、FPGAへのダウンロードおよびPROMへの書き込み
- ■動作検証

学習すること
- ・FPGAの特徴（再構成可能デバス）
- ・エラーメッセージによる解決法
- ・レポートからの情報収集（活用）

論理合成レポート

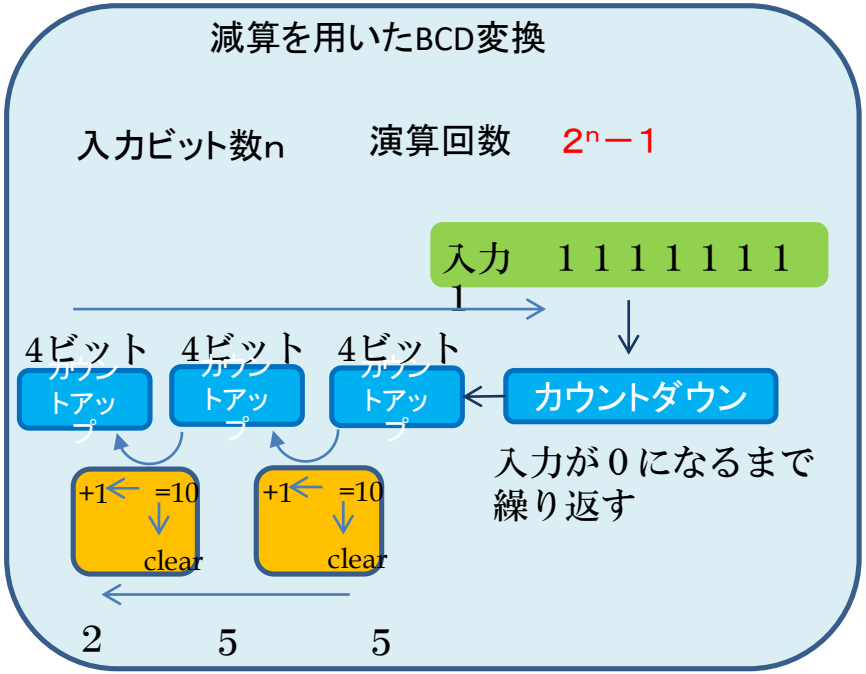
```
Device utilization summary:
---------------------------
Selected Device : 3s50pq208-5
 Number of Slices:              4  out of    768    0%
 Number of 4 input LUTs:        7  out of   1536    0%
 Number of IOs:                13
 Number of bonded IOBs:        13  out of    124   10%
Timing Summary:
---------------
Speed Grade: -5

   Minimum period: No path found
   Minimum input arrival time before clock: No path found
   Maximum output required time after clock: No path
found
   Maximum combinational path delay: 10.613ns
```

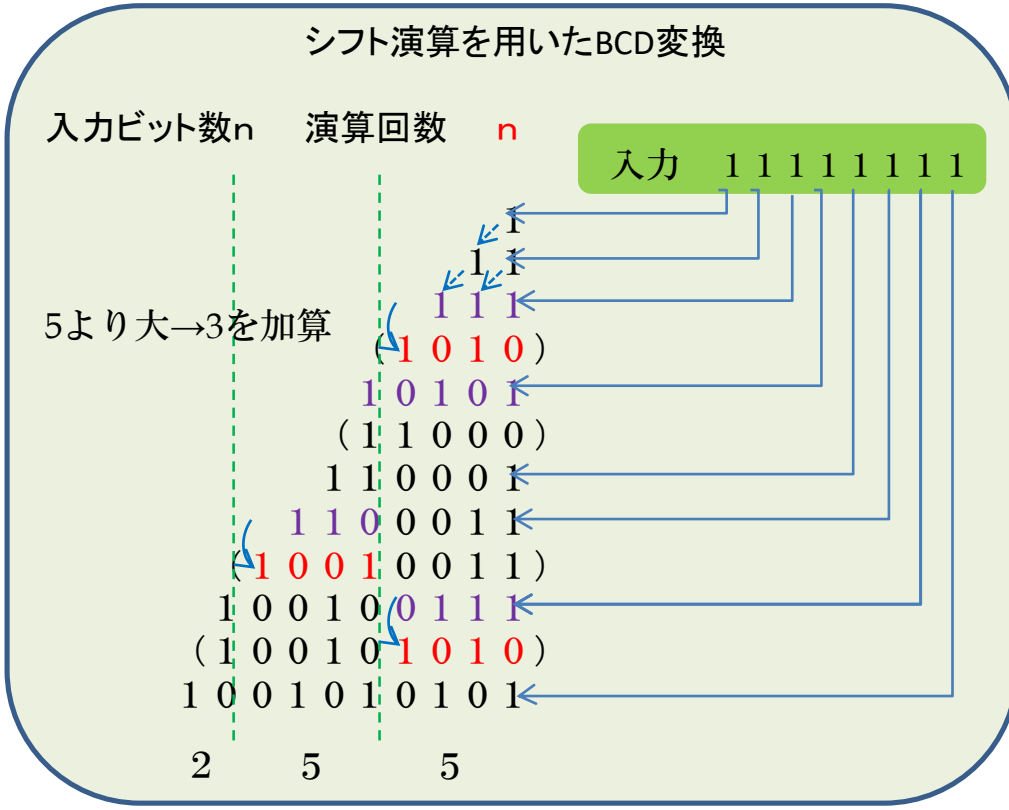
FPGAの特徴と内部構造
- 基本的な論理構造がチップ上に作成
- 設計者が自由にプログラムできるIC
- 回路構造が可変
- 種々の回路を同一ハードウェア上で実現



教育プログラム内容3（アルゴリズム）

- アルゴリズムとは何か？
- アルゴリズム開発はなぜ必要なのか？

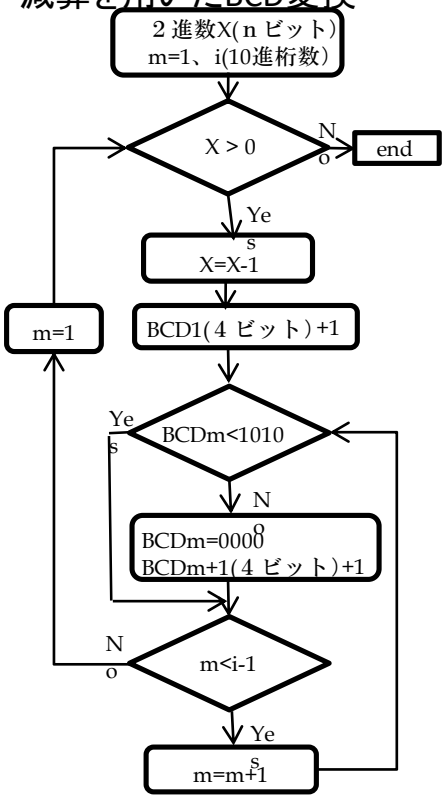




学習すること
- 2つのアルゴリズムの理解、アルゴリズムの重要性
- 演算速度の評価

教育プログラム内容3（BCD変換評価）

減算を用いたBCD変換



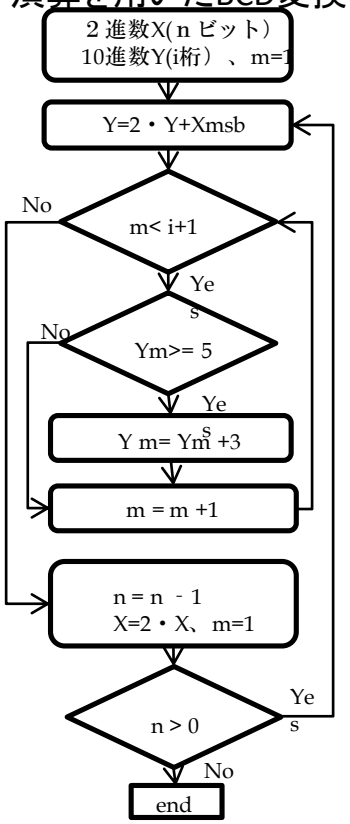
	減算を用いた変換		シフト演算を用いた変換	
Optimization Goal	Speed	Area	Speed	Area
Number of logic	91	76	121	88
Minimum Period	4.52ns	6.372ns	3.878ns	4.737ns

$2^n - 1$ (1,153ns)　　n 31ns

8bitで255のとき(Speed優先)
シフト演算方式は、約37倍の速さ

評価に使用した
FPGAボード

シフト演算を用いたBCD変換



学習すること

- 2つのアルゴリズムをC言語とHDLで記述し、比較
- FPGAで動作させて速度評価（レポートを利用）

教育プログラム内容4（ソートアルゴリズム）

バブルソートアルゴリズム

Ⅰ.	50	60	20	40	80	10	30	70
		①	②	③	④	⑤	⑥	⑦
Ⅱ.	50	20	40	60	10	30	70	80
Ⅲ.	20	40	50	10	30	60	70	80
Ⅳ.	20	40	10	30	50	60	70	80
Ⅴ.	20	10	30	40	50	60	70	80
Ⅵ.	10	20	30	40	50	60	70	80

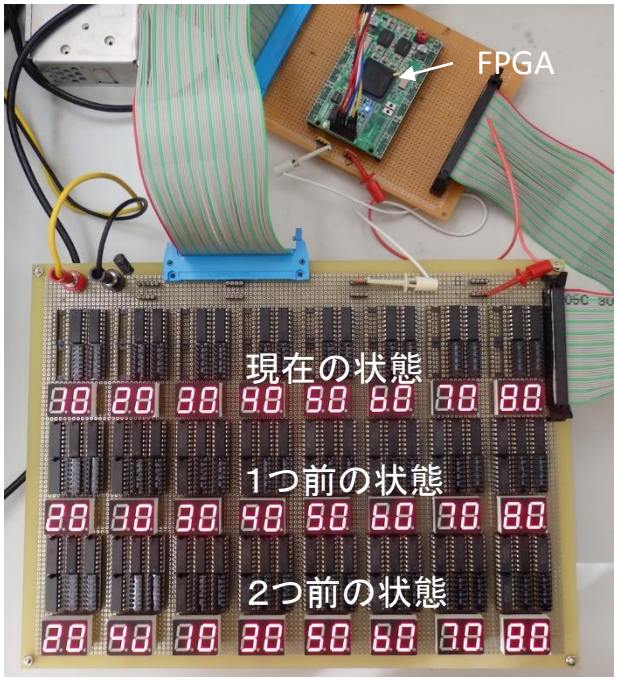
学習すること
- バブルソートアルゴリズムの理解
- C言語とHDLでの記述および
- 可視化ボードを使った動作の確認

C言語プログラム

```
bubblesort (int data[ ], int n)
  {  int temp, i, j;

   for(i=0; i < n-1; i++)
        for(j=0; j<n-i-1; j++)
                if(data[j] > data[j+1])
                {   temp = data[j];
                    data[j] = data[j+1];
                    data[j+1] = temp;
                }
   }
```

例
C言語：8行
↓
HDL:200行

毎回7回比較する単純プログラム



ソートアルゴリズムの可視化

教育プログラム4内容（バブルソート評価）

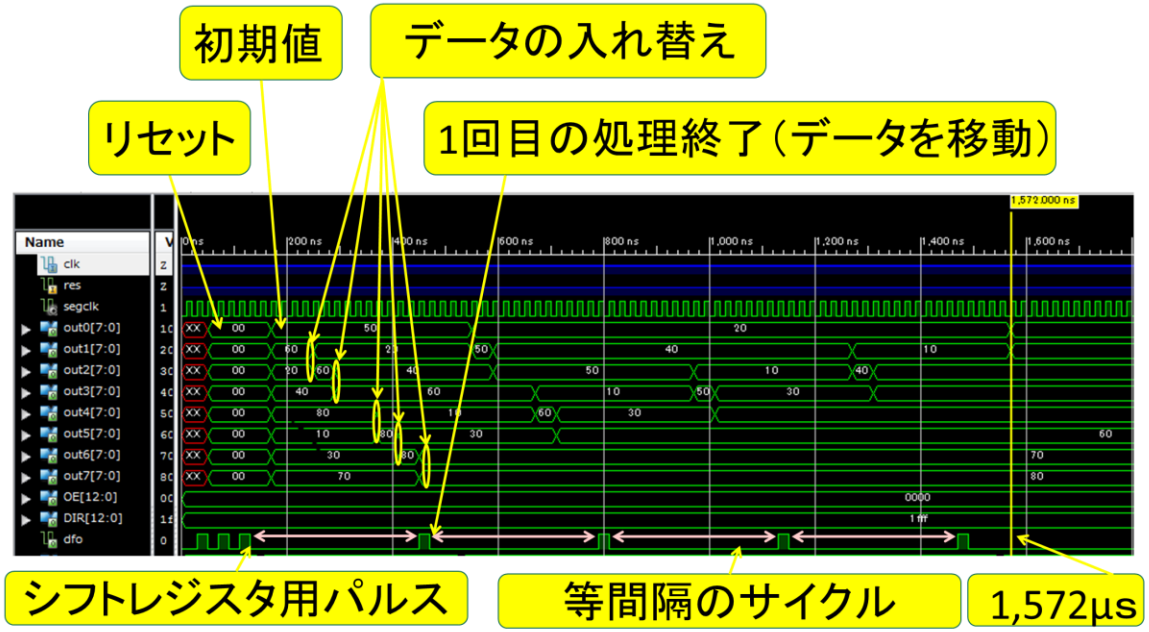
バブルソートアルゴリズム

Ⅰ.	50	60	20	40	80	10	30	70
		①	②	③	④	⑤	⑥	⑦
Ⅱ.	50	20	40	60	10	30	70	80
Ⅲ.	20	40	50	10	30	60	70	80
Ⅳ.	20	40	10	30	50	60	70	80
Ⅴ.	20	10	30	40	50	60	70	80
Ⅵ.	10	20	30	40	50	60	70	80

演算回数：7回の比較を5行　→　35回

1列演算毎に1回減　→　25回

演算回数の少ない方法　→　20回



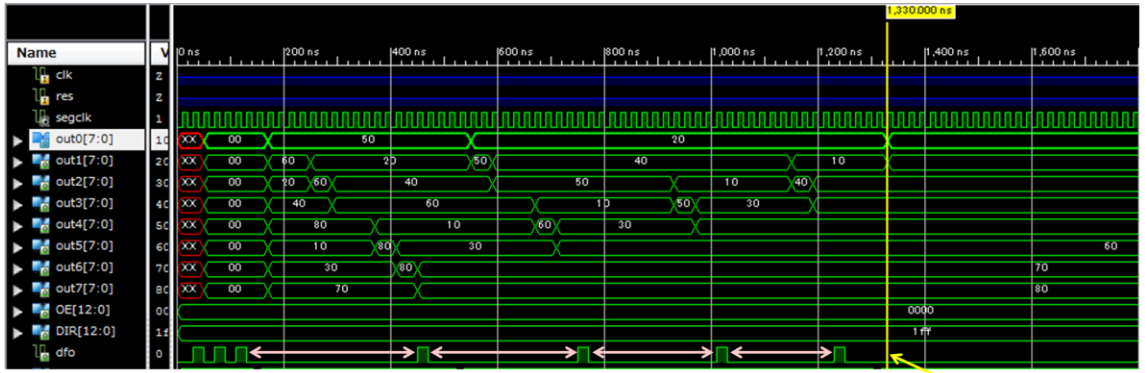
A)　C言語プログラム同様毎回7回の比較演算

学習すること
・シミュレータでの動作確認、HDLでの誤りなどの検証
・シミュレータ波形から、無駄なサイクルを見つけ出し、演算回数の少ないプログラムを作成

教育プログラム4内容（バブルソートの高速化）

1列の処理ごとに右から配置が決定

Ⅰ.	80	70	60	50	40	30	20	10
Ⅱ.	70	60	50	40	30	20	10	80
Ⅲ.	60	50	40	30	20	10	70	80
Ⅳ.	50	40	30	20	10	60	70	80
Ⅴ.	40	30	20	10	50	60	70	80
Ⅵ.	30	20	10	40	50	60	70	80
Ⅶ.	20	10	30	40	50	60	70	80
Ⅷ.	10	20	30	40	50	60	70	80



1回の処理毎に比較演算減

1,330μs

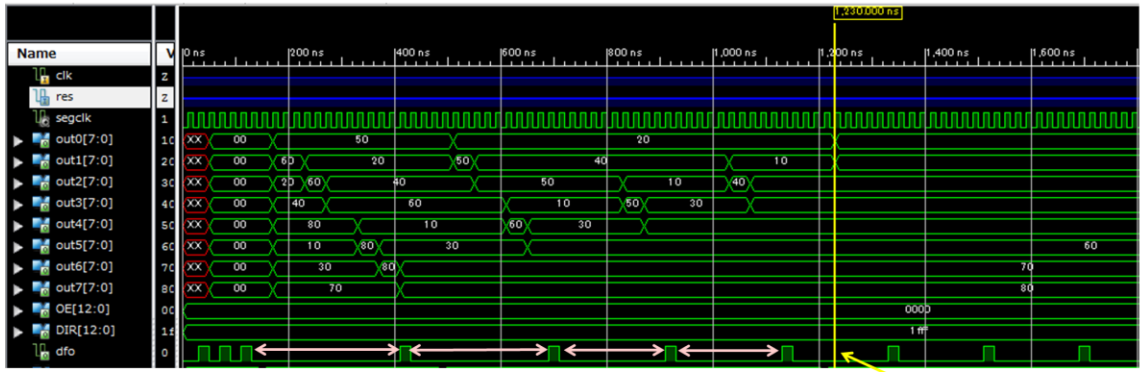
B) 毎回の演算で右から配置が決定（演算を1回づつ減）

1回で複数のデータ配置が決定

Ⅰ.	50	60	20	40	80	10	30	70
	0	1	1	0	1	1	1	
Ⅱ.	50	20	40	60	10	30	70	80
	1	1	0	1	1	0	0	
Ⅲ.	20	40	50	10	30	60	70	80
	0	0	1	1	0	0	0	
Ⅳ.	20	40	10	30	50	60	70	80
	0	1	0	0	0	0	0	
Ⅴ.	20	10	30	40	50	60	70	80
	1	0	0	0	0	0	0	
Ⅵ.	10	20	30	40	50	60	70	80

データ交換が無⇒ 0
データ交換が有⇒ 1

右から連続する0の時にはデータ交換が不要



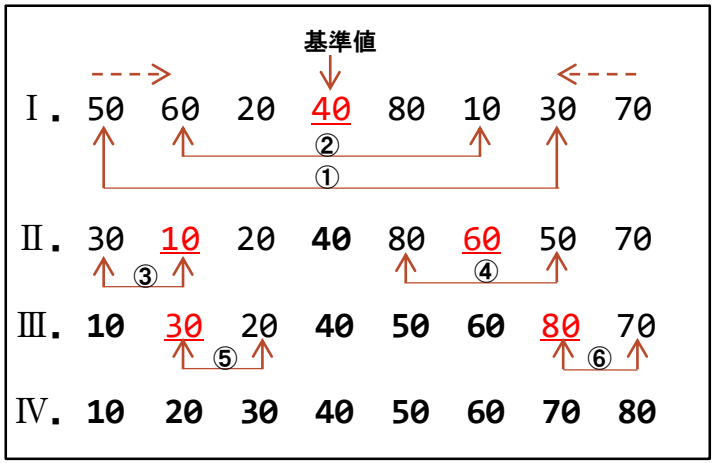
C) 1回の演算で複数の配置が決定

1,230μs

学習すること
・バブルソートの演算回数を減らすアルゴリズムを考える

教育プログラム内容4（クイックソート）

クイックソートアルゴリズム



赤字が基準値

太字が確定

C言語プログラム

```
qiuicksort(int data[ ], int high, int low)
  { int mid, temp i, j;

    i= high;
    j= low;
    mid=data[(low+high)/2];
    do {
        while(data[i]  < mid) i++;
        while(mid  < data[j]) j--;
         if( i <= j)
            { temp = data[i];
              data[i] = data[j];
              data[j] = temp;
               i++;
               j--;
            }
        }
       while(i<=  j);
         if (high  < j ) quicksort(data, high, j);
         if ( i < low) quicksort(data, i, low);
   }
```

再帰的プログラム

学習すること
- クイックソートアルゴリズムの理解
- C言語とHDLによるプログラム作成（再帰的プログラムをHDLでどのようにプログラムするか？）
- バブルソートアルゴリズムとの比較
- FPGAでの動作確認とデバッグ
- 応用としてそのほかのソートアルゴリムとの比較

教育効果

- 論理回路、計算機学、アルゴリズム論、ディジタル制御などを本テーマで実践
- カリキュラムとしてではなく、あくまで自主的な実習（講義の合間、講義終了後）(単位にはならないが参加しやすい）
- 実施時間、実施内容はすべて学生の知識レベルに合わせる

◆アンケートから見える教育効果
（学生実験自由テーマおよび創造工学センター実習より）
・HDL、FPGAを使うことによる短時間でのハードウェア化が面白い
・アルゴリズムによって演算時間が異なることが理解できた
・自分でも新しいアルゴリズムを考えてみたい
・実施時間や学習内容を自分で決めることができ、精神的に楽
・空いている時間を有効利用できる

まとめ

■積極的に技術を習得する姿勢を学ぶ
■社会的ニーズを把握できる力を養う
　→物の考え方・見方を多面的に思考できる人材育成
　→技術力および知識が向上することの楽しさを知る

■「やる気」を出させることができるテーマの発掘が必要
　→やる気のある学生はたくさんいる
　→選択できるテーマが必要
　問題は指導者の不足

■今後は地域貢献も考慮
　→中高生向け教育プログラムへ改良
　→関連する分野の社会人（新入社員）向けプログラムの考案