2023年度（令和5年）版

Ver. 2023-11-14a

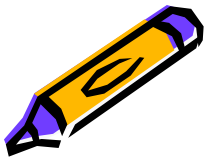
Course number: CSC.T363

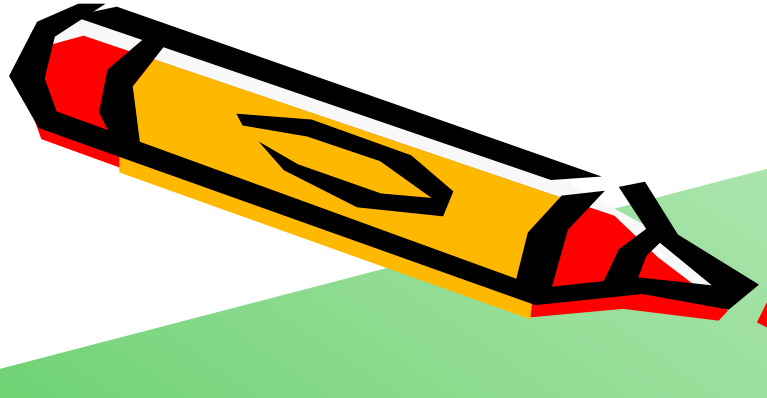
コンピュータアーキテクチャ
Computer Design Contest

情報工学系　吉瀬謙二, Berjab Nesrine

Kenji KISE, Department of Computer Science

kise_at_c.titech.ac.jp





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
$ cp -r /home/u_nesrine/ca/2023/src/contest ./
```

- プロジェクトディレクトリの構成:
  - constrs/
    - 制約ファイルが入っている.
  - prog/
    - contest_program.{dump,elf,bin,hex} ファイルが入っている.
  - src/
    - プロセッサ proc8s と DRAM コントローラのソースコード.
  - vivado/
    - Vivado の Open Project で dram_proc.xpr ファイルを開くと Vivado のプロジェクトが開ける.
  - Makefile
    - make と打つと prog/program0.hex ファイルを読み込んでシミュレーションが実行される.

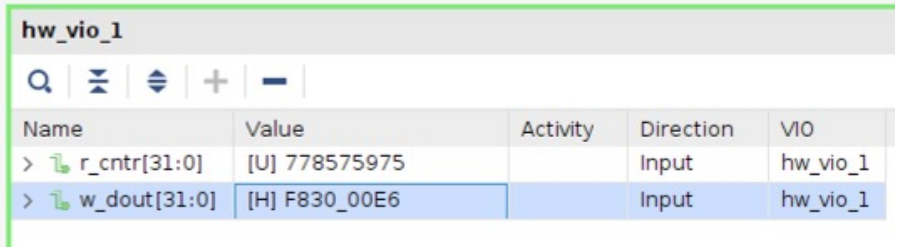
プロジェクトのコピー，論理合成，コンフィギュレーション

```
$ cd ~/ca
$ cp -r /home/u_nesrine/ca/2023/src/contest ./
```

- 上のコマンドで，ファイルをホームディレクトリの下の ca というディレクトリにコピーする．
- Vivado で，~/ca/contest/vivado/dram_proc.xpr を開く．
- 論理合成，配置・配線，ビットストリームファイルを作成する．
- 生成したビットストリームファイルでコンフィギュレーションする．
  - ~/ca/contest/vivado/dram_proc.runs/impl_1/m_main.bit

RISC-V プログラムをシリアル通信で送信して実行

- gtkterm を起動して 1Mbaud に設定する.
  - cat sample_program.bin > /dev/ttyUSB1



sample_program.s

```
#================================================================================
#  Version 2023-11-5                                    Shimooka Noriaki, TOKYO TECH
#--------------------------------------------------------------------------------
    .section .text
    .globl _main
_main:
    addi s0, x0, 0x400  #     max = 0x400
    add  s0, s0, s0     #     max = 0x800
    add  s0, s0, s0     #     max = 0x1000
    add  s0, s0, s0     #     max = 0x2000
    add  s0, s0, s0     #     max = 0x4000
    add  s0, s0, s0     #     max = 0x8000
    add  s0, s0, s0     #     max = 0x10000
    add  s0, s0, s0     #     max = 0x20000
    add  s0, s0, s0     #     max = 0x40000
    add  s0, s0, s0     #     max = 0x80000
    add  s0, s0, s0     #     max = 0x100000
    add  s0, s0, s0     #     max = 0x200000
    addi s1, x0, 0      #     i   = 0
    addi a0, x0, 0      #     sum = 0
```

```
1:  addi s1, s1, 1      # 1:  i   = i + 1
    add  s2, s1, s1     #     adr = i + i
    add  s2, s2, s2     #     adr = adr + adr
    sw   s1, 0(s2)      #     mem[adr] = i
    bne  s1, s0, 1b     #     if (i!=max) goto 1b

    addi s1, x0, 0      #     i   = 0
1:  addi s1, s1, 1      # 1:  i   = i + 1
    add  s2, s1, s1     #     adr = i + i
    add  s2, s2, s2     #     adr = adr + adr
    lw   s3, 0(s2)      #     d   = mem[adr]
    add  a0, a0, s3     #     sum = sum + d
    bne  s1, s0, 1b     #     if (i!=max) goto 1b

    addi s0, s0, -10    #     max = max - 10
    addi s1, x0, 0      #     i   = 0
1:  addi s1, s1, 1      # 1:  i   = i + 1
    addi s4, x0, 0      #     t   = 0
    add  s2, s1, s1     #     adr = i + i
    add  s2, s2, s2     #     adr = adr + adr
    lw   s5, 0(s2)      #     d1  = mem[adr]
    lw   s6, 4(s2)      #     d2  = mem[adr+4]
    lw   s7, 8(s2)      #     d3  = mem[adr+8]
    lw   s8, 12(s2)     #     d4  = mem[adr+12]
    add  s4, s4, s5     #     t   = t + d1
    add  s4, s4, s6     #     t   = t + d2
    add  s4, s4, s7     #     t   = t + d3
    add  s4, s4, s8     #     t   = t + d4
    addi s4, s4, 1      #     t   = t + 1
    add  s9, s5, s6     #     t1  = d1 + d2
    add  s10, s7, s8    #     t2  = d3 + d4
    add  s11, s9, s10   #     t3  = t1 + t2
    add  s4, s4, s11    #     t   = t  + t3
    sw   s4, 0(s2)      #     m[adr] = t
    bne  s1, s0, 1b     #     if (i!=max) goto 1b

    addi s1, x0, 0      #     i   = 0
1:  addi s1, s1, 1      # 1:  i   = i + 1
    add  s2, s1, s1     #     adr = i + i
    add  s2, s2, s2     #     adr = adr + adr
    lw   s3, 0(s2)      #     d   = mem[adr]
    add  a0, a0, s3     #     sum = sum + d
    bne  s1, s0, 1b     #     if (i!=max) goto 1b

_exit:
    add  x30, a0, 0     #     print(sum), halt
    addi t0, x0, 1
1:  bne  t0, x0, 1b
```

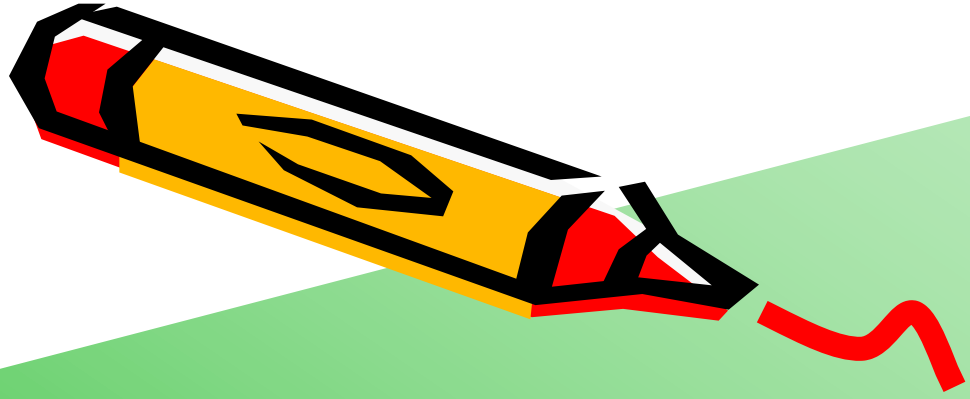
Ref: https://github.com/riscv-non-isa/riscv-asm-manual/blob/master/riscv-asm.md

コンピュータ設計コンテストのルール

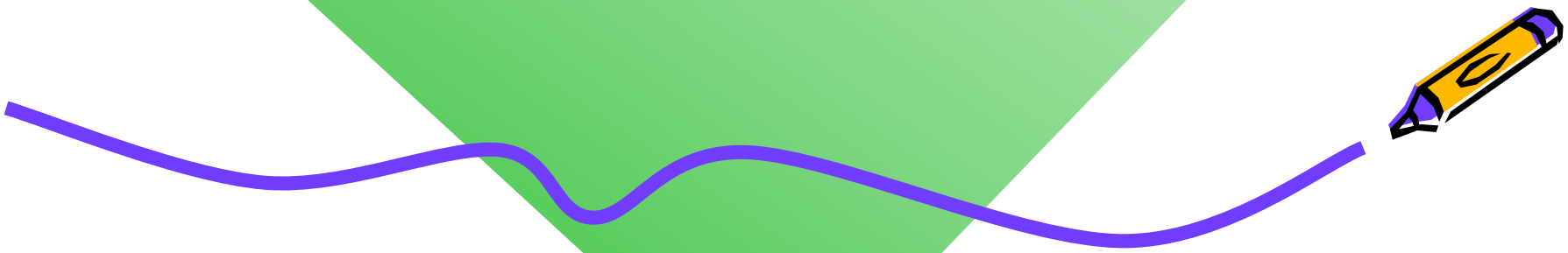
- コンピュータの高速化に取り組み，コンテスト形式で成果を競う．
- RISC-V（のいくつかの命令を処理できる）プロセッサを設計する．
  - サポートする命令: add, addi, lw, sw, bne
  - 与えられたプログラムの実行時間が短いコンピュータを設計すること．
  - Vivadoのタイミング制約を満たす設計とすること．
  - レジスタファイルに書き込まれる値が，正しいことをシミュレーションで確認すること．
  - VIO, Clocking Wizard, MIG により生成される IP を除いて，Verilog HDL で設計すること．
- 利用するFPGAボードは Digilent Arty A7-35T とする．
- 命令メモリのサイズは 2KiB とする．
- 1Mbaudのシリアル通信で，2KiBの与えられたプログラムを送信して処理を開始する．

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.
- 当日はスライドを用いて発表する. 発表時間は1人5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 (160 MHz) 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は氏名）とVerilog HDLのコードを添付する.
  - 2023年11月26日（日） 23:59 までに提出すること.
- Computer Design Contest
  - 2023年11月28日(火) 13:30 – 17:05



References



References (1/2)

- Computer Architecture support page
  - http://www.arch.cs.titech.ac.jp/lecture/CA/
- Computer Logic Design support page
  - http://www.arch.cs.titech.ac.jp/lecture/CLD/
- ACRi Room
  - https://gw.acri.c.titech.ac.jp
- ACRi Blog
  - https://www.acri.c.titech.ac.jp/wordpress/
- 情報工学系計算機室
  - http://www.csc.titech.ac.jp/

References (2/2)

- Xilinx Vivado Design Suite
  - https://japan.xilinx.com/products/design-tools/vivado.html
- Digilent Arty A7-35 A7: FPGA Trainer Board
  - https://reference.digilentinc.com/reference/programmable-logic/arty-a7/start
- Digilent Nexys 4 DDR Atrix-7 FPGA
  - https://store.digilentinc.com/nexys-4-ddr-artix-7-fpga-trainer-board-recommended-for-ece-curriculum/
- Verilog HDL
  - https://ja.wikipedia.org/wiki/Verilog
- Assembly Programmers' Manual
  - https://github.com/riscv-non-isa/riscv-asm-manual/blob/master/riscv-asm.md