池田・飯塚グループトレーニング
～ディジタル編～

池田　誠

¥¥192.168.11.222¥materials¥DigitalTraining2023¥
（もしくは /mnt/kona/materials/DigitalTraining2023/）

内容

① ASIC向け設計：多分数日でできるはず
- 論理合成
- 配置配線
- （設計検証）
- VerilogHDLの記述

② FPGA設計：適宜やってください：２週間程度
- アルゴリズムの選定
- アルゴリズムのVerilogHDLでの記述
- FPGAでの実装
- 最後に実装結果のデモ

③ チップとしての実装：②が完成していれば2週間かからない？
- FPGAで実装したものをチップとして実装
- テープアウトは６月末
- チップが１０月頭に完成：ロジックテスターで測定、（ボードを作ってデモ）

日程：マイルストーン

- X月X日：午後X時ー　論理合成、配置配線の発表

ASIC向け設計

- 論理合成：RTL(VerilogHDL)からネットリストを生成する
  - Design Compiler (Synopsys社）を使用
  - 主に　SYN　のディレクトリで作業
- 配置配線：ネットリストからレイアウトを生成する
  - IC Compiler (Synopsys社）を使用
  - 主に　PR　のディレクトリで作業

ライブラリ・環境の設定

- LabTraining2023.tar.bz2をダウンロードして展開
  - % cd ~/
  - % tar zxf LabTraining2023.tar.bz2
  - % cd LabTraining2023/

以下にパスをとおして実行する

U-2022.12
もしくはV-2023.12
を試してみてください

/mnt/rose/cad/linux/synopsys/syn/Q-2019.12-SP1/linux64/syn/bin
/mnt/rose/cad/linux/synopsys/icc/Q-2019.12-SP1 /linux64/syn/bin
/mnt/rose/cad/linux/cadence/XCELIUMMAIN20.09.011/tools/bin

24.03

HDLシミュレーション

xrun -v ¥
/mnt/rose/cad/linux/synopsys/syn/Q-2019.12-SP1/dw/sim_ver/DW02_mult.v ¥
simmul16.v mul16.v

各自の環境に合わせる

```
Compiling source file "simmul16.v"
Compiling source file "mul16.v"
Scanning library file "******************************/dw/sim_ver/DW02_mult.v"
Highest level modules:
simmulti

xxxx x xxxx (x)= xxxxxxxx(xxxxxxxx)
3524 x 5e81 (0)= xxxxxxxx(xxxxxxxx)
d609 x 5663 (0)= 00000000(xxxxxxxx)
7b0d x 998d (0)= 00000000(139dff24)
8465 x 5212 (0)= 00000000(4839cb7b)
e301 x cd0d (0)= 00000000(49ce8b29)
f176 x cd3d (0)= 00000000(2a71a91a)
57ed x f78c (0)= 00000000(b5d3540d)
e9f9 x 24c6 (0)= 00000000(c195071e)
84c5 x d2aa (0)= 5505c09c(5505c09c)
f7e5 x 7277 (0)= 219bfa96(219bfa96)
d612 x db8f (0)= 6d41c4d2(6d41c4d2)
69f2 x 96ce (0)= 6ed73573(6ed73573)
```

論理合成

dc_shell < syn.tcl > syn.log

syn.tcl

```
set TOP "multi"
set CLK "CLK"
set library_path "../LIB";
lappend search_path $library_path
lappend link_path $library_path
lappend search_path "."
lappend link_path "."
set target_library ri18it182_typ.db
set link_library ri18it182_typ.db
set symbol_library [list "EXD.sdb"]

set designer {Designers Name}
set company {TEST}
set verilogout_no_tri "true"
set verilogout_single_bit "false"
```

合成対象モジュールの指定

クロックネットの指定

RTLファイルの指定

クロック周期の制約条件の指定

```
read_file -rtl -format verilog ./mul16.v
current_design ${TOP}
create_clock -period 20 ${CLK}
set_max_area 0.0
set_dont_touch_network ${CLK}
set_driving_cell  -lib_cell  it18_nand2_0  -pin ZN  [all_inputs]
set_load 0.2 [all_outputs]
set_input_delay -max 2 -clock ${CLK} [all_inputs]
set_output_delay -max 2 -clock ${CLK} [all_outputs]
compile_ultra
define_name_rules verilog -case_insensitive -type net
change_names -rules verilog -hierarchy
current_design ${TOP}
redirect port.txt { report_port }
redirect timing.txt {report_timing -path full_clock -nworst 2 -nets -
transition_time -capacitance -attributes -sort_by slack}
redirect area.txt {report_area}
redirect power.txt {report_power}
current_design ${TOP}
write -format verilog -hierarchy -output "${TOP}out.v" [list ${TOP}]
quit
```

スクリプトを修正

syn.logも参照

timing.txt

合成結果：速度

```
****************************************
Report : timing
Design : mul
Version: Q-2019.12-SP2
Date   : Wed Apr 15 02:32:18 2020
****************************************

Operating Conditions: NCCOM   Library: ri18it182
Wire Load Model Mode: segmented

Path Group: CLK
  Path Type: max

  Point                                    Fanout     Cap     Trans      Incr       Path      Attributes
  ----------------------------------------------------------------------------------------------------------
  clock CLK (rise edge)                                                  0.00       0.00
  clock network delay (ideal)                                            0.00       0.00
  inst_A_FF_reg_2_/CP (it18_dffir_0)                                     0.00       0.00       0.00 r
  ..................
  PRODUCT_inst_reg_31_/D (it18_dffir_0)                                  0.05       0.00      10.79 f
  data arrival time                                                                           10.79

  clock CLK (rise edge)                                                 20.00      20.00
  clock network delay (ideal)                                            0.00      20.00
  PRODUCT_inst_reg_31_/CP (it18_dffir_0)                                 0.00      20.00 r
  library setup time                                                    -0.23      19.77
  data required time                                                               19.77
  ----------------------------------------------------------------------------------------------------------
  data required time                                                               19.77
  data arrival time                                                               -10.79
  ----------------------------------------------------------------------------------------------------------
  slack (MET)                                                                      8.99
```

合成結果に関する考察：
FO4-2NAND換算遅延
このライブラリの場合FO4-2NANDの
遅延はライブラリスペックによると96ps
→ 11.01nsは2NAND 114.7段分

0ns設定に対しての余裕度
20 – 8.99 = 11.01ns
これが最大遅延時間
（動作周期はこの逆数）

area.txt

合成結果：面積

```
****************************************
Report : area
Design : mul0016
Version: Q-2019.12-SP2
Date   : Wed Apr 15 02:32:18 2020
****************************************

Library(s) Used:

    ri18it182 (File:
/home/mikeda/usr2/Work/Rohm018Flow/LIB/ri18it182_typ.db)

Number of ports:                          67
Number of nets:                          691
Number of cells:                         556
Number of combinational cells:           491
Number of sequential cells:               65
Number of macros/black boxes:              0
Number of buf/inv:                        85
Number of references:                     26

Combinational area:              18498.815862
Buf/Inv area:                      906.393587
Noncombinational area:            3983.615990
Macro/Black Box area:                0.000000
Net Interconnect area:              16.677702

Total cell area:                 22482.431851
Total area:                      22499.109553
```

このライブラリの場合には
um^{2}単位の面積を表す

組み合わせ回路部の面積

バッファ・インバータ面積

メモリ（FF、ラッチ）の面積

仮想配線面積

全セル領域の面積

この値から等価ゲート数を求めることができる：最小2入力NAND換算(12.90um^2)

1,742ゲート相当

power.txt

合成結果：電力

ただし、この電力は制約条件で設定したクロック速度で動作した時の電力であるため、この場合は20nsで動作させたときの電力となる。実際には、この合成の場合の最高動作周波数は11.01nsであるため、扱いに注意

1）11.01ns動作に換算

Dynamic Power → 20/11.01倍

Static Power → そのまま

```
Global Operating Voltage = 1.8
Power-specific unit information :
    Voltage Units = 1V
    Capacitance Units = 1.000000pf
    Time Units = 1ns
    Dynamic Power Units = 1mW    (derived from V,C,T units)
    Leakage Power Units = 1nW


  Cell Internal Power  = 655.6787 uW   (61%)
  Net Switching Power  = 412.9781 uW   (39%)
                         ---------
Total Dynamic Power    =   1.0687 mW  (100%)

Cell Leakage Power     =  24.2060 nW
```

11.01nsでは1.94 mW

11.01nsで動作させたときの1周期あたりの消費エネルギー: (11.94mW + 24nW) * 11.01ns

Power Group	Internal Power	Switching Power	Leakage Power	Total Power	(%)	Attrs
io_pad	0.0000	0.0000	0.0000	0.0000	(0.00%)	
memory	0.0000	0.0000	0.0000	0.0000	(0.00%)	
black_box	0.0000	0.0000	0.0000	0.0000	(0.00%)	
clock_network	0.0000	0.0000	0.0000	0.0000	(0.00%)	
register	0.3402	0.1583	4.9400	0.4986	(46.65%)	
sequential	0.0000	0.0000	0.0000	0.0000	(0.00%)	
combinational	0.3154	0.2546	19.2660	0.5701	(53.35%)	
Total	0.6557 mW	0.4130 mW	24.2060 nW	1.0687 mW		

ネットリストのシミュレーション

xrun -v /******/SIM/ri18it182.v ¥**
-v /******/SIM/udp.v ¥**
simmul16.v multiout.v

ompiling source file "simmul16.v"
Compiling source file "multiout.v"
Scanning library file "../SIM/ri18it182.v"
Scanning library file "../SIM/ri18it182.v"
Scanning library file "../SIM/udp.v"

Warning! Too few module port connections [Verilog-TFNPC]
"multiout.v", 250: FS_1(.A({A1_29_, A1_28_,
A1_27_, A1_26_, A1_25_, A1_24_, A1_23_, A1_22_,
A1_21_, A1_20_, A1_19_, A1_18_, A1_17_, A1_16_
, A1_15_, A1_14_, A1_13_, A1_12_, A1_11_,
A1_10_, A1_9_, A1_8_, A1_7_, A1_6_, A1_5_, A1_4_
, A1_3_, A1_2_, A...
Highest level modules:
simmulti

xxxx x xxxx (x)= xxxxxxxx(xxxxxxxx)
3524 x 5e81 (0)= xxxxxxxx(xxxxxxxx)
d609 x 5663 (0)= 00000000(xxxxxxxx)
7b0d x 998d (0)= 00000000(139dff24)
8465 x 5212 (0)= 00000000(4839cb7b)
e301 x cd0d (0)= 00000000(49ce8b29)
f176 x cd3d (0)= 00000000(2a71a91a)
57ed x f78c (0)= 00000000(b5d3540d)
e9f9 x 24c6 (0)= 00000000(c195071e)
84c5 x d2aa (0)= 5505c09c(5505c09c)
f7e5 x 7277 (0)= 219bfa96(219bfa96)
d612 x db8f (0)= 6d41c4d2(6d41c4d2)
69f2 x 96ce (0)= 6ed73573(6ed73573)

論理合成の最適化

- create_clock -period **20** $CLK
- の周期 “20”を変えて合成を行い合成結果（面積）、遅延時間（動作周波数）の変化を調べる
- 動作速度が最も速くなるものを配置配線に採用する

合成の最適化の一例

制約条件	最大遅延[ns]	面積[um2](kG)	消費電力[mW]	消費エネルギー[pJ]	合成時間(user time) [s]
20.0	10.54	29,062 (2.25)	2.20	23.2	3.32
10.0	9.8	29,169 (2.26)	2.35	23.0	3.44
5.5	5.5	33,211 (2.57)	4.66	25.6	7.16
5.0	5.1	37,456 (2.90)	5.48	28.0	36.69
4.5	5.2	37,420 (2.90)	6.04	31.4	31.91
4.0	5.2	37,278 (2.89)	6.75	35.1	27.11

最速

横軸　ゲート規模、縦軸1最大遅延（クロック速度）、縦軸2消費エネルギーでグラフを書くこと

課題

- 課題１：16bit乗算器に対して、前頁のような最適化を行い表にまとめよ。その際、それぞれの合成に要したCPU時間もまとめておくこと
- 課題２：8bit、32bit、64bit, 128bitの乗算器に対して課題１同様の最適化を行い、ビット幅ー動作周波数、ビット幅ーゲート規模、ビット幅ー消費電力を表もしくはグラフにまとめよ
- 課題３：64bit乗算器に関してmul16_stg2.vを参考にして、2段～6段パイプラインに関して合成結果を求め、動作周波数、演算遅延時間、面積、消費電力をまとめよ

まずは配置配線

icc_shell -shared_license -f PR0.tcl

PR0.tcl

```
start_gui
set top "multi"
set sh_output_log_file "${top}.log"
set sh_command_log_file "${top}.cmd"
set_tlu_plus_files ¥
        -max_tluplus "/********************/LIB/rohm018.tluplus ¥
        -min_tluplus "/********************/LIB/rohm018.tluplus ¥
        -tech2itf_map "/home/ikeda/LabTraining/LIB/starrc_layermap
lappend search_path "/home/ikeda/LabTraining/LIB/"
lappend search_path "./"
set target_library {ri18it182_typ.db}
set link_library "ri18it182_typ.db"
create_mw_lib ¥
        -technology "/********************/LIB/rohm018_astro.tf" ¥
        -mw_reference_library {"/********************/LIB/RI18IT" } ¥
        -hier_separator {/} -bus_naming_style {[%d]} ${top}
open_mw_lib ${top}
set mw_logic0_net VSS
set mw_logic1_net VDD
import_designs -format verilog "../SYN/${top}out.v"
create_clock -period 5 -waveform {0 2.5} -name CLK
create_floorplan -core_utilization 0.8 ¥
    -left_io2core 100 -bottom_io2core 100 -right_io2core 100 -top_io2core 100
derive_pg_connection -power_net VDD -power_pin VDD:P -ground_net VSS -ground_pin VSS:G -create_port all
derive_pg_connection -tie -power_net VDD -ground_net VSS
```

GUIの起動

配置配線用ライブラリの作製

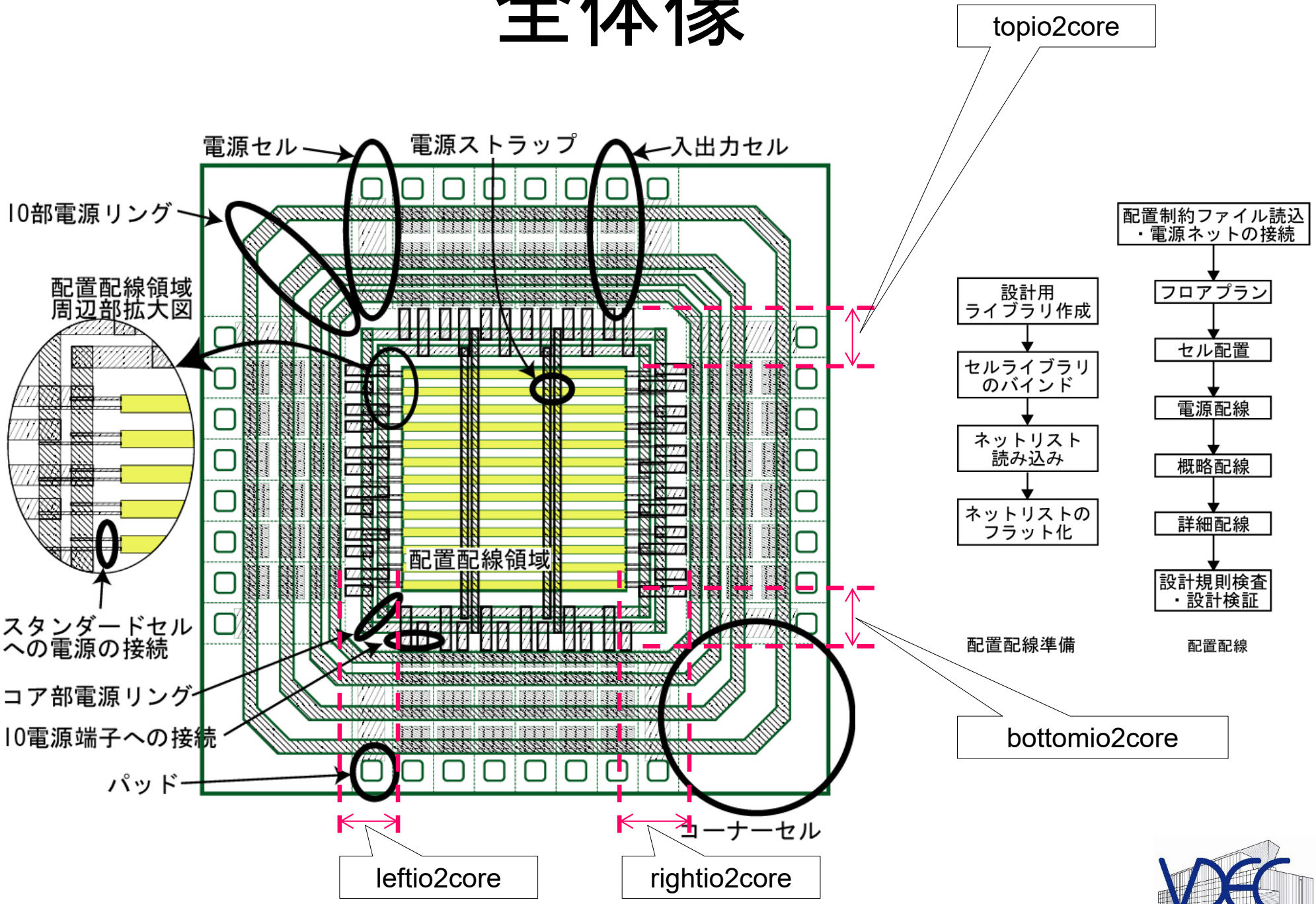
ネットリストの読み込み。課題1もしくは2で論理合成したいずれかのネットリストを使用（32ビットを使用するのを標準とする）

クロック速度は合成結果に合わせる（あまりきつい制約にすると配置配線の際にバッファが大量に挿入されてしまい、制約がゆるいと性能劣化する

フロアプランの作製

電源グランドの定義・論理的な接続

全体像


topio2core
電源セル
電源ストラップ
入出力セル
IO部電源リング
配置配線領域
周辺部拡大図
配置配線領域
スタンダードセル
への電源の接続
コア部電源リング
IO電源端子への接続
パッド
コーナーセル
leftio2core
rightio2core
配置制約ファイル読込
・電源ネットの接続
設計用
ライブラリ作成
セルライブラリ
のバインド
ネットリスト
読み込み
ネットリストの
フラット化
フロアプラン
セル配置
電源配線
概略配線
詳細配線
設計規則検査
・設計検証
配置配線準備
配置配線
bottomio2core

PR0.tcl 続き

```
create_rectangular_rings  -nets  {VDD} ¥
                -left_offset   5 -left_segment_width   20 -left_segment_layer M2 ¥
                -right_offset 30 -right_segment_width  20 -right_segment_layer M2 ¥
                -bottom_offset 5 -bottom_segment_width 20 -bottom_segment_layer M3 ¥
                -top_offset   30 -top_segment_width    20 -top_segment_layer M3 ¥
                -offsets absolute
create_rectangular_rings  -nets  {VSS} ¥
                -left_offset  30 -left_segment_width   20 -left_segment_layer M2 ¥
                -right_offset  5 -right_segment_width  20 -right_segment_layer M2 ¥
                -bottom_offset 30 -bottom_segment_width 20 -bottom_segment_layer M3 ¥
                -top_offset    5 -top_segment_width    20 -top_segment_layer M3 ¥
                -offsets absolute
place_opt
clock_opt -only_hold_time -fix_hold_all_clocks
set_route_options -same_net_notch check_and_fix
route_opt
optimize_zrt_wire_via -max_detail_route_iterations 20
```

電源リングの作成

配置

クロックツリー合成

配線

配線の最適化

PR0.tcl 続き

```
remove_antenna_rules $top
set_parameter -module droute -name dontMergeGateForAntenna -value 0
set_parameter -module droute -name lowerLayerAntennaAccumMode -value 1
set_parameter -module droute -name useSideWallForAntenna -value 0

define_antenna_accumulation_mode -cut_to_metal -metal_to_cut
define_antenna_rule $top -mode 4 -diode_mode 4 -metal_ratio 380 -cut_ratio 18
define_antenna_layer_rule $top -mode 4 -layer "M1"   -ratio 380 -diode_ratio {0.203 0 400 2200}
define_antenna_layer_rule $top -mode 4 -layer "M2"   -ratio 380 -diode_ratio {0.203 0 400 2200}
define_antenna_layer_rule $top -mode 4 -layer "M3"   -ratio 380 -diode_ratio {0.203 0 400 2200}
define_antenna_layer_rule $top -mode 4 -layer "M4"   -ratio 380 -diode_ratio {0.203 0 400 2200}
define_antenna_layer_rule $top -mode 4 -layer "M5"   -ratio 380 -diode_ratio {0.203 0 8000 30000}
define_antenna_rule $top -mode 1 -diode_mode 4 -metal_ratio 380 -cut_ratio 18
define_antenna_layer_rule $top -mode 1 -layer "VIA1" -ratio 18 -diode_ratio {0.203 0 83.33 75}
define_antenna_layer_rule $top -mode 1 -layer "VIA2" -ratio 18 -diode_ratio {0.203 0 83.33 75}
define_antenna_layer_rule $top -mode 1 -layer "VIA3" -ratio 18 -diode_ratio {0.203 0 83.33 75}
define_antenna_layer_rule $top -mode 1 -layer "VIA4" -ratio 18 -diode_ratio {0.203 0 83.33 75}

set_parameter -name doAntennaConx -value 4
optimize_zrt_wire_via -max_detail_route_iterations 20
verify_zrt_route -antenna true
```

アンテナルールの考慮

PR0.tcl 続き

```
insert_stdcell_filler ¥
    -cell_with_metal "it18_fillcap_8 it18_fillcap_4 it18_fillcap_2 it18_fillcap_1" ¥
    -cell_without_metal "it18_fillcap_8 it18_fillcap_4 it18_fillcap_2 it18_fillcap_1" ¥
    -ignore_hard_placement_blockage -ignore_soft_placement_blockage ¥
    -connect_to_power {VDD}  -connect_to_ground {VSS}

derive_pg_connection -power_net VDD -ground_net VSS  -create_port all
derive_pg_connection -tie -power_net VDD -ground_net VSS

preroute_standard_cells -connect horizontal  -fill_empty_rows  -fill_empty_sites ¥
    -port_filter_mode off -cell_master_filter_mode off -cell_instance_filter_mode off ¥
    -voltage_area_filter_mode off -route_type {P/G Std. Cell Pin Conn}

optimize_zrt_wire_via -max_detail_route_iterations 20
verify_zrt_route -antenna true

insert_stdcell_filler ¥
    -cell_without_metal "it18_fillcap_8 it18_fillcap_4 it18_fillcap_2 it18_fillcap_1" ¥
    -cell_with_metal "it18_fillcap_8 it18_fillcap_4 it18_fillcap_2 it18_fillcap_1" ¥
    -ignore_hard_placement_blockage -ignore_soft_placement_blockage ¥
    -connect_to_power {VDD}  -connect_to_ground {VSS}

preroute_standard_cells -connect horizontal  -fill_empty_rows  -fill_empty_sites ¥
    -port_filter_mode off -cell_master_filter_mode off -cell_instance_filter_mode off ¥
    -voltage_area_filter_mode off -route_type {P/G Std. Cell Pin Conn}

optimize_zrt_wire_via -max_detail_route_iterations 20
verify_zrt_route -antenna true
```

フィラーセル挿入

セル領域の電源接続

信号配線の最適化

フィラーセル再挿入

信号配線の再最適化

PR0.tcl 続き

```
save_mw_cel  -design  multi
verify_lvs
verify_drc

insert_metal_filler  -purge -out self -from_metal 1 -to_metal 5

insert_metal_filler -out roulette_top.FILL ¥
        -width {M1 0.22 M2 0.28 M3 0.28 M4 0.28 M5 0.44}  ¥
        -space {M1 0.24 M2 0.28 M3 0.28 M4 0.28 M5 0.46}  ¥
        -min_length {M1 2 M2 2 M3 2 M4 2 M5 2}  ¥
        -max_length {M1 10 M2 10 M3 10 M4 10 M5 10}  ¥
        -space_to_route {M1 0.36 M2 0.42 M3 0.42 M4 0.42 M5 0.69}  ¥
        -space_to_pg {M1 0.36 M2 0.42 M3 0.42 M4 0.42 M5 0.69}  ¥
        -bounding_box { {$FilAreaX0 $FilAreaY0} {$FilAreaX1 $FilAreaY1} }  ¥
        -from_metal 1 -to_metal 5  -dont_snap_fill_to_track

define_name_rules verilog -allowed "A-Z0-9_" -type net
change_names -rules verilog -hierarchy
write  -format ddc  -hierarchy   -output ${top}.ddc
write_verilog -no_unconnected_cells -no_physical_only_cells -pg ${top}_pg_out.v
write_verilog -no_unconnected_cells -no_physical_only_cells ${top}_out.v
set_write_stream_options  -child_depth 20 ¥
        -contact_prefix GENVIA -output_filling {fill} ¥
        -output_outdated_fill ¥
        -output_pin {text geometry} -output_design_intent
write_stream -format gds -lib_name ${top} -cells ${top} ${top}.str
write_sdf -version 2.1 ${top}.sdf
redirect ${top}_timing.log { report_timing -max_paths 20 }
save_mw_cel  -design ${top}
quit
```

配置配線ツール内の
LVS／DRCの実行

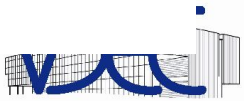
密度ルール向け配線
フィラーの挿入

レイアウト（GDS）の出力

途中で必ずログを確認しながら

```
****************************************
  Report : Chip Summary
  Design : multi
  Version: G-2012.06-ICC-SP5
  Date   : Thu Apr 24 11:56:51 2014
****************************************
Std cell utilization: 70.10%  (9952/(14196-0))
(Non-fixed + Fixed)
Std cell utilization: 70.10%  (9952/(14196-0))
(Non-fixed only)
Chip area:            14196    sites, bbox (100.00 100.00 316.32 311.68) um
Std cell area:        9952     sites, (non-fixed:9952   fixed:0)
                      833      cells, (non-fixed:833    fixed:0)
Macro cell area:      0        sites
                      0        cells
Placement blockages:  0        sites, (excluding fixed std cel
                      0        sites, (include fixed std cells
                      0        sites, (complete p/g net blocka
Routing blockages:    0        sites, (partial p/g net blockag
                      0        sites, (routing blockages and s
Lib cell count:       34
Avg. std cell width:  6.81 um
Site array:           unit     (width: 0.64 um, height: 5.04 u
Physical DB scale:    1000 db_unit = 1 um
```

```
DR finished with 0 open nets, of which 0 are frozen

DR finished with 0 violations

DRC-SUMMARY:
        @@@@@@@ TOTAL VIOLATIONS =      0



Total Wire Length =                    43266 micron
Total Number of Contacts =             6110
Total Number of Wires =                5292
Total Number of PtConns =              756
Total Number of Routable Wires =       5292
Total Routable Wire Length =           43083 micron
        Layer           M1 :       1015 micron
        Layer           M2 :      21589 micron
        Layer           M3 :      17914 micron
        Layer           M4 :       2748 micron
        Layer           M5 :          0 micron
        Via        VIA34NN :        121
        Via        VIA23NN :       2955
        Via        VIA12NN :       3024
        Via   VIA12NN(rot) :         10
```



配置配線の最適化

- create_floorplan -core_utilization **0.8** ￥
  -left_io2core 100 -bottom_io2core 100 ￥
  -right_io2core 100 -top_io2core 100
  各パラメータ変えて配置配線を行い、パラメータの意味を理解する
- core_utilizationがどこまで配置配線可能か確認する

なお、配置配線を実行する際には、毎回 multiというディレクトリ（配置配線のライブラリ）を削除してから開始すると問題がなくてよい

課題

- 課題４：前頁のCore Utilizationを0.5 ~ 1.0まで幾つか振って、その結果（配置配線が完成したorViolationが残った、合成結果のタイミングレポート値がどうなったか）をまとめる

- 課題５：課題４のタイミングレポートが最も早くなったものCore Utilizationを使用し、スライド２０のClock制約を変化させたときのタイミングレポートを表にまとめる

レイアウトの完成・・IOセル

~/SYN において、ポート情報の取得とIOセルの挿入

./Ro18GetPinPos.pl I port.txt PinPos.dat 18 QFP160

論理合成で出力される
（ポートの名前は短く）

仮の信号配置

IO信号数の関係で
QFP160が必要

18: 2.5mm角チップ

PinPos.dat を編集してピン配置を確定する

32bitの場合には信号数の関係でチップサイズ22、パッケージはQFP208とする

./Ro18GetPinPos.pl O port.txt PinPos.dat 18 QFP160 CLK PR1

クロック信号名

生成される配置配線スクリプトのPrefix

PR1_pr.cmd: 配置配線スクリプトのメイン
PR1_iopos.tcl: IOセル配置（位置決め）
PR1_iodef.tcl: ネットリストに存在しないIOセルの定義
PR1_padpos.tcl
multi_top.v: IOセルが含まれたネットリスト：これを配置配線に使用
→ PR1_pr.cmd を ../PR にコピーする

IOセルが含まれたネットリスト

```
module multi_top (CLK, RST, inA, inB, inTC, outPROD);
input   CLK;
・・・
input   inTC;
output  [31:0]  outPROD;
wire    in_CLK;
wire    in_RST;
・・・・・・
wire    in_inTC;
wire    [31:0]  in_outPROD;

        multi top (   .CLK(in_CLK), .RST(in_RST), .inA(in_inA),
                .inB(in_inB), .inTC(in_inTC), .outPROD(in_outPROD));
        PC30D01 IO_002 (.PAD(CLK), .CIN(in_CLK) );
        PC30D01 IO_003 (.PAD(RST), .CIN(in_RST) );
        PC30D01 IO_004 (.PAD(inA[0]), .CIN(in_inA[0]) );
        PC30D01 IO_005 (.PAD(inA[1]), .CIN(in_inA[1]) );
        PC30D01 IO_006 (.PAD(inA[2]), .CIN(in_inA[2]) );
        PC30D01 IO_007 (.PAD(inA[3]), .CIN(in_inA[3]) );
```

IO込ネットリストのシミュレーション

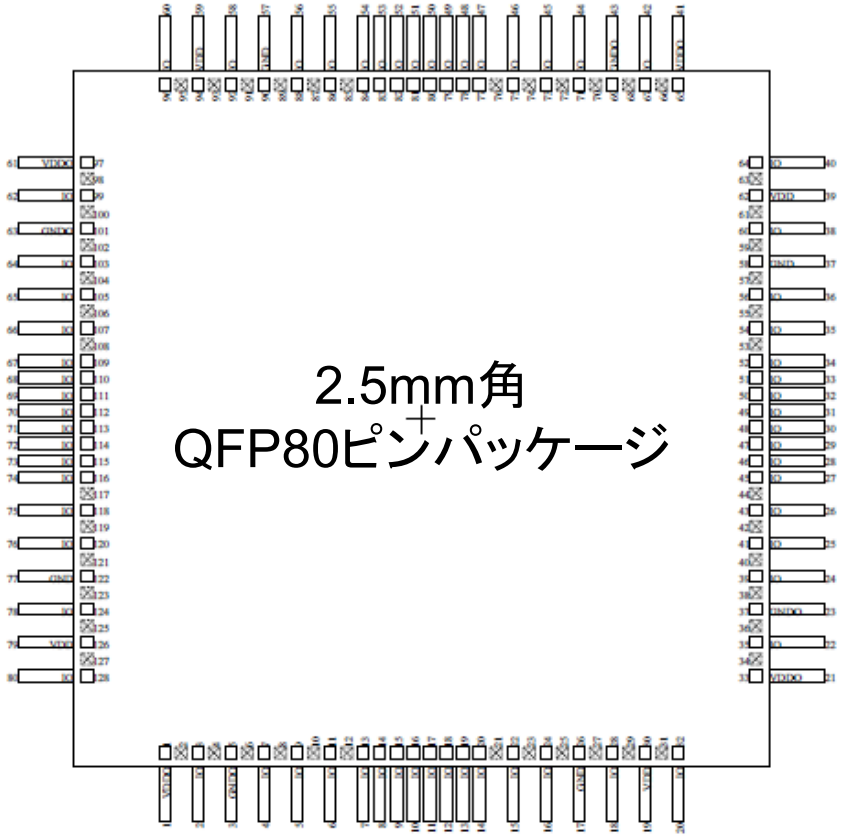
verilog +define+SIMIO -v /******/SIM/ri18it182.v ¥**
-v /*******/SIM/rip18io18300.v -v /********/SIM/udp.v ¥**
simmul16.v multi_top.v multiout.v

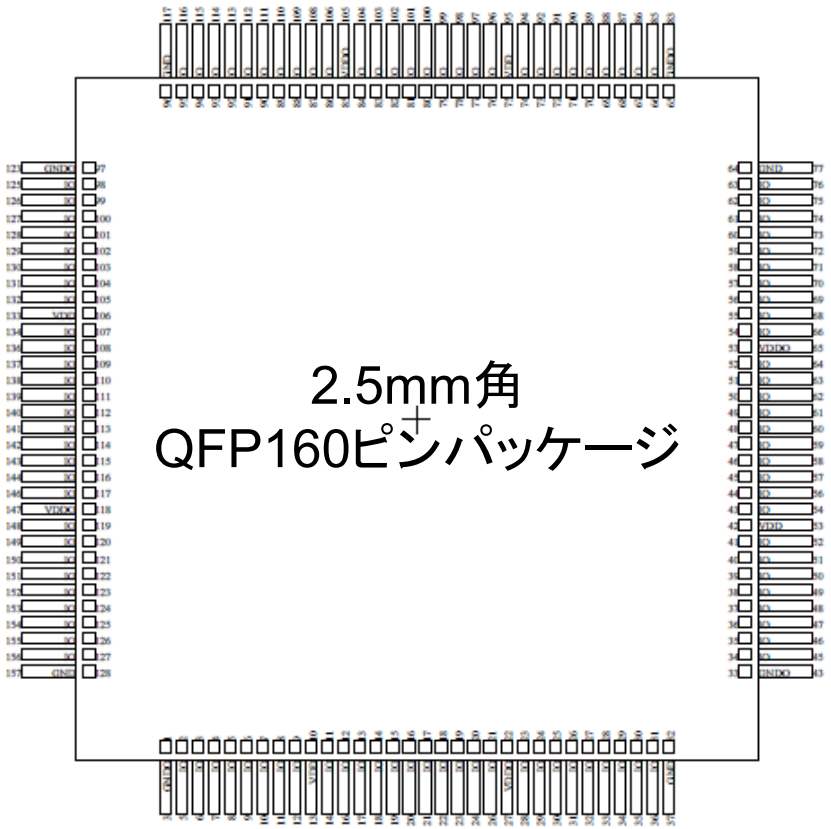
```
ompiling source file "simmul16.v"
Compiling source file "multi_top.v"
Compiling source file "multiout.v"
Scanning library file "../SIM/ri18it182.v"
Scanning library file "../SIM/ri18it182.v"
Scanning library file "../SIM/rip18io18300.v"
Scanning library file "../SIM/udp.v"
Warning!  Too few module port connections                    [Verilog-TFNPC]
        "multiout.v", 654: FS_1(.A({A1_29_, A1_28_,
        A1_27_, A1_26_, A1_25_, A1_24_, A1_23_, A1_22_,
        A1_21_,   A1_20_, A1_19_, A1_18_, A1_17_, A1_16_
        , A1_15_, n70, A1_13_, A1_12_, A1_11_,   A1_10_,
        A1_9_, A1_8_, A1_7_, A1_6_, A1_5_, A1_4_, A1_3_,
        A1_2_, A1_1_,   A...
Highest level modules:
simmulti
```

```
xxxx x xxxx (x)= xxxxxxxx(xxxxxxxx)
3524 x 5e81 (0)= xxxxxxxx(xxxxxxxx)
d609 x 5663 (0)= 00000000(xxxxxxxx)
7b0d x 998d (0)= 00000000(139dff24)
8465 x 5212 (0)= 00000000(4839cb7b)
e301 x cd0d (0)= 00000000(49ce8b29)
f176 x cd3d (0)= 00000000(2a71a91a)
57ed x f78c (0)= 00000000(b5d3540d)
e9f9 x 24c6 (0)= 00000000(c195071e)
84c5 x d2aa (0)= 5505c09c(5505c09c)
f7e5 x 7277 (0)= 219bfa96(219bfa96)
d612 x db8f (0)= 6d41c4d2(6d41c4d2)
69f2 x 96ce (0)= 6ed73573(6ed73573)
```

信号・電源配置

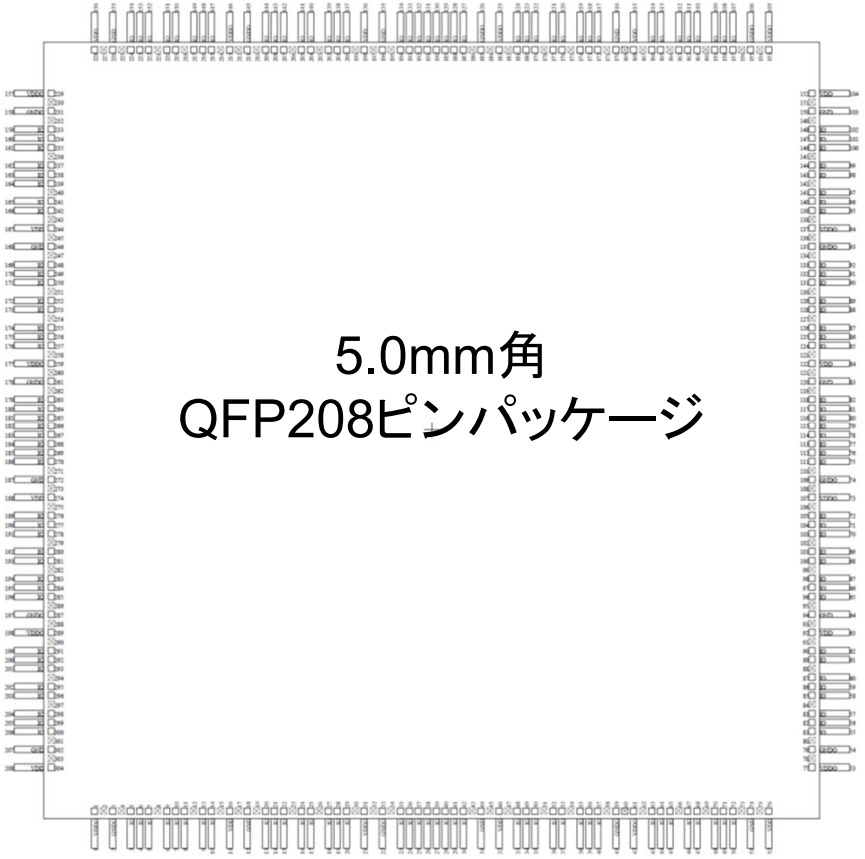
- VDECWEBに掲載の電源グランド配置に従っておくと測定の際に何かと便利





信号・電源配置

- VDECWEBに掲載の電源グランド配置に従っておくと測定の際に何かと便利



IOセルを含めた配置配線

~/PR において、配置配線の実施

icc_shell -f PR1_pr.cmd

```
initialize_floorplan -control_type width_and_height ¥
        -core_width 1822.08 -core_height 1819.44 ¥
        -left_io2core 100 -bottom_io2core 100 ¥
        -right_io2core 100 -top_io2core 102.64 ¥
        -keep_macro_place

preroute_instances  -ignore_macros -ignore_cover_cells ¥
        -select_net_by_type specified -nets  {VSS} ¥
        -route_pins_on_layer M2 ¥
        -primary_routing_layer specified -specified_horizontal_layer M2 ¥
        -route_boundary_coinciding_edges_only
preroute_instances  -ignore_macros -ignore_cover_cells ¥
        -select_net_by_type specified -nets {VDD} ¥
        -route_pins_on_layer M2 ¥
        -primary_routing_layer specified -specified_horizontal_layer M2 ¥
        -customize -routing_width 10 -route_boundary_coinciding_edges_only

create_rectangular_rings  -nets  {VDD} ¥
        -left_offset  85 -left_segment_width   10 -left_segment_layer M4 ¥
        -right_offset 85 -right_segment_width  10 -right_segment_layer M4 ¥
        -bottom_offset 85 -bottom_segment_width 10 -bottom_segment_layer M4 ¥
        -top_offset   87.64 -top_segment_width    10 -top_segment_layer M4 ¥
        -offsets absolute

preroute_instances  -ignore_macros -ignore_cover_cells ¥
        -select_net_by_type specified -nets  {VDD} ¥
        -route_pins_on_layer M2 ¥
        -primary_routing_layer specified -specified_horizontal_layer M2 ¥
        -route_boundary_coinciding_edges_only
```

IOセルの配置により配置配線領域が変わってしまうと、変なところに配線が来てしまって困るので、そうならないように、コア寸法を調整することが不可欠

IOセルからコアへの電源線

IOセルにVDDをコア側から供給する必要があるため、そのための電源リングを貼り、そこに対して電源の接続を行う

```
create_power_straps  -direction vertical  -nets  {VDD}  -layer M2   -width 7 ¥
      -start_at 359.58 -configure step_and_stop  -step 199.68 -stop 2031.82 ¥
      -do_not_route_over_macros
create_power_straps  -direction vertical  -nets  {VSS}  -layer M2   -width 7 ¥
      -start_at 460.06 -configure step_and_stop  -step 199.68 -stop 2031.82 ¥
      -do_not_route_over_macros
create_power_straps  -direction vertical  -nets  {VDD}  -layer M4   -width 7 ¥
      -start_at 460.06 -configure step_and_stop  -step 199.68 -stop 2031.82 ¥
      -do_not_route_over_macros
create_power_straps  -direction vertical  -nets  {VSS}  -layer M4   -width 7 ¥
      -start_at 359.58 -configure step_and_stop  -step 199.68 -stop 2031.82 ¥
      -do_not_route_over_macros
#
create_power_straps  -direction horizontal  -nets  {VDD}  -layer M3   -width 7 ¥
      -start_at 350.7 -configure step_and_stop  -step 191.52 -stop 2029.18 ¥
      -do_not_route_over_macros
create_power_straps  -direction horizontal  -nets  {VSS}  -layer M3   -width 7 ¥
      -start_at 446.46 -configure step_and_stop  -step 191.52 -stop 2029.18 ¥
      -do_not_route_over_macros
```

コア領域の電源ストラップの生成

レイアウトの確認

- 配置配線が終了したら multi.str を ../LAYにコピーして、レイアウトエディタで中身を確認
- このあと、calibreでの DRC, LVSを行う（今回は省略）

DRCの実行

- DRC
  drc.rulの該当箇所を編集して

  % caliber –drc –hier drc.rul > drc.log

drc.sum

LVSの実行

- LVS

  % v2lvs -i -sn -sl -lsr ri18it.spi -s ri18it.spi ¥
  -lsr rip18io18300.spi -s rip18io18300.spi ¥
  -v multi_out.v -o multi_top_out.spi

- cbu018m5n1a_mod.lvsの該当箇所を編集

  LAYOUT PATH “multi.str“
  LAYOUT PRIMARY "multi_top“
  SOURCE PATH "multi_top_out.spi“
  SOURCE PRIMARY "multi_top“
  INCLUDE "lvs_box_add.rul"

  % calibre -lvs -hier -automatch -hcell RI18IT.hcell¥
  -spice multi10top.net cbu018m5n1a_mod.lvs

池田・飯塚グループトレーニング
～ディジタルFPGA編～

池田　誠

VerilogHDL記述課題

- VerilogHDLを記述してFPGAへの実装を習得する
  - 課題：自然対数の低を100桁以上求め、演算終了後1クロックごとに10進数4桁ずつ外部（7-seg LED）に出力する（その際残り2桁でどの桁を表示しているかを示す）→最終的にVGAでディスプレイに表示
  - TOPモジュールは、クロック入力、LED出力（LED表示桁選択入力）、内部演算結果出力（演算結果桁選択入力）で構成し、シミュレーションにおいて計算終了後に演算結果全桁が外部に出力されることを示すこと
  - 以下のアルゴリズムを実装
  - Quartusにより、面積(ALM数：DSP、BRAMはいくら使ってもカウント外)（A）、動作周波数(f[MHz])、消費電力(P[mW])を求める → DesignCompilerでの比較もオプションとして歓迎
  - 評価①：
    - 100桁を求め出力するのに要した時間: $T_{50}=N_{50}$(必要クロックサイクル数)$/f_{100}$
    - ゲートあたりの処理速度：$1/(A_{100}\cdot T_{100})$ [Operations/ALM数]
    - 電力あたりの処理速度： $1/(P_{100}\cdot T_{100})$ [Operations/mW]
  - 評価②：
    - 合成できた最大桁数
    - その際の要した時間: $T_{max}=N_{max}$(必要クロックサイクル数)$/f_{max}$
    - ゲートあたりの処理速度：$1/(A_{max}\cdot T_{max})$ [Operations/ALM数]
    - 電力あたりの処理速度： $1/(P_{max}\cdot T_{max})$ [Operations/mW]

自然対数の底

- e=2.71828…….

$$e = \sum \frac{1}{n!}$$

- このまま計算しても10桁程度であれば普通に求められる――>
- 今回の課題では、任意の桁数まで求めたい
  - 演算器・データパスのビット幅を広げることでは任意の桁数までの演算は保障できない
  - 固定長のデータパスを繰り返し使用することで任意長の演算器を実装する必要がある

```
int y[MAX],x[MAX];
main(argc,argv) int argc; char **argv;
{
        register int n,i;
        if(argc==1) n = 10; else n = atoi(*++argv);
        clr(n,y); clr(n,x); x[0]=1;
        i=0;
        do {
                add(n,y,x);
                div(n,x,++i);
        } while(!zero(n,x));
        if(argc==2) out(n,y);
}
clr(n,a) int n,*a;
{
        while(n--) *a++ = 0;
}
add(n,a,b) int n,*a,*b;
{
        register int *p,*q,r;
        p = a+n; q = b+n; r = 0;
        while(n--) {
                r = *--p + *--q + r;
                *p = r%10; r = r/10;
        }
}
```

```
div(n,a,d) int n,*a,d;
{
        register int *p,m,r;
        p = a; m = d; r = 0;
        while(n--) {
                r = r*10 + *p;
                *p++ = r/m; r = r%m;
        }
}
zero(n,a) int n,*a;
{
        register int *p,m;
        m = n; p = a;
        while(m--) if(*p++) return(0);
        return(1);
}
out(n,a) int n,*a;
{
        register int i;
        for(i=0;i<n;i++) {
                if(i%50==0) putchar('¥n');
                putchar('0'+ *a++);
        }
        putchar('¥n');
}
```

例のように、数字一桁もしくは数桁毎にデータを格納し、その単位で演算を繰り返す
特に多ビットの計算の場合には不可欠な手法

4ビットの数を7-segLEDに数字として表示するには？

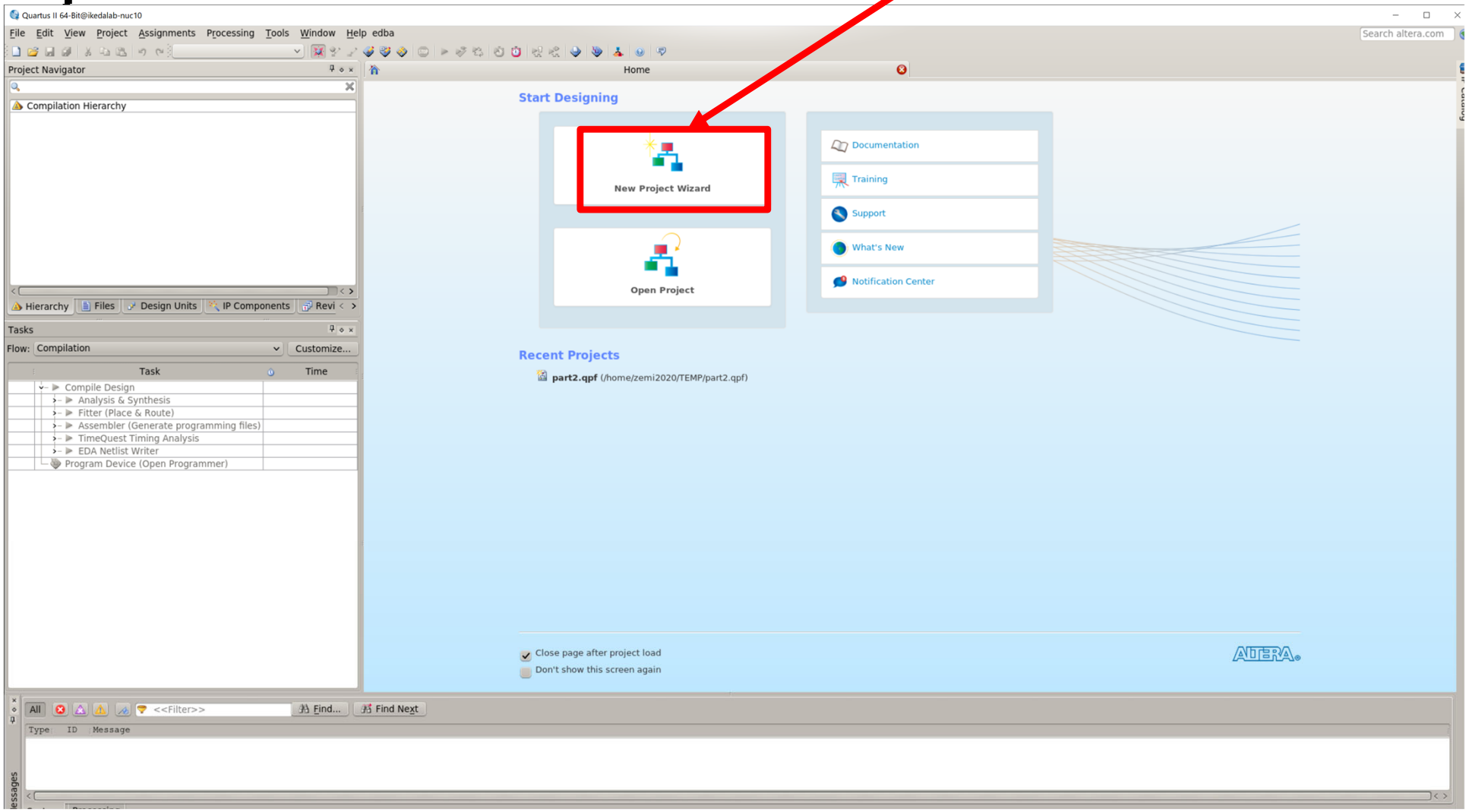
```
`define SEG_OUT_0 7'b011_1111
`define SEG_OUT_1 7'b000_0110
`define SEG_OUT_2 7'b101_1011
`define SEG_OUT_3 7'b100_1111
`define SEG_OUT_4 7'b110_0110
`define SEG_OUT_5 7'b110_1101
`define SEG_OUT_6 7'b111_1101
`define SEG_OUT_7 7'b010_0111
`define SEG_OUT_8 7'b111_1111
`define SEG_OUT_9 7'b110_1111
`define SEG_OUT_A 7'b111_0111
`define SEG_OUT_B 7'b111_1100
`define SEG_OUT_C 7'b011_1001
`define SEG_OUT_D 7'b101_1110
`define SEG_OUT_E 7'b111_1001
`define SEG_OUT_F 7'b111_0001

module SEG_HEX (
            ////////////////////    4 Binary bits Input          ////////////////////
            iDIG,
            ////////////////////    HEX 7-SEG Output             ////////////////////
            oHEX_D
            );

  input         [3:0]   iDIG;
  output [6:0]          oHEX_D;

  reg [6:0]             oHEX_D;
```

```
always @(iDIG)
    begin
      case(iDIG)
        4'h0: oHEX_D <= ~(`SEG_OUT_0);
        4'h1: oHEX_D <= ~(`SEG_OUT_1);
        4'h2: oHEX_D <= ~(`SEG_OUT_2);
        4'h3: oHEX_D <= ~(`SEG_OUT_3);
        4'h4: oHEX_D <= ~(`SEG_OUT_4);
        4'h5: oHEX_D <= ~(`SEG_OUT_5);
        4'h6: oHEX_D <= ~(`SEG_OUT_6);
        4'h7: oHEX_D <= ~(`SEG_OUT_7);
        4'h8: oHEX_D <= ~(`SEG_OUT_8);
        4'h9: oHEX_D <= ~(`SEG_OUT_9);
        4'ha: oHEX_D <= ~(`SEG_OUT_A);
        4'hb: oHEX_D <= ~(`SEG_OUT_B);
        4'hc: oHEX_D <= ~(`SEG_OUT_C);
        4'hd: oHEX_D <= ~(`SEG_OUT_D);
        4'he: oHEX_D <= ~(`SEG_OUT_E);
        4'hf: oHEX_D <= ~(`SEG_OUT_F);
        default: oHEX_D <= ~(`SEG_OUT_0);
      endcase
    end
endmodule
```

設計ツール(Quartus II)の起動

% quartus &

New Project Wizard

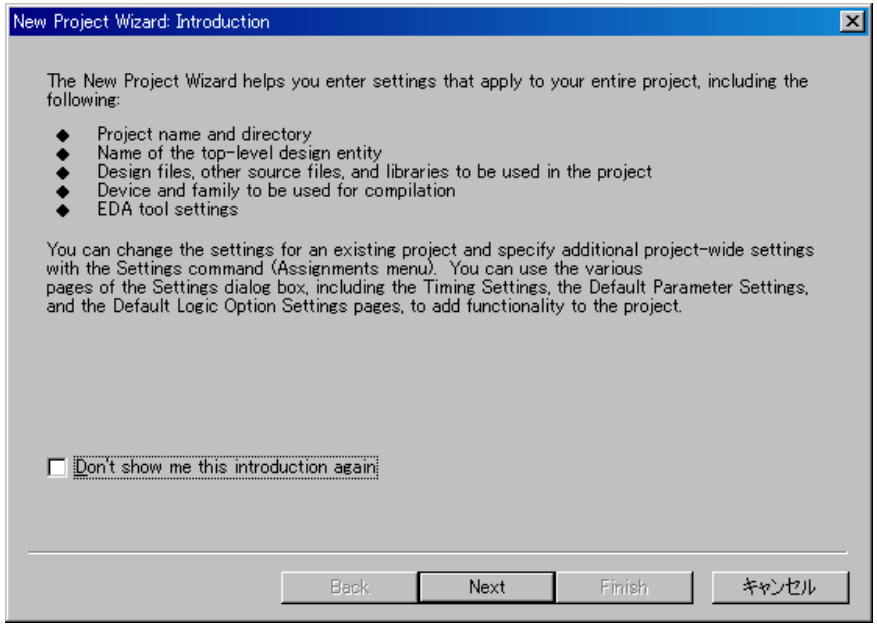


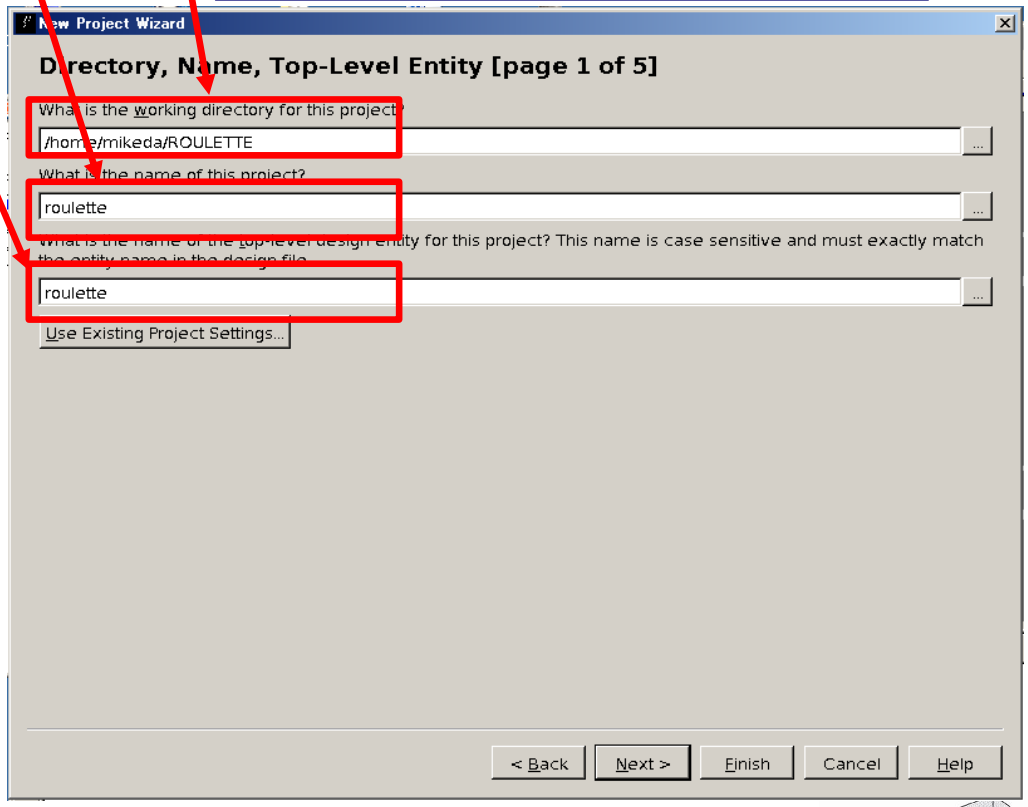
Projectの作成

- File-New Project Wizard
- Working directory　　どこでもよい
- Project name:　**EXE1**
- top-level entity :　**top**
- Nextを選択して次へ

なんでも構わない

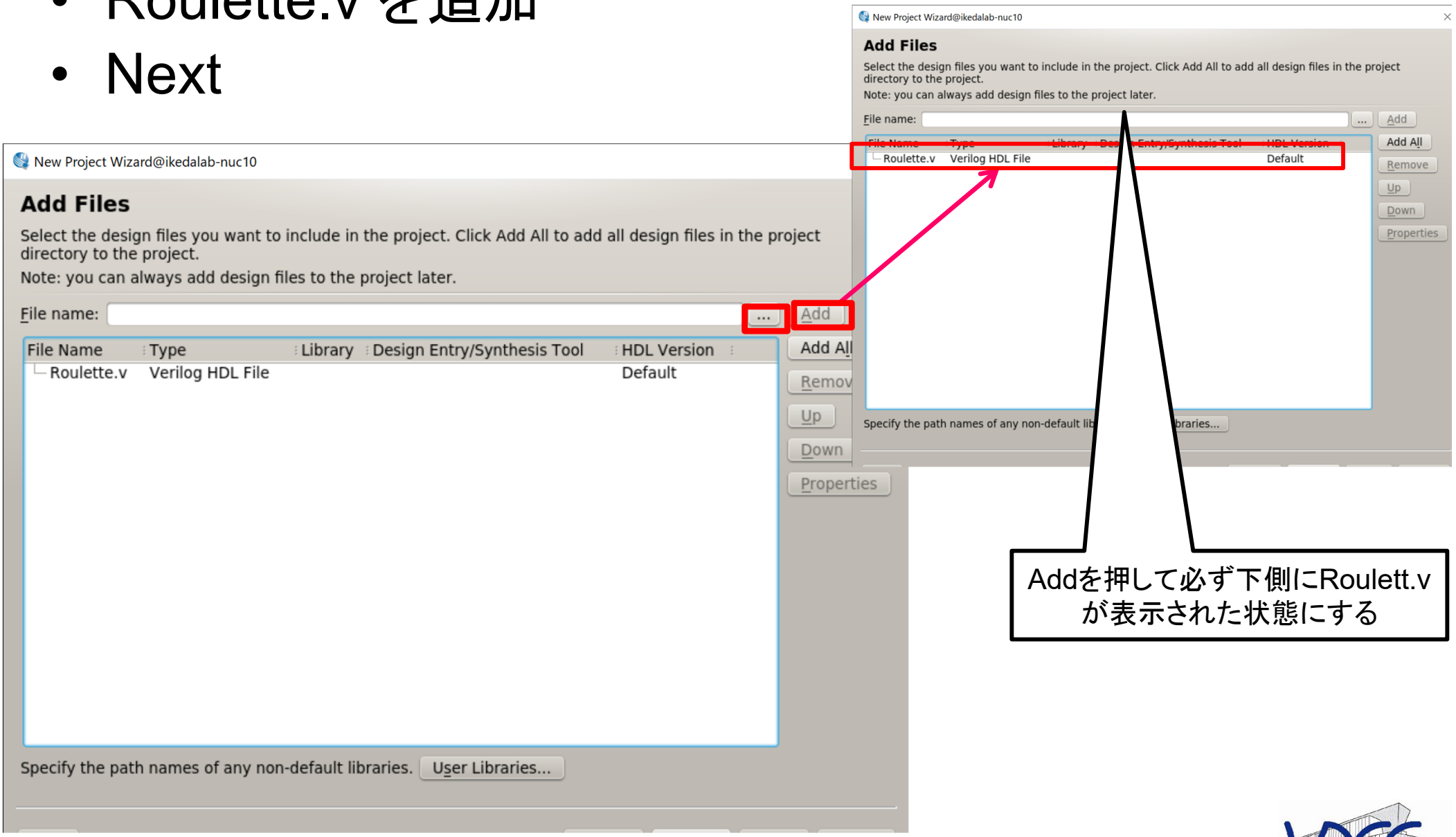
最上位モジュールのモジュール名とすること。デフォルトはProject Nameになるので注意が必要





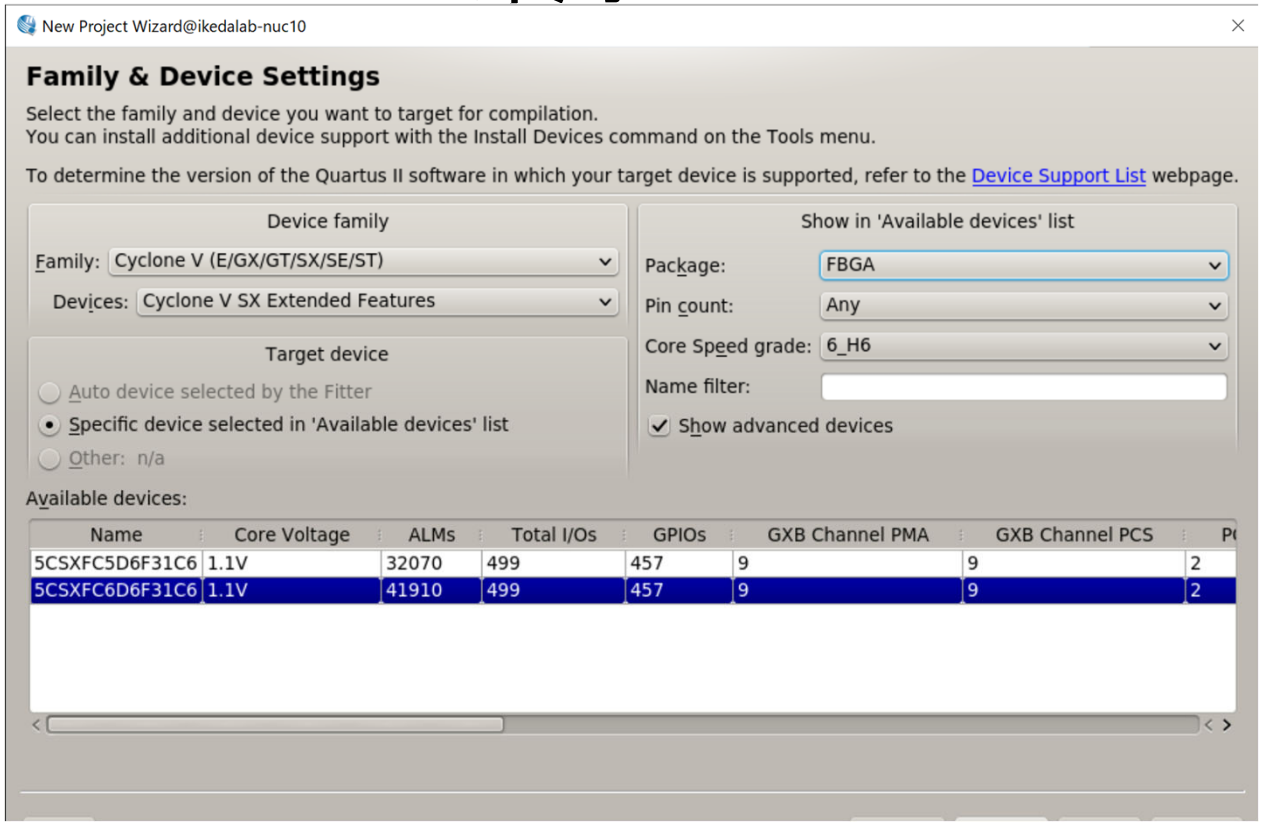
Verilogファイルの追加

- Roulette.v を追加
- Next



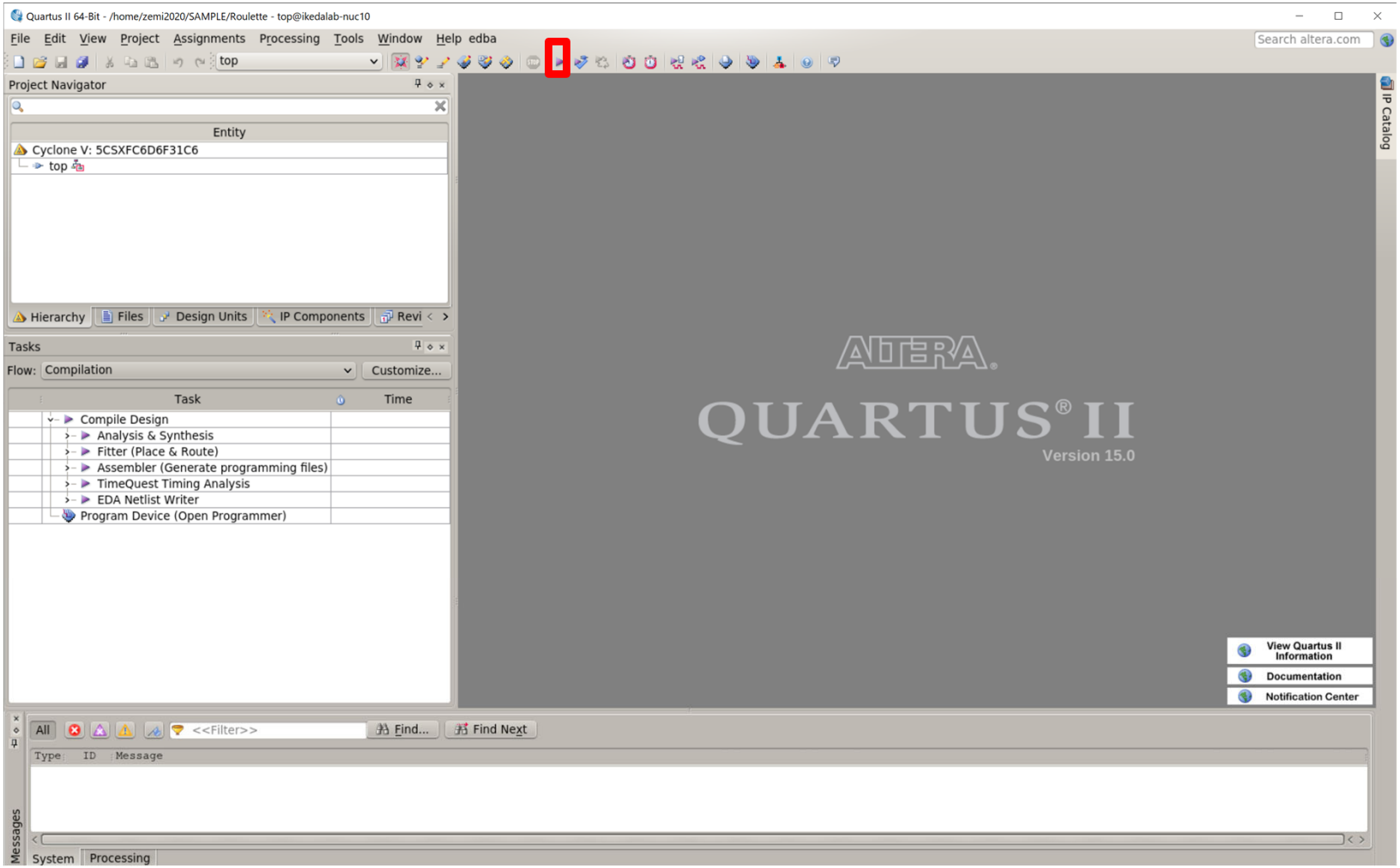
FPGAデバイスの指定

- Assignments-Device
  - FamilyをCycloneV, Cyclone V SX, Package: FBGA, Speed grade 6_H6
  - Available Devicesから5CSXFC6D6F31C6を選択
  - Finishで終了



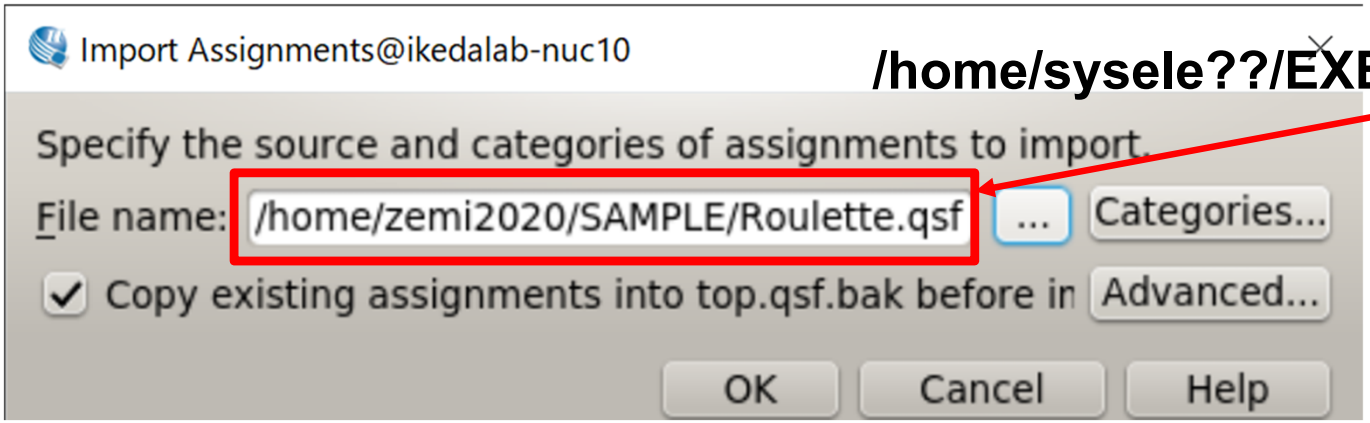
コンパイル

- Processing-Start Compilation(ツールバーの紫色の矢印)
- Full compilation was successfulと表示されたらOKをクリック

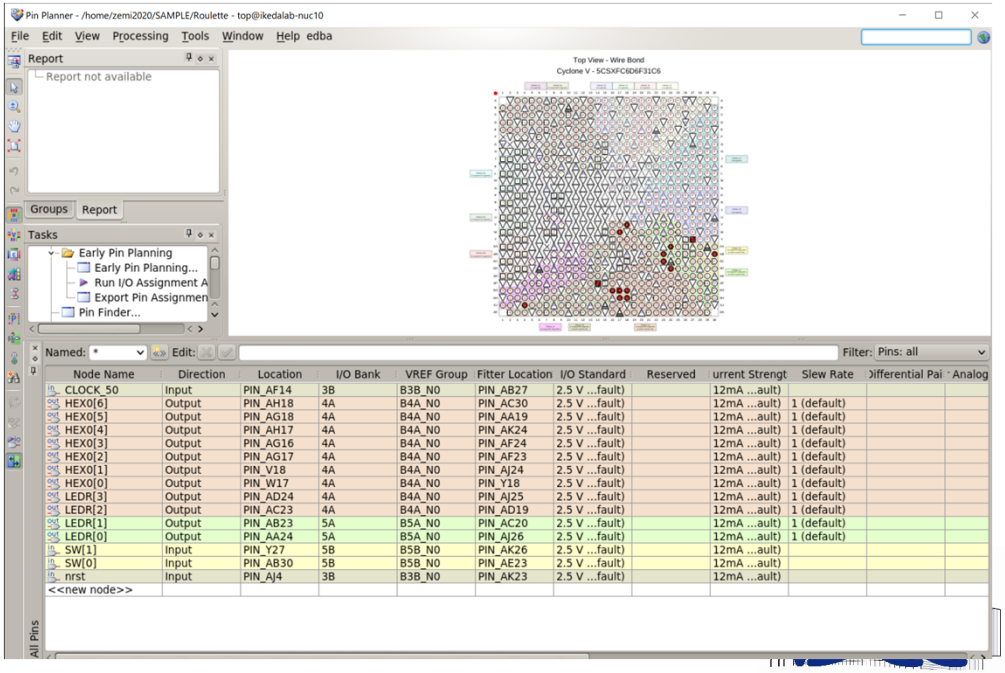


ピンの指定

- Assignments -> Import Assignments

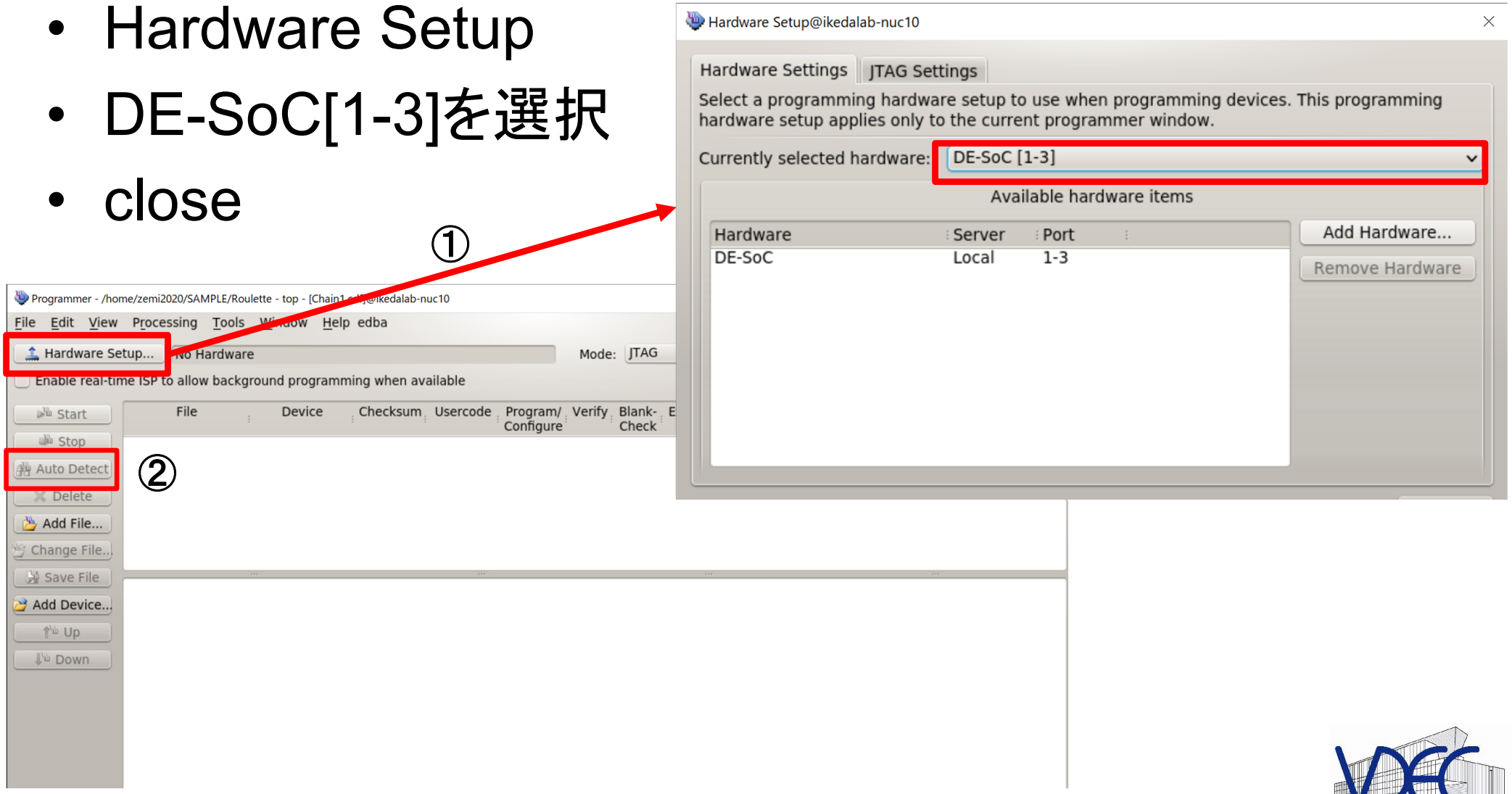


- Assignments -> Pin Plannerでピン番号が設定されていることを確認

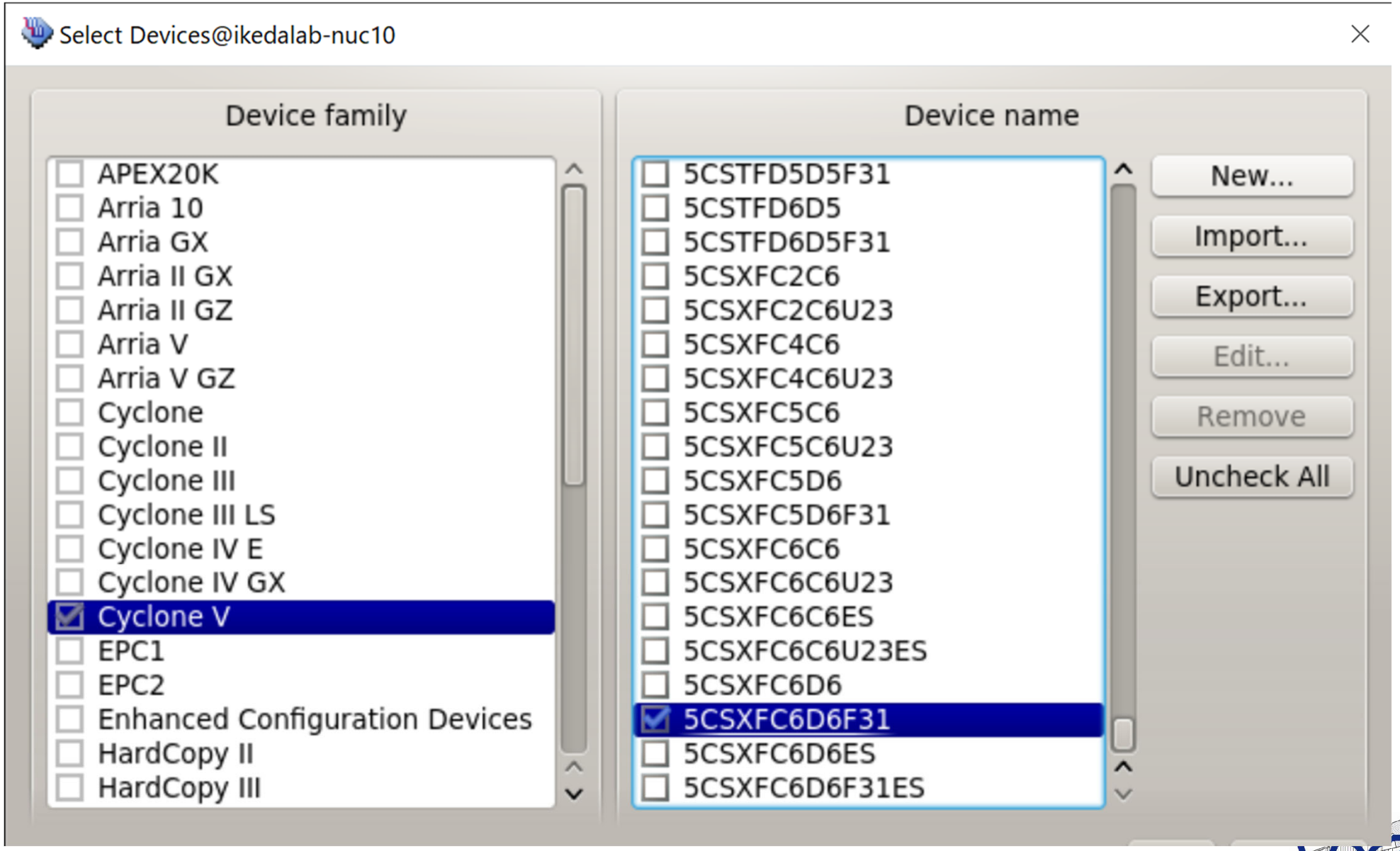


再コンパイル　ボードへのダウンロード

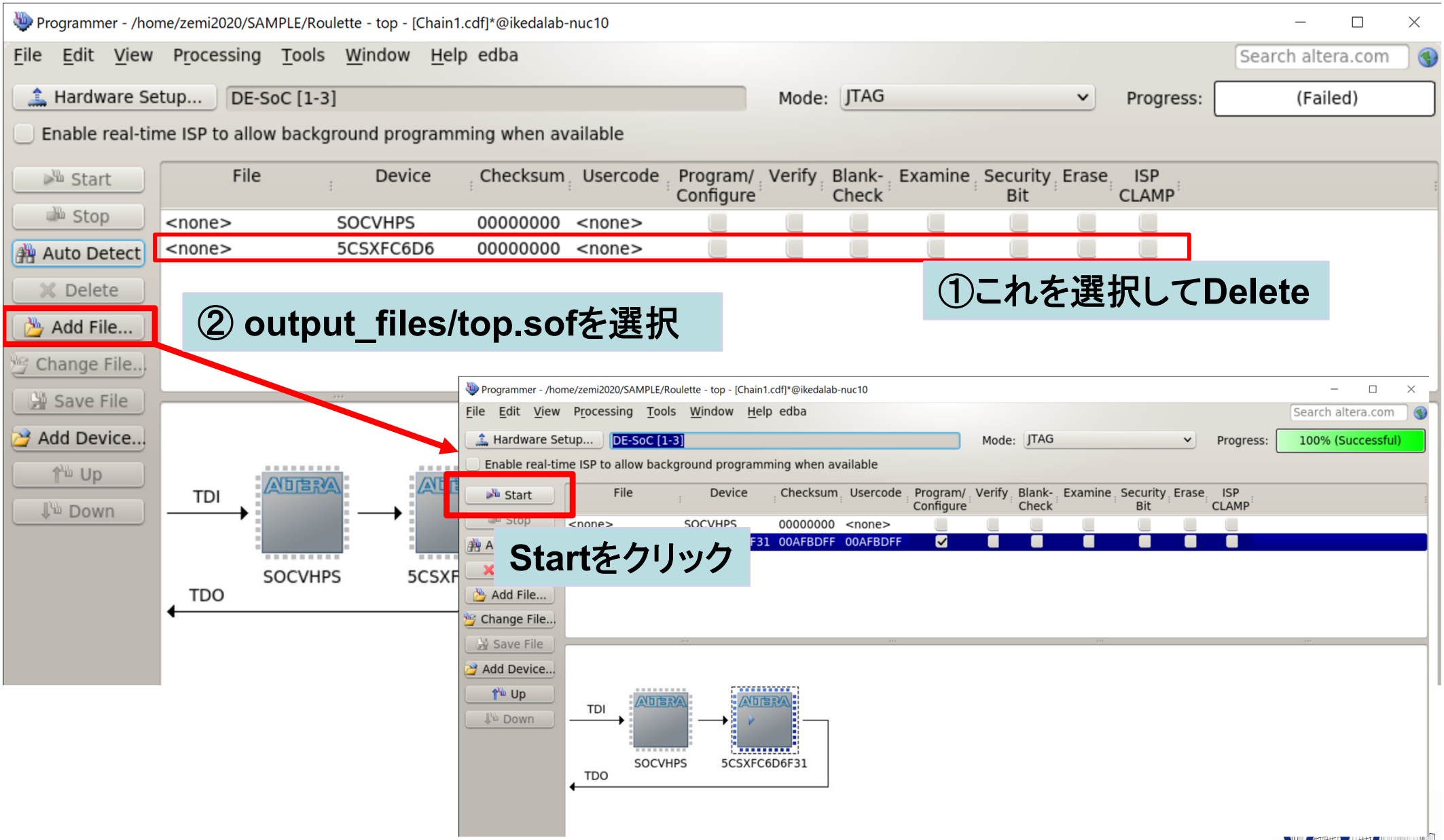
- もう一度Start Compilationを行う
- Tools-Programmerを選択
- Hardware Setup
- DE-SoC[1-3]を選択
- close



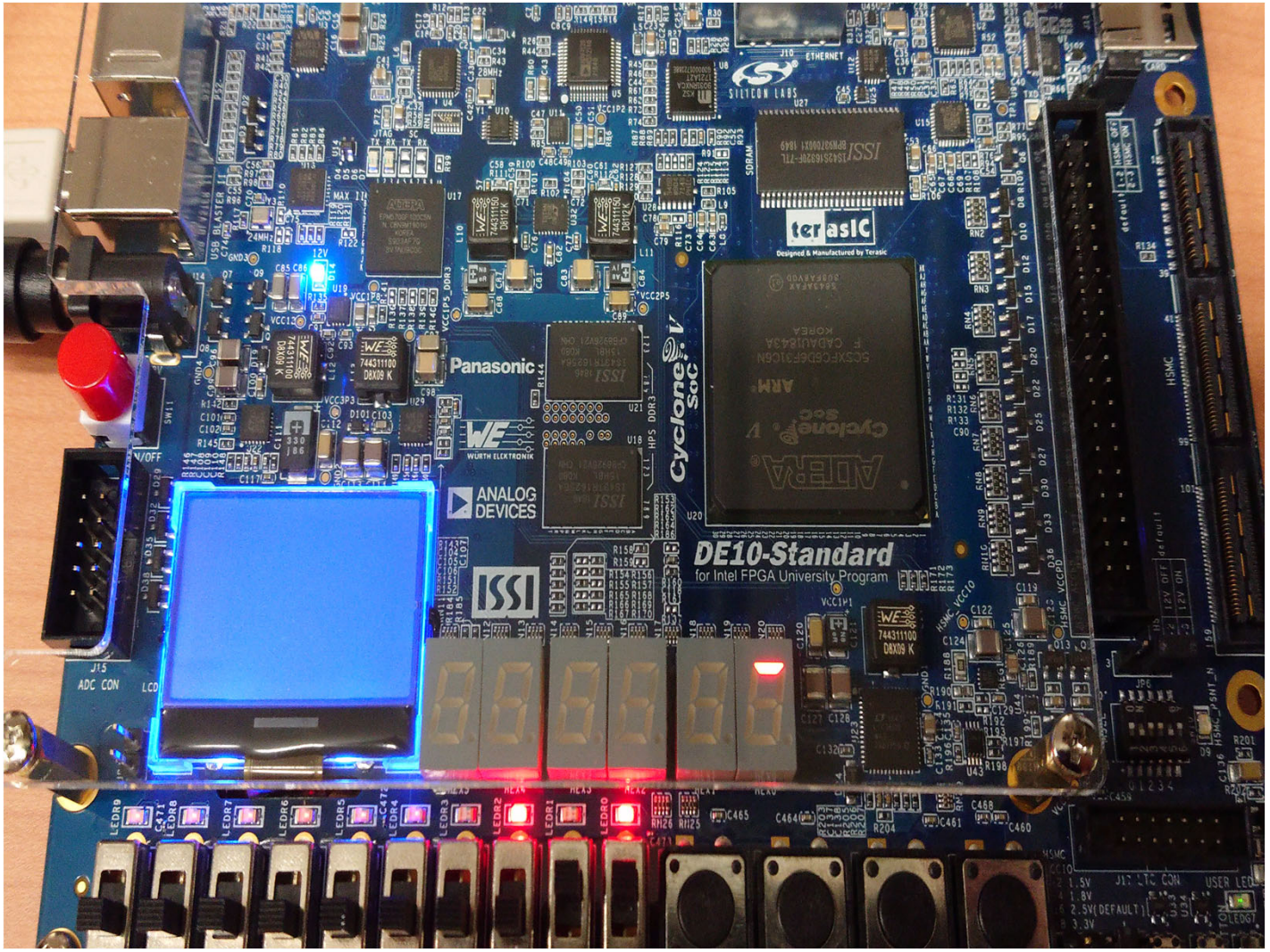
デバイスの選択


Select Devices@ikedalab-nuc10
Device family
APEX20K
Arria 10
Arria GX
Arria II GX
Arria II GZ
Arria V
Arria V GZ
Cyclone
Cyclone II
Cyclone III
Cyclone III LS
Cyclone IV E
Cyclone IV GX
Cyclone V
EPC1
EPC2
Enhanced Configuration Devices
HardCopy II
HardCopy III
Device name
5CSTFD5D5F31
5CSTFD6D5
5CSTFD6D5F31
5CSXFC2C6
5CSXFC2C6U23
5CSXFC4C6
5CSXFC4C6U23
5CSXFC5C6
5CSXFC5C6U23
5CSXFC5D6
5CSXFC5D6F31
5CSXFC6C6
5CSXFC6C6U23
5CSXFC6C6ES
5CSXFC6C6U23ES
5CSXFC6D6
5CSXFC6D6F31
5CSXFC6D6ES
5CSXFC6D6F31ES
New...
Import...
Export...
Edit...
Remove
Uncheck All

ダウンロードファイルの選択

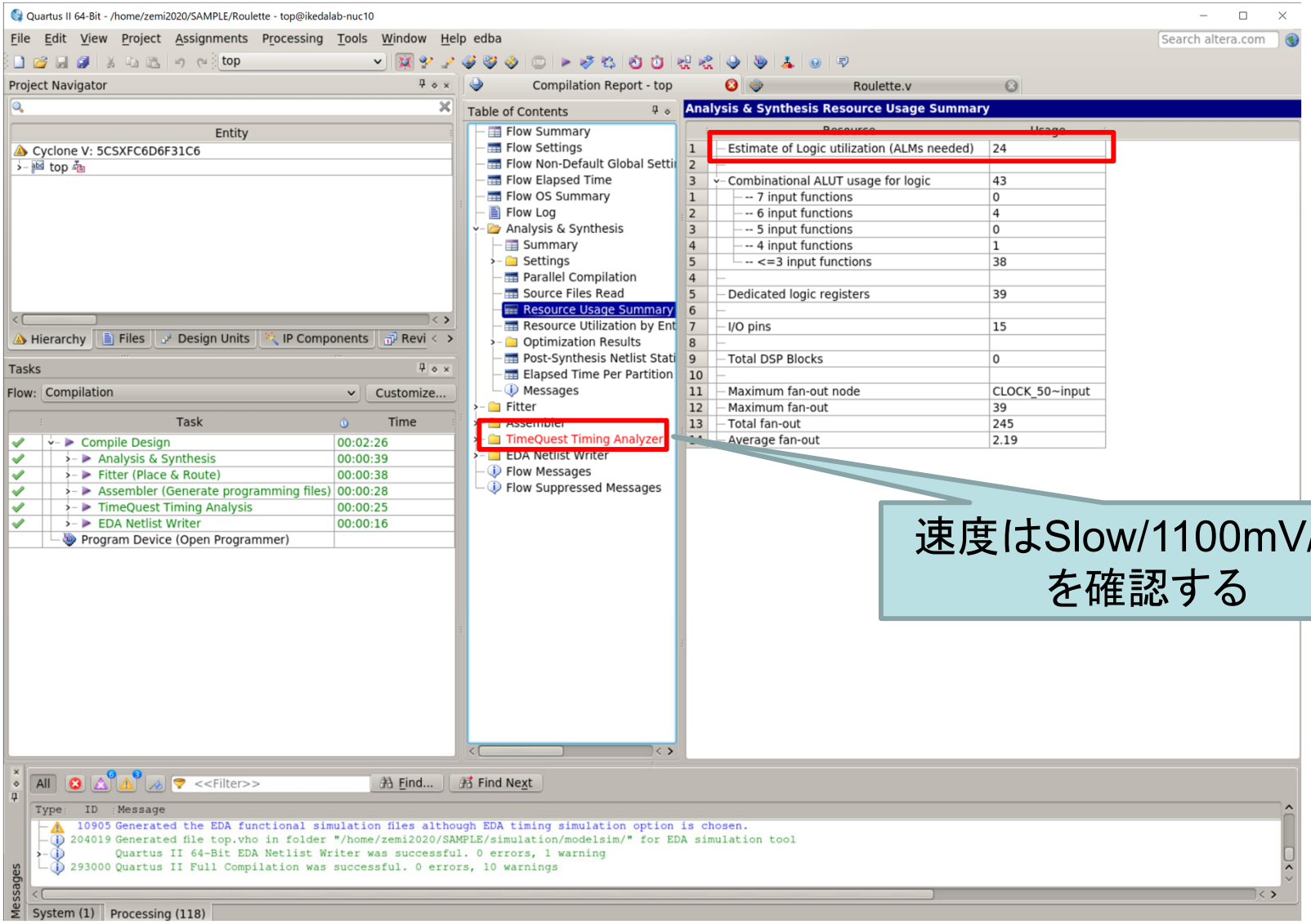


結果

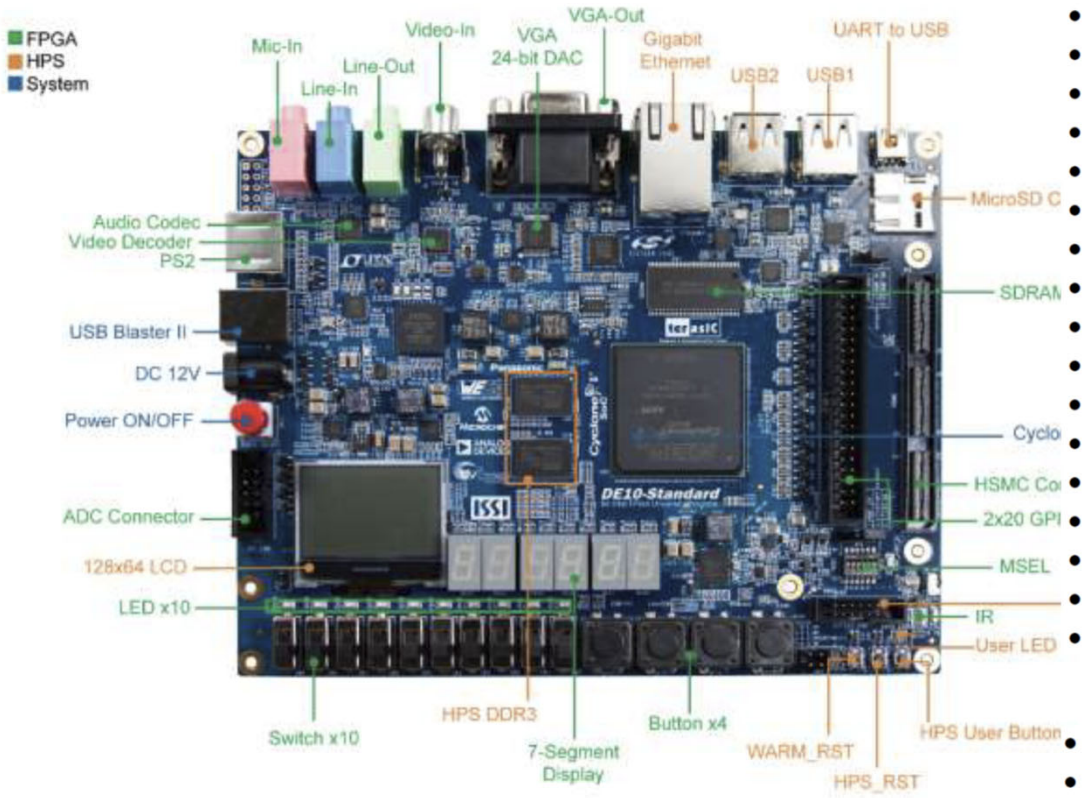

terasIC
Designed & Manufactured by Terasic
Panasonic
ANALOG DEVICES
ISSI
DE10-Standard
for Intel FPGA University Program
Cyclone V SoC
ALTERA
ARM
ETHERNET
SILICON LABS
ADC CON
LCD

結果の確認

- ゲート規模、および動作速度を記録しておく



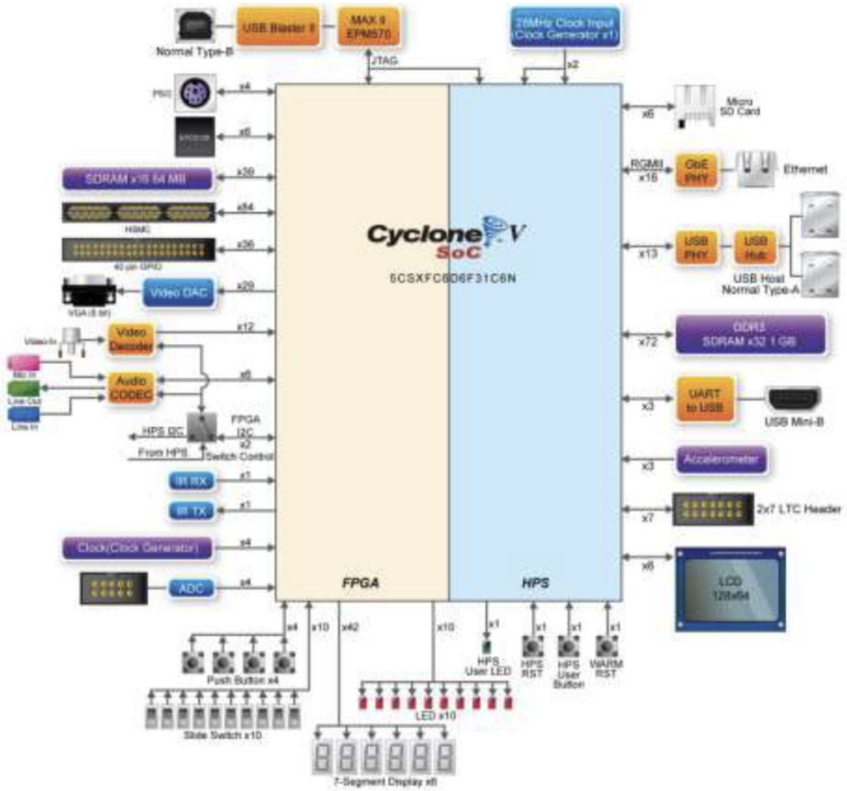
FPGAの仕様



- Intel Cyclone® V SE 5CSXFC6D6F31C6N device
- Serial configuration device – EPCS128
- USB-Blaster II onboard for programming; JTAG Mode
- 64MB SDRAM (16-bit data bus)
- 4 push-buttons
- 10 slide switches
- 10 red user LEDs
- Six 7-segment displays
- Four 50MHz clock sources from the clock generator
- 24-bit CD-quality audio CODEC with line-in, line-out, and microphone-in jacks
- VGA DAC (8-bit high-speed triple DACs) with VGA-out connector
- TV decoder (NTSC/PAL/SECAM) and TV-in connector
- PS/2 mouse/keyboard connector
- IR receiver and IR emitter
- One HSMC with Configurable I/O standard 1.5/1.8/2.5/3.3
- One 40-pin expansion header with diode protection
- A/D converter, 4-pin SPI interface with FPGA

- 925MHz Dual-core ARM Cortex-A9 MPCore processor
- 1GB DDR3 SDRAM (32-bit data bus)
- 1 Gigabit Ethernet PHY with RJ45 connector
- 2-port USB Host, normal Type-A USB connector
- Micro SD card socket
- Accelerometer (I2C interface + interrupt)
- UART to USB, USB Mini-B connector
- Warm reset button and cold reset button
- One user button and one user LED
- LTC 2x7 expansion header
- 128x64 dots LCD Module with Backlight

FPGAボードの構成



FPGA Device

- Cyclone V SoC 5CSXFC6D6F31C6N Device
- Dual-core ARM Cortex-A9 (HPS)
- 110K programmable logic elements
- 5,761 Kbits embedded memory
- 6 fractional PLLs
- 2 hard memory controllers

Memory Device

- 64MB (32Mx16) SDRAM on FPGA
- 1GB (2x256Mx16) DDR3 SDRAM on HPS
- Micro SD card socket on HPS

Display

- 24-bit VGA DAC
- 128x64 dots LCD Module with Backlight

Audio

- 24-bit CODEC, Line-in, Line-out, and microphone-in jacks

Communication

- Two port USB 2.0 Host (ULPI interface with USB type A connector)
- UART to USB (USB Mini-B connector)
- 10/100/1000 Ethernet
- PS/2 mouse/keyboard
- IR emitter/receiver
- I2C multiplexer

ADC

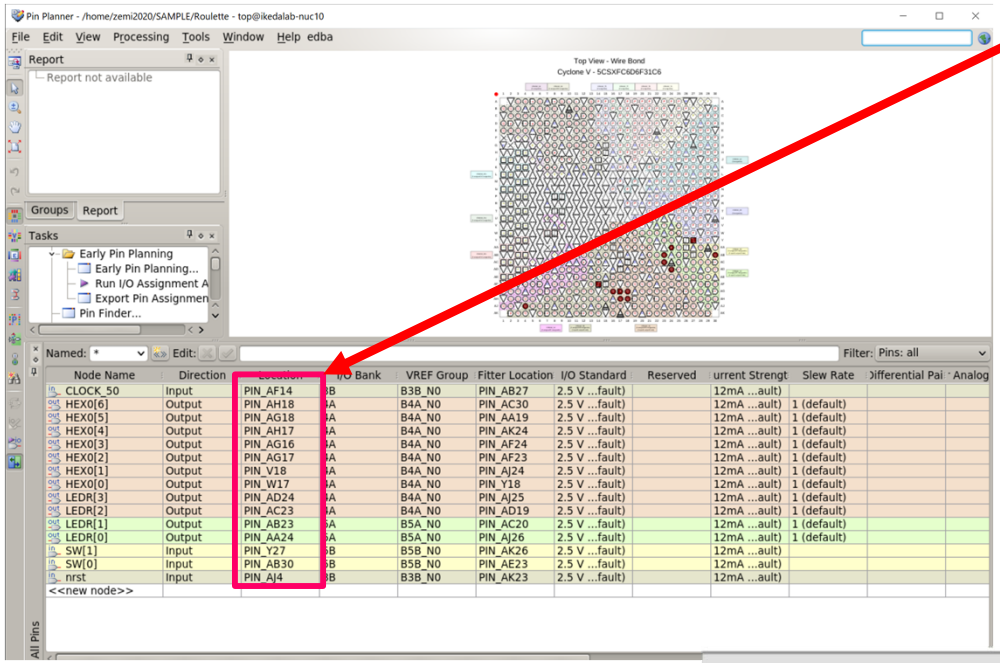
- Interface: SPI
- Fast throughput rate: 500 KSPS
- Channel number: 8
- Resolution: 12-bit
- Analog input range : 0 ~ 4.096

Sensors

- G-Sensor on HPS

新たなピンを使用する場合の設定

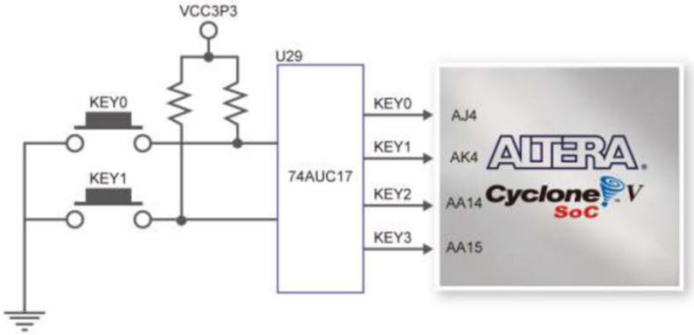
- Assignments -> Pin Plannerでピン番号を設定する



通常クロックはCLOCK_50を利用

Signal Name	*FPGA Pin No.*	*Description*	*I/O Standard*
CLOCK_50	PIN_AF14	50 MHz clock input	3.3V
CLOCK2_50	PIN_AA16	50 MHz clock input	3.3V
CLOCK3_50	PIN_Y26	50 MHz clock input	3.3V
CLOCK4_50	PIN_K14	50 MHz clock input	3.3V
HPS_CLOCK1_25	PIN_D25	25 MHz clock input	3.3V
HPS_CLOCK2_25	PIN_F25	25 MHz clock input	3.3V

ボタン・スイッチの使用

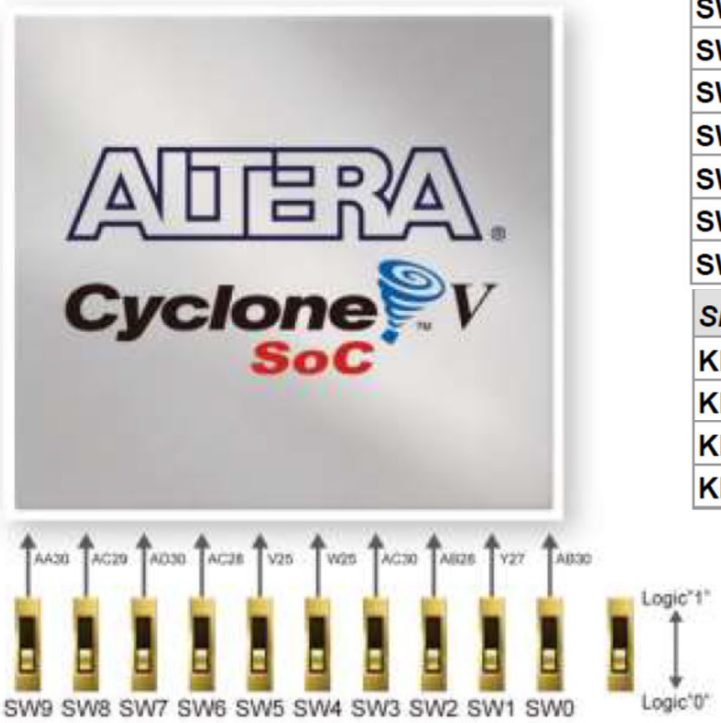


Keyは押したとき 「0」、離すと「1」

SWは下が 「0」、上が「1」

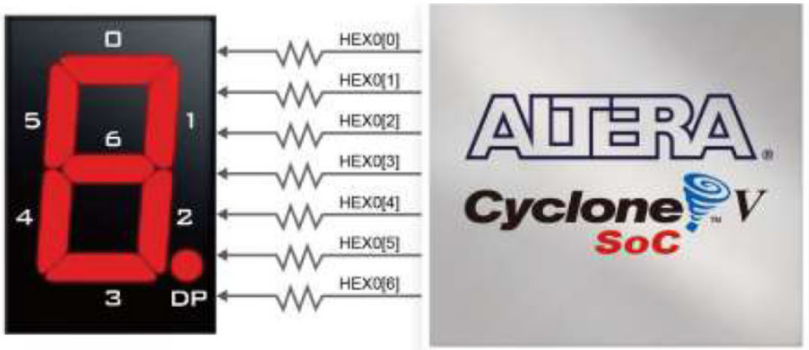
Signal Name	*FPGA Pin No.*	*Description*	*I/O Standard*
SW[0]	PIN_AB30	Slide Switch[0]	Depend on JP3
SW[1]	PIN_Y27	Slide Switch[1]	Depend on JP3
SW[2]	PIN_AB28	Slide Switch[2]	Depend on JP3
SW[3]	PIN_AC30	Slide Switch[3]	Depend on JP3
SW[4]	PIN_W25	Slide Switch[4]	Depend on JP3
SW[5]	PIN_V25	Slide Switch[5]	Depend on JP3
SW[6]	PIN_AC28	Slide Switch[6]	Depend on JP3
SW[7]	PIN_AD30	Slide Switch[7]	Depend on JP3
SW[8]	PIN_AC29	Slide Switch[8]	Depend on JP3
SW[9]	PIN_AA30	Slide Switch[9]	Depend on JP3

Signal Name	*FPGA Pin No.*	*Description*	*I/O Standard*
KEY[0]	PIN_AJ4	Push-button[0]	3.3V
KEY[1]	PIN_AK4	Push-button[1]	3.3V
KEY[2]	PIN_AA14	Push-button[2]	3.3V
KEY[3]	PIN_AA15	Push-button[3]	3.3V



LEDの使用

Signal Name	FPGA Pin No.	Description	I/O Standard
LEDR[0]	PIN_AA24	LED [0]	3.3V
LEDR[1]	PIN_AB23	LED [1]	3.3V
LEDR[2]	PIN_AC23	LED [2]	3.3V
LEDR[3]	PIN_AD24	LED [3]	3.3V
LEDR[4]	PIN_AG25	LED [4]	3.3V
LEDR[5]	PIN_AF25	LED [5]	3.3V
LEDR[6]	PIN_AE24	LED [6]	3.3V
LEDR[7]	PIN_AF24	LED [7]	3.3V
LEDR[8]	PIN_AB22	LED [8]	3.3V
LEDR[9]	PIN_AC22	LED [9]	3.3V



Signal Name	FPGA Pin No.	Description	I/O Standard
HEX0[0]	PIN_W17	Seven Segment Digit 0[0]	3.3V
HEX0[1]	PIN_V18	Seven Segment Digit 0[1]	3.3V
HEX0[2]	PIN_AG17	Seven Segment Digit 0[2]	3.3V
HEX0[3]	PIN_AG16	Seven Segment Digit 0[3]	3.3V
HEX0[4]	PIN_AH17	Seven Segment Digit 0[4]	3.3V
HEX0[5]	PIN_AG18	Seven Segment Digit 0[5]	3.3V
HEX0[6]	PIN_AH18	Seven Segment Digit 0[6]	3.3V
HEX1[0]	PIN_AF16	Seven Segment Digit 1[0]	3.3V
HEX1[1]	PIN_V16	Seven Segment Digit 1[1]	3.3V
HEX1[2]	PIN_AE16	Seven Segment Digit 1[2]	3.3V
HEX1[3]	PIN_AD17	Seven Segment Digit 1[3]	3.3V
HEX1[4]	PIN_AE18	Seven Segment Digit 1[4]	3.3V
HEX1[5]	PIN_AE17	Seven Segment Digit 1[5]	3.3V
HEX1[6]	PIN_V17	Seven Segment Digit 1[6]	3.3V

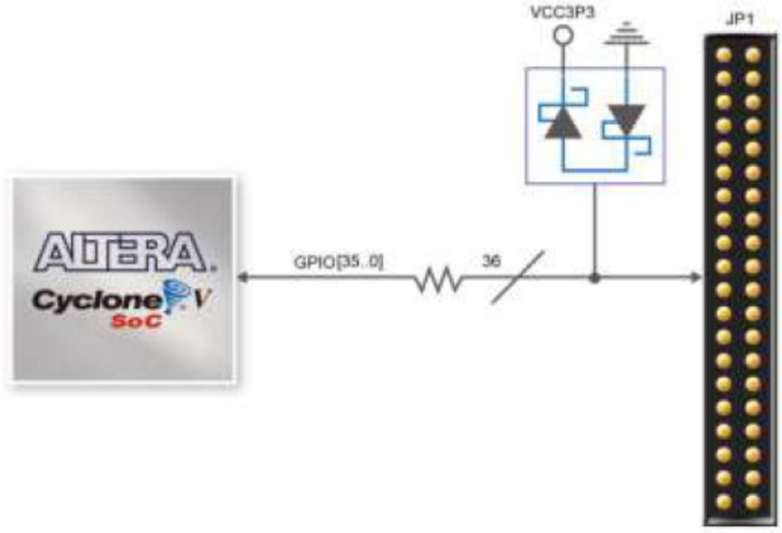
LEDは「1」で点灯

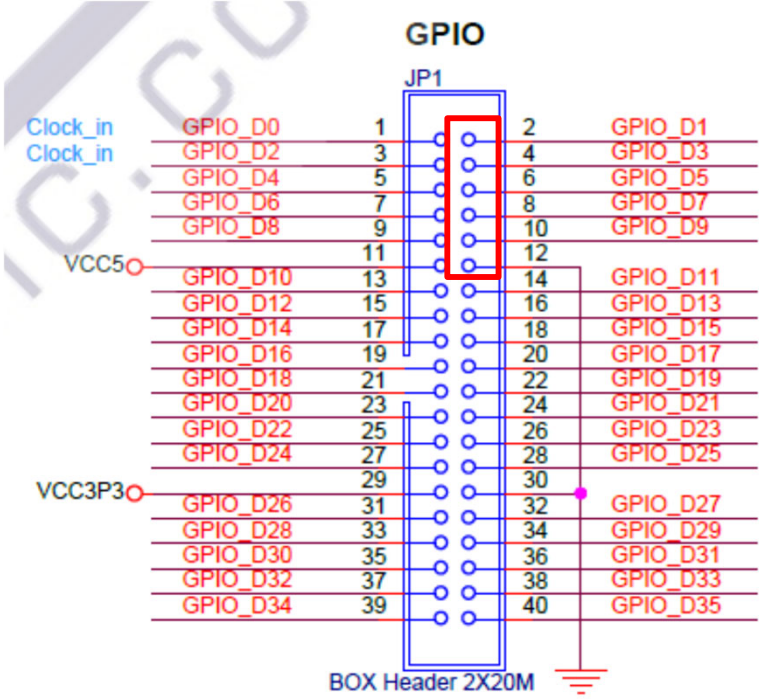
HEX(7セグ) 「0」で点灯

HEX2[0]	PIN_AA21	Seven Segment Digit 2[0]	3.3V
HEX2[1]	PIN_AB17	Seven Segment Digit 2[1]	3.3V
HEX2[2]	PIN_AA18	Seven Segment Digit 2[2]	3.3V
HEX2[3]	PIN_Y17	Seven Segment Digit 2[3]	3.3V
HEX2[4]	PIN_Y18	Seven Segment Digit 2[4]	3.3V
HEX2[5]	PIN_AF18	Seven Segment Digit 2[5]	3.3V
HEX2[6]	PIN_W16	Seven Segment Digit 2[6]	3.3V
HEX3[0]	PIN_Y19	Seven Segment Digit 3[0]	3.3V
HEX3[1]	PIN_W19	Seven Segment Digit 3[1]	3.3V
HEX3[2]	PIN_AD19	Seven Segment Digit 3[2]	3.3V
HEX3[3]	PIN_AA20	Seven Segment Digit 3[3]	3.3V
HEX3[4]	PIN_AC20	Seven Segment Digit 3[4]	3.3V
HEX3[5]	PIN_AA19	Seven Segment Digit 3[5]	3.3V
HEX3[6]	PIN_AD20	Seven Segment Digit 3[6]	3.3V
HEX4[0]	PIN_AD21	Seven Segment Digit 4[0]	3.3V
HEX4[1]	PIN_AG22	Seven Segment Digit 4[1]	3.3V
HEX4[2]	PIN_AE22	Seven Segment Digit 4[2]	3.3V
HEX4[3]	PIN_AE23	Seven Segment Digit 4[3]	3.3V
HEX4[4]	PIN_AG23	Seven Segment Digit 4[4]	3.3V
HEX4[5]	PIN_AF23	Seven Segment Digit 4[5]	3.3V
HEX4[6]	PIN_AH22	Seven Segment Digit 4[6]	3.3V
HEX5[0]	PIN_AF21	Seven Segment Digit 5[0]	3.3V
HEX5[1]	PIN_AG21	Seven Segment Digit 5[1]	3.3V
HEX5[2]	PIN_AF20	Seven Segment Digit 5[2]	3.3V
HEX5[3]	PIN_AG20	Seven Segment Digit 5[3]	3.3V
HEX5[4]	PIN_AE19	Seven Segment Digit 5[4]	3.3V
HEX5[5]	PIN_AF19	Seven Segment Digit 5[5]	3.3V
HEX5[6]	PIN_AB21	Seven Segment Digit 5[6]	3.3V

GPIOの使用

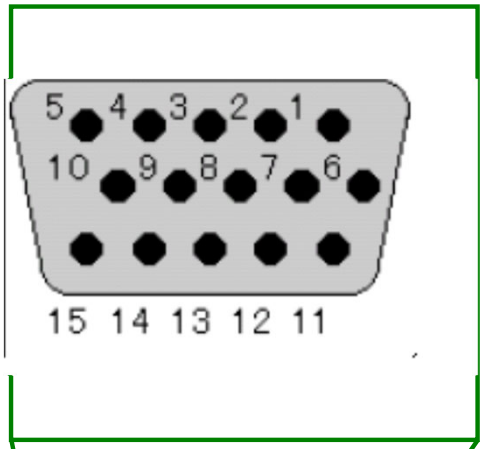
Signal Name	FPGA Pin No.	Description	I/O Standard
GPIO[0]	PIN_W15	GPIO Connection 0[0]	3.3V
GPIO[1]	PIN_AK2	GPIO Connection 0[1]	3.3V
GPIO[2]	PIN_Y16	GPIO Connection 0[2]	3.3V
GPIO[3]	PIN_AK3	GPIO Connection 0[3]	3.3V
GPIO[4]	PIN_AJ1	GPIO Connection 0[4]	3.3V
GPIO[5]	PIN_AJ2	GPIO Connection 0[5]	3.3V
GPIO[6]	PIN_AH2	GPIO Connection 0[6]	3.3V
GPIO[7]	PIN_AH3	GPIO Connection 0[7]	3.3V
GPIO[8]	PIN_AH4	GPIO Connection 0[8]	3.3V
GPIO[9]	PIN_AH5	GPIO Connection 0[9]	3.3V
GPIO[10]	PIN_AG1	GPIO Connection 0[10]	3.3V
GPIO[11]	PIN_AG2	GPIO Connection 0[11]	3.3V
GPIO[12]	PIN_AG3	GPIO Connection 0[12]	3.3V
GPIO[13]	PIN_AG5	GPIO Connection 0[13]	3.3V
GPIO[14]	PIN_AG6	GPIO Connection 0[14]	3.3V
GPIO[15]	PIN_AG7	GPIO Connection 0[15]	3.3V
GPIO[16]	PIN_AG8	GPIO Connection 0[16]	3.3V
GPIO[17]	PIN_AF4	GPIO Connection 0[17]	3.3V
GPIO[18]	PIN_AF5	GPIO Connection 0[18]	3.3V
GPIO[19]	PIN_AF6	GPIO Connection 0[19]	3.3V
GPIO[20]	PIN_AF8	GPIO Connection 0[20]	3.3V
GPIO[21]	PIN_AF9	GPIO Connection 0[21]	3.3V
GPIO[22]	PIN_AF10	GPIO Connection 0[22]	3.3V
GPIO[23]	PIN_AE7	GPIO Connection 0[23]	3.3V
GPIO[24]	PIN_AE9	GPIO Connection 0[24]	3.3V
GPIO[25]	PIN_AE11	GPIO Connection 0[25]	3.3V
GPIO[26]	PIN_AE12	GPIO Connection 0[26]	3.3V
GPIO[27]	PIN_AD7	GPIO Connection 0[27]	3.3V
GPIO[28]	PIN_AD9	GPIO Connection 0[28]	3.3V
GPIO[29]	PIN_AD10	GPIO Connection 0[29]	3.3V
GPIO[30]	PIN_AD11	GPIO Connection 0[30]	3.3V
GPIO[31]	PIN_AD12	GPIO Connection 0[31]	3.3V
GPIO[32]	PIN_AC9	GPIO Connection 0[32]	3.3V
GPIO[33]	PIN_AC12	GPIO Connection 0[33]	3.3V
GPIO[34]	PIN_AB12	GPIO Connection 0[34]	3.3V
GPIO[35]	PIN_AA12	GPIO Connection 0[35]	3.3V





VGA (Video Graphics Array)

- ディスプレイ表示のための規格
- R(赤),G(緑),B(青)の3つの輝度信号
- 水平同期信号
- 垂直同期信号



VGAコネクタ

VGAケーブル

PC

モニタ

VGAコネクタピン配列

1 RED: 赤色信号入力

2 GREEN: 緑色信号入力

3 BLUE: 青色信号入力

4 ID2

5 GND

6 RGND

7 GGND

8 BGND

9 KEY

10 SGND

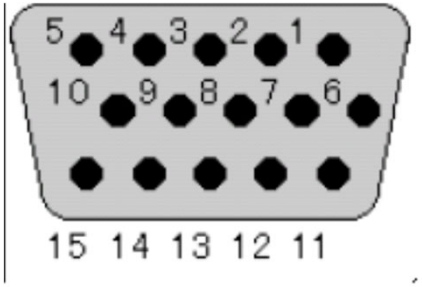
11 ID0

12 ID1

13 HSYNC: 水平同期信号入力

14 VSYNC: 垂直同期信号入力

15 ID3



映像を表示させるために必要な信号は、赤青緑の信号線と、その帰線。そして、同期信号とその帰線です。帰線はすべてGNDにつないでしまってOKです。一見して、GNDが多いように見えますが、高速な電気信号を扱う場合にはGNDは多くなるもので、VGAコネクタのGNDは妥当な数です。
RGBの信号は75Ω、0.7Vppです。

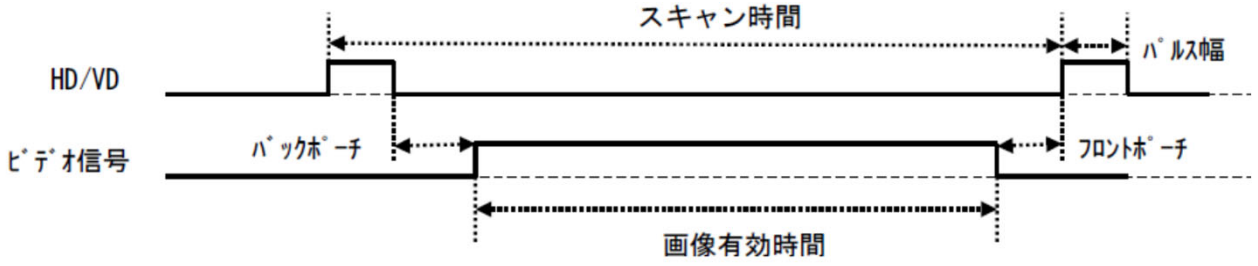
赤色: 必須信号

VGAの同期信号

モニタモード	水平 周波数	垂直 周波数（Hz）	ピクセルク ロック	同期極性 （水平 /
DOS 720 x 400	31.5	70.1	28.3	-/+
VGA 640 x 480	31.5	60.0	25.2	-/-
VESA 640 x 480	37.5	75.0	31.5	-/-
VESA 800 x 600	37.9	60.0	40.0	+/+
VESA 800 x 600	46.9	75.0	49.5	+/+
VESA 1024 x 768	48.4	60.0	65.0	-/-
VESA 1024 x 768	60.0	75.0	78.8	+/+
VESA 1152 x 864	67.5	75.0	108.0	+/+
VESA 1280 x 1024	64.0	60.0	108.0	+/+
VESA 1280 x 1024	80.0	75.0	135.0	+/+

電気

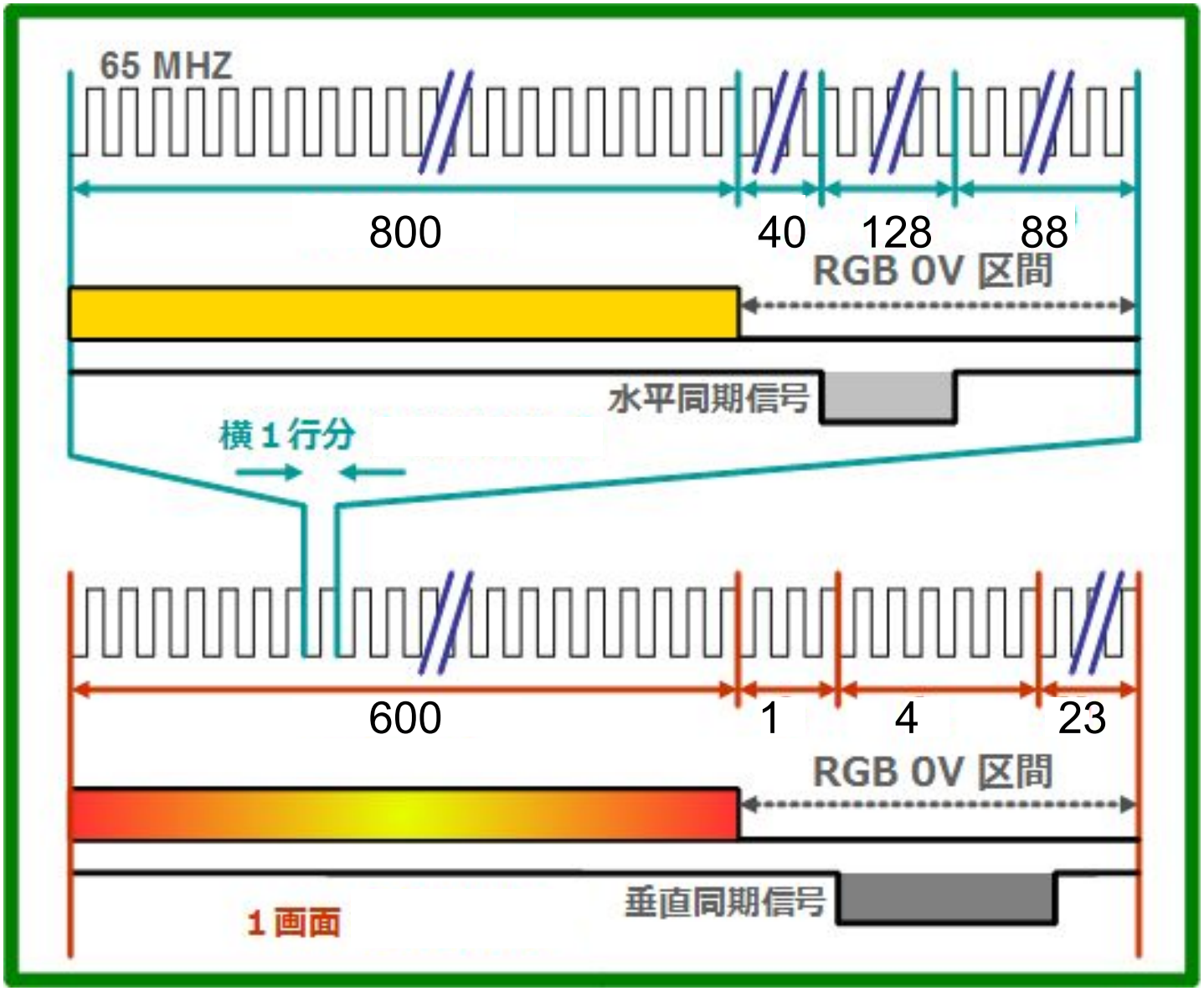
ビデオ入力信号	アナログ RGB、0.7 V +/-5 %、75 Ω入力インピーダンス
同期入力信号	別々の水平信号と垂直信号： 3.3V Cmos または 5V TTL レベル、正同期または負同期



同期信号と解像度

	VGA 640x480				SVGA 800x600				
水平ｽｷｬﾝﾚｰﾄ(KHz)	31.469	37.861	37.5	43.269	35.156	37.879	48.077	46.875	53.674
垂直ｽｷｬﾝﾚｰﾄ(Hz)	59.94	72.809	75	85.008	56.25	60.317	72.188	75	85.061
ﾋﾟｸｾﾙｸﾛｯｸ(MHz)	25.175	31.5	31.5	36	36	40	50	49.5	56.25
水平ｽｷｬﾝ時間	800	832	840	832	1024	1056	1040	1056	1048
水平ﾌﾛﾝﾄﾎﾟｰﾁ	8	16	16	56	24	40	56	16	32
水平同期ﾊﾟﾙｽ幅	96	40	64	56	72	128	120	80	64
水平ﾊﾞｯｸﾎﾟｰﾁ	40	120	120	80	128	88	64	160	152
垂直ｽｷｬﾝ期間	525	520	500	509	625	628	666	625	631
垂直ﾌﾛﾝﾄﾎﾟｰﾁ	2	1	1	1	1	1	37	1	1
垂直同期ﾊﾟﾙｽ幅	2	3	3	3	2	4	6	3	3
垂直ﾊﾞｯｸﾎﾟｰﾁ	25	20	16	25	22	23	23	21	27
同期ﾊﾟﾙｽ極性	負	負	負	負	正	正	正	正	正

	XGA 1024x768				SXGA 1280x1024			UXGA 1600x1200	
水平ｽｷｬﾝﾚｰﾄ(KHz)	48.363	56.476	60.023	68.677	63.981	79.976	91.146	75	81.25
垂直ｽｷｬﾝﾚｰﾄ(Hz)	60.004	70.069	75.029	84.997	60.02	75.025	85.024	60	65
ﾋﾟｸｾﾙｸﾛｯｸ(MHz)	65	75	78.75	94.5	108	135	157.5	162	175.5
水平ｽｷｬﾝ時間	1344	1328	1312	1376	1688	1688	1728	2160	2160
水平ﾌﾛﾝﾄﾎﾟｰﾁ	24	24	16	48	48	16	64	64	64
水平同期ﾊﾟﾙｽ幅	136	136	96	96	112	144	160	192	192
水平ﾊﾞｯｸﾎﾟｰﾁ	160	144	176	208	248	248	224	304	304
垂直ｽｷｬﾝ時間	806	806	800	808	1066	1066	1072	1250	1250
垂直ﾌﾛﾝﾄﾎﾟｰﾁ	3	3	1	1	1	1	1	1	1
垂直同期ﾊﾟﾙｽ幅	6	6	3	3	3	3	3	3	3
垂直ﾊﾞｯｸﾎﾟｰﾁ	29	29	28	36	38	38	44	46	46
同期ﾊﾟﾙｽ極性	負	負	正	正	正	正	正	正	正


65 MHZ
800
40
128
88
RGB 0V 区間
水平同期信号
横１行分
600
1
4
23
RGB 0V 区間
垂直同期信号
１画面

同期信号の生成

```
`define HDOT    1056
`define HDOT0   1055
`define HFP     40
`define HSYNC   128
`define HBP     88

module hsync(ck,hvalid,hsync,rst);
input ck, rst;
output hvalid, hsync;

reg[10:0] hcnt;
reg hvalid;
reg hsync;

always @(posedge ck) begin
  if( !rst ) begin
      hcnt <= `HDOT0;
      hvalid <= 0;
      hsync <= 0;
  end else begin
      if( hcnt == 0 ) begin
            hvalid <= 0;
            hcnt <= `HDOT0;
      end else begin
            if( hcnt == (`HDOT-`HFP) ) begin
                  hsync <= 1;
            end else if( hcnt == (`HDOT-`HFP-`HSYNC) ) begin
                  hsync <= 0;
            end else if( hcnt == (`HDOT-`HFP-`HSYNC-`HBP) ) begin
                  hvalid <= 1;
            end
            hcnt <= hcnt - 1;
      end
  end

end

endmodule
```

```
`define VDOT    628
`define VDOT0   627
`define VFP     1
`define VSYNC   4
`define VBP     23

module vsync(ck,vvalid,vsync,hvalid,rst);
input ck, rst,hvalid;
output vvalid, vsync;

reg[10:0] vcnt;
reg vvalid;
reg vsync;

always @(posedge hvalid or !rst) begin
  if( !rst ) begin
      vcnt <= `VDOT0;
      vvalid <= 0;
      vsync <= 0;
  end else begin
      if( vcnt == 0 ) begin
            vvalid <= 0;
            vcnt <= `VDOT0;
      end else begin
            if( vcnt == (`VDOT-`VFP) ) begin
                  vsync <= 1;
            end else if( vcnt == (`VDOT-`VFP-`VSYNC) ) begin
                  vsync <= 0;
            end else if( vcnt == (`VDOT-`VFP-`VSYNC-`VBP) ) begin
                  vvalid <= 1;
            end
            vcnt <= vcnt - 1;
      end
  end

end

endmodule
```

でもこれでは合成できない

同期信号の生成・・改

```
module sync(ck,rst,hsync,vsync,hvalid,vvalid,hcnt,vcnt);
input ck, rst;
output hsync, vsync, hvalid, vvalid;
output [10:0] hcnt, vcnt;
reg [10:0] hcnt, vcnt;
reg hvalid, vvalid, hsync, vsync;
always @(posedge ck) begin
	if( !rst ) begin
		hcnt <= `HDOT0;	hvalid <= 0;	hsync <= 0;
		vcnt <= `VDOT0;	vvalid <= 0;	vsync <= 0;
	end else begin
		if( hcnt == 0 ) begin
			hvalid <= 0;	hcnt <= `HDOT0;
		end else begin
			if( hcnt == (`HDOT-`HFP)) begin
				hsync <= 1;
				if( vcnt == 0 ) begin
					vvalid <= 0;	vcnt <= `VDOT0;
				end else begin
					if( vcnt == (`VDOT-`VFP) ) vsync <= 1;
					else if( vcnt == (`VDOT-`VFP-`VSYNC) ) vsync <= 0;
					else if( vcnt == (`VDOT-`VFP-`VSYNC-`VBP) ) vvalid <= 1;
					vcnt <= vcnt -1;
				end
			end else if( hcnt == (`HDOT-`HFP-`HSYNC)) hsync <= 0;
			else if( hcnt == (`HDOT-`HFP-`HSYNC-`HBP)) hvalid <= vvalid;
			hcnt <= hcnt -1;
		end
	end
end
endmodule
```

```
`define HDOT 1056
`define HDOT0 1055
`define HFP 40
`define HSYNC 128
`define HBP 88

`define VDOT 628
`define VDOT0 627
`define VFP 1
`define VSYNC 4
`define VBP 23
```

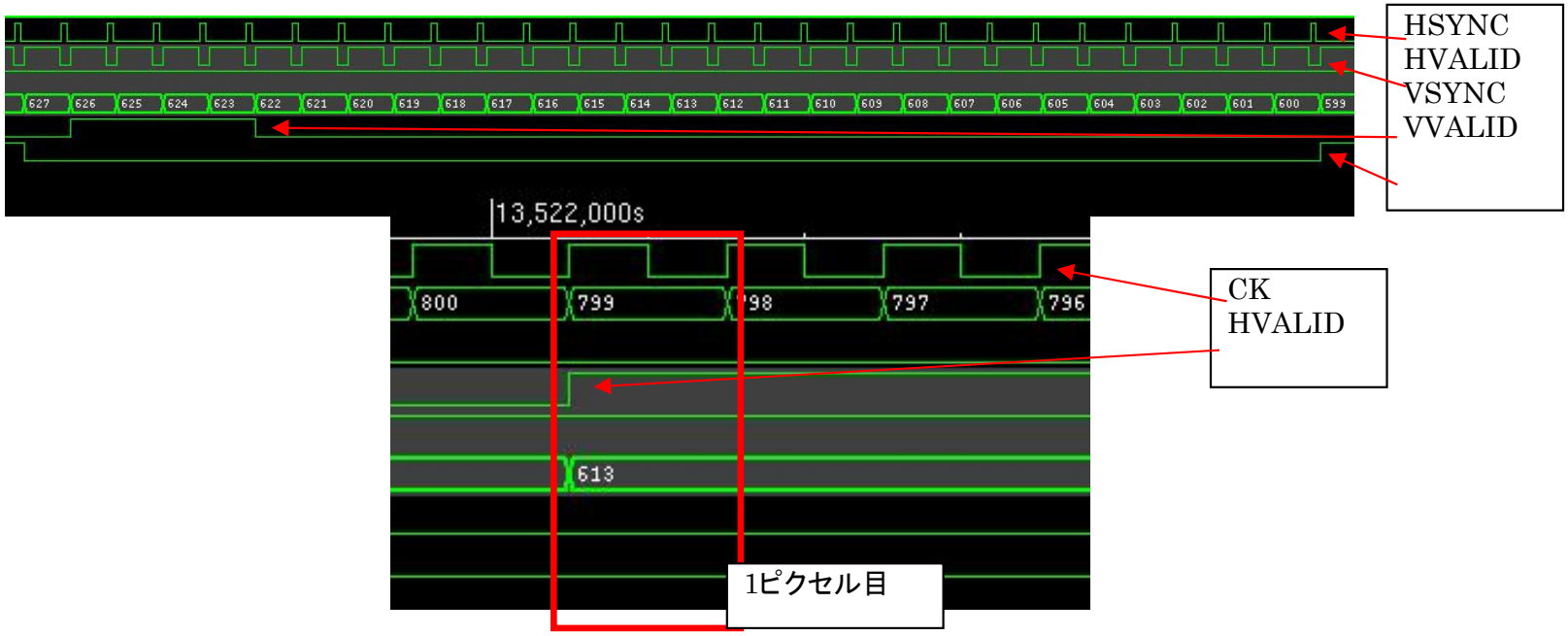
単一クロックでの同期回路として実装

画像信号の生成

```
module VGA(ck,rst,hsync,vsync,hvalid,vvalid,r,g,b,key);
input ck, rst;
input [9:0] key;
output hsync, vsync, hvalid, vvalid;
output r, g, b;
reg r, g, b;
wire [10:0] hcnt, vcnt;
wire hsync, vsync, hvalid, vvalid;
sync sync(.ck(ck), .rst(rst), .hsync(hsync), .vsync(vsync), .hvalid(hvalid), .vvalid(vvalid), .hcnt(hcnt), .vcnt(vcnt) );
always @(posedge ck) begin
        if( !rst ) begin
                r <= 0;         g <= 0;         b <= 0;
        end else begin
                if( hvalid == 1 ) begin
                        case (key)
                                'b 000000: begin
                                                r<= 1;   g<= 0;    b<= 0;
                                        end
                                'b 000010: begin
                                                r<= 0;    g<= 1;    b<= 0;
                                        end
.....................................................................
                                endcase
                end else begin
                        r <= 0;  g <= 0;  b <= 0;
                end
        end
end
```

ここを変えればいろいろなパタンを
出力できる筈

シミュレーション

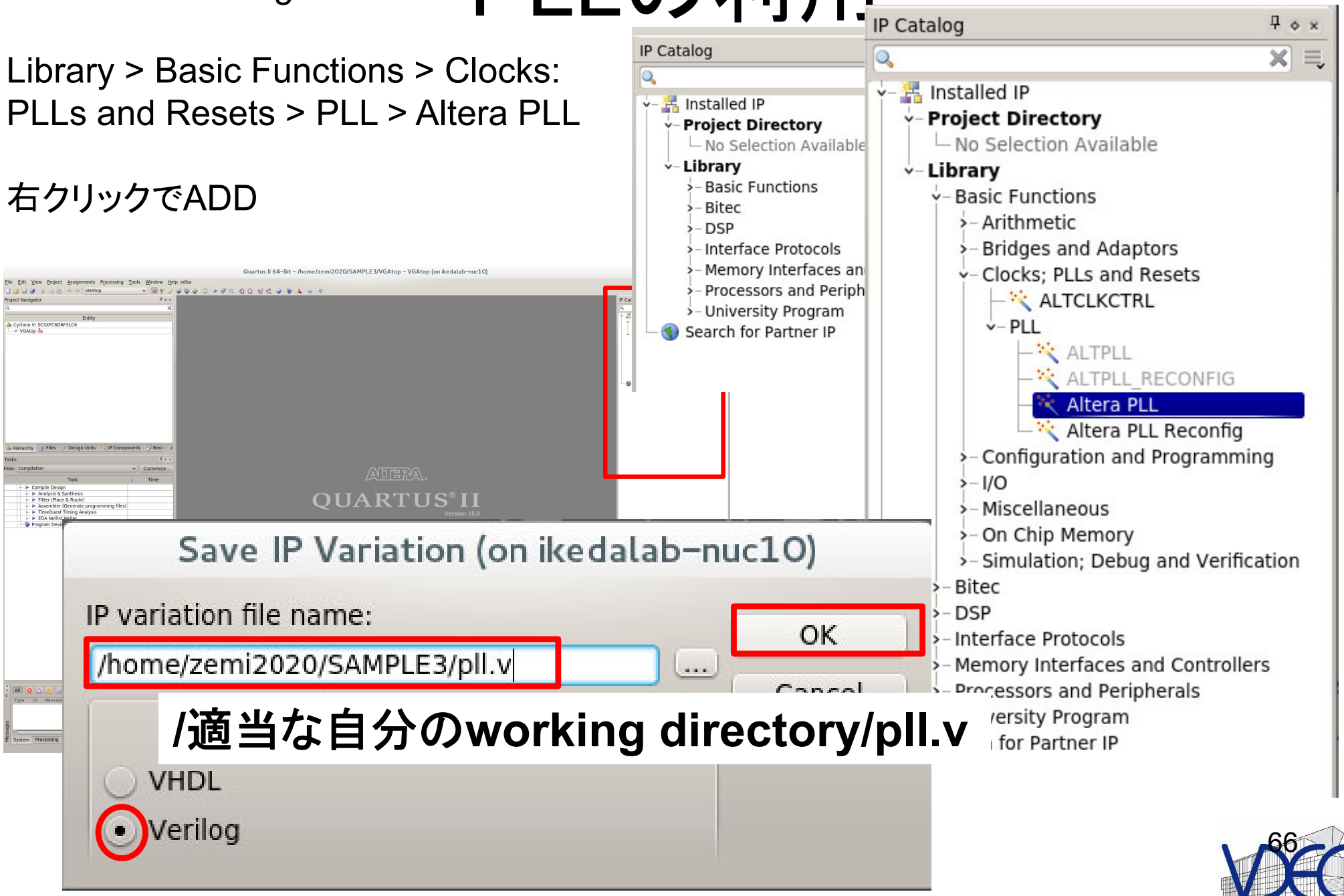

HSYNC
HVALID
VSYNC
VVALID
13,522,000s
800
799
798
797
796
CK
HVALID
613
1ピクセル目

プロジェクト作成後
Tools > IP Catalog

PLLの利用

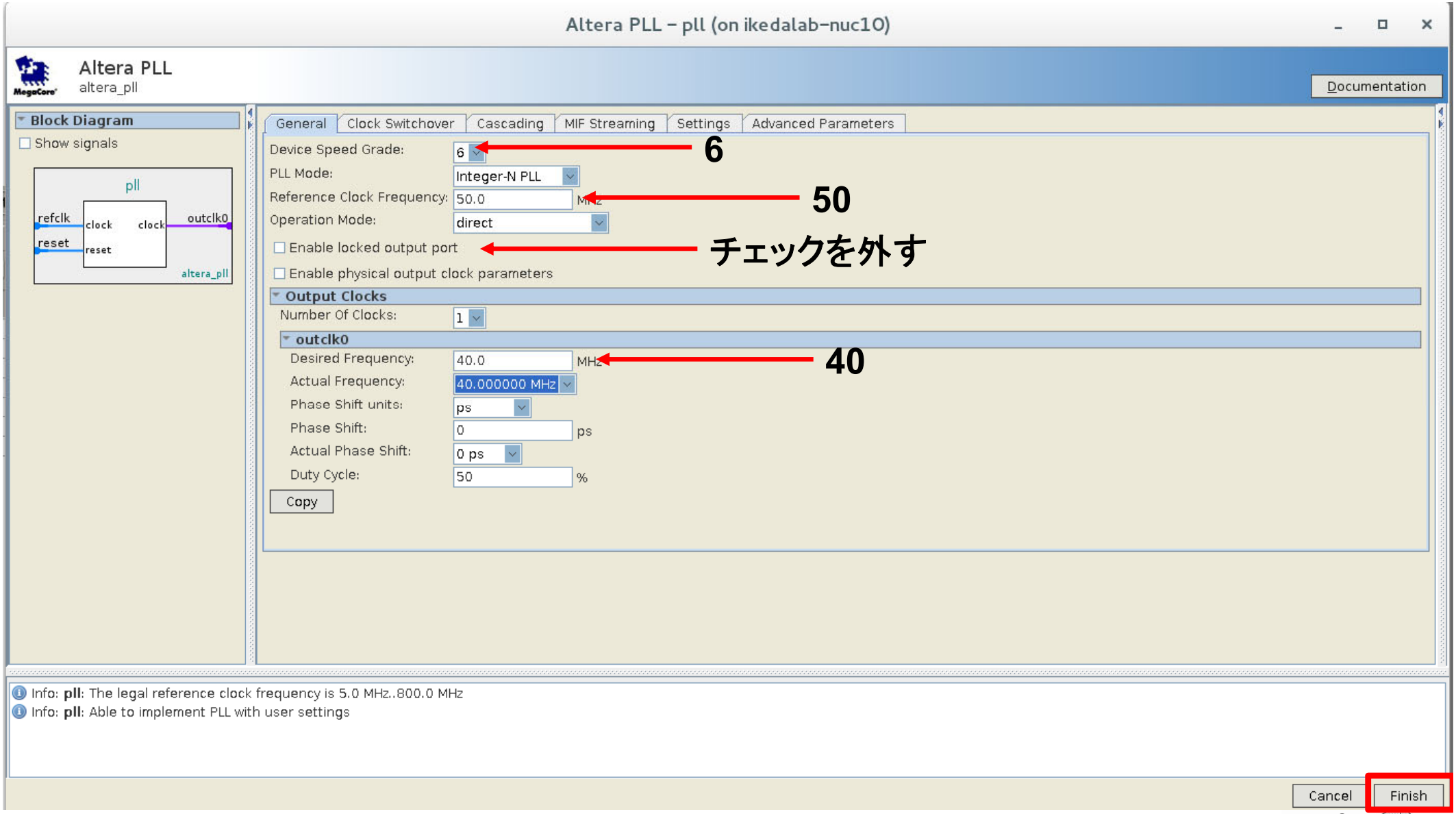
Library > Basic Functions > Clocks:
PLLs and Resets > PLL > Altera PLL

右クリックでADD

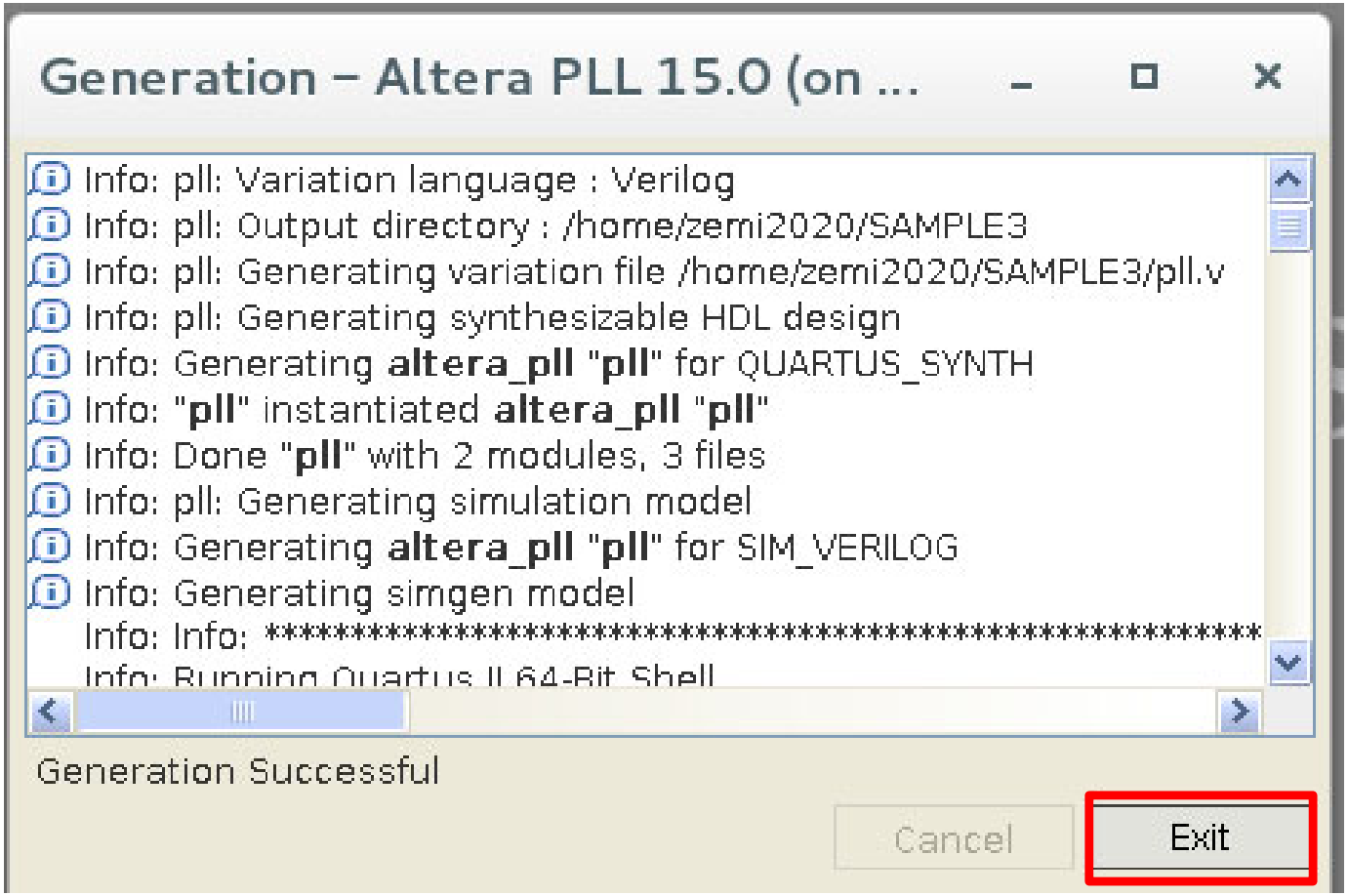


/適当な自分のworking directory/pll.v

PLLの設定


Altera PLL – pll (on ikedalab-nuc10)
Altera PLL
altera_pll
Documentation
Block Diagram
Show signals
pll
refclk
reset
clock
reset
clock
outclk0
altera_pll
General
Clock Switchover
Cascading
MIF Streaming
Settings
Advanced Parameters
Device Speed Grade: 6
6
PLL Mode: Integer-N PLL
Reference Clock Frequency: 50.0 MHz
50
Operation Mode: direct
Enable locked output port
チェックを外す
Enable physical output clock parameters
Output Clocks
Number Of Clocks: 1
outclk0
Desired Frequency: 40.0 MHz
40
Actual Frequency: 40.000000 MHz
Phase Shift units: ps
Phase Shift: 0 ps
Actual Phase Shift: 0 ps
Duty Cycle: 50 %
Copy
Info: pll: The legal reference clock frequency is 5.0 MHz..800.0 MHz
Info: pll: Able to implement PLL with user settings
Cancel
Finish

PLL設定完了

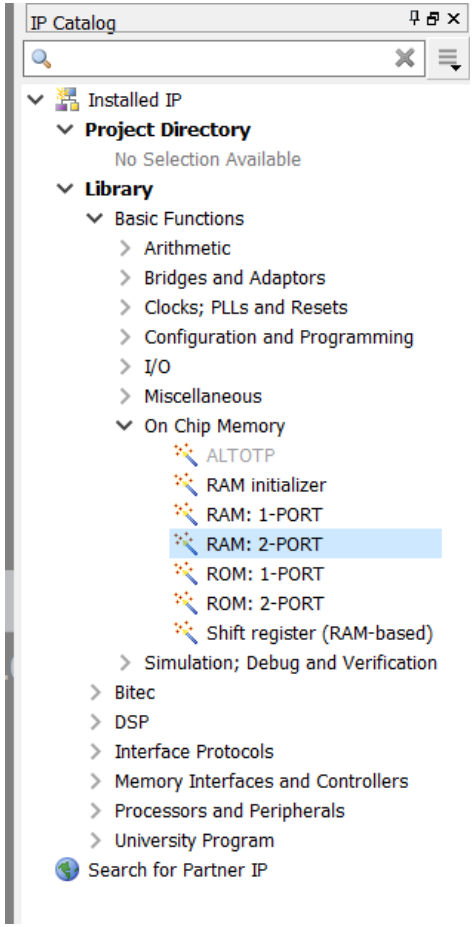

Generation – Altera PLL 15.0 (on ...
Info: pll: Variation language : Verilog
Info: pll: Output directory : /home/zemi2020/SAMPLE3
Info: pll: Generating variation file /home/zemi2020/SAMPLE3/pll.v
Info: pll: Generating synthesizable HDL design
Info: Generating altera_pll "pll" for QUARTUS_SYNTH
Info: "pll" instantiated altera_pll "pll"
Info: Done "pll" with 2 modules, 3 files
Info: pll: Generating simulation model
Info: Generating altera_pll "pll" for SIM_VERILOG
Info: Generating simgen model
Info: Info: **********
Info: Running Quartus II 64-Bit Shell
Generation Successful
Cancel
Exit

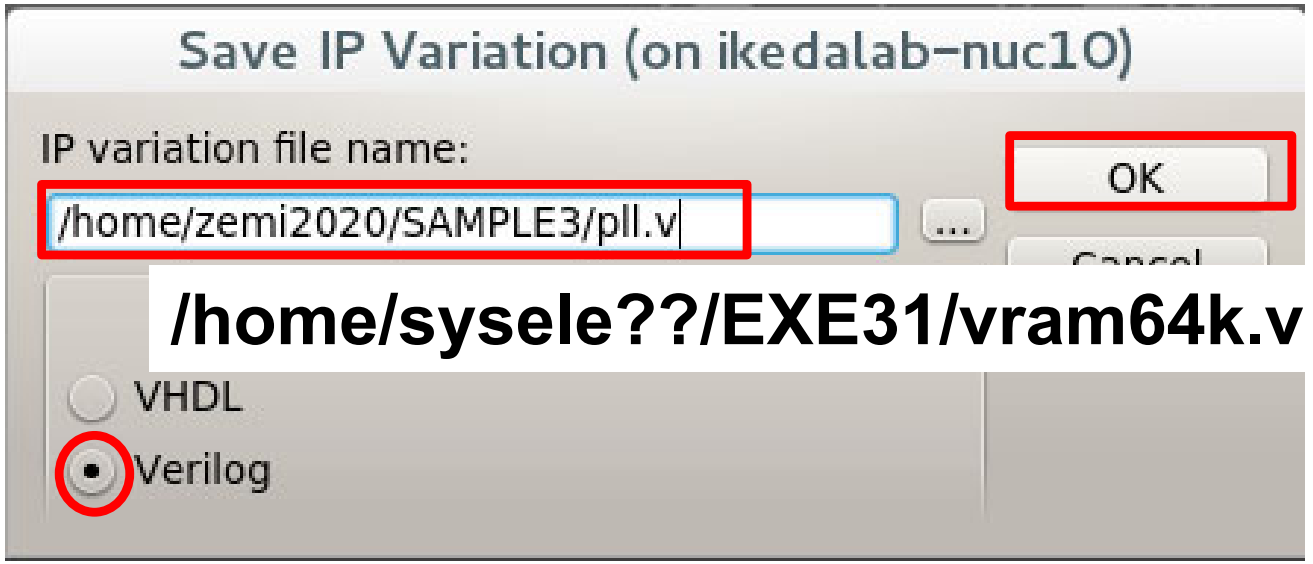
プロジェクト作成後
Tools > IP Catalog

メモリの利用

Library > Basic Functions > On Chip
Memory > RAM: 2-PORT

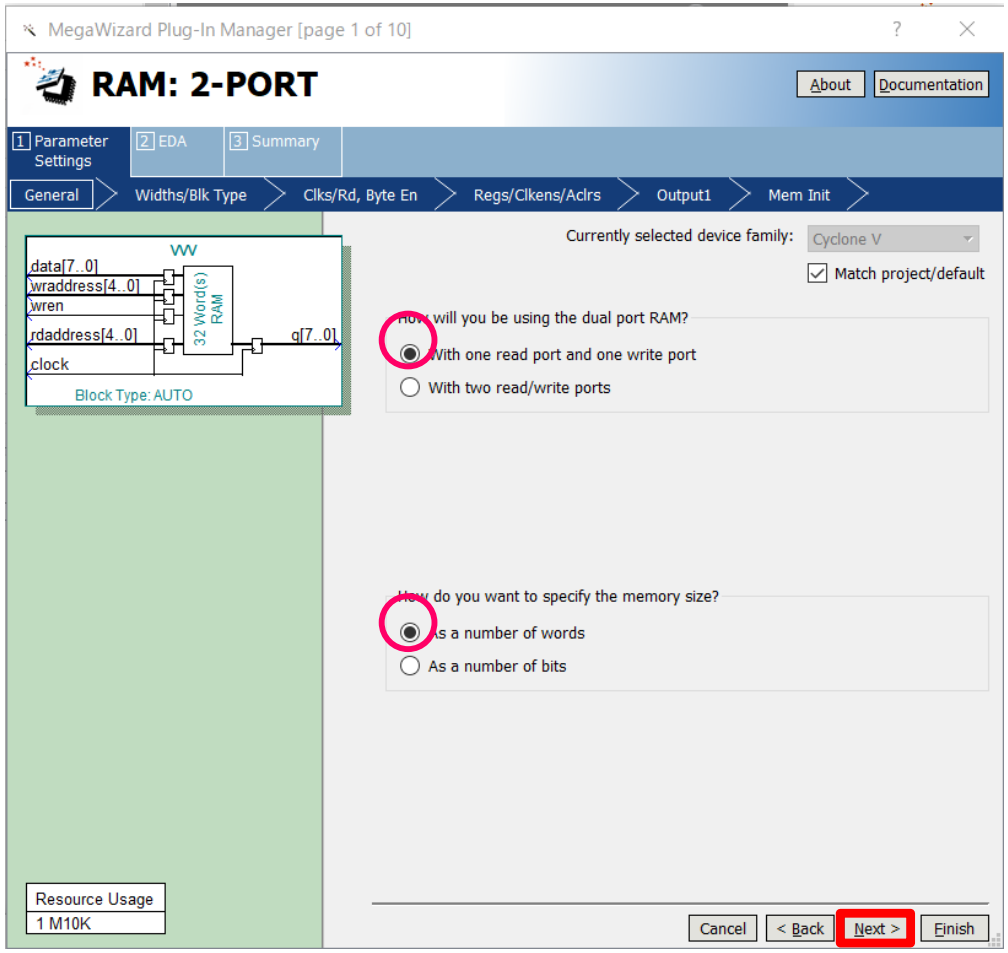
右クリックでADD

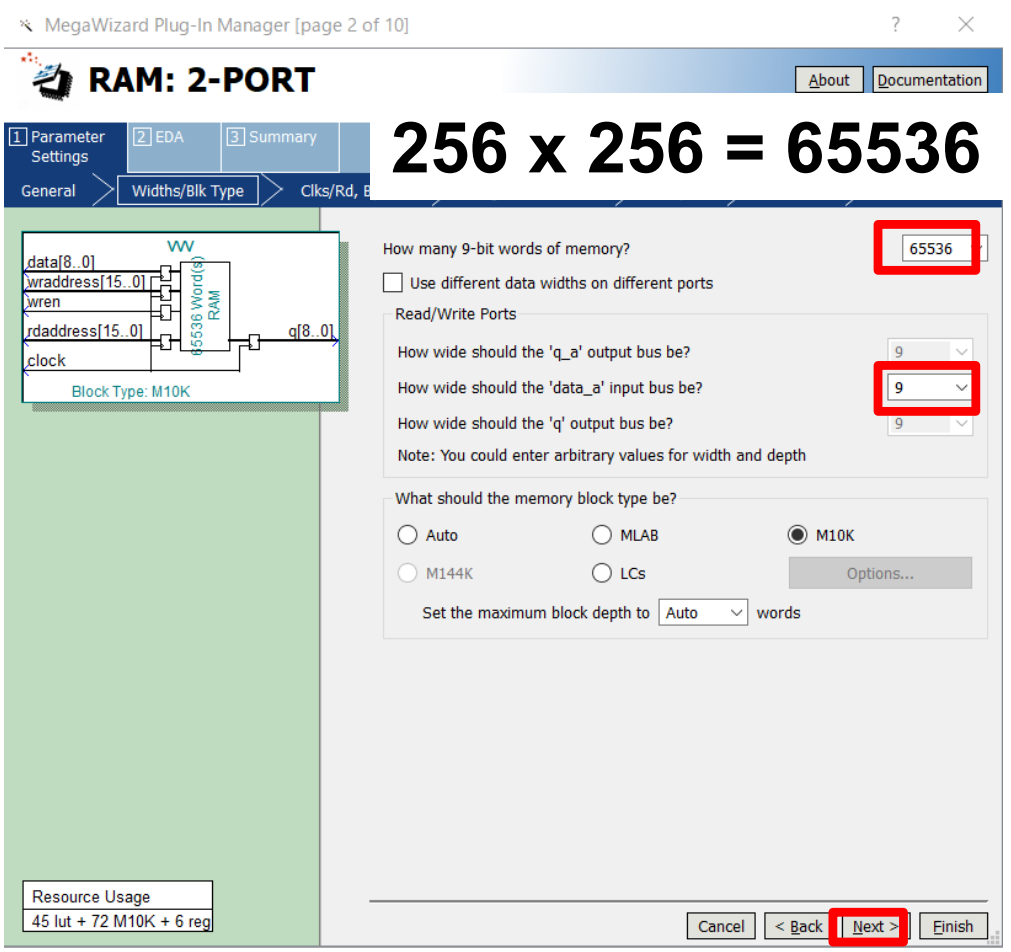




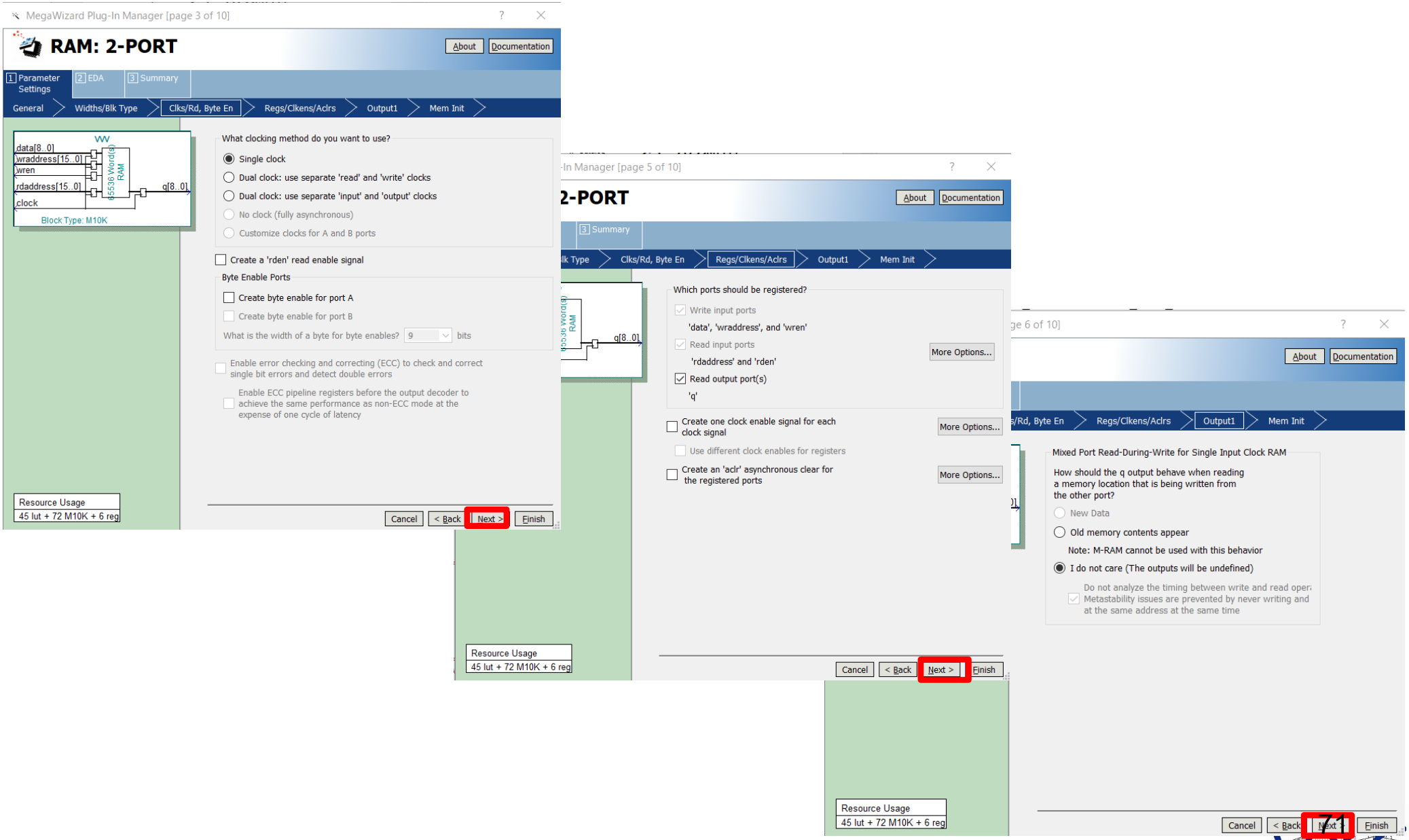
メモリのコンフィグ

今回は、256 x 256 x 9 (R, G, Bそれぞれ3bit)のメモリを生成

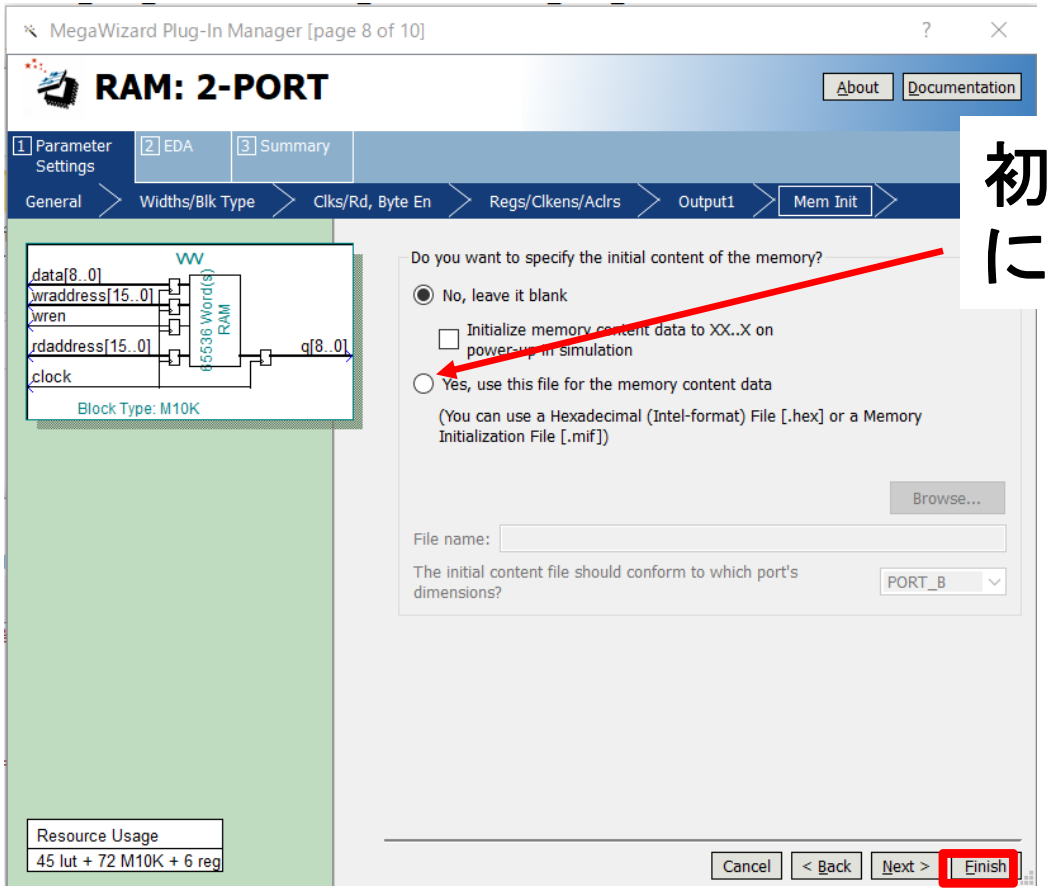




メモリのコンフィグ・・続


MegaWizard Plug-In Manager [page 3 of 10]
RAM: 2-PORT
About
Documentation
1 Parameter Settings
2 EDA
3 Summary
General
Widths/Blk Type
Clks/Rd, Byte En
Regs/Clkens/Aclrs
Output1
Mem Init
data[8..0]
wraddress[15..0]
wren
rdaddress[15..0]
clock
q[8..0]
Block Type: M10K
What clocking method do you want to use?
Single clock
Dual clock: use separate 'read' and 'write' clocks
Dual clock: use separate 'input' and 'output' clocks
No clock (fully asynchronous)
Customize clocks for A and B ports
Create a 'rden' read enable signal
Byte Enable Ports
Create byte enable for port A
Create byte enable for port B
What is the width of a byte for byte enables? 9 bits
Enable error checking and correcting (ECC) to check and correct single bit errors and detect double errors
Enable ECC pipeline registers before the output decoder to achieve the same performance as non-ECC mode at the expense of one cycle of latency
Resource Usage
45 lut + 72 M10K + 6 reg
Cancel
< Back
Next >
Finish
Which ports should be registered?
Write input ports
'data', 'wraddress', and 'wren'
Read input ports
'rdaddress' and 'rden'
Read output port(s)
'q'
More Options...
Create one clock enable signal for each clock signal
Use different clock enables for registers
Create an 'aclr' asynchronous clear for the registered ports
Mixed Port Read-During-Write for Single Input Clock RAM
How should the q output behave when reading a memory location that is being written from the other port?
New Data
Old memory contents appear
Note: M-RAM cannot be used with this behavior
I do not care (The outputs will be undefined)
Do not analyze the timing between write and read oper
Metastability issues are prevented by never writing and at the same address at the same time

メモリのコンフィグ・・続2



初期値を入れたい場合にはここをチェックする

後はFinishを何回か押して終了

池田・飯塚グループトレーニング
～テープアウト編～

池田　誠

検討課題

- ディジタルFPGA編で作成したVerilogを用いて、0.18um向けに論理合成、配置配線を行い、テープアウトを行う
  - 画像はRGBそれぞれ1bitとして表現する
  - FPGAで用いたBRAMはサイズの近いSRAMに置き換える
  - PLLは取り除き、40MHzのクロック入力を仮定した設計とする（クロック制約は25nsを満たすようにする）
  - ピン数制約を満たすように入出力を調整する
  - LED出力だけでなく、内部の結果のデータを直接読み出せるような出力信号を必ず作っておく

SRAMを含んだ配置配線

- フロアプラン後SRAMを手配置する必要がある
  - SRAMの位置・向きによって配線性が大きく変化するのでそのあたりを上手に考える
  - 配線ストラップ等がSRAMを突き抜けないように考えてみる