Vivado/インタフェース生成ツールチュートリアル

高木研究室 谷

準備

- Vivado 2015.4のインストール
  - SDKのインストールも必要
  - Vivado 2015.4，Vivado HLS 2015.4, Xillinx SDK の3種類がインストールされているはずです
- スクリプトの動作環境
  - LinuxなりCygwinなりでpythonとpython-clangが動く環境を作っておいてください

準備

- スクリプトの動作環境：実行を確認してるもの
  - Linux Mint 17.3/18 (Debian系なら大丈夫だと思います)
  - Cygwin
- Linuxでの環境構築
  - $ apt-get install python-pip
  - $ apt-get install clang
  - $ apt-get install python-clang-3.8
  - $ pip install jsonschema
  - $ pip install clang
  - 別のバージョンのpython-clangを使用する場合は extractparameter.py と divider/analyzer.py の Config.set_library_fileの設定パスのバージョンを変更してください

準備

- Cygwinでの環境構築
  - Cygwin64インストーラから，python 2.7.10/clang 3.8.1がインストールされていることを確認
  - $ easy_install-2.7 –version
    setuptools 15.2
  - $ easy_install-2.7 pip
    Searching for pip …
  - $ pip --version
    pip 8.1.2 from /usr/lib/python2.7/site-packages/pip-8.1.2-py2.7.egg (python 2.7)
  - $ pip install jsonschema
  - $ pip install clang

準備

入力ソースC

- C言語サンプル(matrixmul.c/matrixmul_cs.c)の仕様
  - HW化する関数 matrixmul
  - SWで同じ計算をする関数 matrixmul_soft
  - メイン関数 main
  - 次頁のJSONファイルでHW化とそのインタフェースを指定
- (実行時間測定のためのライブラリ)
  - timer.c およびtimer.h

入力ソースJSONの仕様

- software_tasks でSWの関数指定
- Hardware_tasks でHW化する関数の指定
  - “name”　で関数名
  - “mode” は 全体の制御で s_axilite固定
  - “argments”内でそれぞれの関数の通信インタフェースを記述
    - “name”　は引数名
    - “mode” は通信プロトコル “s_axilite” “m_axi” “axis” どれか
    - “bundle” で同じプロトコルのポートをまとめられる(“axis”以外)
    - “m_axi”のときoffset = slave を追加
    - “direction” で入出力どちらかを指定

```
{
  "software_tasks": [
    {
      "name": "main"
    },
    {
      "name": "matrixmul_soft"
    }
  ],
  "hardware_tasks": [
    {
      "name": "matrixmul",
      "mode": "s_axilite",
      "arguments": [
        {
          "name": "a",
          "mode": "m_axi",
          "bundle": "port_a",
          "offset": "slave",
          "direction": "in"
        },
        {
          "name": "b",
          "mode": "m_axi",
          "bundle": "port_a",
          "offset": "slave",
          "direction": "in"
        },
        {
          "name": "c",
          "mode": "s_axilite",
          "bundle": "port_b",
          "direction": "out"
        }
      ]
    }
  ]
}
```

IF生成ツールの仕様

- シェルで
  ./ifmake.sh matrixmul /usr/lib/llvm-3.8/lib/libclang-3.8.so (Linux)
  ./ifmake.sh matrixmul /bin/cygclang-3.8.dll (Cygwwin)
  を実行すれば
  SWインタフェースレイヤ：matrixmul_if.c
  HWインタフェース情報付HWタスク：matrixmul_hw_re.c
  SWタスク：matrixmul_sw.c
  が出力される
  - 中身では,
    python divide.py matrixmul.c matrixmul.json　(SW/HW分割)
    python ifmake.py matrixmul.c matrixmul.json　(IFレイヤ生成)
    python renamehwparms.py matrixmul_hw.c matrixmul.json
    (HWタスクのAXIプロトコルを使用する引数の通信処理の付加)

    を順に実行している.

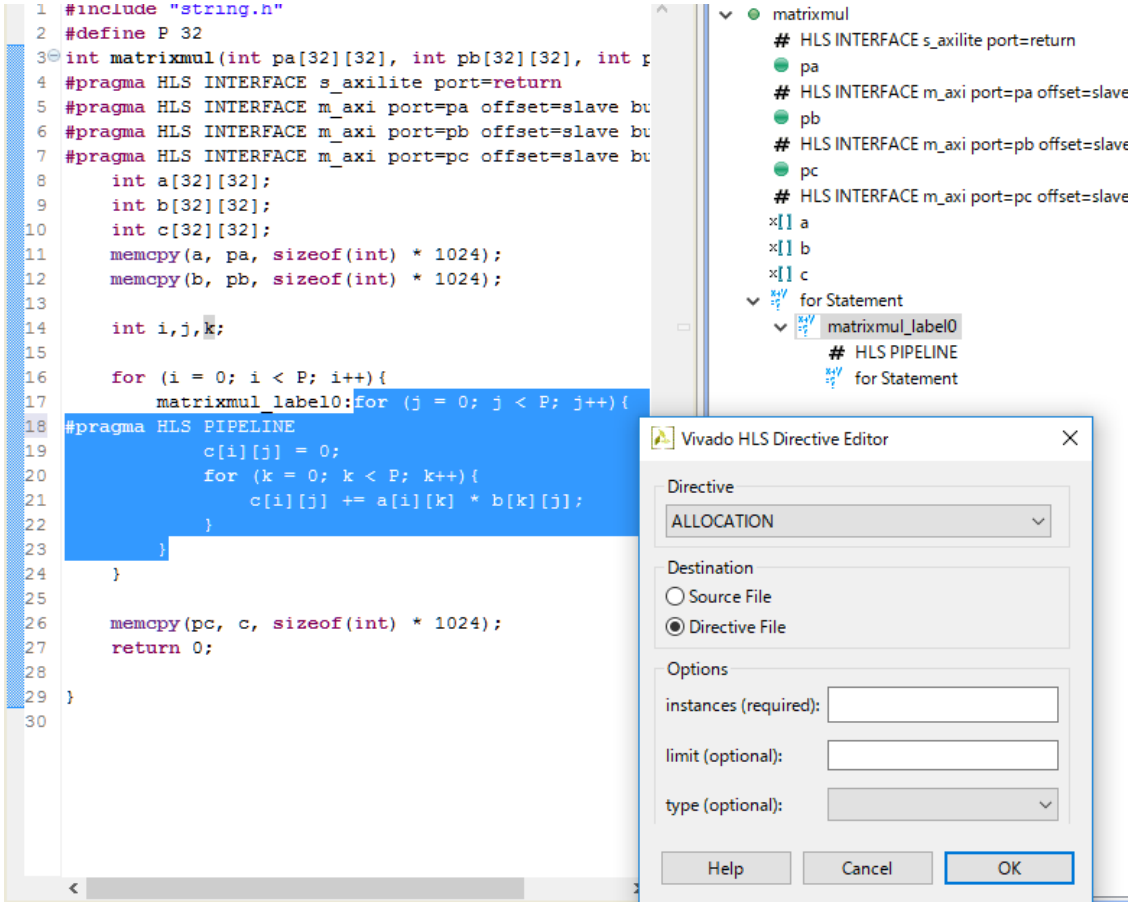
Vivado HLSによる高位合成

- プロジェクト作成・設定
  - Vivado HLSの起動画面でCreate New Projectを選択
  - Project Configuration でプロジェクト名と配置ディレクトリを選択
    (今回は matrixmul_ambmcm_2015.4 とする)
  - Add/Remove files で matrixmul_hw_re.c を選択
    Top Function に matrixmul を設定
  - Test bench は特になし
  - Solution Configuration で
    Please Select part→board→Zedboard で使用ボードを選択
    →finish

Vivado HLSによる高位合成

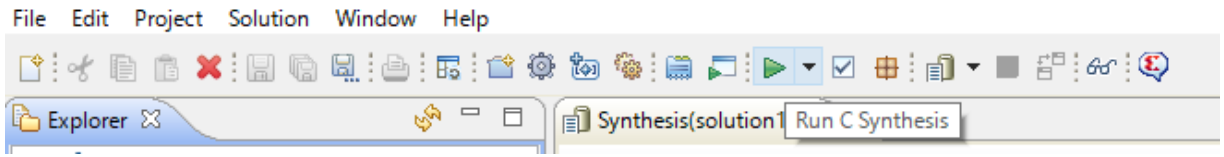
- パイプライン化プラグマの追加
  - ループを
    ダブルクリックし，
    パイプライン化の
    プラグマを追加できる

    (ループにはラベルが
    張られる)

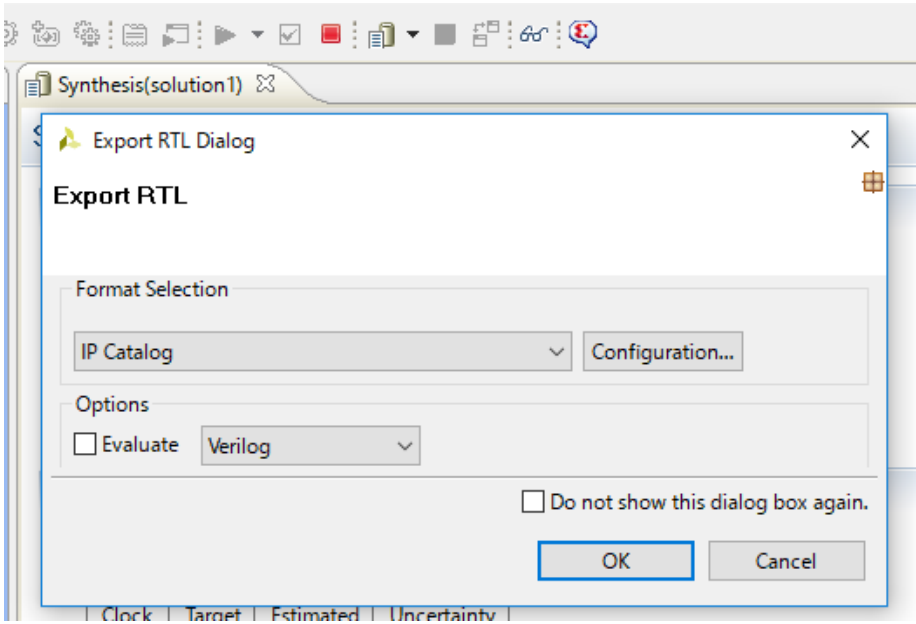


Vivado HLSによる高位合成

- 合成
  Run C Synthesis



- IPとして出力
  - Export RTL
    (Run C Synthesisの2つ隣)
    からIP Catalogを選択し
    Vivadoで使用できるIPとして
    出力



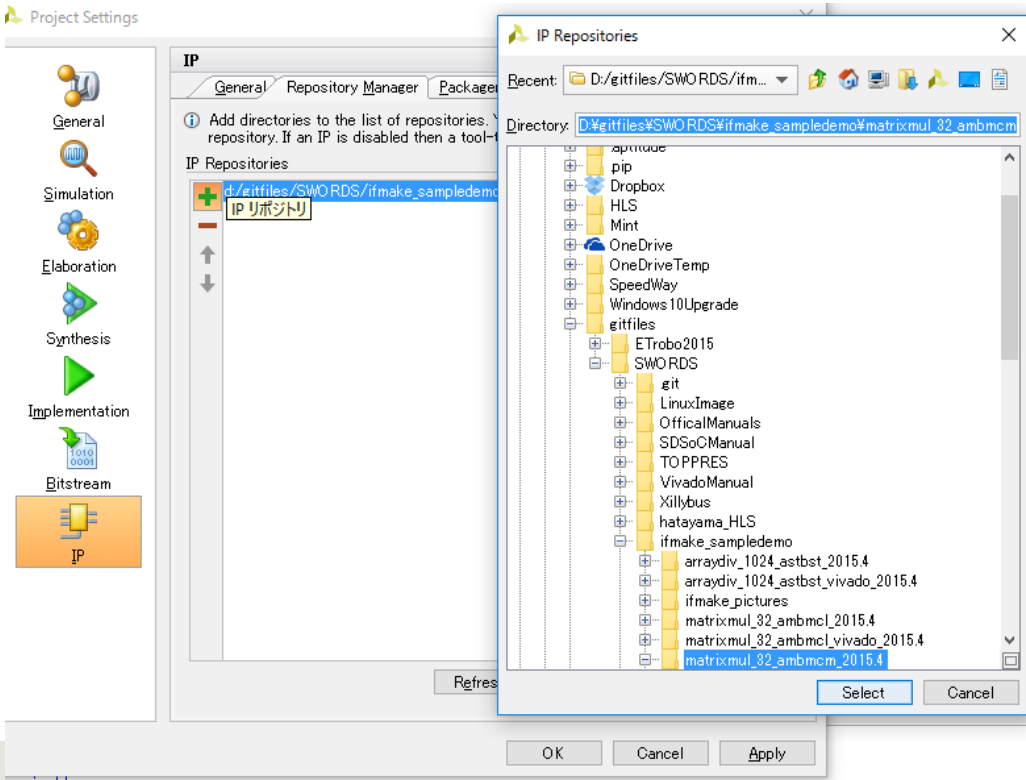
- Vivado HLSにおける作業は終了

Vivadoによるデザイン

- プロジェクト作成
  - Vivado の起動画面でCreate New Projectを選択
  - Project Nameでプロジェクト名
    (今回は matrixmul_ambmcm_vivado_2015.4 とする) を入力,
    ディレクトリを指定
  - Project Type → RTL Project
  - Add Sources/Existing IP/Constraints → 特になし
  - Default Part で Board→ZedBoard を対象に選択
    →finish

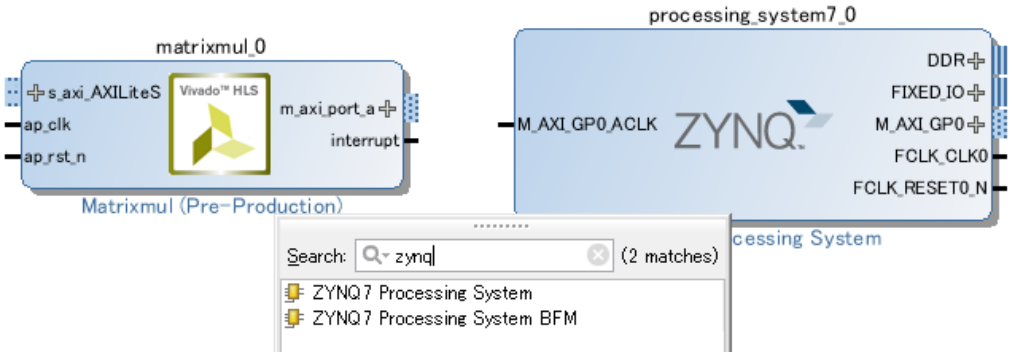
Vivadoによるデザイン

- デザインの作成
  - 左の IP Integrator → Create Block Design から
    デザイン名 (ここでは matrixmul_system) としてデザインを作成
- ライブラリへの高位合成IPの追加
  - 空のデザインを
    右クリックすると
    メニューが現れるので
    IP Settings を選択
  - IP→Repository Manager
    →＋　から
    Vivado HLSで作った
    プロジェクトの
    ディレクトリを追加,
    IPが認識される

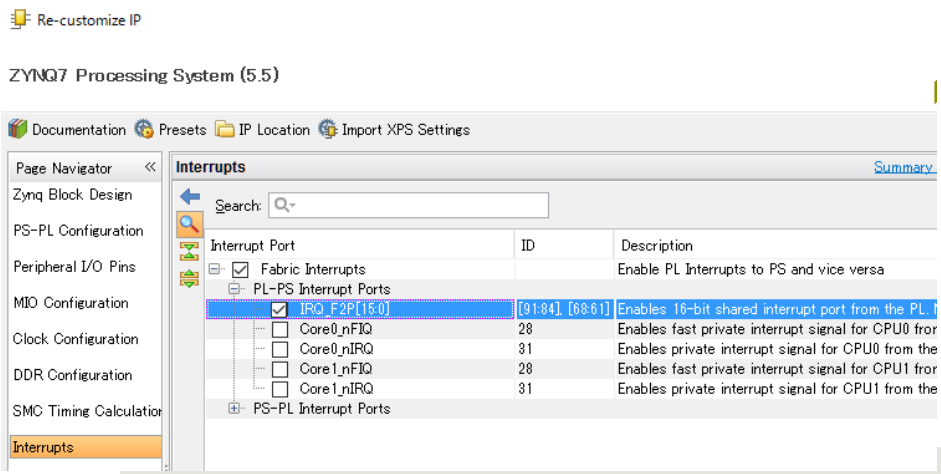


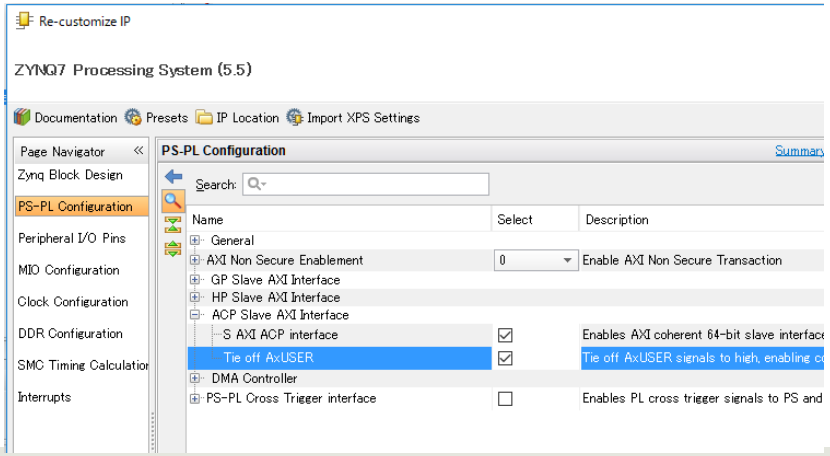
Vivadoによるデザイン

- デザインにIPを追加
  - 右クリック → Add IP… からZynq7 Processing System と Matrixmul を追加



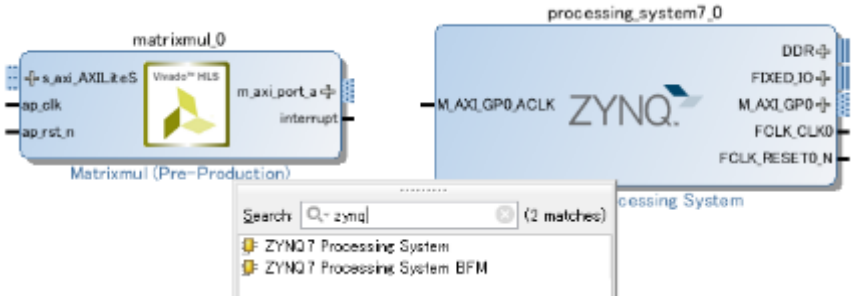
- ARM-FPGA間
  通信ポートの追加
  - Zynqを
    ダブルクリックし，
    割込み信号とACPポートの追加を行う



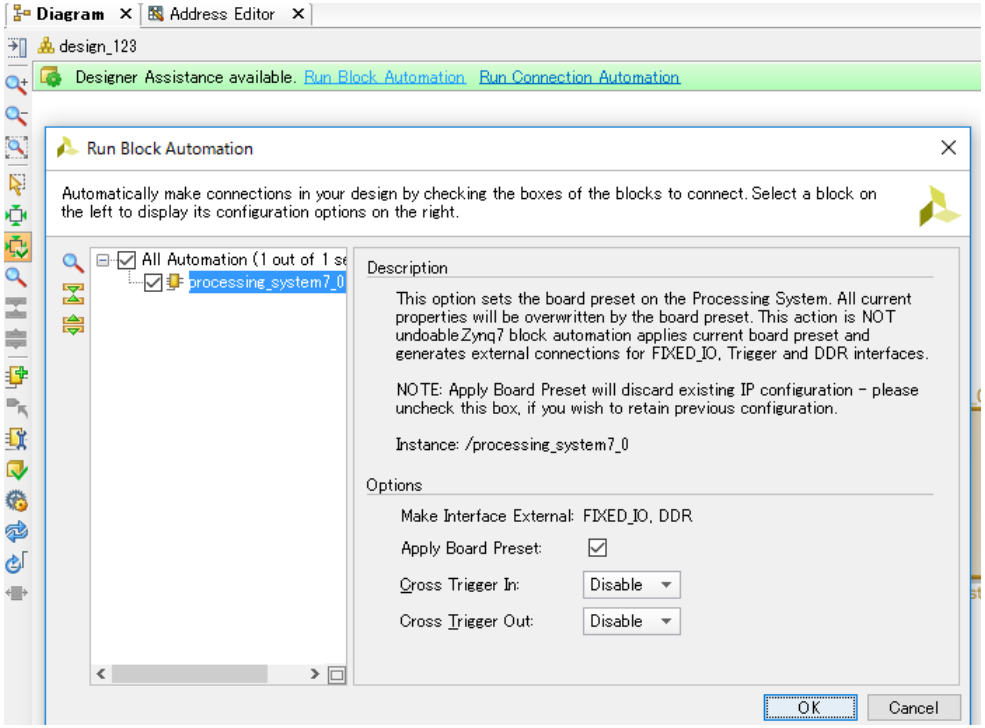


Vivadoによるデザイン

- デザインにIPを追加
  - 右クリック → Add IP… から Zynq7 Processing System と Matrixmul を追加

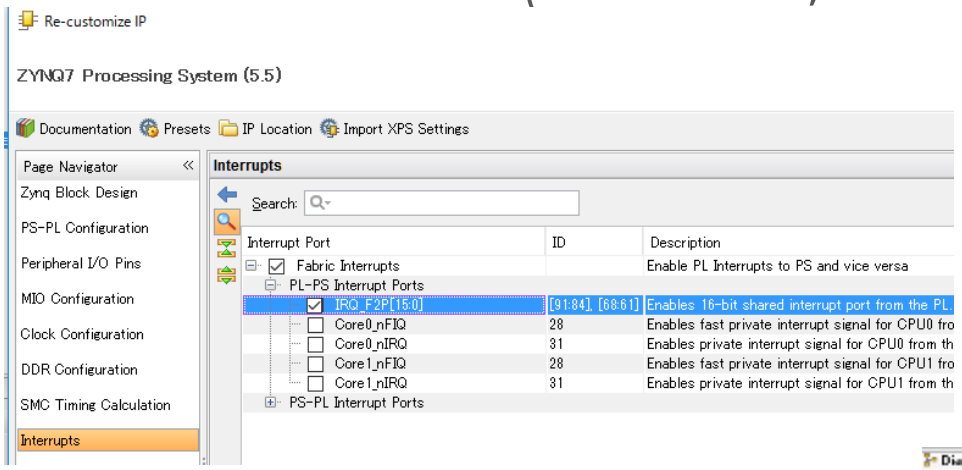


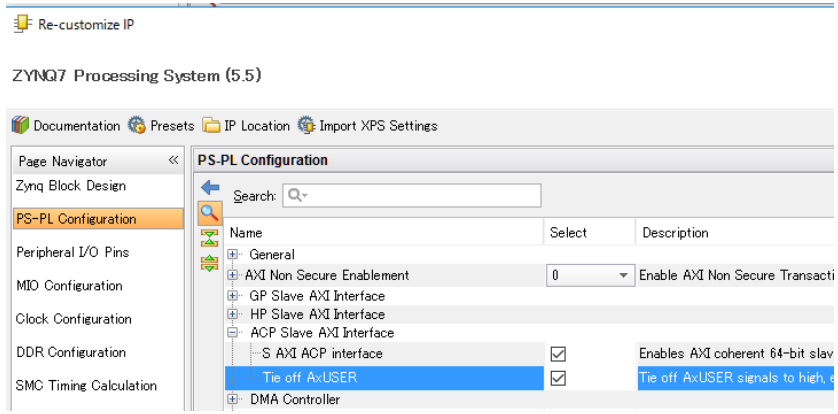
- 自動配線①
  - Run Block Automation を実行



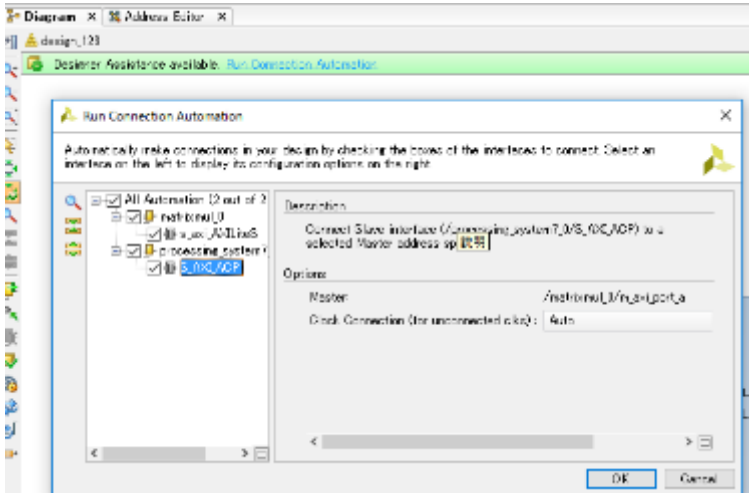
Vivadoによるデザイン

- ZynqのProcessing Systemをダブルクリックし，ARM-FPGA間通信(割込み信号，ACPポート)を用意



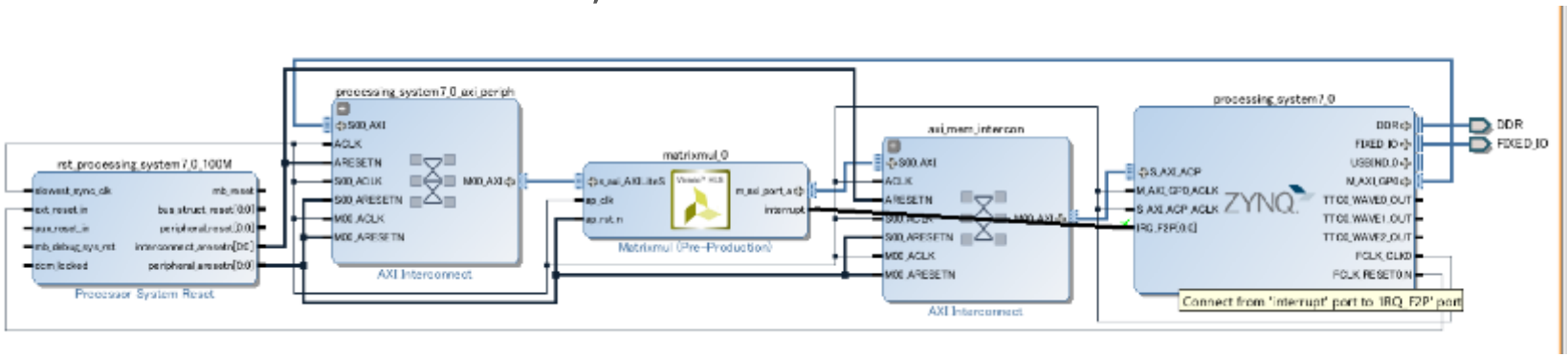


- 自動配線②
  - Run Connection Automation を実行(チェックをつける)

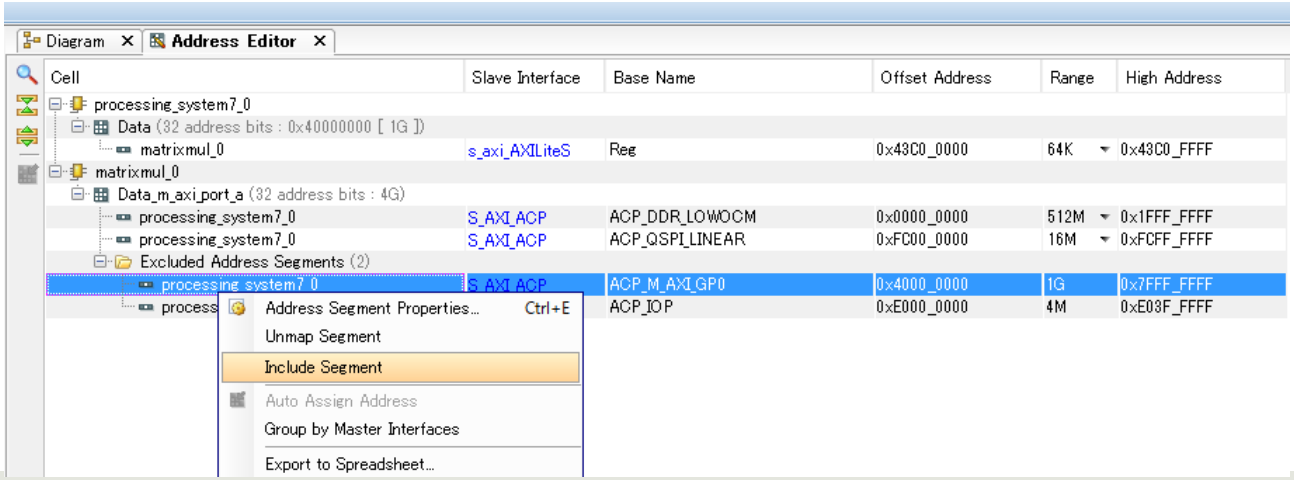


Vivadoによるデザイン

- 割込み信号を手動接続し，ダイアグラムの完成

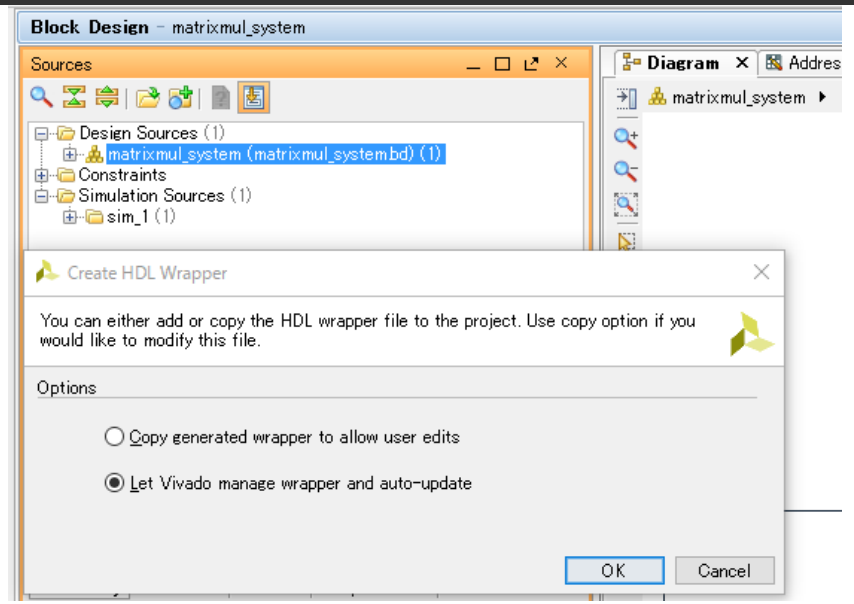


- Address Editor で
  Exclude … となっているなっているアドレスをInclude

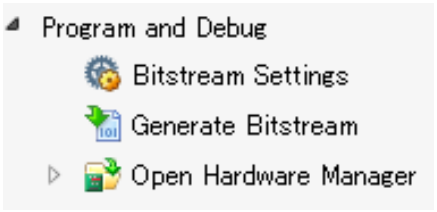


Vivadoによるデザイン

- HDL Wapper の作成
  - Sources にある.bdファイルを選択し，右クリックして Create HDL Wapper... を実行

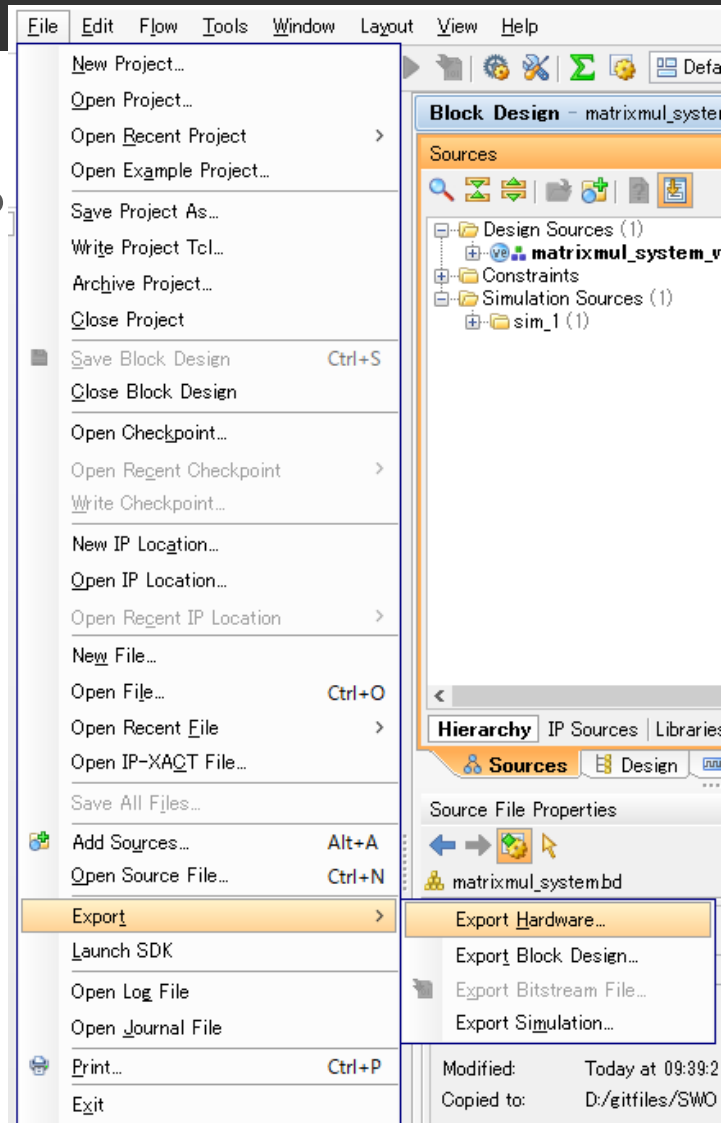


- 論理合成・配置配線・ビットストリーム生成
  - Program and Debug のGenerate Bitstream... を実行すれば途中の論理合成・配置配線も通しで実行される



Vivadoによるデザイン

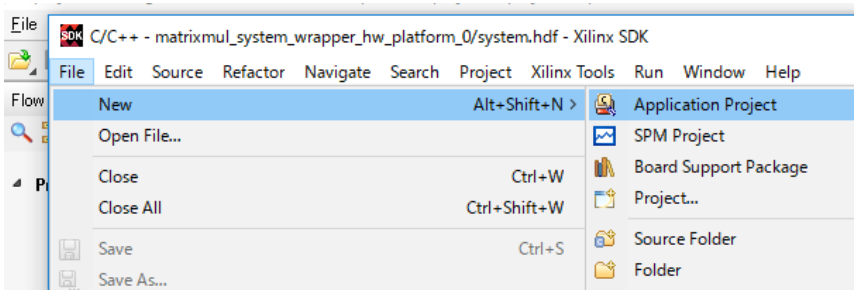
- ビットストリーム出力・SDKの呼び出し
  - File→Export→Export Hardware…から
    ビットストリームファイルを出力
    (Include bitstreamにチェック)
  - Launch SDK からSDKを起動
- これでVivadoでの作業は終了です

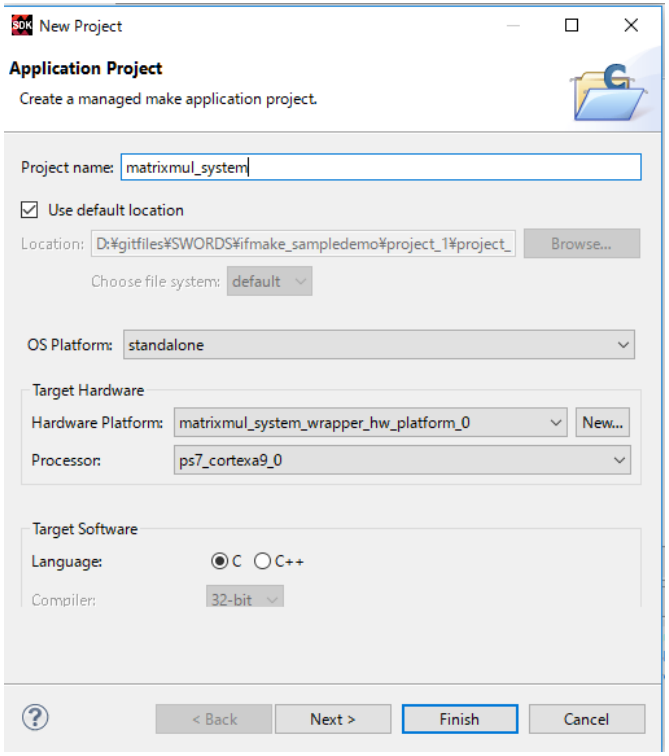


SDKでのSW作成・実行

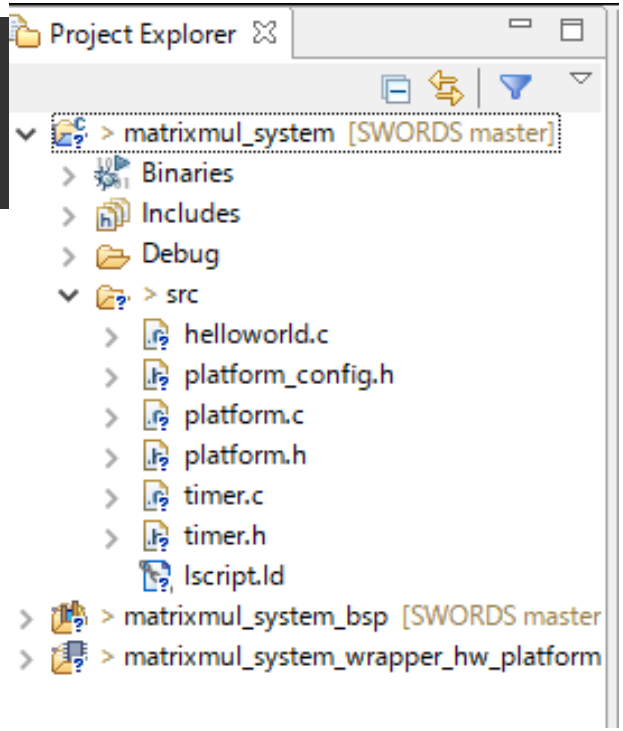
- アプリケーションプロジェクトの作成
  - File→NEW→Application Project から Project nameを matrixmul_system として作成

    スタンドアロン(OSなし)とする





SDKでのSW作成・実行



- 時間計測ライブラリの追加
  - matrixmul system→srcに timer.c/timer.h をドラッグドロップして追加(copyで)
- SWの追加
  - matrixmul_sw.c の中身で helloworld.c を置換
- SWインタフェースレイヤの追加
  - matrixmul_if.c の中身を先ほどのhelloworld.c の下に追加

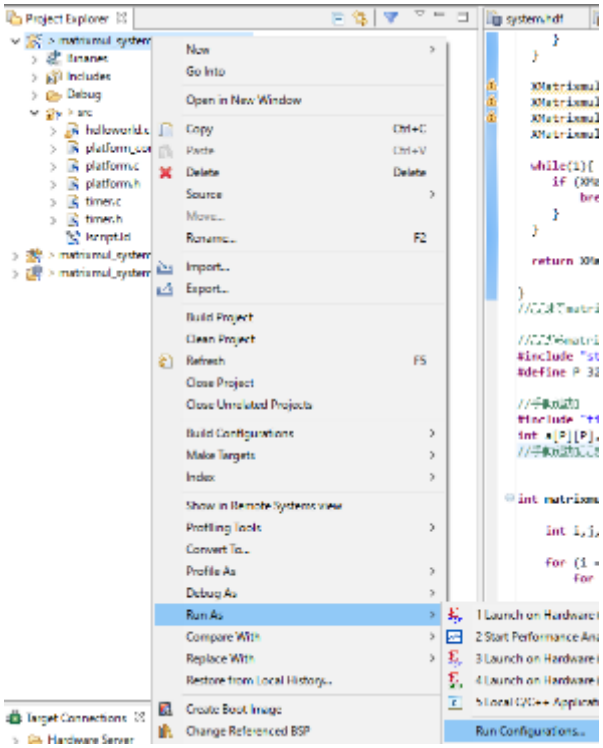
SDKでのSW作成・実行

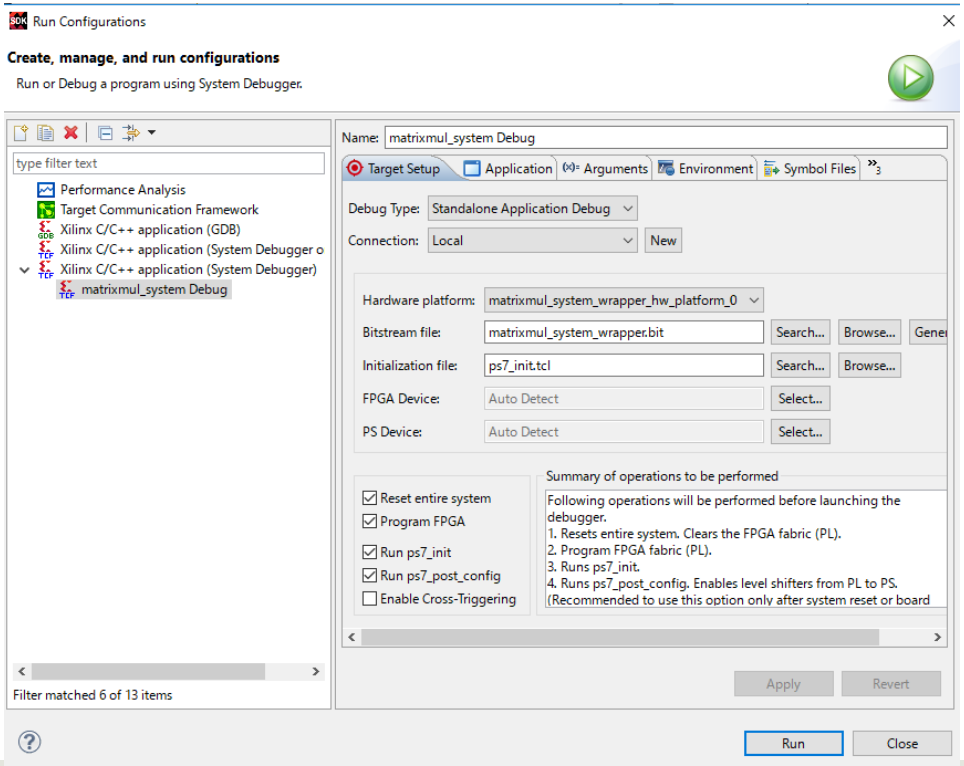
- SWへの追加
  - グローバル変数a~eの宣言と timer.h のインクルードを追加

```
  XMatrixmul_Set_pa(&matrixmulx, a);
  XMatrixmul_Set_pb(&matrixmulx, b);
  XMatrixmul_Set_pc(&matrixmulx, c);
  XMatrixmul_Start(&matrixmulx);

  while(1){
     if (XMatrixmul_IsIdle(&matrixmulx) == 1){
         break;
     }
  }

  return XMatrixmul_Get_return(&matrixmulx);

}
//ここまでmatrixmul_if.c

//ここからmatrixmul_sw.c
#include "string.h"
#define P 32

//手動追加
#include "timer.h"
int a[P][P], b[P][P], c[P][P], d[P][P], e[P][P];
//手動追加ここまで


int matrixmul_soft(int a[32][32], int b[32][32], int c[32][32]){

    int i,j,k;

    for (i = 0; i < P; i++){
        for (j = 0; j < P; j++){
            c[i][j] = 0;
            for (k = 0; k < P; k++){
                c[i][j] += a[i][k] * b[k][j];
            }
        }
    }

    return 0;

}
int main()
{


    int tmp1,tmp2,tmp3,tmp4,tmp5,tmp6;
```

SDKでのSW作成・実行

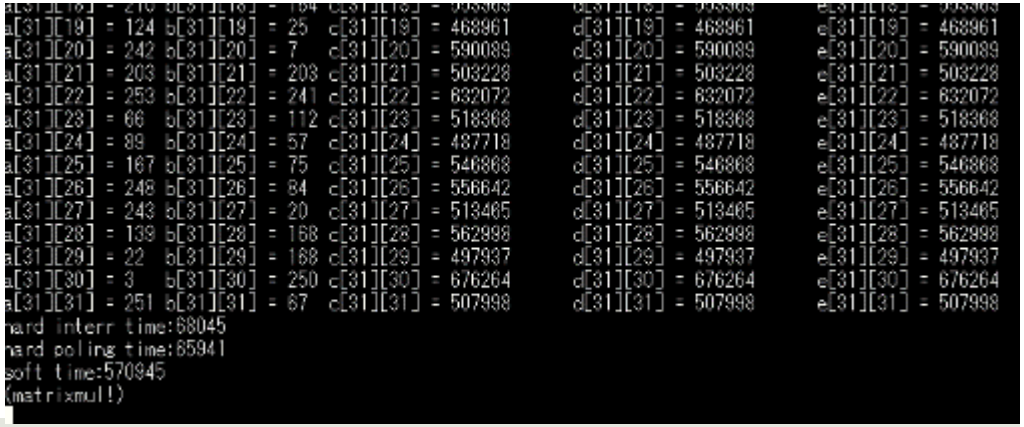
- Run Configurations の作成
  - Xilinx C/C++ application (System Debugger)をダブルクリックするとコンフィギュレーションが作成できる
    - Reset entire system および Program FPGA にチェックしApply





SDKでのSW作成・実行

- PCとZedBoardの接続
  - UART と PROG と書いているZedBoardのmicroUSBポート2つをPCとmicroUSBケーブルで接続
  - Tera Term で COM1以外にポートが出現しているのでそれを選択
- 実行
  - 先ほど生成したRun Configurations で Run を押し実行
  - HW(割込みを使用するもの) HW(割込みを使用しないもの) SW の実行時間と結果がTera Termに出力されます



matrixmul とmatrixmul_cs の違い

- matrimxul_cs は 出力引数cをAXI4-Liteプロトコルで
  通信するようにしています(matrixmul_cs.jsonを参照)(元ソースは同一)
- プラグマやIFレイヤが少し異なってきます
- Vivadoでのデザインも異なってきます