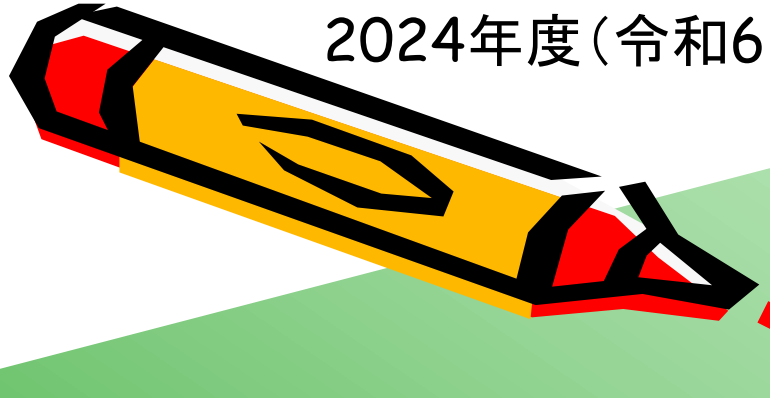
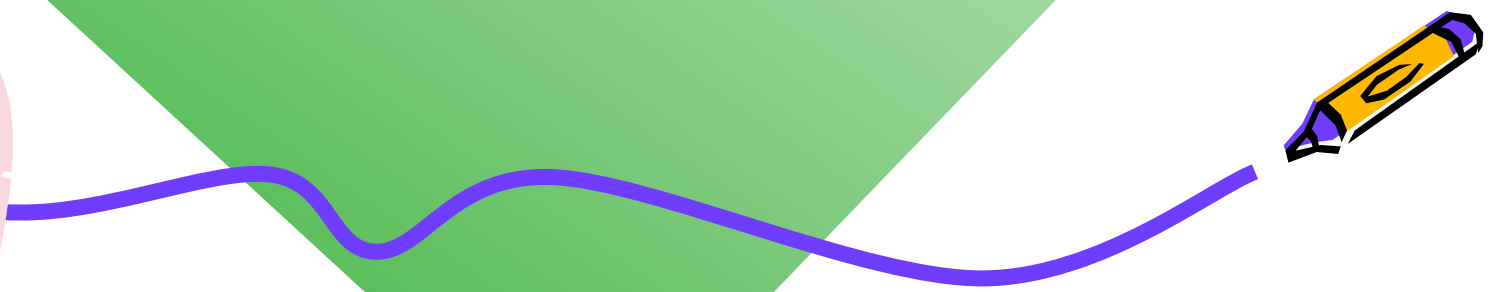
2024年度（令和6年）版

Ver. 2024-11-18a



Computer Design Contest

- The purpose of the contest
  - Design *a high-performance computer system* of your own which at least achieves better performance than the baseline



リファレンスデザインのプロジェクトのコピー

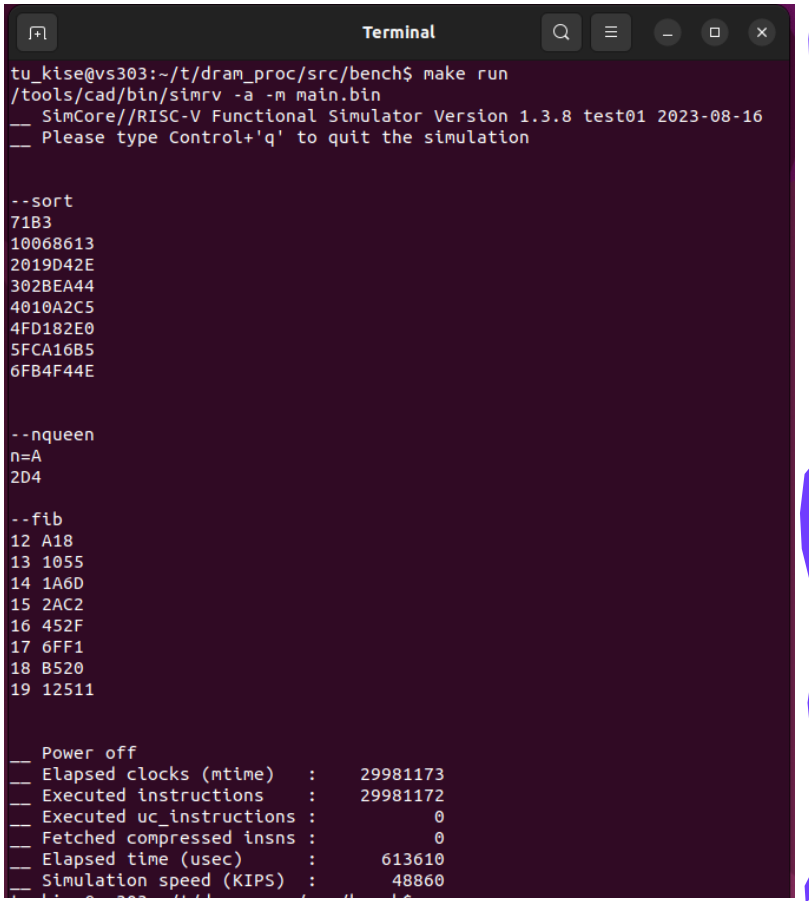
```
$ cd ~/ca
$ tar xfz /home/tu_kise/ca/dram_proc_V12.tgz
```

- 自分のホームディレクトリに Computer Architecture用のディレクトリ ca を作成しておくこと.
- 上のコマンドで，自分のホームディレクトリの下の ca というディレクトリに，ファイルを展開する.
- ca のディレクトリの中に，dram_proc というフォルダが作成されて，その中に必要となるファイルが格納される.

使い方 (1) ベンチマークプログラムの作成とテスト

```
$ cd ~/ca/dram_proc/src/bench
$ make
$ make run
```

- 上の最初のコマンドで，ベンチマークプログラムの開発環境が格納されているディレクトリに移動する.
- 2番目のコマンド make で，ベンチマークプログラムをコンパイルする.
- ~/ca/dram_proc/src に、次のファイルがコピーされる.
  - sample1.mem 生成されたベンチマークの内容を16進数で出力したテキスト形式のファイル
  - **sample1.bin** 生成されたベンチマークをバイナリ形式で保存したファイル
- 3番目のコマンド make run で、機能レベルのシミュレータでベンチマークプログラムを実行する。
- main.c を修正して試してみる.



ベンチマークプログラム main.c

指示される最新の版
を用いること.

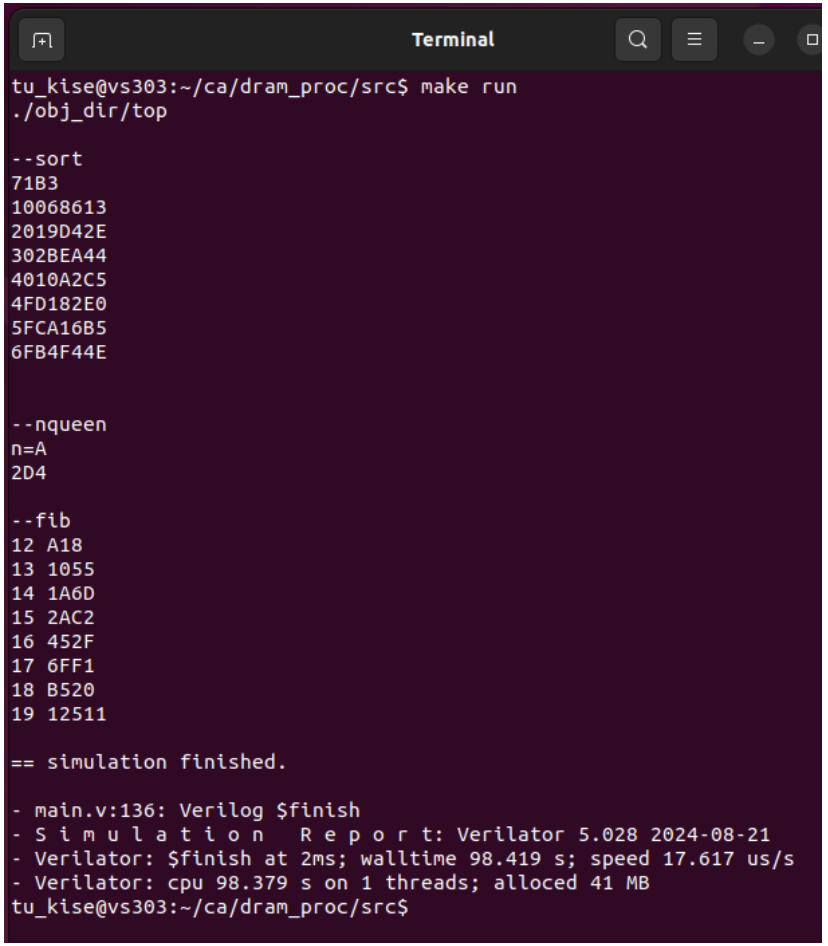
```
/*****************************************************************************/
/**** benchmark prog. for Computer Design Contest V1.0 ArchLab,Science Tokyo */
/*****************************************************************************/
#include "simrv.h"

#define SORTN (1024*128) // 128K word = 512KB data
#define QUEEN 10
#define FIBN  25

/*****************************************************************************/
int xorshift(int a)
{
    int x = a;
    x ^= x << 13;
    x ^= x >> 17;
    x ^= x << 5;
    a = x;
    return x;
}

/*****************************************************************************/
int nqueen (int i, int a[], int b[], int c[], int x[], int N)
{
    int j;
    int ret = 0;
    for (j = 0; j < N; j++) {
        if (a[j] && b[i + j] && c[i - j + N - 1]) {
            x[i] = j;
            if (i < N - 1) {
                a[j] = b[i + j] = c[i - j + N - 1] = 0;
                ret += nqueen (i + 1, a, b, c, x, N);
                a[j] = b[i + j] = c[i - j + N - 1] = 1;
            } else {
                return 1;
            }
        }
    }
    return ret;
}

/*****************************************************************************/
int fibonacci (int n)
{
    if (n == 1) return 1;
    if (n == 2) return 1;
    return fibonacci(n - 1) + fibonacci(n - 2);
    return n + 3;
}

/*****************************************************************************/
void qsort (int a[], int first, int last)
{
    int i,j;
    int x,t;
    x = a[(first + last) / 2];
    i = first;
    j = last;
    for ( ; ; ) {
        while (a[i] < x) i++;
        while (x < a[j]) j--;
        if (i >= j) break;
        t = a[i];
        a[i] = a[j];
        a[j] = t;
        i++;
        j--;
    }
    if (first < i - 1) qsort(a, first, i - 1);
    if (j + 1 < last)  qsort(a, j + 1, last);
}
```

```
int main(void)
{
    int i, j;
    /*****************************************************/
    int seq[SORTN];

    simrv_putc('\n');
    simrv_putc('-');
    simrv_putc('-');
    simrv_putc('s');
    simrv_putc('o');
    simrv_putc('r');
    simrv_putc('t');
    simrv_putc('\n');
    seq[0] = 463534242;
    for (i = 1; i < SORTN; i++) {
        seq[i] = xorshift(seq[i-1]) & 0x7fffffff;
    }

    qsort(seq, 0, SORTN - 1);

    for (i = 0; i < SORTN; i=i+SORTN/8) {
        simrv_puth(seq[i]);
        simrv_putc('\n');
    }
    simrv_putc('\n');

    /*****************************************************/
    int a[QUEEN];
    int b[2 * QUEEN - 1];
    int c[2 * QUEEN - 1];
    int x[QUEEN];

    simrv_putc('\n');
    simrv_putc('-');
    simrv_putc('-');
    simrv_putc('n');
    simrv_putc('q');
    simrv_putc('u');
    simrv_putc('e');
    simrv_putc('e');
    simrv_putc('n');
    simrv_putc('\n');
    simrv_putc('n');
    simrv_putc('=');
    simrv_puth(QUEEN);
    simrv_putc('\n');
    for (i = 0; i < QUEEN; i++)         a[i] = 1;
    for (i = 0; i < 2 * QUEEN - 1; i++) b[i] = 1;
    for (i = 0; i < 2 * QUEEN - 1; i++) c[i] = 1;
    for (i = 0; i < QUEEN; i++)         x[i] = 0;
    int num = nqueen (0, a, b, c, x, QUEEN);
    simrv_puth(num);
    simrv_putc('\n');
    simrv_putc('\n');

    /*****************************************************/
    simrv_putc('-');
    simrv_putc('-');
    simrv_putc('f');
    simrv_putc('i');
    simrv_putc('b');
    simrv_putc('\n');
    for (i = FIBN-7; i <= FIBN; i++) {
        int val = fibonacci(i);
        simrv_puth(i);
        simrv_putc(' ');
        simrv_puth(val);
        simrv_putc('\n');
    }
    simrv_putc('\n');

    simrv_exit();
}
```

使い方 (2) RTLシミュレーション

```
$ cd ~/ca/dram_proc/src
$ make
$ make run
```

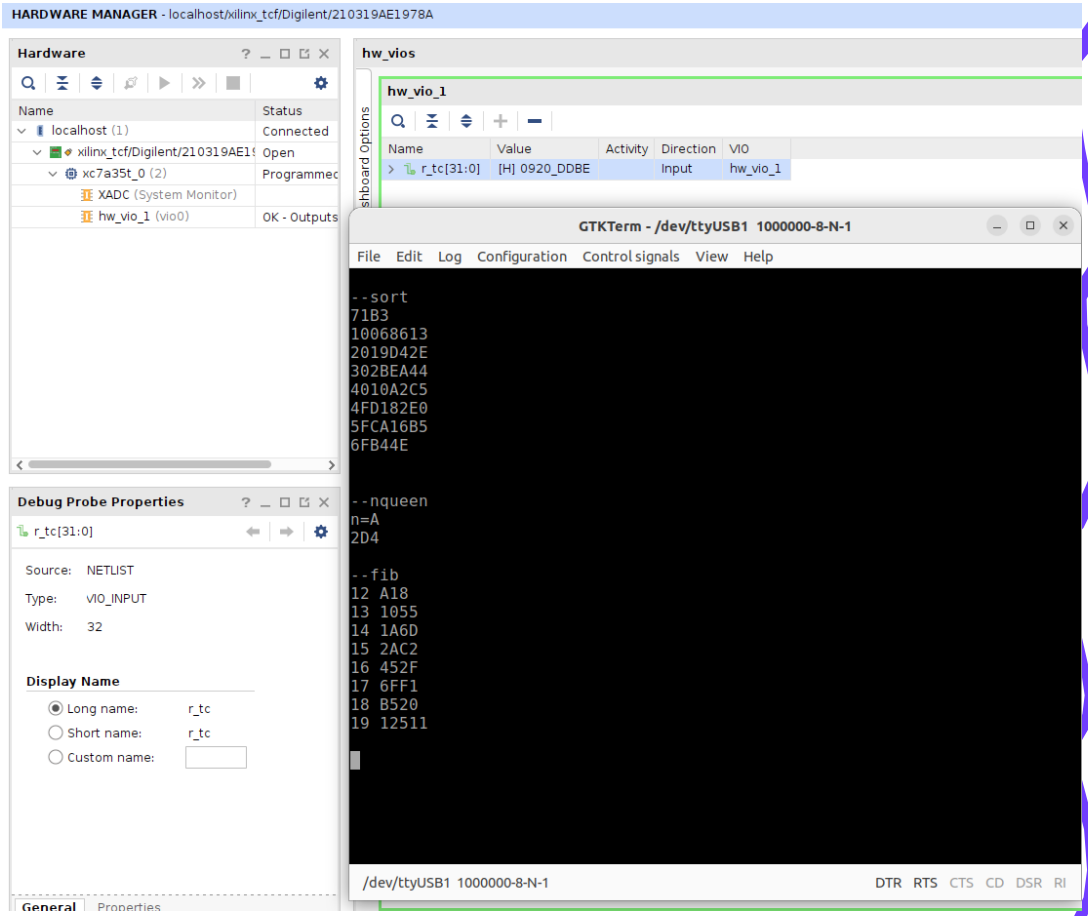
- 上の最初のコマンドで，Verilog HDLのコード等が格納されているディレクトリに移動する．
- 2番目のコマンド make で，verilator を用いて，Verilog HDLのコードをコンパイルする．
- 3番目のコマンド make run で、サイクルレベルのRTLシミュレーションとしてベンチマークプログラムを実行する．（この実行は遅いので注意すること．）
- このフォルダには，正しいPCの系列をテキスト形式で出力して，gzip で圧縮した verify.txt.gz があるので，これを用いて，設計したシステムの正しさを確認すること．



使い方 (3) FPGAで動かす

```
$ cd ~/ca/dram_proc/vivado
$ /tools/Xilinx/Vivado/2024.1/bin/vivado dram_proc.xpr
```

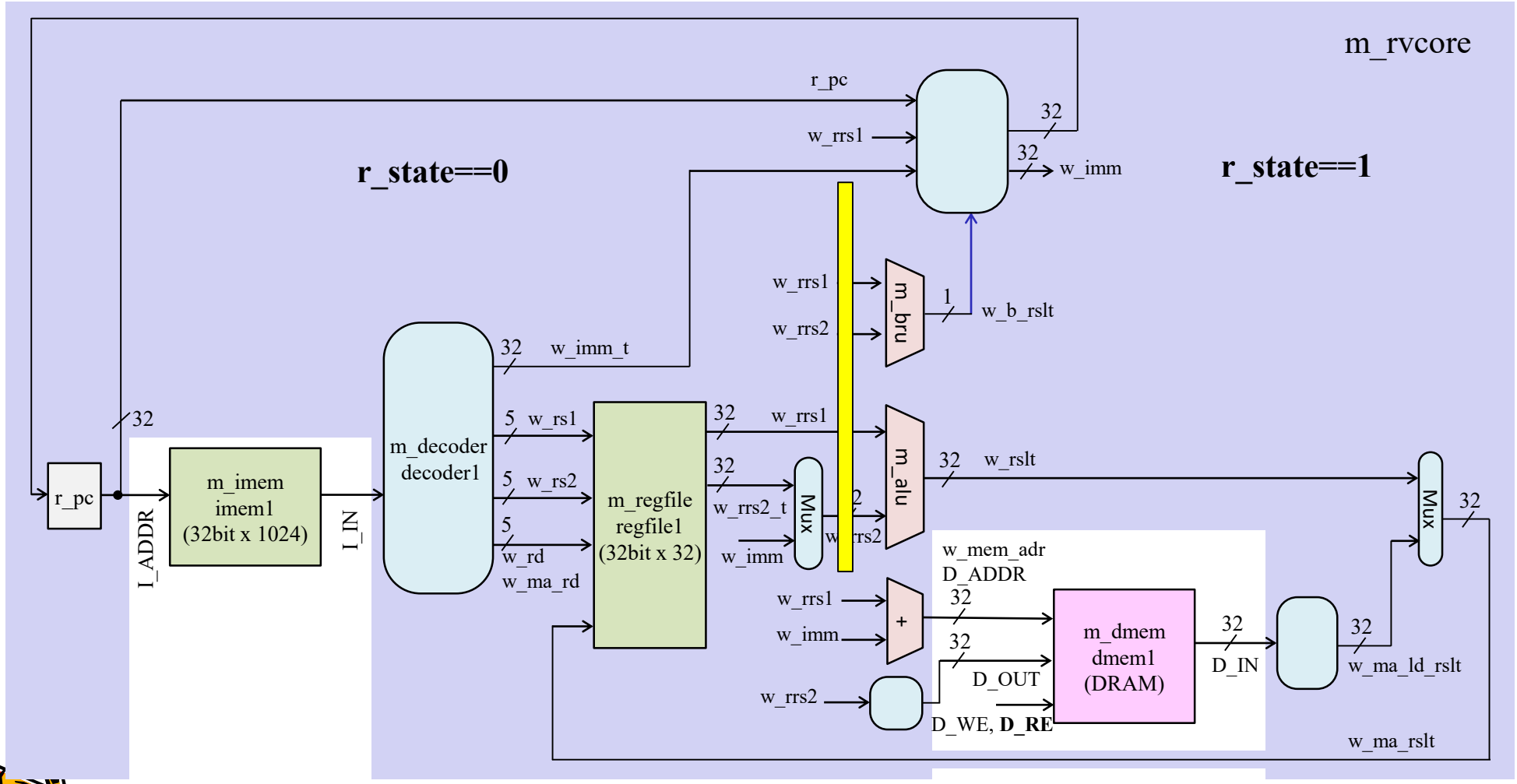
- 上の最初のコマンドで，Vivadoのプロジェクトファイルが格納されているディレクトリに移動する．
- 2番目のコマンドで，プロジェクトファイル dram_proc.xpr を指定して，Vivado を起動する．
- Vivado が起動したら，bitstream ファイルを生成する．（初回はIPのための論理合成，配置・配線が必要なので少し時間がかかる．）
  - 全ての論理合成，配置・配線，bitstreamファイルの生成まで6分程度．
- Hardware Manager を開いて，FPGAをコンフィギュレーションする．
- gtkterm を開いて，~/ca/dram_proc/src の sample1.bin を転送する．



module m_rvcore (RV32I, multi-cycle processor)

40MHz operating frequency

lb, lbu, lh, lhu, sb, sh are not supported, DRAM is not initialized through UART



コンピュータ設計コンテストのルール

- コンピュータの高速化に取り組み，コンテスト形式で成果を競う．
- RISC-Vプロセッサを含むコンピュータを設計する．
  - サポートする命令: ベンチマークプログラムに含まれるRISC-Vの命令．
  - 与えられたプログラムの**実行時間が短いコンピュータの設計・実装**を目指すこと．
  - Vivadoのタイミング制約を満たすように設計すること．
  - PCの系列が，提供する verify.txt の系列と同じことをシミュレーションで確認すること．
  - VIO, Clocking Wizard, MIG により生成される IP を除いて，Verilog HDL で設計すること．
  - 記述したソースコード等のファイル一式を提出すること．
- 利用するFPGAボードは Digilent Arty A7-35T とする．
- 命令メモリのサイズは1024ワード (4KB) とする．
- 1Mbaudのシリアル通信で，4KBの与えられたプログラムを送信して処理を開始する．
- リファレンスデザインに含まれる Verilog HDL のコードを利用してもよい．それ以外の Verilog HDL のコードは自分で記述（他人のものからのコピー＆ペーストはNG）すること．
- VIOに表示される値から，実行に必要となったサイクル数を取得すること．
- Clocking Wizard で指定した動作周波数と，実行サイクル数から実行時間を求めること．

Ver. 2024-11-18a

Presentation slide (PowerPoint file) and code

- 設計コンテストのためのスライドを準備する．サンプルファイルを参考にすること．
- 当日はスライドを用いて発表する．発表時間は1人 4分以内とする．
- スライドには次を含めること．
  - リファレンスデザイン (40MHz) に対する速度向上率 speedup（性能）．
    - 実行時間 (msec)．
  - プロセッサの動作周波数 (MHz)．
  - 設計したコンピュータのアーキテクチャ（ブロック図など）
  - 工夫した点．
- 性能の高いプロセッサを設計・実装した者を「優勝」とする．
- スライド（PowerPointファイル）とソースコードの提出方法
  - Slackで担当TAにダイレクトメッセージで，作成した ContestSlide_XX.pptx（XXは名前）と Verilog HDL のコード一式（zipファイル）を添付する．
  - 2024年11月24日（日）23:59 までに提出すること．
- Computer Design Contest
  - 2024年11月26日（火）13:30 - 17:05