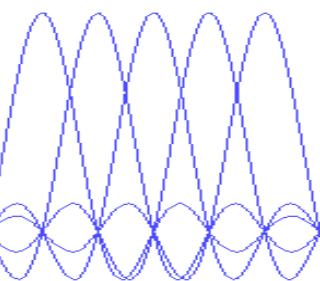
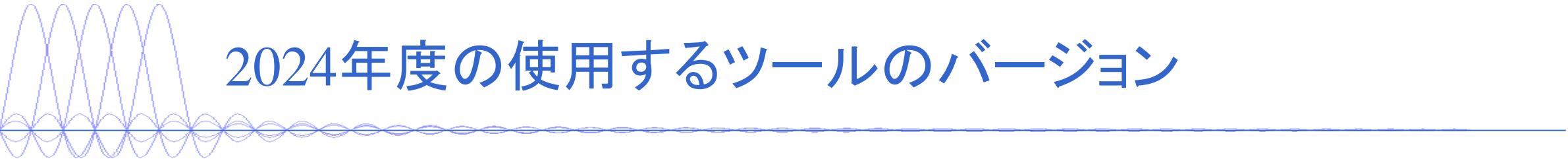


LSI Design Contest in Okinawa/Hokkaido

九州工業大学大学院情報工学研究院
情報・通信工学研究系
黒崎 正行

Mail : kuroasaki@csn.kyutech.ac.jp





2024年度の使用するツールのバージョン

■ 使用するツール(沖縄ポリテク)

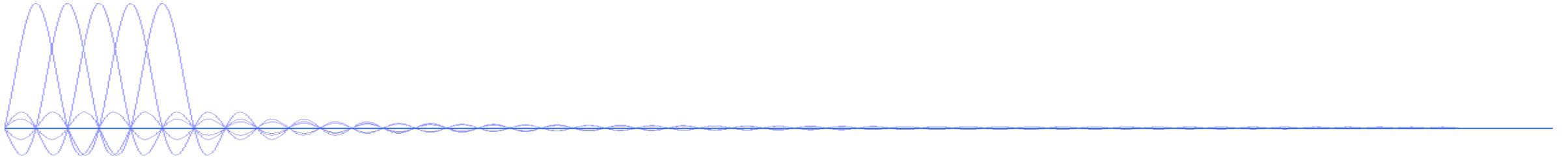
- MATLAB R2020b
- VIVADO 2020.2
- 作業フォルダ: G:\LSI2025\

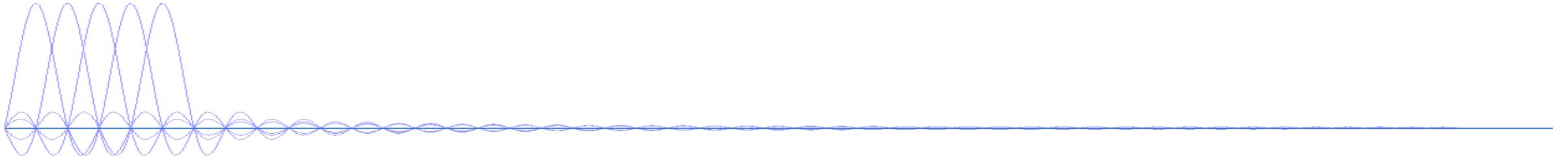
■ 使用するツール(九州ポリテク)

- MATLAB R2022b
- VIVADO 2022.2
- 作業フォルダ: C:\Users\xxx\LSI\

■ 大学使用ツール

- MATLAB R2021b / MATLAB R2022b
- VIVADO 2022.2 / VIVADO 2022.1
- 作業フォルダ: C:\LSI2025\

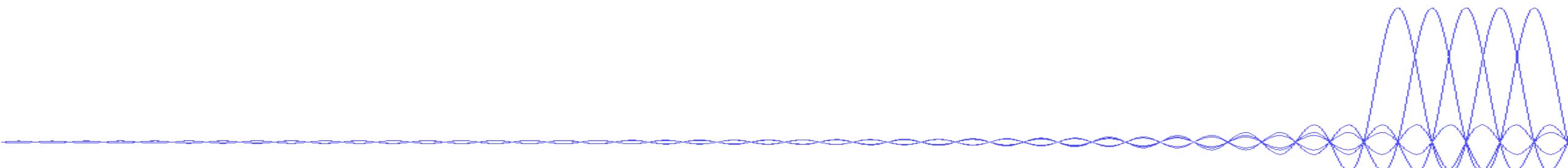


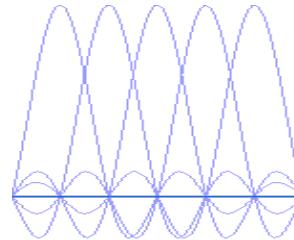


Auto Encoder & Variational Auto Encoder

The 28th LSI Design Contest

in Hokkaido/Okinawa 2025





LSI Design Contest

■ AE (Auto Encoder): オートエンコーダ

□ ニューラルネットワークを使用した圧縮アルゴリズム

□ 応用例

- 異常検出
- 雑音除去

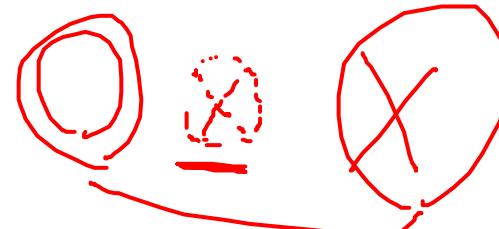
■ VAE (Variational Auto Encoder): 変分オートエンコーダ

□ 潜在空間を確率分布で表現した次元圧縮アルゴリズム

□ 応用例

- 画像生成
- 画像変換

覚冗の間の中間的なかせに(可)



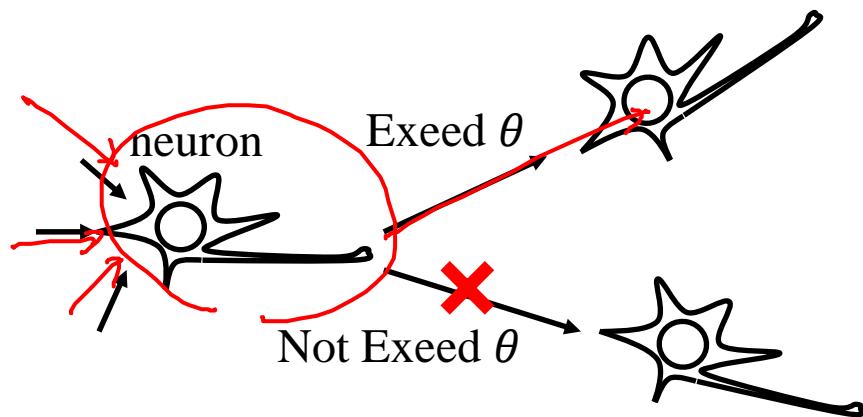
ニューラルネットワークとは

■ ニューラルネットワーク

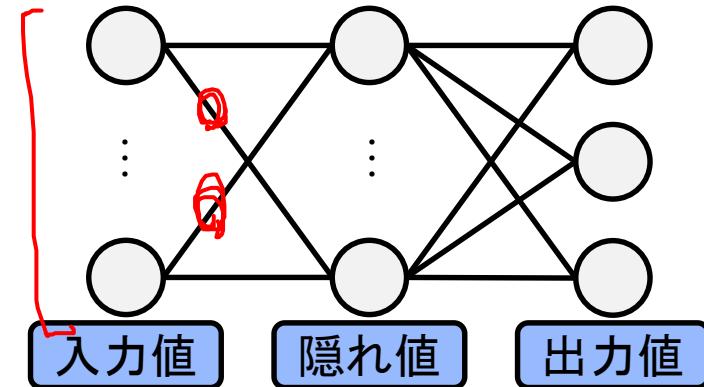
人間の神経細胞の働きを
数学モデルで表現したもの

■ ニューロンのはたらき

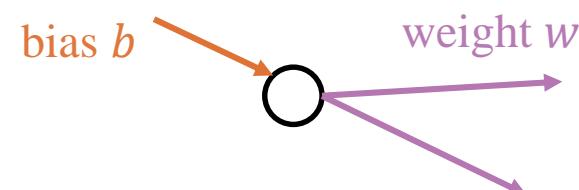
受けた伝達信号を合成
閾値を超えると伝達(発火)

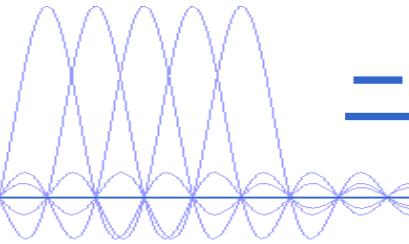


■ 三層パーセプトロン



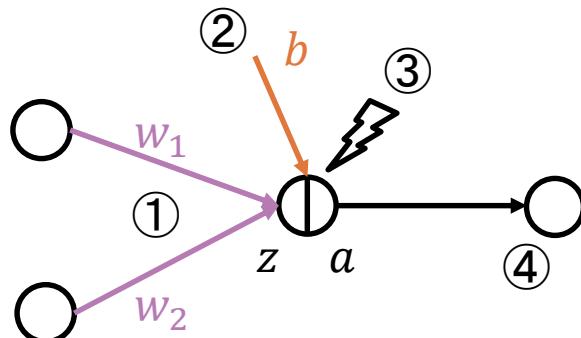
- ◆ ユニットをニューロンと見立て伝達と発火の特性をモデル化したもの
- ◆ 入力値を任意の出力値に近似させる
- ◆ NNの学習には重み w とバイアス b を変更する





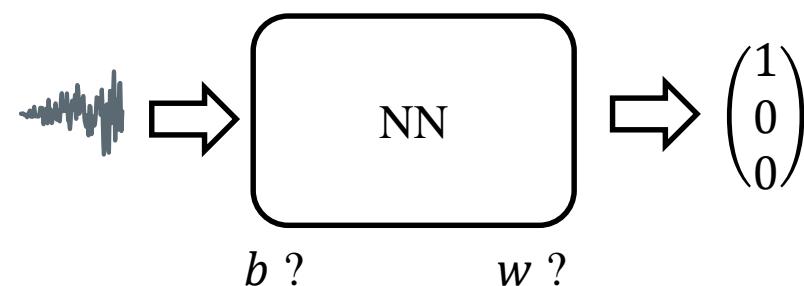
ニューラルネットワークの構造と学習目的

■ 三層パーセプトロンの仕組み



- ① 前層の出力に重み w をかけたものを次のユニットで合計（重み付き入力 z ）
- ② バイアス b がユニットを発火
- ③ z が発火関数 a に変化
- ④ a を新たな入力として次のユニットに伝送（①に戻る）

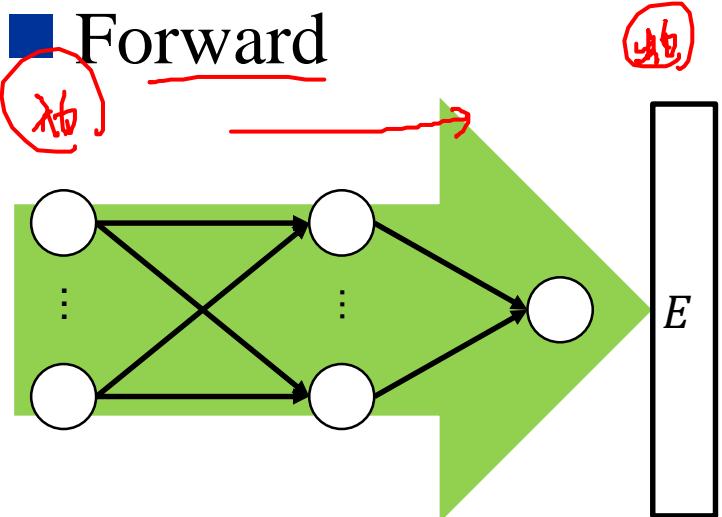
■ 三層パーセプトロンの学習



- 学習させたいデータ
... 教師データ
- 学習させたいデータの期待する出力
... 教師値
- 出力値が教師値に近似されるような NNをつくることが目的
⇒ w と b のパラメータを変化させる

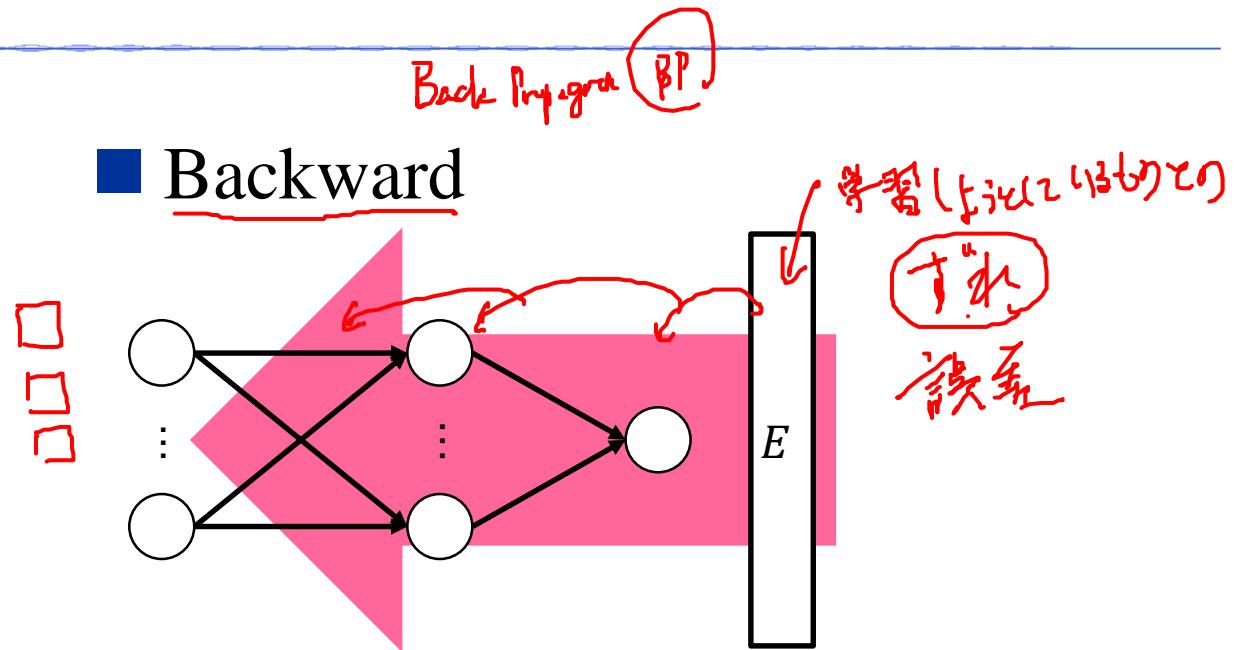
バックプロパゲーションの学習手順

■ Forward



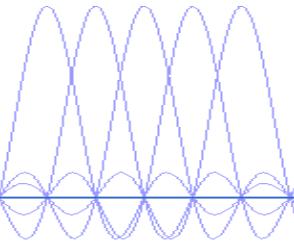
- 教師値を入力し、重みとバイアスの初期値に従って演算する
- 導出した出力値と教師値を比較する
⇒ 誤差関数 E の導出

■ Backward



- 誤差関数 E から逆伝搬し演算する
- 更新すべき変化量 Δ を導出
⇒ パラメータ w, b の更新

出力値と教師値が近似されるまで
ForwardとBackwardを繰り返す



オートエンコーダの学習

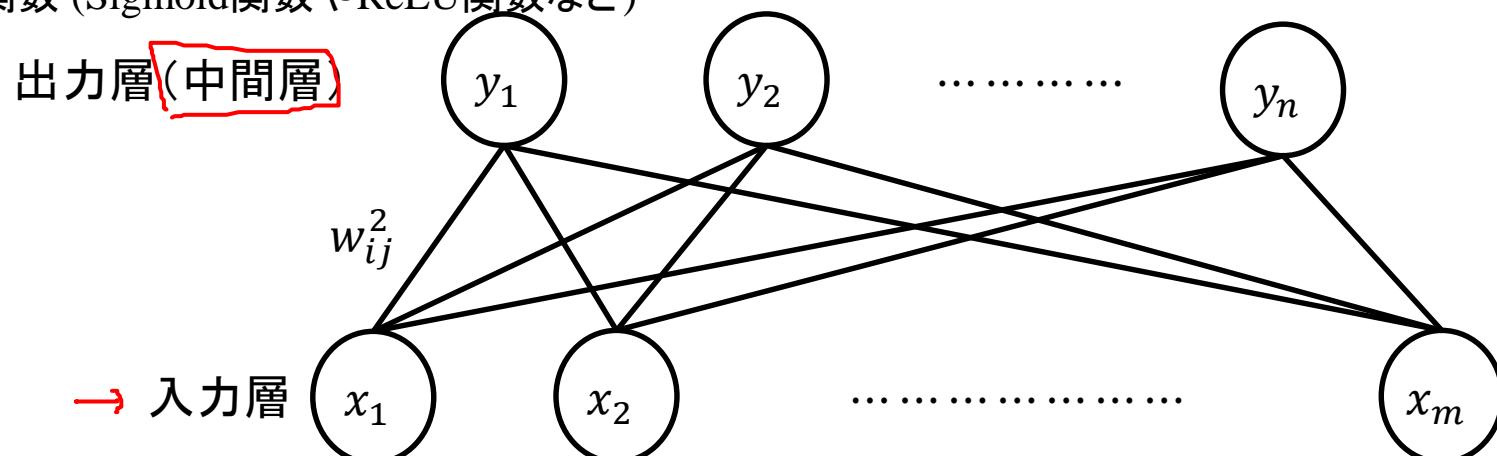
- オートエンコーダ (AE) の基本形は以下のようなネットワーク
- AEの出力層の出力 y_i^2 は以下のように示される

$$z_i^2 = \sum_{j=1}^9 [w_{ij}^2 x_j + b_i^2]$$

$$y_i^2 = a_i^2 = f(z_i^2)$$

x_j :入力データ, w_{ij}^2 :重み, b_i^2 :バイアス, z_i^2 :潜在変数, a_i^2 :出力値

$f(\cdot)$:活性化関数 (Sigmoid関数やReLU関数など)



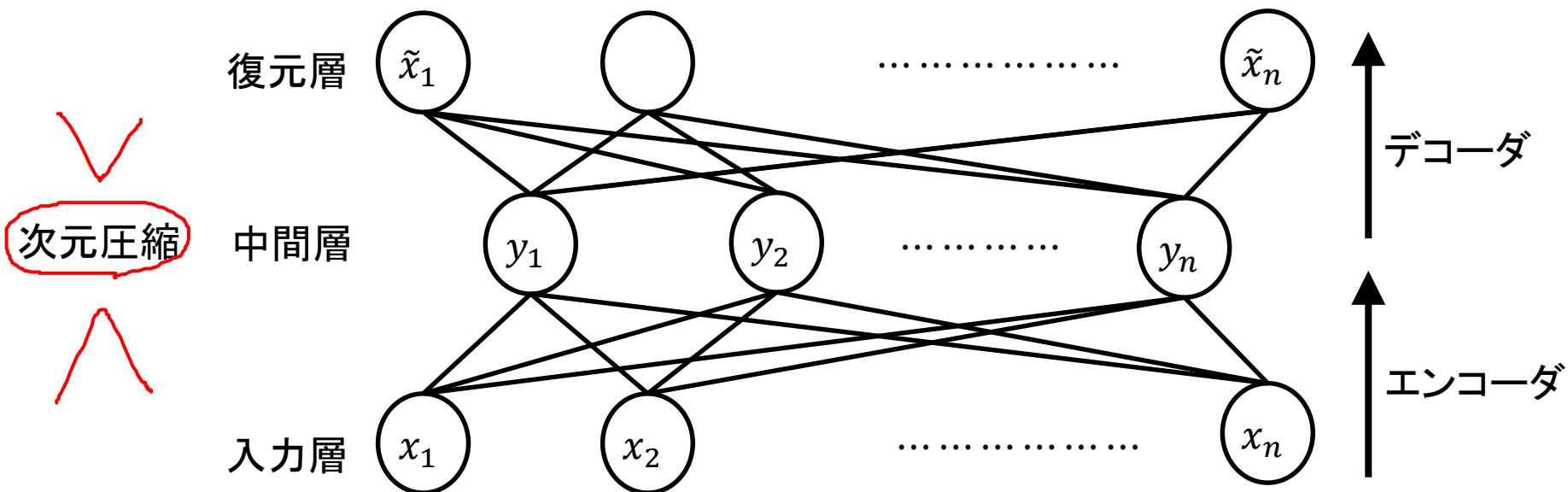
オートエンコーダの学習

■ オートエンコーダの目的

□ 入力サンプルの次元を圧縮しても復元できるような

■ 三層パラメータの学習

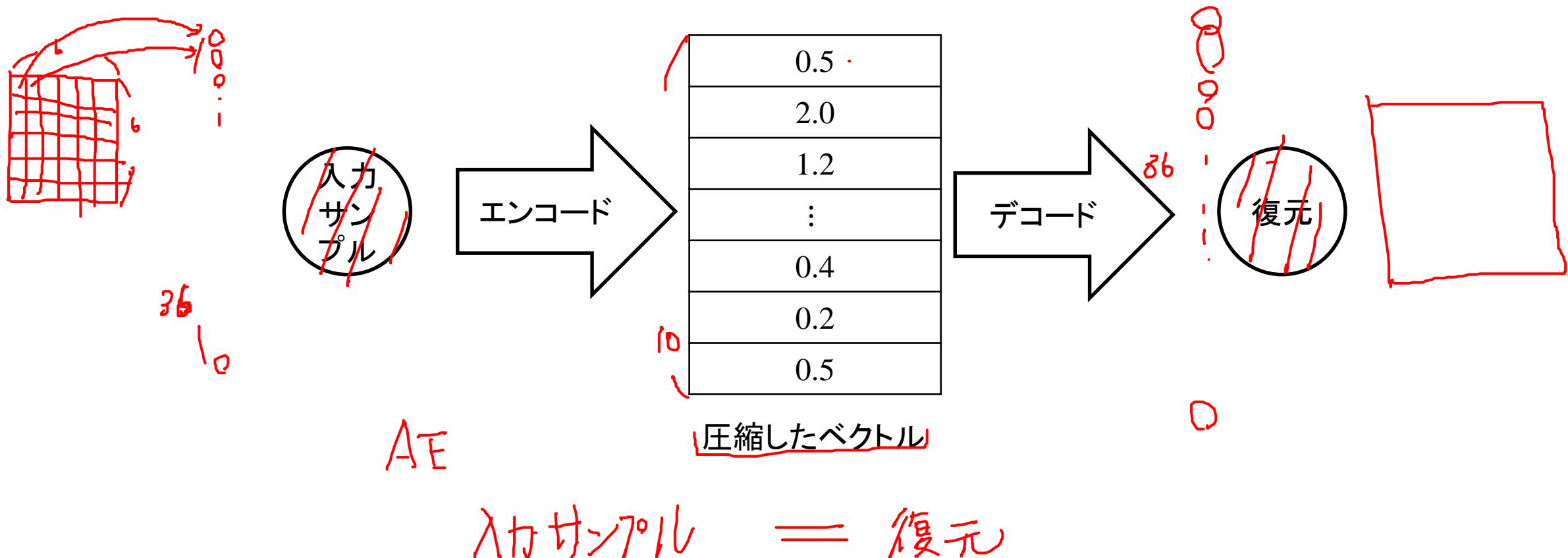
パラメータの学習

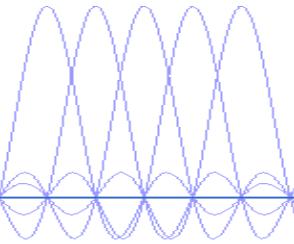


オートエンコーダの学習

■ エンコーダとデコーダ

- 入力サンプルから圧縮したベクトルを生成しそれを復元





オートエンコーダの学習

- 中間層から復元層の間にも重みとバイアスが存在

$$z_j^3 = \sum_{i=1}^2 [w_{ji}^3 a_i^2 + b_j^3]$$

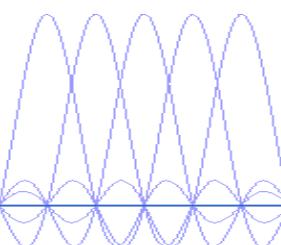
$$\tilde{x}_j = a_j^3 = \underline{\tilde{f}(z_j^3)}$$

w_{ji}^3 :重み, b_j^3 :バイアス, z_j^3 :潜在変数, a_j^3, \tilde{x}_j :復元値, $\tilde{f}(\cdot)$:活性化関数

- 復元層の \tilde{x}_j は以下となる

$$\tilde{x}_j = \tilde{f} \left(\sum_{i=1}^2 \left[w_{ji}^3 f \left(\sum_{j=1}^9 [w_{ij}^2 x_j + b_i^2] \right) + b_j^3 \right] \right)$$

$f(\cdot)$:エンコード側の活性化関数, $\tilde{f}(\cdot)$:デコーダ側の活性化関数



オートエンコーダの学習

- AEの学習では、エンコーダ側及びデコーダ側のパラメータ($w, \tilde{w}, b, \tilde{b}$)を決定
- 入力サンプル x に対し、 \tilde{x} を求め、誤差関数を最小化するようにパラメータを学習
 - 誤差逆伝播法により繰り返し更新して最適なパラメータを更新
- 誤差関数 E には二乗誤差関数や交差エントロピー関数を使用

$$E = \frac{1}{2} \sum_{j=1}^N \|x_j - \tilde{x}_j\|^2$$

入力
出力(復元)

$$E = - \sum_{j=1}^N [x_j \log \tilde{x}_j + (1 - x_j) \log(1 - \tilde{x}_j)]$$

入力
出力

$$x \log \frac{1}{2} + (1-x) \log \frac{1}{2}$$

注

$$0 < z < 1$$

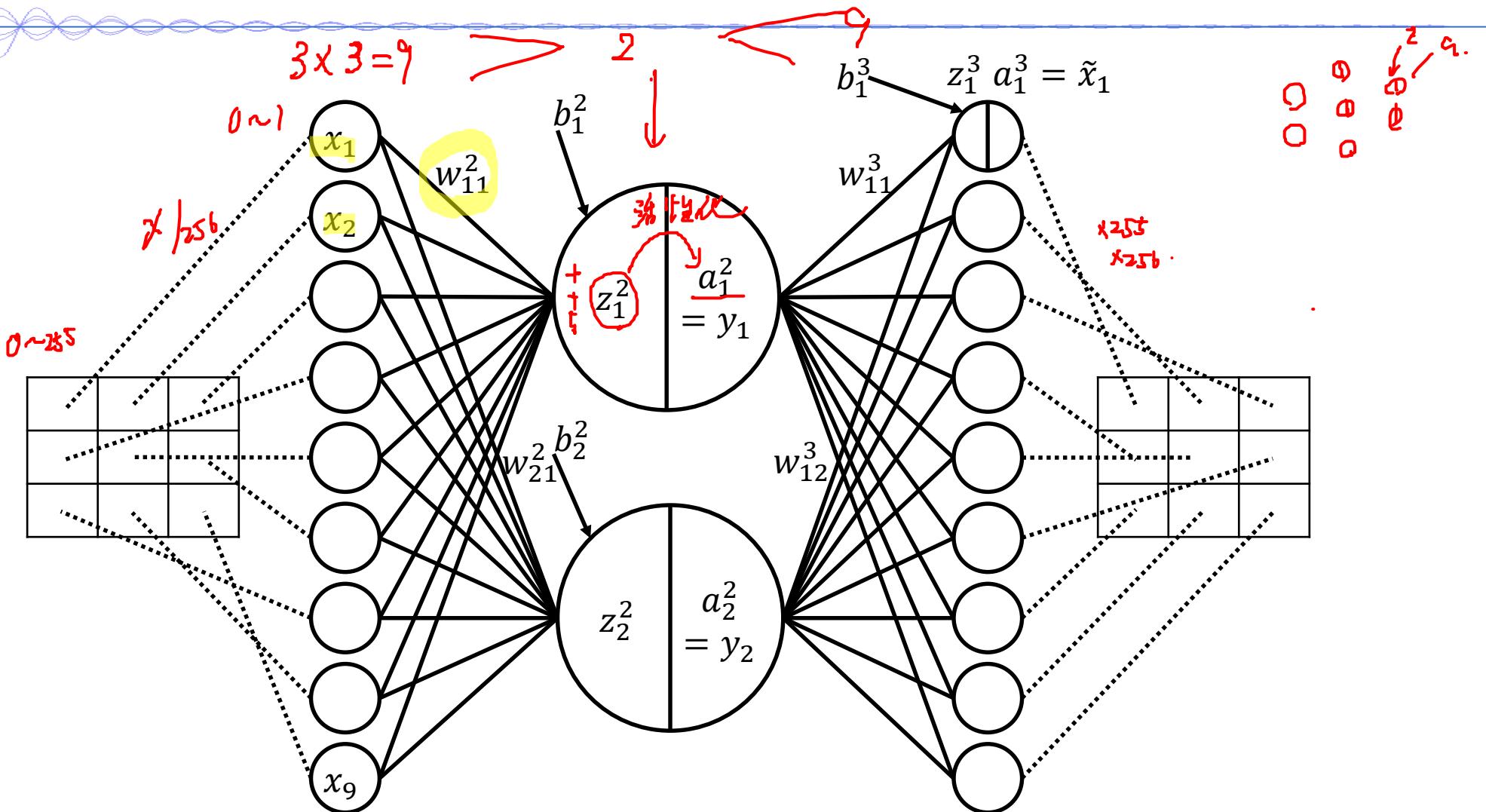
画像 0 ~ 255

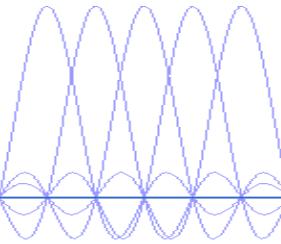
$$\frac{255}{256}$$

二乗誤差関数

交差エントロピー関数
クロス

AEの例





活性化関数と誤差関数

■ 活性化関数

■ 中間層(隠れ層)

□ ReLU (Rectified Linear Unit) 関数

$$f(z) = \begin{cases} 0, & z \leq 0 \\ z, & z > 0 \end{cases}$$

$$\frac{df}{dz} = \begin{cases} 0, & z \leq 0 \\ 1, & z > 0 \end{cases}$$

■ 復元層(出力層)

□ Sigmoid 関数

$$f(z) = \frac{1}{1 + e^{-z}}$$

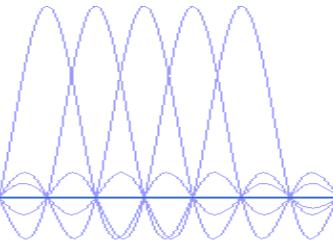
$$\frac{df}{dz} = f(z)(1 - f(z))$$

■ 誤差関数

□ 交差エントロピー関数

$$E = - \sum_{j=1}^N [x_j \log \tilde{x}_j + (1 - x_j) \log(1 - \tilde{x}_j)]$$

$$\begin{aligned} \frac{\partial E}{\partial \tilde{x}_j} &= -\frac{x_j}{\tilde{x}_j} + \frac{1 - x_j}{1 - \tilde{x}_j} \\ &= -\frac{x_j}{a_j^3} + \frac{1 - x_j}{1 - a_j^3} \end{aligned}$$



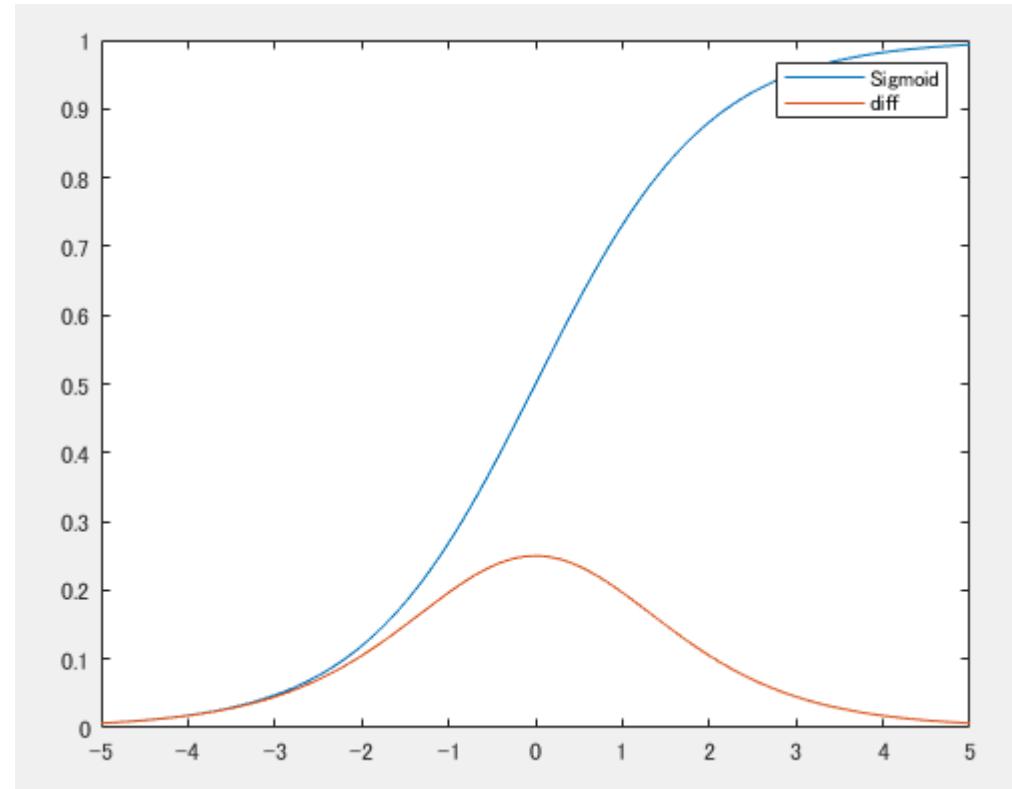
シグモイド関数と微分

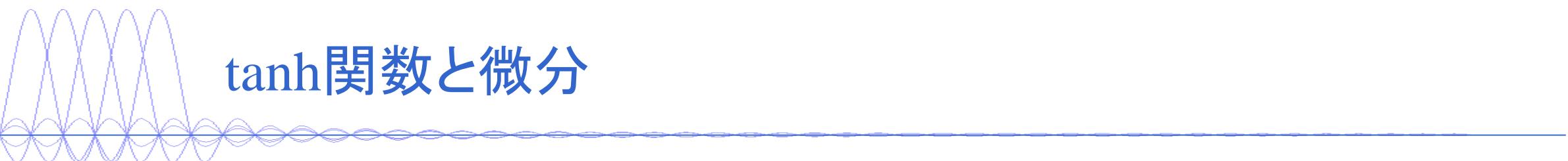
■ シグモイド関数

$$a_i^2 = \frac{1}{1 + \exp(-z_i^2)}$$

■ 微分

$$(1 - a_i^2)a_i^2$$





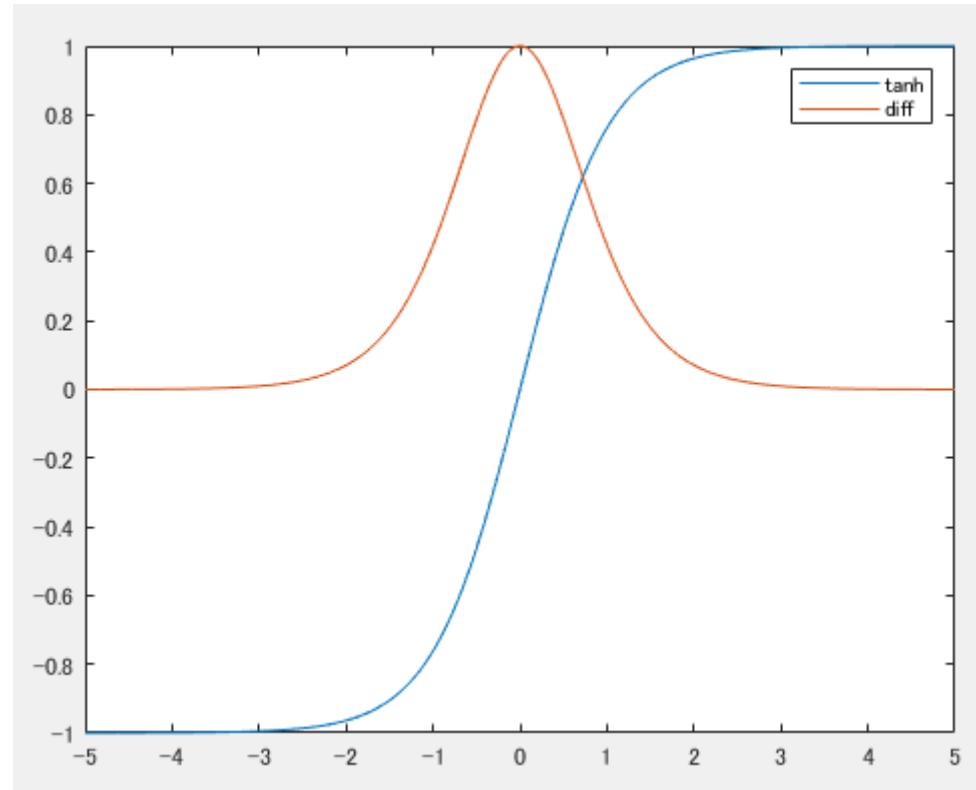
tanh関数と微分

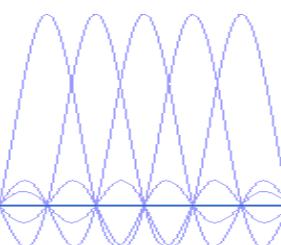
■ tanh (hyperbolic tangent) 関数

$$a_i^2 = \frac{e^{z_i^2} - e^{-z_i^2}}{e^{z_i^2} + e^{-z_i^2}}$$

■ 微分

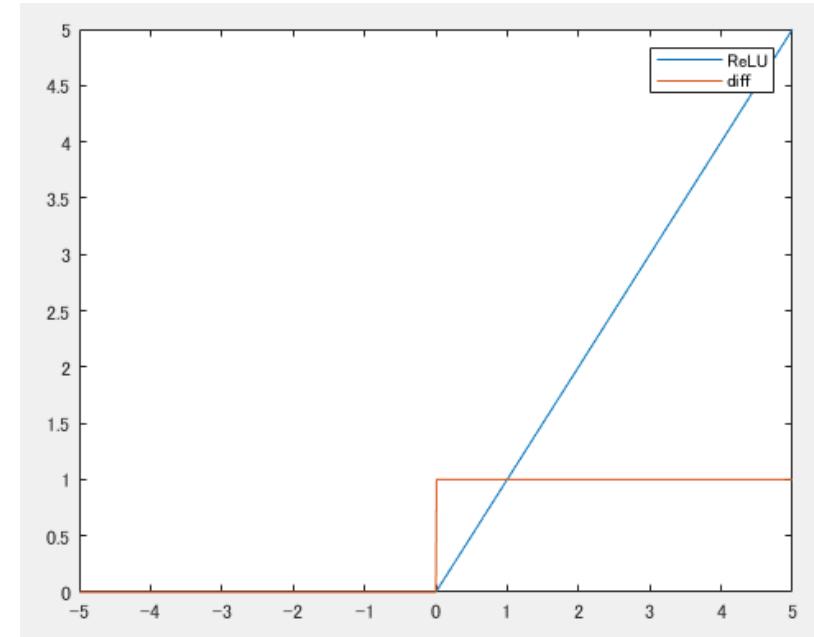
$$\frac{4}{\left(e^{z_i^2} + e^{-z_i^2}\right)^2}$$

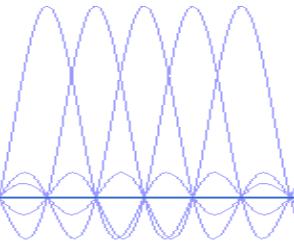




ReLU関数と微分

- 現在よく使用されている関数
- ReLU (Rectified Linear Unit) 関数
$$a_i^2 = \max(0, z_i^2)$$
- 微分
$$\begin{cases} 0, & z_i^2 < 0 \\ 1, & z_i^2 \geq 0 \end{cases}$$
- 勾配消失しにくい
- z が0以下であると微分値も0になるため、学習がうまくいかないことがある。





Gradient descent

■ Gradient descent

□ E : 誤差関数の値

□ E を最小化する重み W とバイアス b を見つけるアルゴリズム

$$(\Delta w_{11}^2 \dots, \Delta w_{11}^3 \dots, \Delta b_1^2 \dots, \Delta b_1^3 \dots) = -\eta \left(\frac{\partial E}{\partial w_{11}^2}, \frac{\partial E}{\partial w_{11}^3}, \frac{\partial E}{\partial b_1^2}, \frac{\partial E}{\partial b_1^3} \right)$$

この値が小さいほど, E は小さくなる. (η は学習率)

しかしながら, この数式を計算することは非常に困難である.

入力ユニット数: 10, 隠れユニット数: 10, 出力ユニット数: 3 のとき,

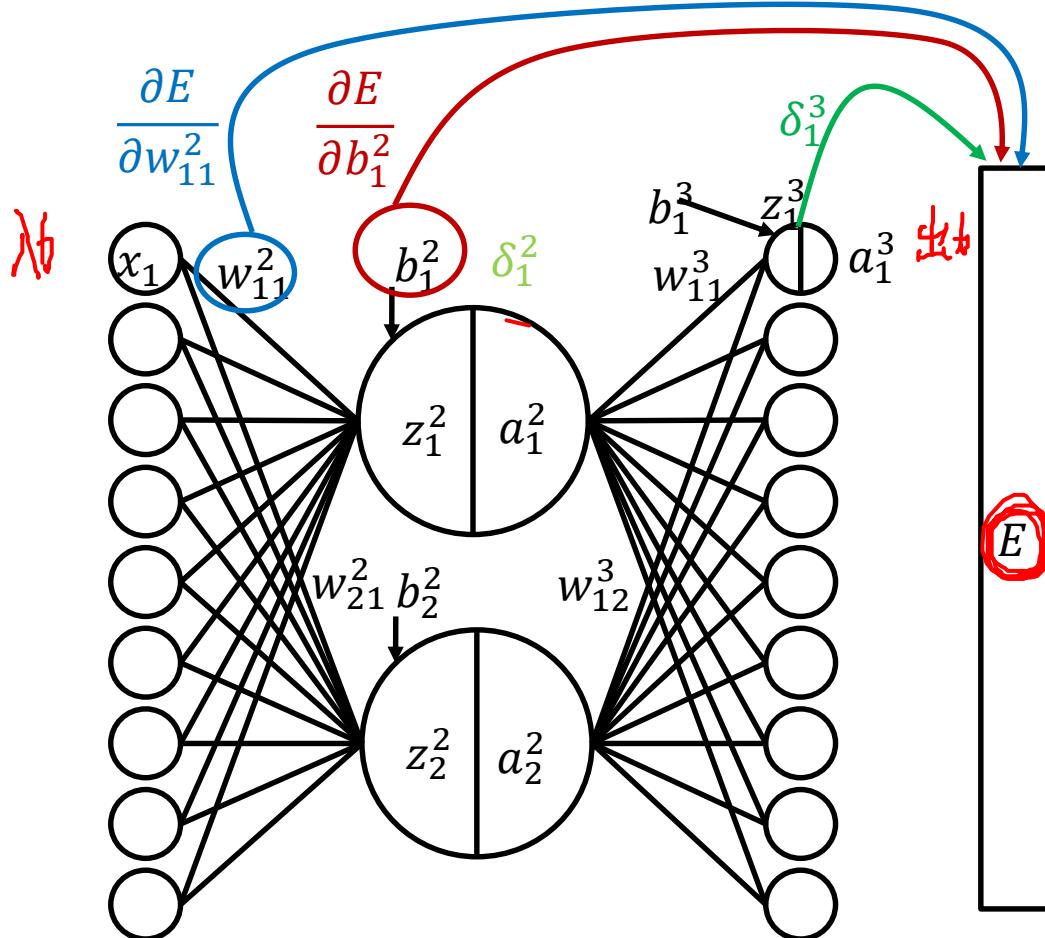
パラメータの総数は $10*10+10(\text{bias})+10*3+3(\text{bias})=143$ となる.

→できるだけ早く誤差を縮めたいが, 一度にこれを計算しなければならない.

⇒バックプロパゲーションを行う.

誤差関数の偏微分

■ 誤差関数 E に対する w, b の偏微分



$$\frac{\partial E}{\partial w_{11}^2} = \frac{\partial E}{\partial z_1^2} \frac{\partial z_1^2}{\partial w_{11}^2} = \frac{\partial E}{\partial z_1^2} x_1$$

($\because z_1^2 = w_{11}^2 x_1 + w_{12}^2 x_2 + \dots + w_{19}^2 x_9 + b_1^2$)

$$\delta_1^2 := \frac{\partial E}{\partial z_1^2}$$

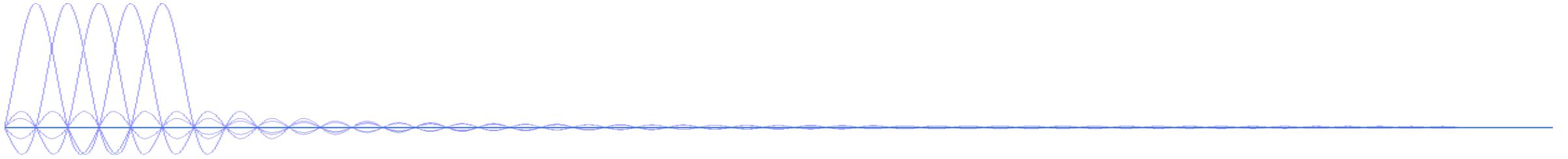
$$\frac{\partial E}{\partial w_{11}^2} = \delta_1^2 x_1$$

$$\frac{\partial E}{\partial b_1^2} = \frac{\partial E}{\partial z_1^2} \frac{\partial z_1^2}{\partial b_1^2} = \delta_1^2$$

$$\therefore \frac{\partial E}{\partial w_{n_m n_{m-1}}^m} = \delta_{n_m}^m a_{n_{m-1}}^{m-1},$$

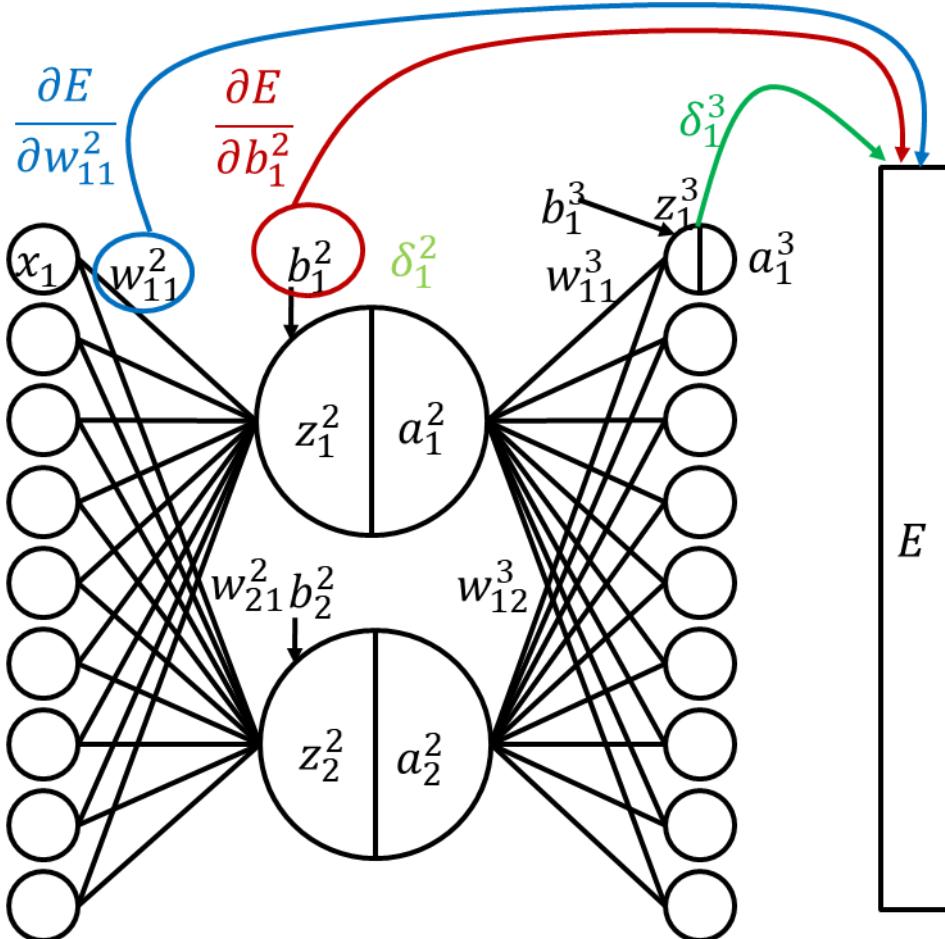
$$\frac{\partial E}{\partial b_{n_m}^m} = \delta_{n_m}^m$$

$$\left(m = 2 \text{ or } 3, a_{n_1}^1 = x_{n_1}, n_m = \begin{cases} 1, 2, \dots, 9 & m = 1, 3 \\ 1, 2 & m = 2 \end{cases} \right)$$



ユニット誤差の計算

- ユニット誤差 δ により複雑な偏微分を計算せずに済む



簡単に出力層エラーを計算できる。

$$\delta_1^3 = \frac{\partial E}{\partial z_1^3} = \frac{\partial E}{\partial a_1^3} \frac{\partial a_1^3}{\partial z_1^3} = \frac{\partial E}{\partial a_1^3} f'(z_1^3)$$

$$\therefore \delta_1^3 = -x_1(1 - a_1^3) + (1 - x_1)a_1^3$$

($f(\cdot)$:シグモイド関数)

しかし、中間層は計算が難しい。

$$\delta_1^2 = \frac{\partial E}{\partial z_1^2} = \frac{\partial E}{\partial z_1^3} \frac{\partial z_1^3}{\partial a_1^2} \frac{\partial a_1^2}{\partial z_1^2} + \frac{\partial E}{\partial z_2^3} \frac{\partial z_2^3}{\partial a_1^2} \frac{\partial a_1^2}{\partial z_1^2} + \dots$$
$$\left(\frac{\partial E}{\partial z_1^3} = \delta_1^3, \frac{\partial E}{\partial z_2^3} = \delta_2^3, \dots, \frac{\partial z_1^3}{\partial a_1^2} = w_{11}^3, \frac{\partial z_2^3}{\partial a_1^2} = w_{21}^3, \dots \right)$$

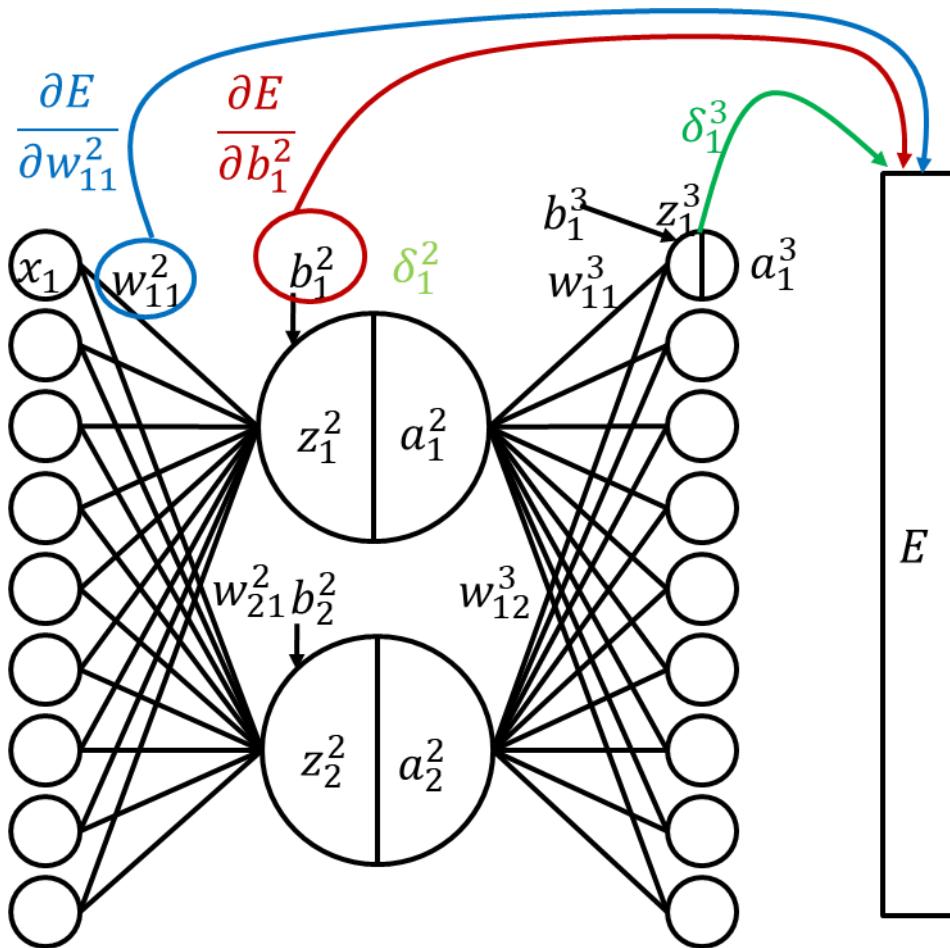
$$\delta_1^2 = \delta_1^3 w_{11}^3 a'(z_1^2) + \delta_2^3 w_{21}^3 a'(z_1^2) + \dots$$

$$\therefore \delta_1^2 = (\delta_1^3 w_{11}^3 + \delta_2^3 w_{21}^3 + \dots) a'(z_1^2)$$

バックワード処理

潜在変数次元: $i = 1, 2$
入力&出力画素数: $j = 1, \dots, 9$

- 全体の誤差の大きさから、ユニット毎の誤差を求める



$$\delta_j^3 = -x_j(1 - a_j^3) + (1 - x_j)a_j^3 \dots (1)$$

$$a'(z_j^3) = a(z_j^3)(1 - a(z_j^3))$$

$$\frac{\partial E}{\partial w_{ji}^3} = \delta_j^3 a_i^2 \dots (2) \quad \frac{\partial E}{\partial b_j^3} = \delta_j^3 \dots (3)$$

$$\delta_i^2 = \sum_{j=1}^9 [\delta_j^3 w_{ji}^3] a'(z_i^2) \dots (4)$$

$$\frac{\partial E}{\partial w_{ij}^2} = \delta_i^2 x_j \dots (5) \quad \frac{\partial E}{\partial b_i^2} = \delta_i^2 \dots (6)$$

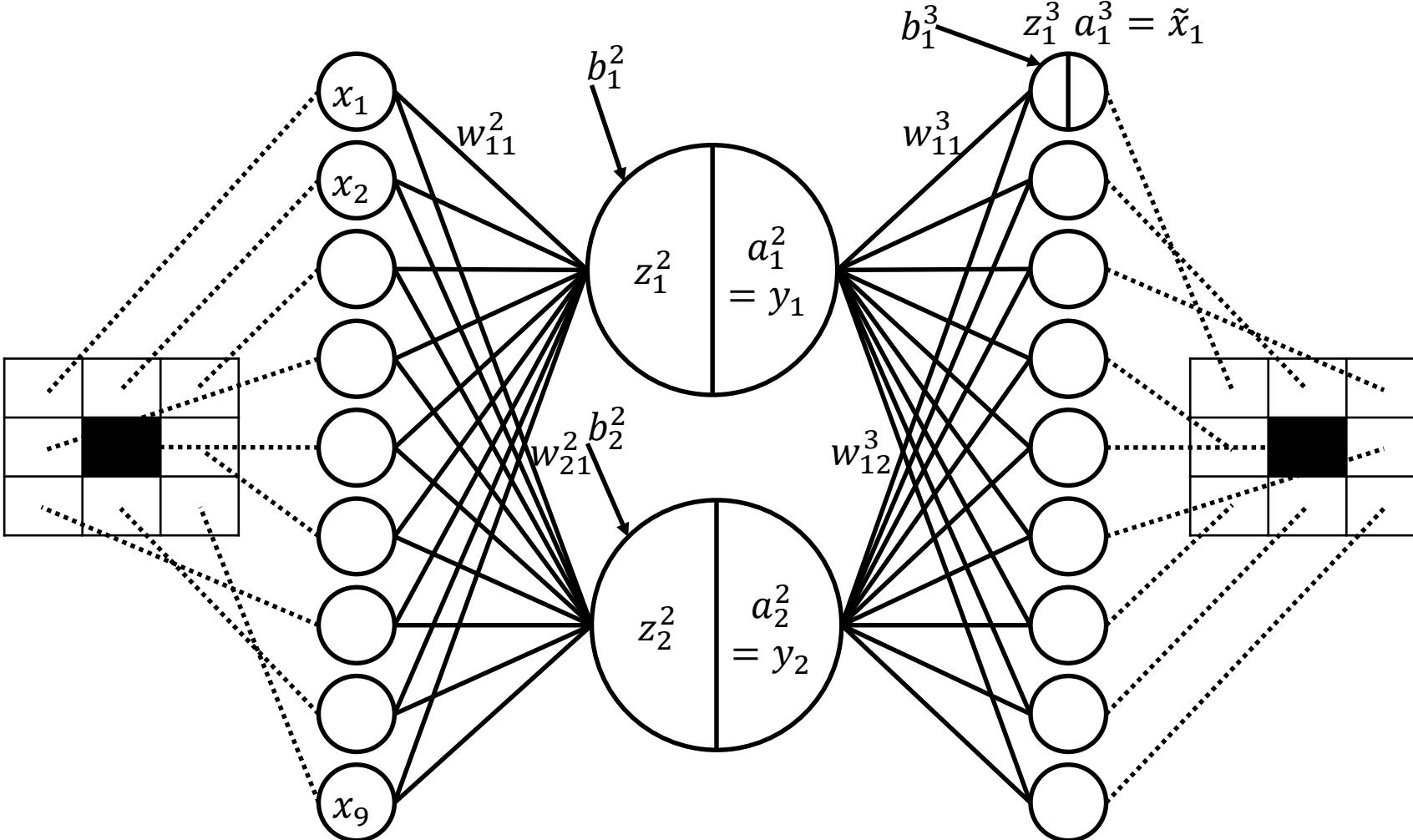
$$w_{ji}^3 = w_{ji}^3 - \eta \frac{\partial E}{\partial w_{ji}^3} \dots (7)$$

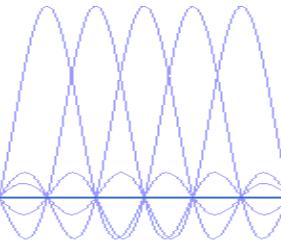
$$b_j^3 = b_j^3 - \eta \frac{\partial E}{\partial b_j^3} \dots (8)$$

$$w_{ij}^2 = w_{ij}^2 - \eta \frac{\partial E}{\partial w_{ij}^2} \dots (9)$$

$$b_i^2 = b_i^2 - \eta \frac{\partial E}{\partial b_i^2} \dots (10)$$

AEの例 (○の場合)





wとbの初期値

w2 (w^2)

0.1355	0.9326	0.3882	0.6574	0.9642	0.4552	0.0417	0.0032	0.6109	-0.5000
0.8879	0.4456	0.2576	0.4926	0.8010	0.8011	0.7695	0.2928	0.9130	-0.5000

b2 (b^2)

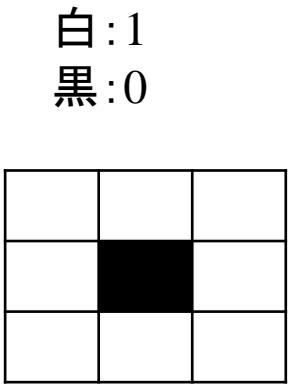
w3 (w^3)

0.3001	0.5197
0.2486	0.1547
0.6664	0.0146
0.9875	0.3242
0.4683	0.9909
0.1233	0.5131
0.9160	0.8765
0.9461	0.0674
0.2777	0.2842

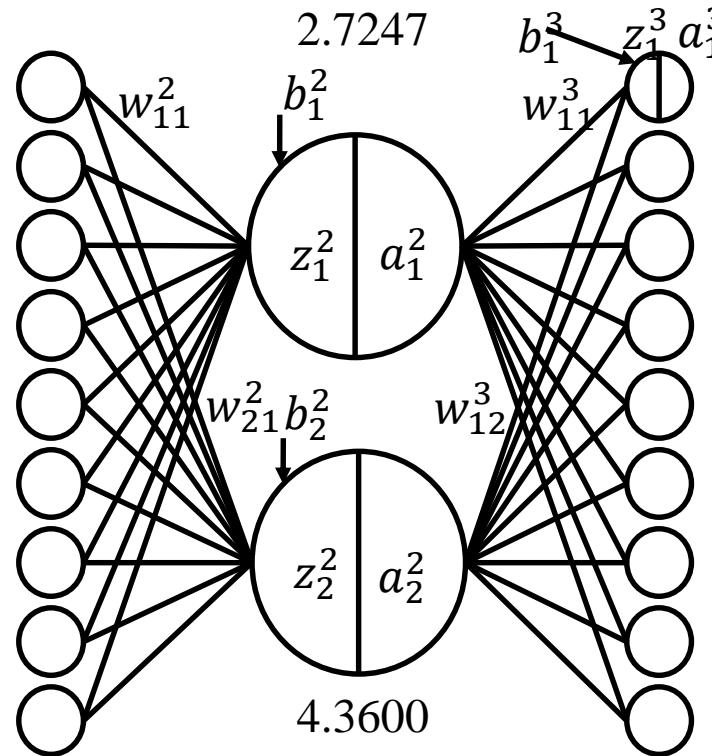
b3 (b^3)

-0.5000
-0.5000
-0.5000
-0.5000
-0.5000
-0.5000
-0.5000
-0.5000
-0.5000

AEの例 (○の場合, 初期値)



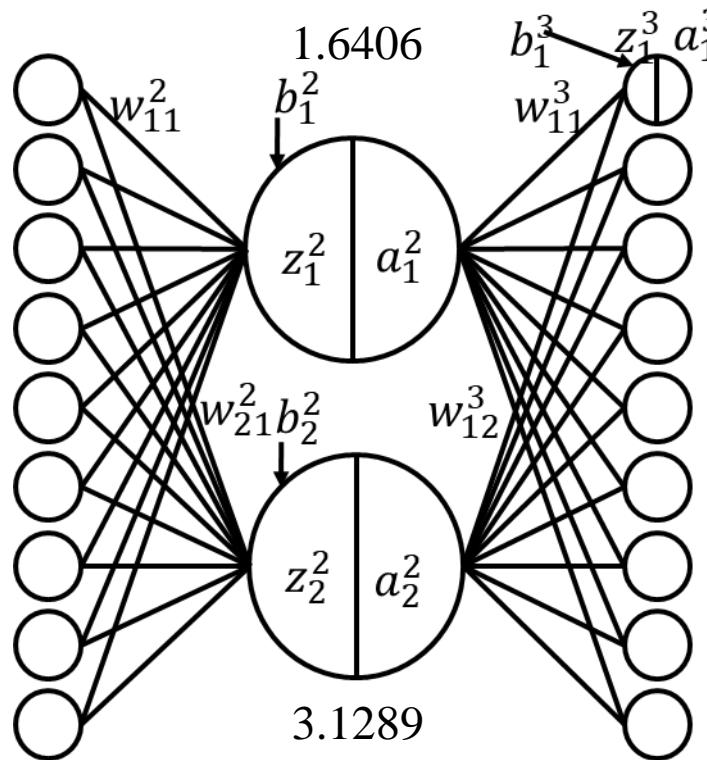
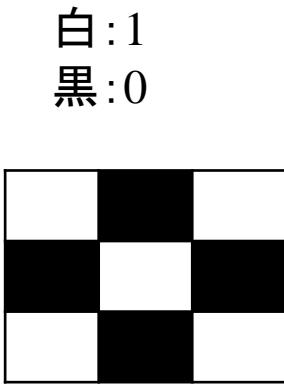
1
1
1
1
0
1
1
1
1



0.9298
0.7010
0.7989
0.9735
0.9939
0.8883
0.9970
0.9147
0.8169

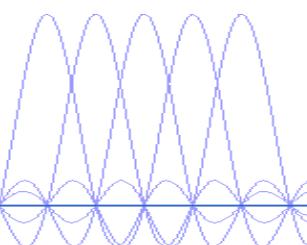


AEの例（×の場合、初期値）

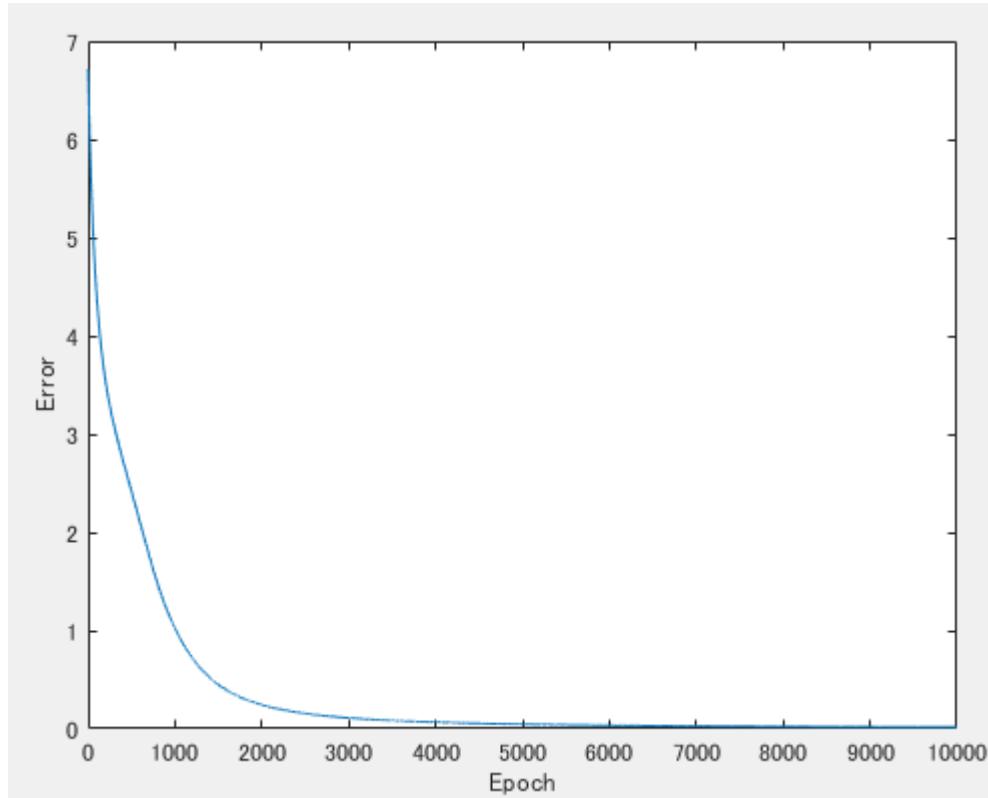


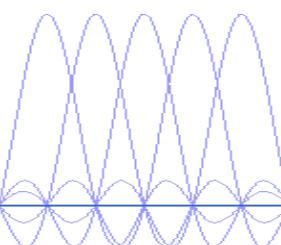
0.8346
0.5968
0.6545
0.8942
0.9667
0.7872
0.9769
0.7796
0.6995

E



誤差関数の履歴





10000回学習後の w と b

w2 (w^2)

0. 0636	2. 1421	0. 3164	1. 8668	-0. 3171	1. 6647	-0. 0301	1. 2126	0. 5391	-0. 5719
1. 4179	0. 0681	0. 7876	0. 1151	1. 7085	0. 4236	1. 2995	-0. 0847	1. 4431	0. 0300

b2 (b^2)

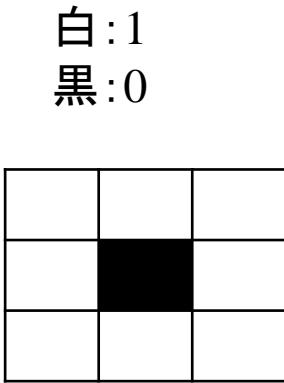
w3 (w^3)

0. 4572	1. 1117
1. 4884	-0. 7096
0. 8875	1. 0320
1. 5475	-0. 7296
-1. 4697	0. 9091
1. 4746	-0. 6863
0. 9472	1. 1527
1. 5595	-0. 7469
0. 5013	1. 0800

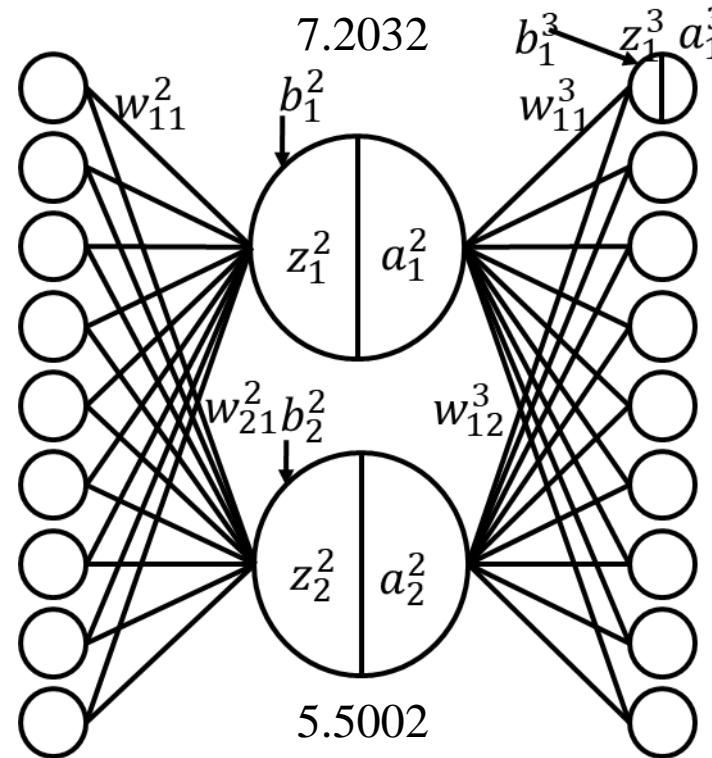
b3 (b^3)

-0. 3430
-0. 7622
-0. 2275
-0. 8162
-0. 4913
-0. 8597
-0. 4318
-0. 7462
-0. 2863

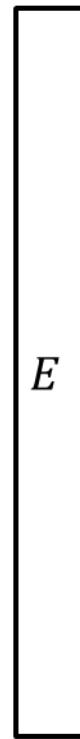
AEの例 (○の場合, 10000回学習後)



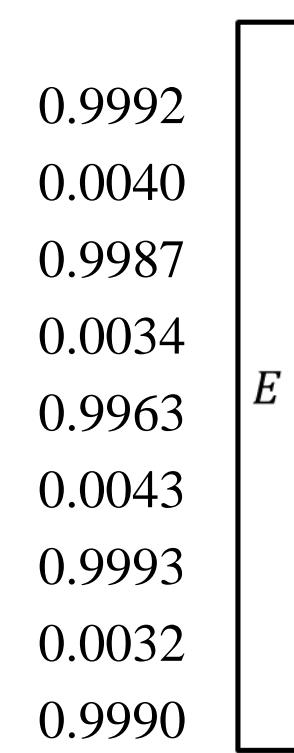
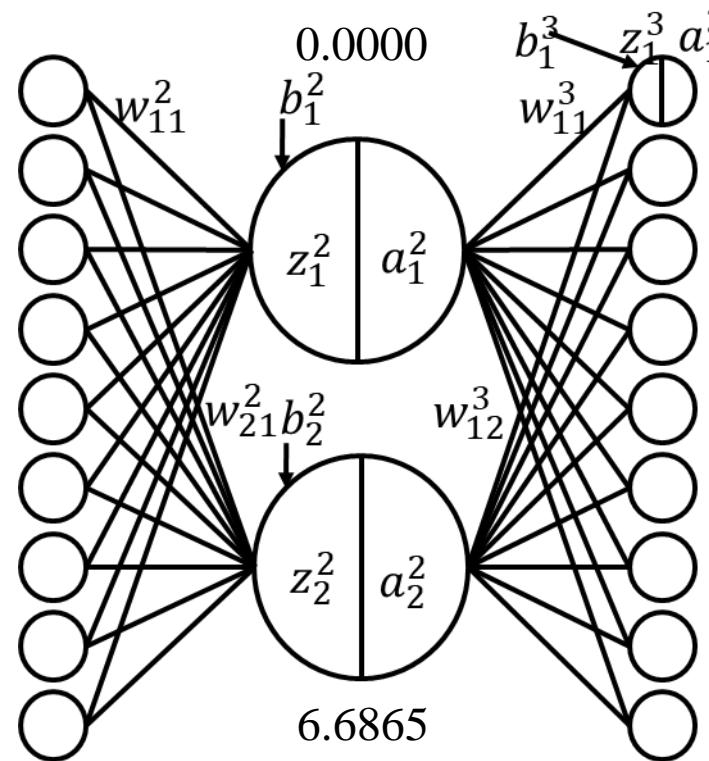
1
1
1
1
0
1
1
1
1



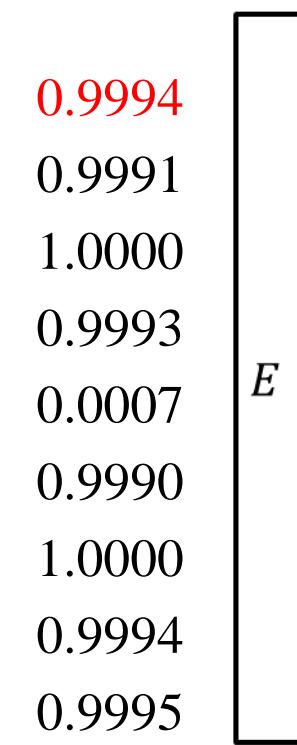
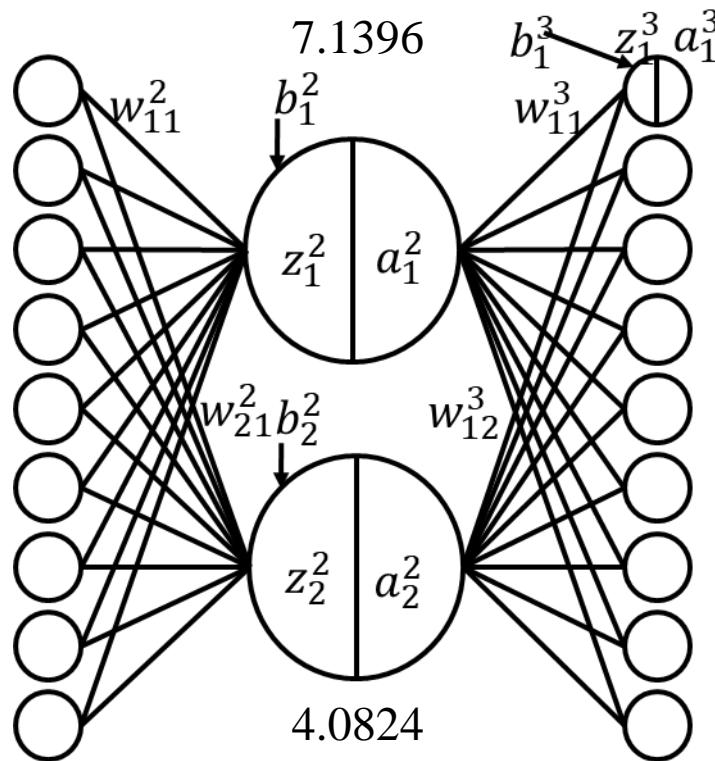
0.9999
0.9977
1.0000
0.9982
0.0023
0.9975
1.0000
0.9983
0.9999

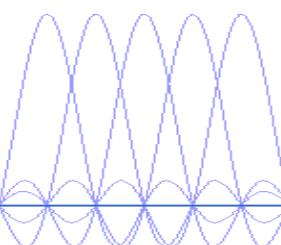


AEの例（×の場合、10000回学習後）



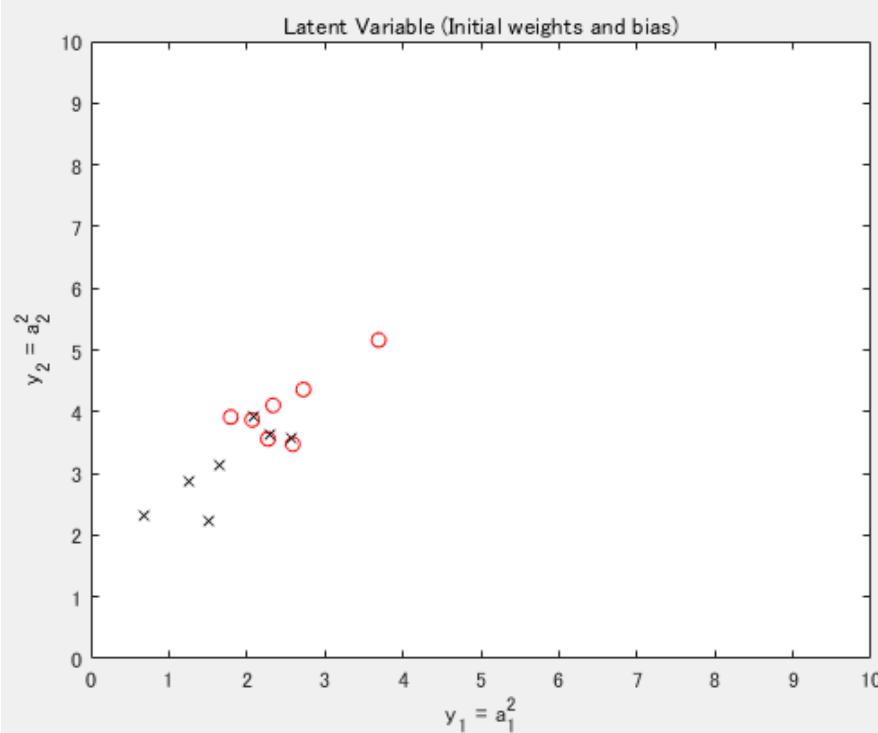
1画素誤りがあった場合の出力



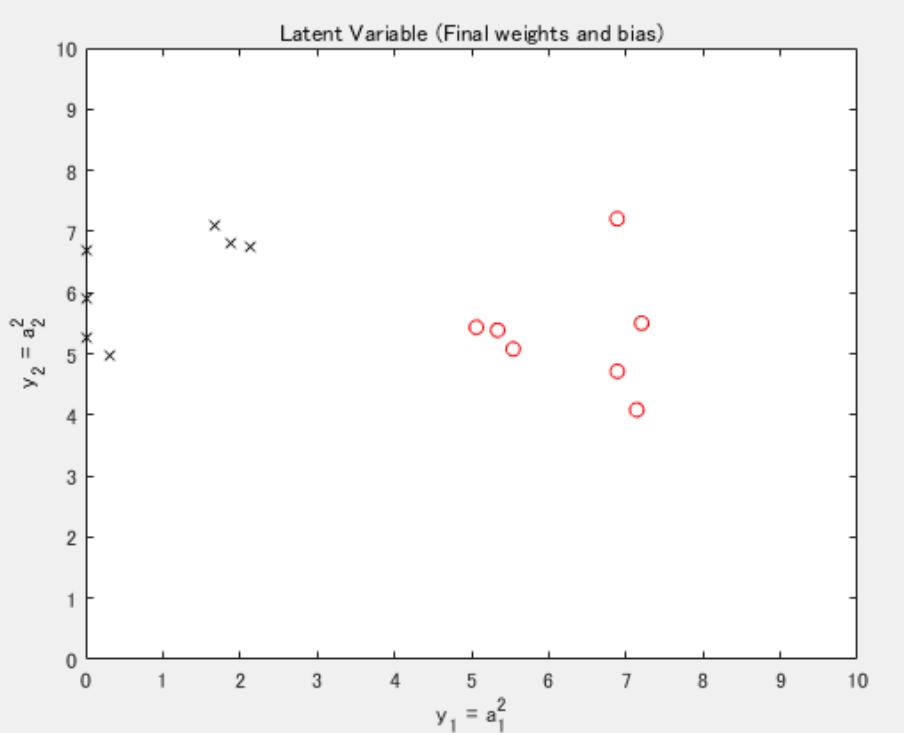


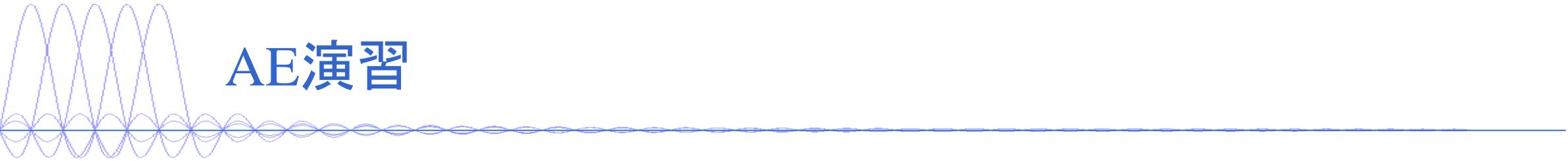
潜在変数 (1画素誤り6パターン)

Initial parameters



Final parameters





AE演習

■ AEを動作させよう

□ 作業フォルダ : ¥AE

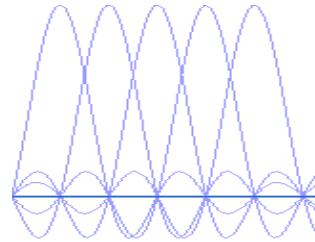
- OX_judge_AE.m を読み込み、実行。

- リーダ : 2025

- 副リーダ : 2026

- 総務 : 2027

- 書記 : 2028



LSI Design Contest

■ AE (Auto Encoder): オートエンコーダ

□ ニューラルネットワークを使用した圧縮アルゴリズム

□ 応用例

■ 異常検出

■ 雑音除去

■ VAE (Variational Auto Encoder): 変分オートエンコーダ

□ 潜在空間を確率分布で表現した次元圧縮アルゴリズム

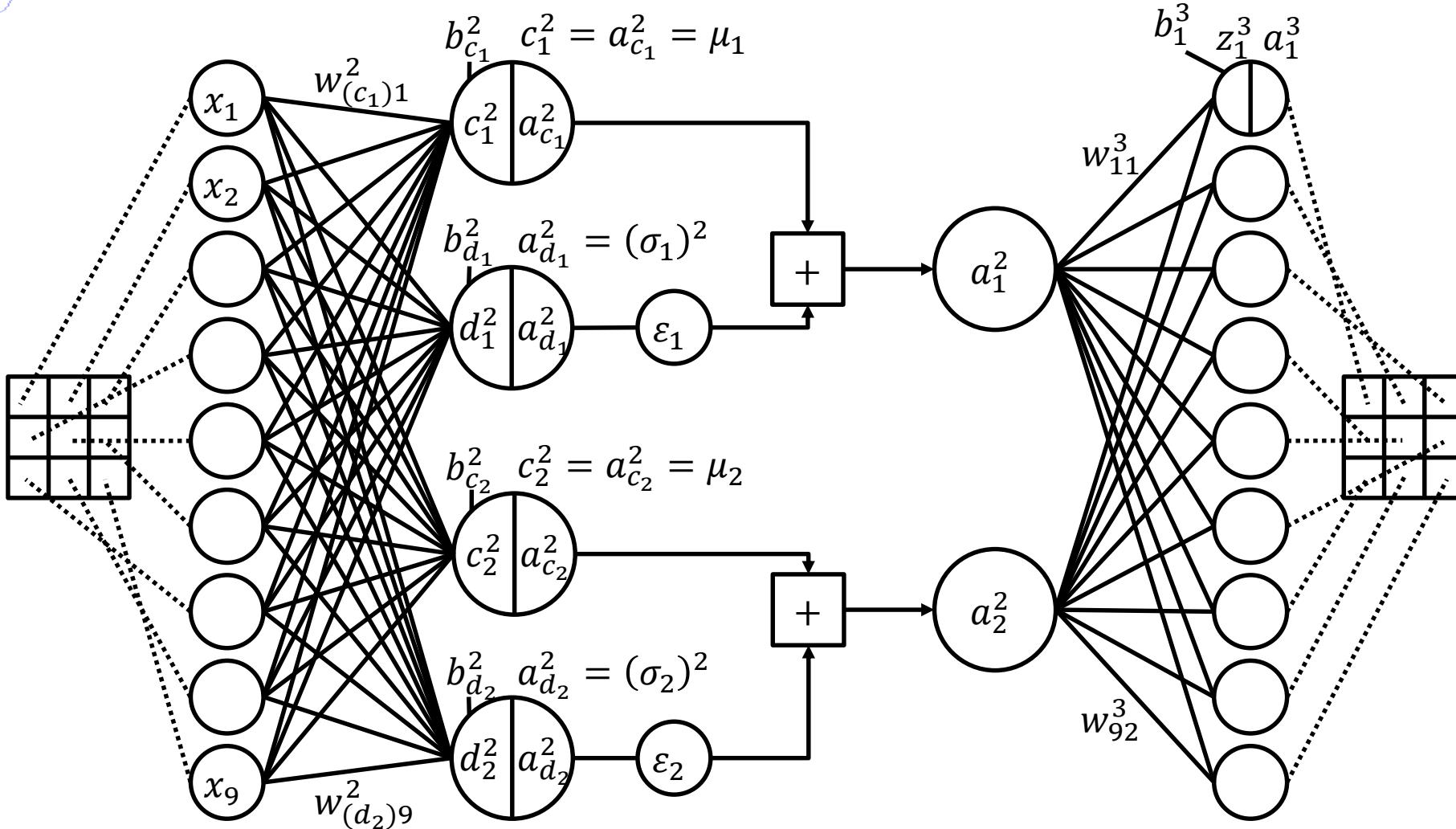
□ 応用例

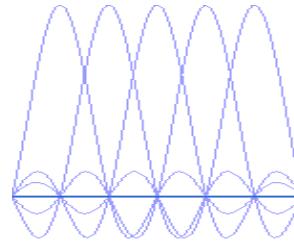
■ 画像生成

■ 画像変換

VAEの例

- ・潜在変数 z^2 の平均を c^2 , 偏差を d^2 と表記している
- ・下付きの c_i^2, d_i^2 は c_i, d_i と表記している





順伝搬の流れ

$a_{c_1}^2 \leftarrow 2\text{乗ではない} \Rightarrow 2\text{番目の層}$

潜在変数次元: $i = 1, 2$
入力&出力画素数: $j = 1, \dots, 9$

$$(1) c_i^2 = \sum_{j=1}^9 [w_{(c_i)j}^2 x_j] + b_{c_i}^2 \quad a_{c_1}^2 = f^2(c_1^2) = c_1^2$$

$$(2) c_i^2 = a_{c_i}^2$$

$$(3) d_i^2 = \sum_{j=1}^9 [w_{(d_i)j}^2 x_j] + b_{d_i}^2$$

$$(4) f^2(d_i^2) = a_{d_i}^2 = \log(1 + e^{d_i^2}) = \sigma_i^2$$

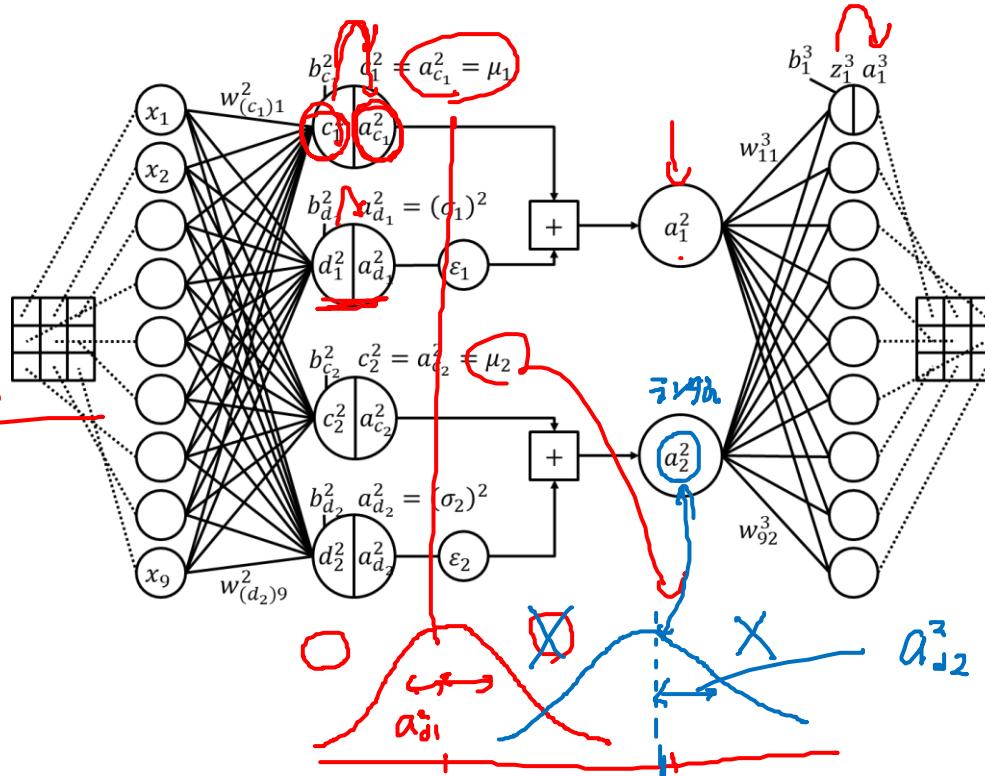
$$(5) \varepsilon_i \sim N(0, 1)$$

$$(6) a_i^2 = N(\mu_i, \sigma_i^2) = \mu_i + \sigma_i \varepsilon_i$$

$$= N(a_{c_i}^2, \sqrt{a_{d_i}^2}) = a_{c_i}^2 + \sqrt{a_{d_i}^2} \varepsilon_i$$

$$(7) c_j^3 = \sum_{i=1}^2 [w_{ji}^3 a_i^2] + b_{c_j}^2$$

$$(8) f^3(c_j^3) = a_j^3 = \frac{1}{1 + e^{-c_j^3}}$$



$$f^2(x) = \log(1 + e^x) : \text{ソフトプラス関数}$$

$$f^3(x) = \frac{1}{1 + e^{-x}} : \text{シグモイド関数}$$

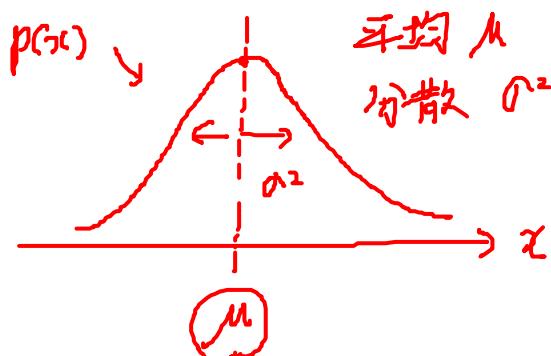
復習: 平均と分散

■ 公平なサイコロにおける出目の平均と分散を求めよ.

出目	1	2	3	4	5	6
確率	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$

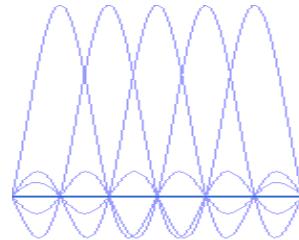
$$16.66\% = \frac{6}{36} = \frac{1}{6}$$

ガウス分布 (正規分布)



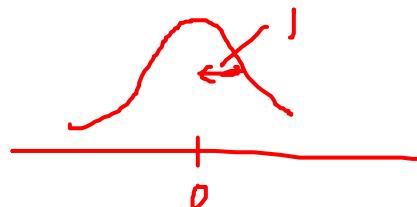
$$\begin{aligned} E[X] &= 1 \times \frac{1}{6} + 2 \times \frac{1}{6} + \dots + 6 \times \frac{1}{6} \quad (\leftarrow \sum_{i=1}^{N-1} x_i p_i) \\ &= \frac{21}{6} = 3.5 = \mu \end{aligned}$$

$$\begin{aligned} V[X] &= E[(X-\mu)^2] = (1-3.5)^2 \frac{1}{6} + (2-3.5)^2 \frac{1}{6} + (3-3.5)^2 \frac{1}{6} \\ &\quad + (4-3.5)^2 \frac{1}{6} + (5-3.5)^2 \frac{1}{6} + (6-3.5)^2 \frac{1}{6} \\ &= \frac{70}{24} = 2.916 \end{aligned}$$



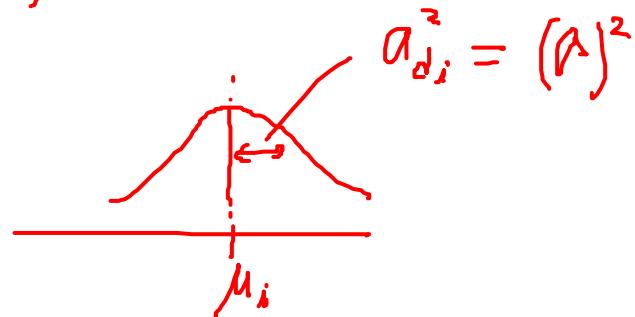
ある平均と分散を持ったランダムな信号を作りたい

① 平均 μ , 分散 σ^2 の ガウス分布(正規分布)にしたがうランダム信号を作る。

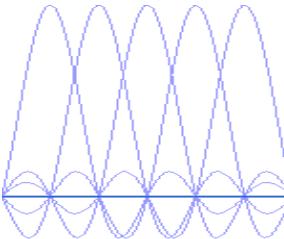


⇒ このようなランダム信号 ξ

② $\xi = \text{欲しい分散} \times \text{ガウス} + \text{欲しい平均} \times \text{足す}$



$$\sigma_{d,i}^2 = (\alpha)^2$$



実習: 平均, 分散を指定したランダム信号の生成 ex_rand.m

■ 作業フォルダ: G:\YLSI2025

■ 作業1: 平均0, 分散1のガウス分布のランダム信号(ep)を100,000個作成する

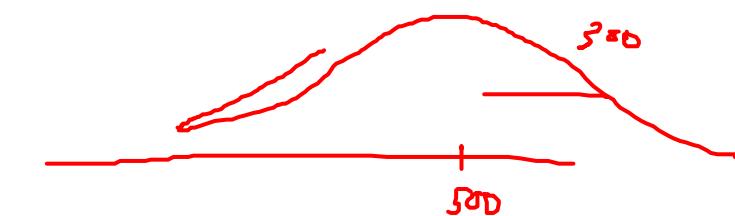
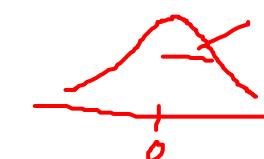
■ 作業2: ランダム信号の分布を確認する

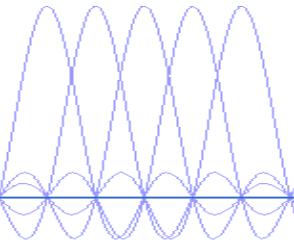
■ 作業3: ランダム信号に $\sqrt{\text{分散}}$ をかける

■ 作業4: 作業3の信号に平均をたす

■ 作業5: 作業4の信号の分布を確認する

■ 今回作成する信号: 平均: 500 , 分散: 300





VAEの損失関数

■ 再構築ロス

- 復元画像と入力画像の誤差
- バイナリクロスエントロピーを使用

$$E_{rec} = - \sum_{j=1}^9 [x_j \log a_j^3 + (1 - x_j) \log(1 - a_j^3)]$$

■ 潜在ロス

- ガウス分布とプログラムによって生成された分布の誤差

$$E_{reg} = -\frac{1}{2} \sum_{i=1}^2 [1 + \log(\sigma_i)^2 - (\mu_i)^2 - (\sigma_i)^2]$$

■ 総合ロス

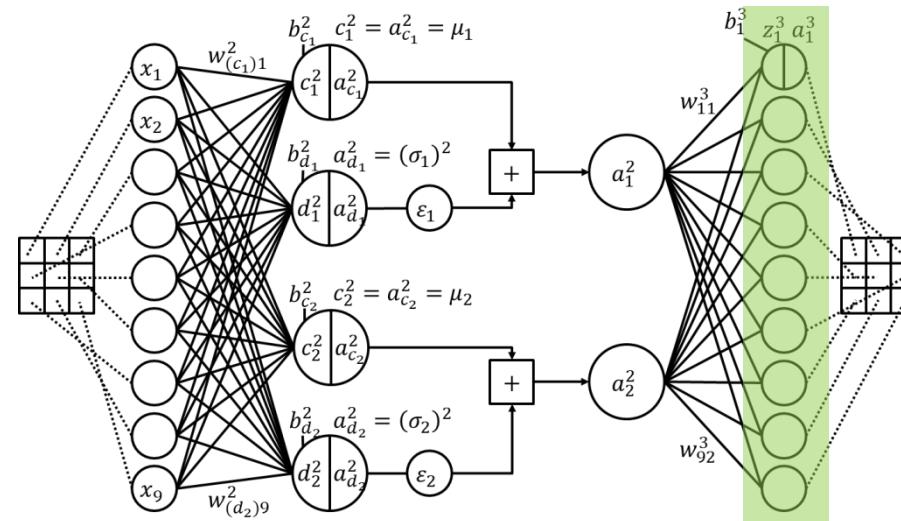
$$E = E_{rec} + E_{reg}$$

逆伝搬の流れ

$$\frac{\partial E}{\partial a_j^3} = \frac{\partial E_{rec}}{\partial a_j^3} = -\frac{x_j}{a_j^3} + \frac{1-x_j}{1-a_j^3}$$

$$\begin{aligned}\delta_j^3 &= \frac{\partial E}{\partial z_j^3} = \frac{\partial E}{\partial a_j^3} \frac{\partial a_j^3}{\partial z_j^3} \\ &= \left(-\frac{x_j}{a_j^3} + \frac{1-x_j}{1-a_j^3} \right) (1-a_j^3)a_j^3 \\ &= -x_j(1-a_j^3) + (1-x_j)a_j^3\end{aligned}$$

$$\frac{da_j^3}{dz_j^3} = (1 - f^3(z_j^3))f^3(z_j^3) = (1 - a_j^3)a_j^3 = \left(1 - \frac{1}{1 + e^{-z_j^3}}\right) \frac{1}{1 + e^{-z_j^3}}$$

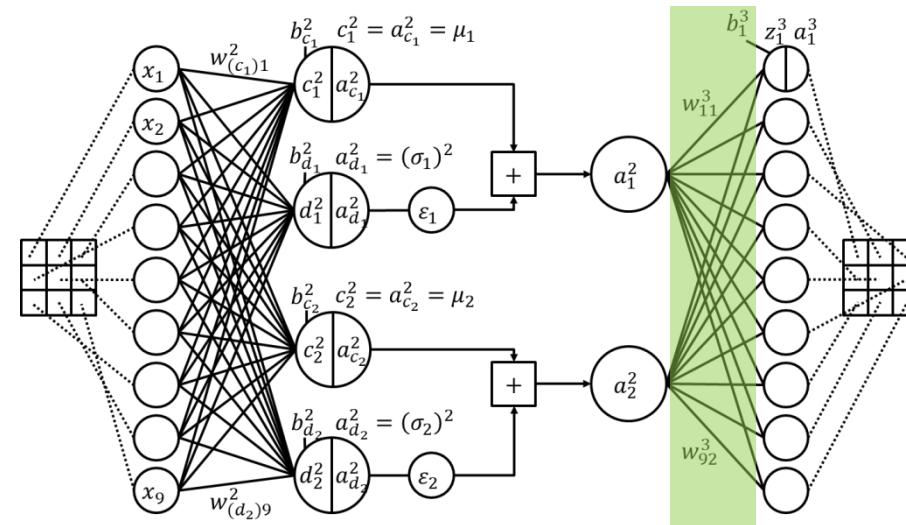


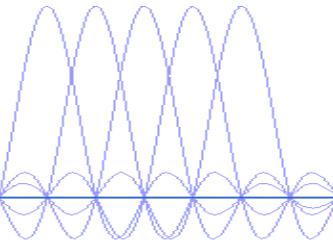
逆伝搬の流れ

$$\frac{\partial E}{\partial w_{ji}^3} = \frac{\partial E}{\partial z_j^3} \frac{\partial z_j^3}{\partial w_{ji}^3} = \delta_j^3 a_i^2$$

$$\frac{\partial E}{\partial b_j^3} = \frac{\partial E}{\partial z_j^3} \frac{\partial z_j^3}{\partial b_j^3} = \delta_j^3$$

$$\frac{\partial z_j^3}{\partial w_{ji}^3} = a_i^2, \quad \frac{\partial z_j^3}{\partial b_j^3} = 1$$

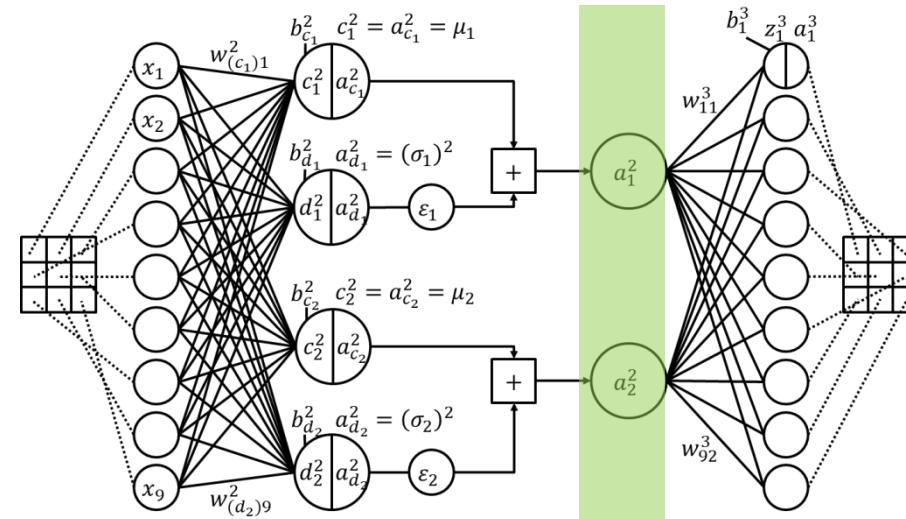




逆伝搬の流れ

$$\frac{\partial E}{\partial a_i^2} = \sum_{j=1}^9 \frac{\partial E}{\partial z_j^3} \frac{\partial z_j^3}{\partial a_i^2} = \sum_{j=1}^9 \delta_j^3 w_{ji}^3$$

$$\frac{\partial z_j^3}{\partial a_i^2} = w_{ji}^3$$

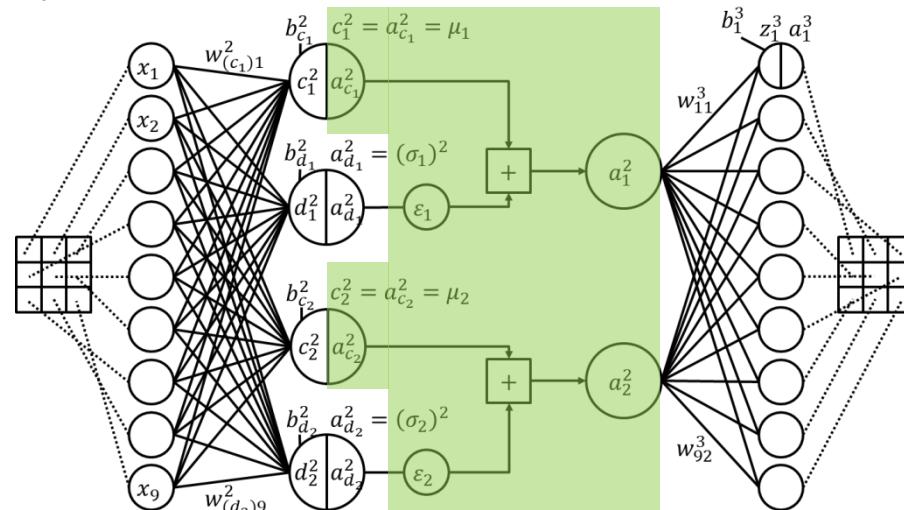


逆伝搬の流れ

$$\begin{aligned}\frac{\partial E}{\partial \mu_i^2} &= \frac{\partial E_{rec}}{\partial \mu_i^2} + \frac{\partial E_{reg}}{\partial \mu_i^2} = \frac{\partial E_{rec}}{\partial a_i^2} \frac{\partial a_i^2}{\partial a_{c_i}^2} + \frac{\partial E_{reg}}{\partial a_{c_i}^2} \\ &= \sum_{j=1}^9 [\delta_j^3 w_{ji}^3] + \mu_i\end{aligned}$$

$$\frac{\partial a_i^2}{\partial a_{c_i}^2} = \frac{\partial}{\partial a_{c_i}^2} \left[a_{c_i}^2 + \sqrt{a_{d_i}^2} \cdot \varepsilon_1 \right] = 1$$

$$\frac{\partial E_{reg}}{\partial a_{c_i}^2} = \frac{\partial}{\partial a_{c_i}^2} \left[-\frac{1}{2} \sum_{i=1}^2 \left[1 + \log a_{d_i}^2 - (a_{c_i}^2)^2 - a_{d_i}^2 \right] \right] = a_{c_i}^2 = \mu_i$$



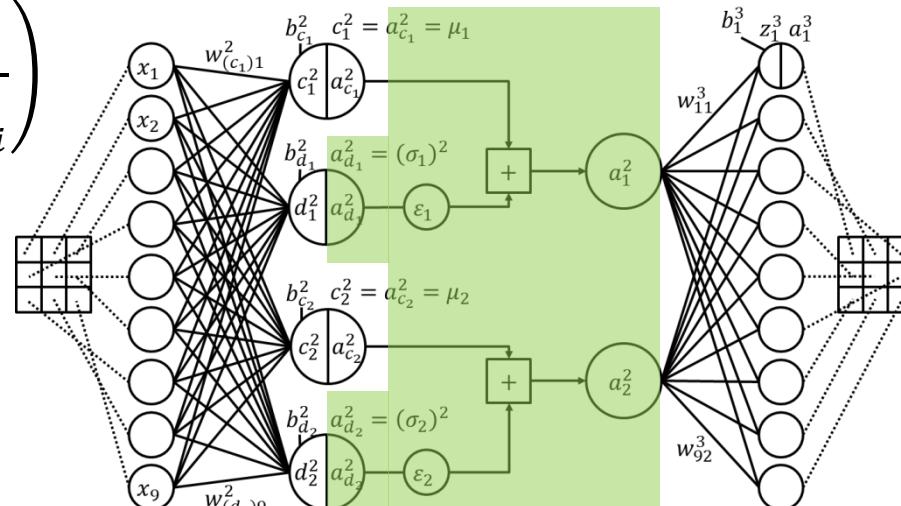
逆伝搬の流れ

$$\frac{\partial E}{\partial \sigma_i^2} = \frac{\partial E_{rec}}{\partial \sigma_i^2} + \frac{\partial E_{reg}}{\partial \sigma_i^2} = \frac{\partial E_{rec}}{\partial a_i^2} \frac{\partial a_i^2}{\partial a_{d_i}^2} + \frac{\partial E_{reg}}{\partial a_{d_i}^2}$$

$$= \sum_{j=1}^9 [\delta_j^3 w_{ji}^3] \frac{\varepsilon_i}{2\sqrt{a_{d_i}^2}} + \frac{1}{2} \left(1 - \frac{1}{a_{d_i}^2} \right)$$

$$\frac{\partial a_i^2}{\partial a_{d_i}^2} = \frac{\partial}{\partial a_{d_i}^2} [a_{c_i}^2 + \sqrt{a_{d_i}^2} \cdot \varepsilon_1] = \frac{\varepsilon_i}{2\sqrt{a_{d_i}^2}}$$

$$\frac{\partial E_{reg}}{\partial a_{d_i}^2} = \frac{\partial}{\partial a_{d_i}^2} \left[-\frac{1}{2} \sum_{i=1}^2 \left[1 + \log a_{d_i}^2 - (a_{c_i}^2)^2 - a_{d_i}^2 \right] \right] = -\frac{1}{2} \left(\frac{1}{a_{d_i}^2} - 1 \right) = \frac{1}{2} \left(1 - \frac{1}{a_{d_i}^2} \right)$$

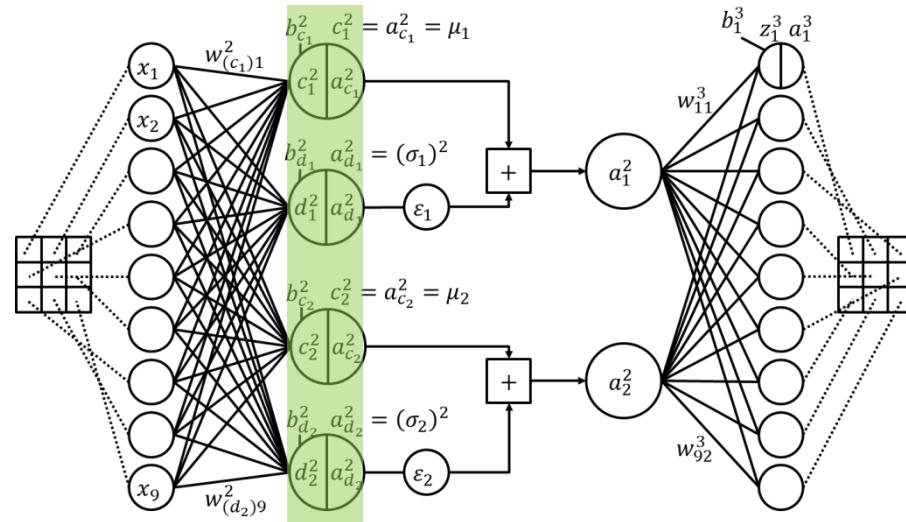


逆伝搬の流れ

$$\frac{\partial E}{\partial c_i^2} = \frac{\partial E}{\partial a_{c_i}^2}$$

$$\begin{aligned}\frac{\partial E}{\partial d_i^2} &= \frac{\partial E_{rec}}{\partial a_{d_i}^2} \frac{\partial a_{d_i}^2}{\partial d_i^2} + \frac{\partial E_{reg}}{\partial a_{d_i}^2} \frac{\partial a_{d_i}^2}{\partial d_i^2} \\ &= \sum_{j=1}^9 [\delta_j^3 w_{ji}^3] \frac{\varepsilon_i}{2\sqrt{a_{d_i}^2}} \frac{1}{1+e^{-d_i^2}} \\ &\quad + \frac{1}{2} \left(1 - \frac{1}{a_{d_i}^2}\right) \frac{1}{1+e^{-d_i^2}}\end{aligned}$$

$$\frac{\partial a_{d_i}^2}{\partial d_i^2} = \frac{e^{d_i^2}}{1+e^{d_i^2}} = \frac{1}{1+e^{-d_i^2}}$$



逆伝搬の流れ

$$\frac{\partial E}{\partial w_{(c_i)j}^2} = \frac{\partial E_{rec}}{\partial c_i^2} \frac{\partial c_i^2}{\partial w_{(c_i)j}^2} + \frac{\partial E_{reg}}{\partial c_i^2} \frac{\partial c_i^2}{\partial w_{(c_i)j}^2}$$

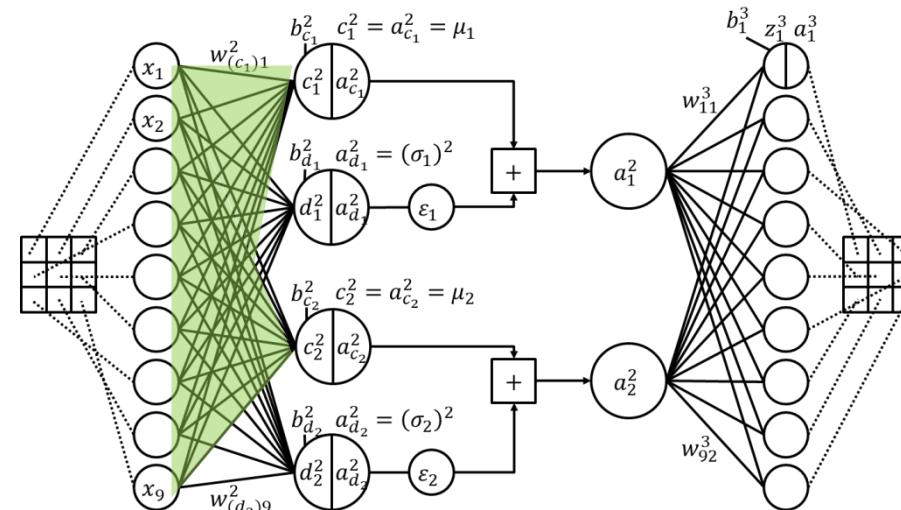
$$= \sum_{j=1}^9 [\delta_j^3 w_{ji}^3] x_j + \mu_i x_j$$

$$\frac{\partial E}{\partial b_{c_i}^2} = \frac{\partial E_{rec}}{\partial c_i^2} \frac{\partial c_i^2}{\partial b_{c_i}^2} + \frac{\partial E_{reg}}{\partial c_i^2} \frac{\partial c_i^2}{\partial b_{c_i}^2}$$

$$= \sum_{j=1}^9 [\delta_j^3 w_{ji}^3] + \mu_i$$

$$\frac{\partial c_i^2}{\partial w_{(c_i)j}^2} = \frac{\partial}{\partial w_{(c_i)j}^2} \left[\sum_{j=1}^9 [w_{(c_i)j}^2 x_j + b_{c_i}^2] \right] = x_j$$

$$\frac{\partial c_i^2}{\partial b_{c_i}^2} = \frac{\partial}{\partial b_{c_i}^2} \left[\sum_{j=1}^9 [w_{(c_i)j}^2 x_j + b_{c_i}^2] \right] = 1$$



逆伝搬の流れ

$$\frac{\partial E}{\partial w_{(d_i)j}^2} = \frac{\partial E_{rec}}{\partial d_i^2} \frac{\partial d_i^2}{\partial w_{(d_i)j}^2} + \frac{\partial E_{reg}}{\partial d_i^2} \frac{\partial d_i^2}{\partial w_{(d_i)j}^2}$$

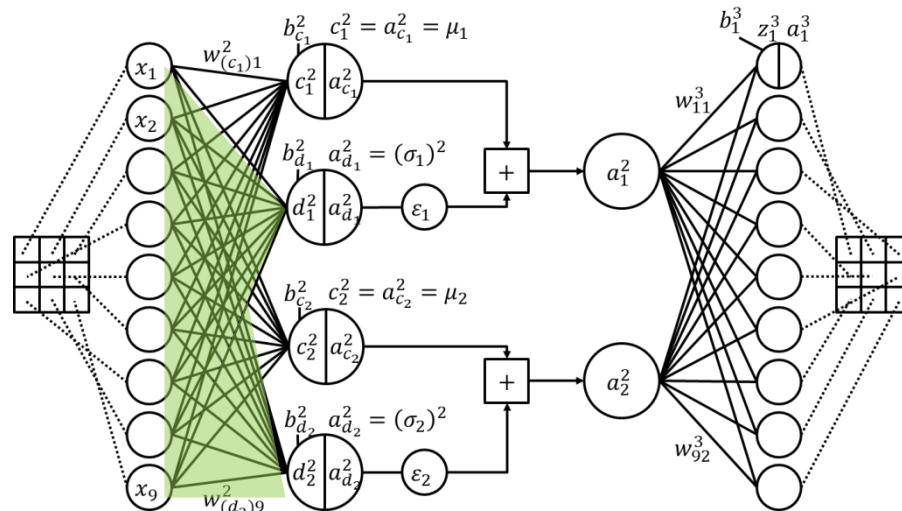
$$= \sum_{j=1}^9 [\delta_j^3 w_{ji}^3] \frac{\varepsilon_i}{2\sqrt{a_{d_i}^2}} \frac{x_j}{1 + e^{-d_i^2}}$$

$$+ \frac{1}{2} \left(1 - \frac{1}{a_{d_i}^2} \right) \frac{x_j}{1 + e^{-d_i^2}}$$

$$\frac{\partial E}{\partial b_{d_i}^2} = \frac{\partial E_{rec}}{\partial d_i^2} \frac{\partial d_i^2}{\partial b_{d_i}^2} + \frac{\partial E_{reg}}{\partial d_i^2} \frac{\partial d_i^2}{\partial b_{d_i}^2}$$

$$= \sum_{j=1}^9 [\delta_j^3 w_{ji}^3] \frac{\varepsilon_i}{2\sqrt{a_{d_i}^2}} \frac{1}{1 + e^{-d_i^2}}$$

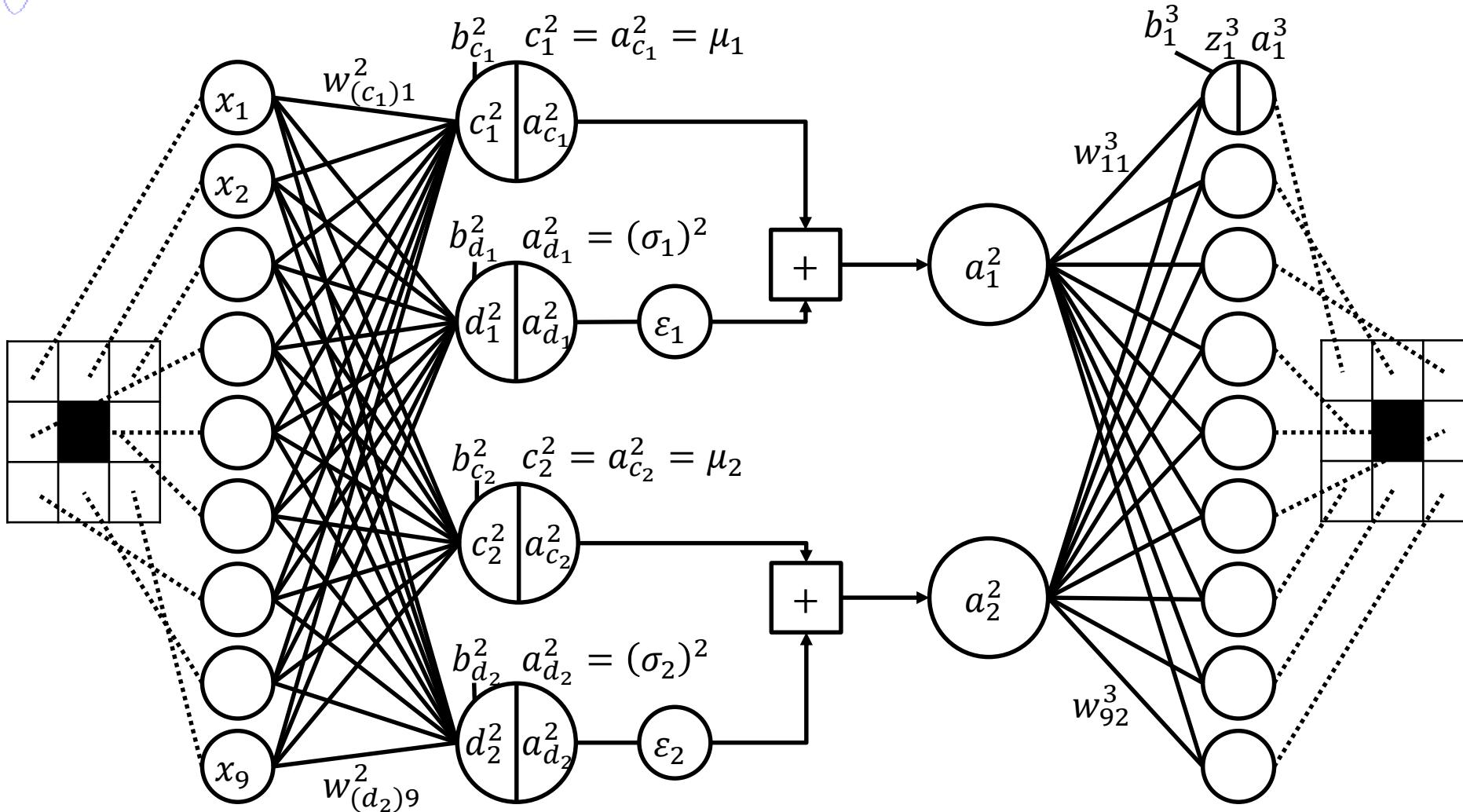
$$+ \frac{1}{2} \left(1 - \frac{1}{a_{d_i}^2} \right) \frac{1}{1 + e^{-d_i^2}}$$

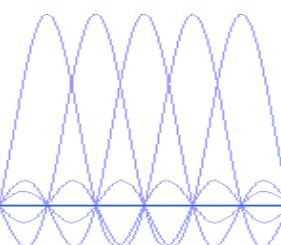


$$\frac{\partial d_i^2}{\partial w_{(d_i)j}^2} = \frac{\partial}{\partial w_{(d_i)j}^2} \left[\sum_{j=1}^9 [w_{(d_i)j}^2 x_j + b_{d_i}^2] \right] = x_j$$

$$\frac{\partial d_i^2}{\partial b_{d_i}^2} = \frac{\partial}{\partial b_{d_i}^2} \left[\sum_{j=1}^9 [w_{(d_i)j}^2 x_j + b_{d_i}^2] \right] = 1$$

VAEの例 (○の場合)





w と b の初期値(rng(2025);)

w2_mean (w_c^2)

0.1355	0.9326	0.3882	0.6574	0.9642	0.4552	0.0417	0.0032	0.6109
0.8879	0.4456	0.2576	0.4926	0.8010	0.8011	0.7695	0.2928	0.9130

b2_mean (b_c^2)

-0.5000
-0.5000

w2_var (w_d^2)

0.3001	0.6664	0.4683	0.9160	0.2777	0.1547	0.3242	0.5131	0.0674
0.2486	0.9875	0.1233	0.9461	0.5197	0.0146	0.9909	0.8765	0.2842

b2_var (b_d^2)

-0.5000
-0.5000

w3 (w^3)

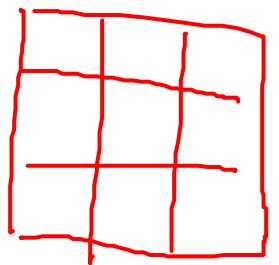
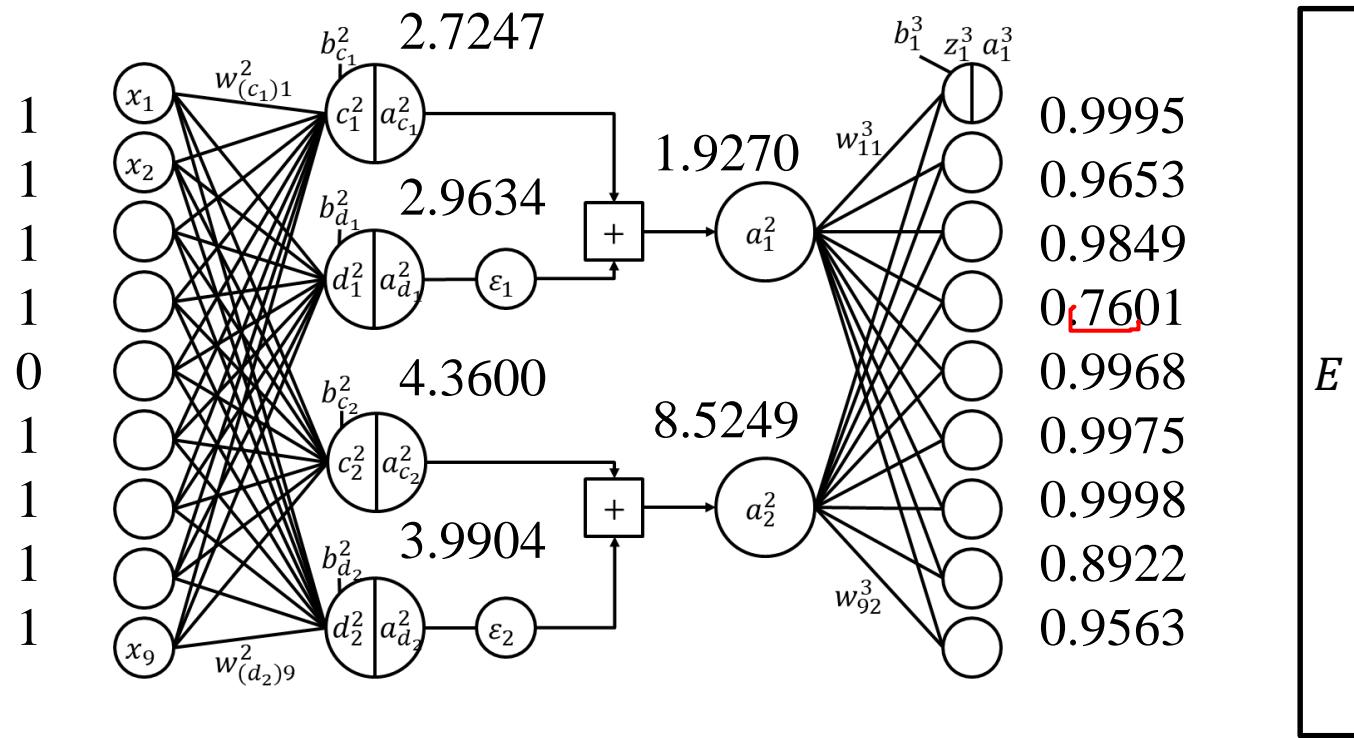
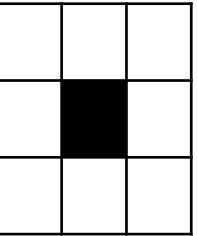
0.4689	0.8479
0.7618	0.2767
0.9226	0.3402
0.3930	0.1051
0.9291	0.5231
0.4996	0.6485
0.8020	0.8996
0.8966	0.1039
0.4821	0.3117

b3 (b^3)

-0.5000
-0.5000
-0.5000
-0.5000
-0.5000
-0.5000
-0.5000
-0.5000
-0.5000
-0.5000

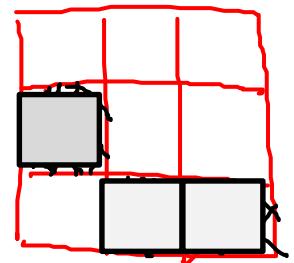
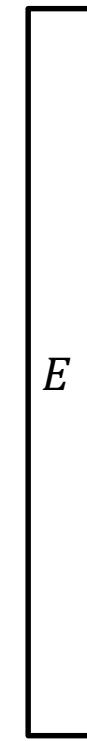
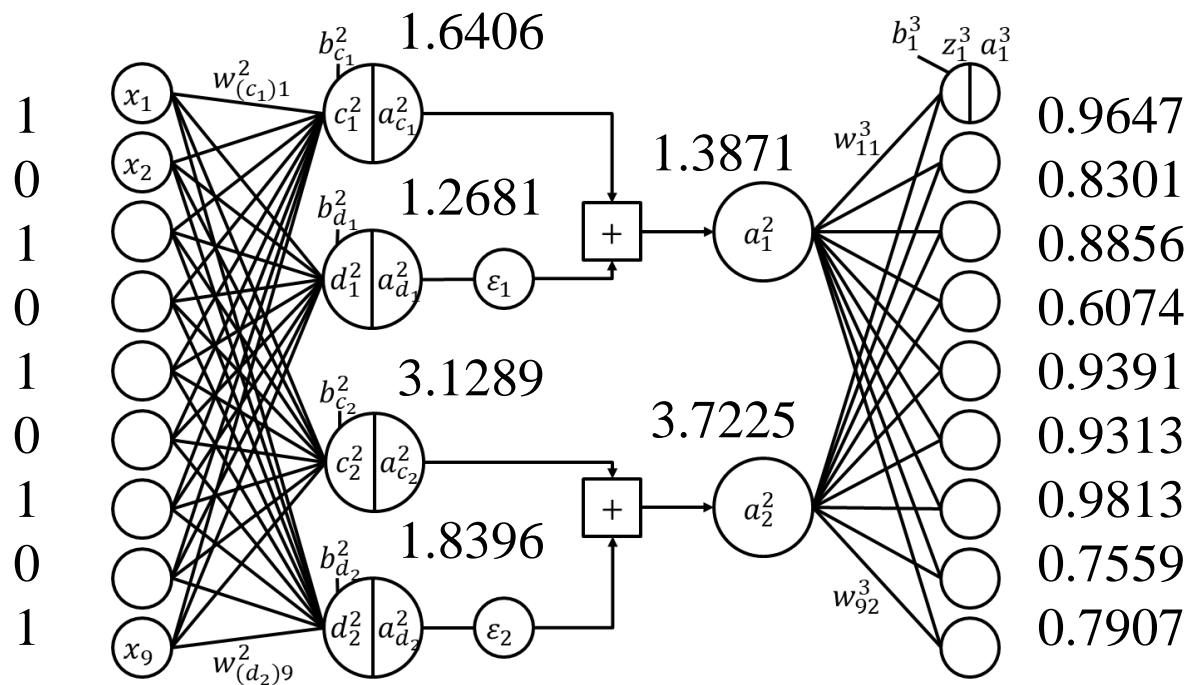
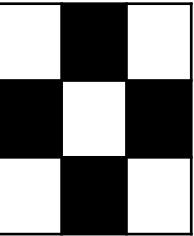
VAEの例 (○の場合, 初期値)

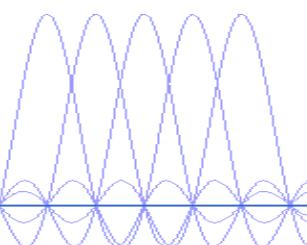
白:1
黒:0



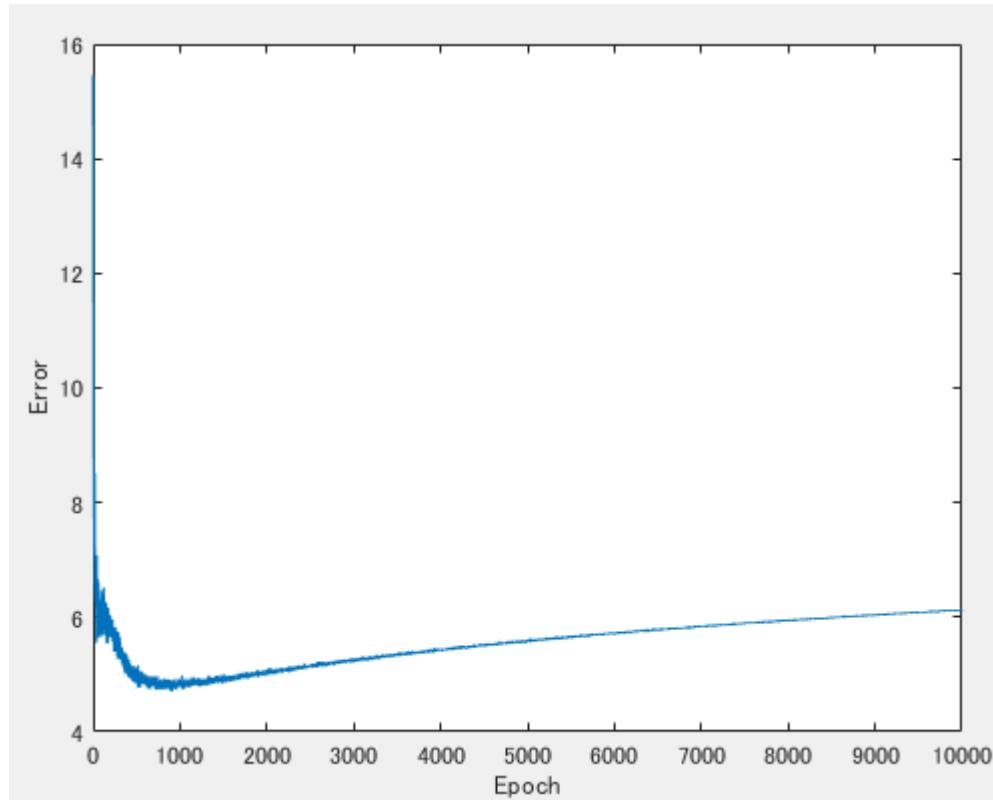
VAEの例（×の場合、初期値）

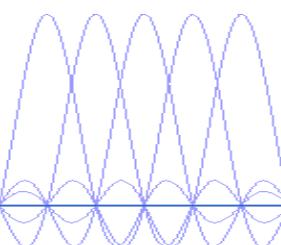
白:1
黒:0





誤差関数の履歴





10000回学習後の w と b

w2_mean (w_c^2)

-0.0004	-0.0028	-0.0004	-0.0012	0.0024	-0.0036	-0.1861	1.2185	0.3831
-0.0034	0.0038	-0.0032	0.0034	-0.0056	0.0036	0.4888	-0.4737	0.6324

w2_var (w_d^2)

-0.9178	-0.1435	-0.7811	0.0728	-0.4137	-0.5869	-1.1672	-0.3001	-1.4240
-1.1124	0.0357	-1.2106	-0.0005	-0.2634	-0.8148	-0.6815	-0.0534	-1.3883

w3 (w^3)

1.2638	2.7963
5.6302	-1.7521
1.2328	2.6412
5.5842	-1.8523
-5.2076	2.7393
5.6714	-1.6399
1.3139	2.8903
5.6133	-1.8008
1.1772	2.5499

b2_mean (b_c^2)

-0.7278
-0.7807

b2_var (b_d^2)

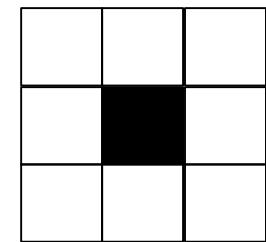
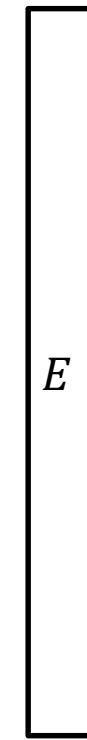
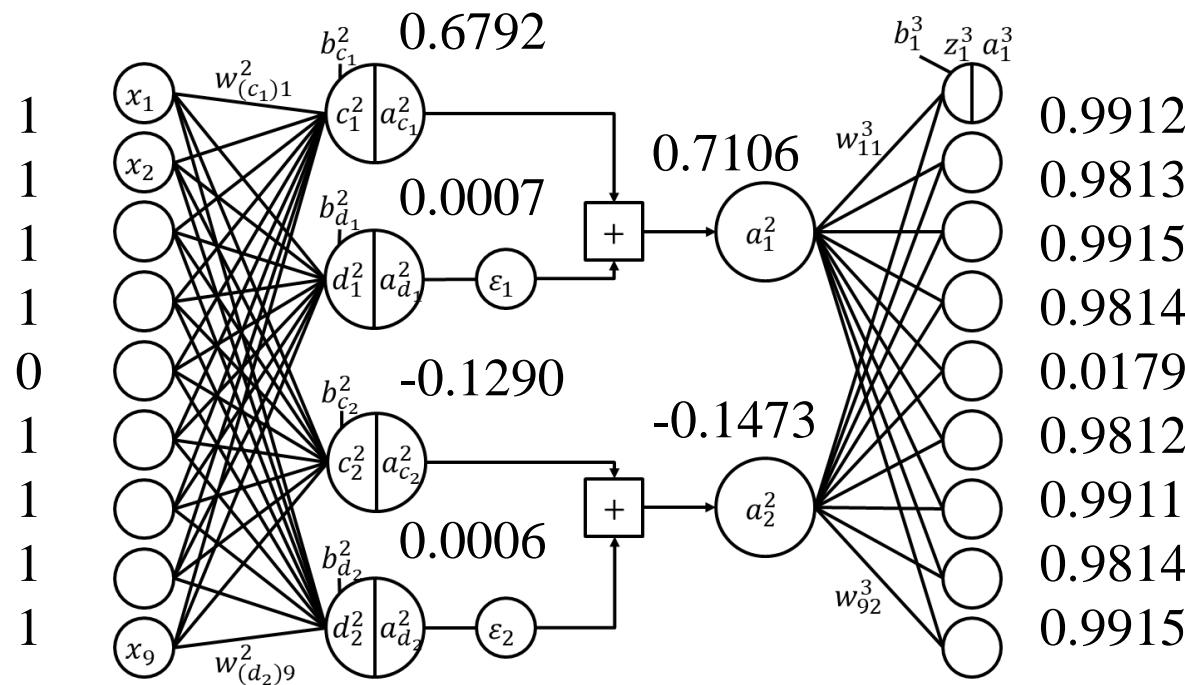
-1.9914
-2.1724

b3 (b^3)

4.2329
-0.2964
4.2722
-0.2755
0.0995
-0.3173
4.2099
-0.2876
4.2941

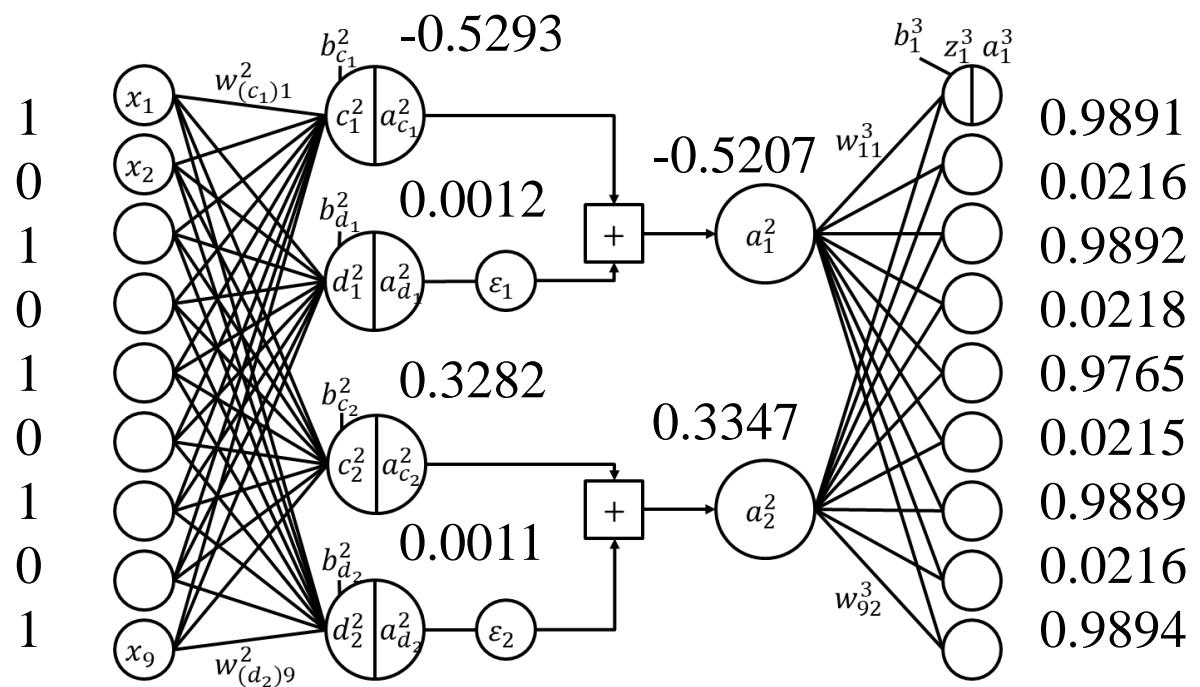
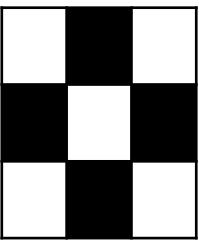
VAEの例 (○の場合, 10000回学習後)

白:1
黒:0



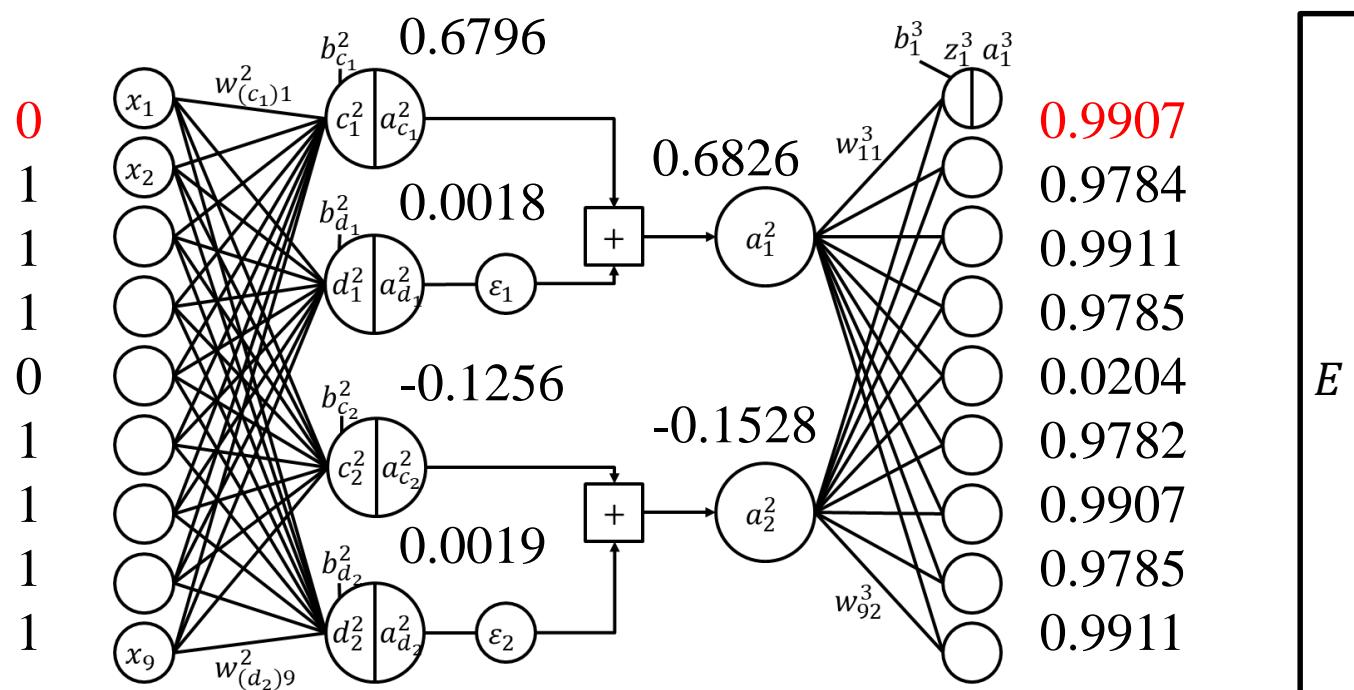
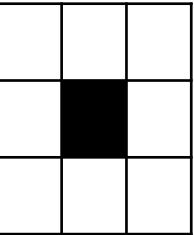
VAEの例（×の場合、10000回学習後）

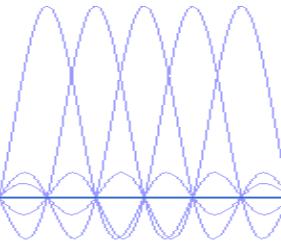
白:1
黒:0



1画素誤りがあった場合の出力

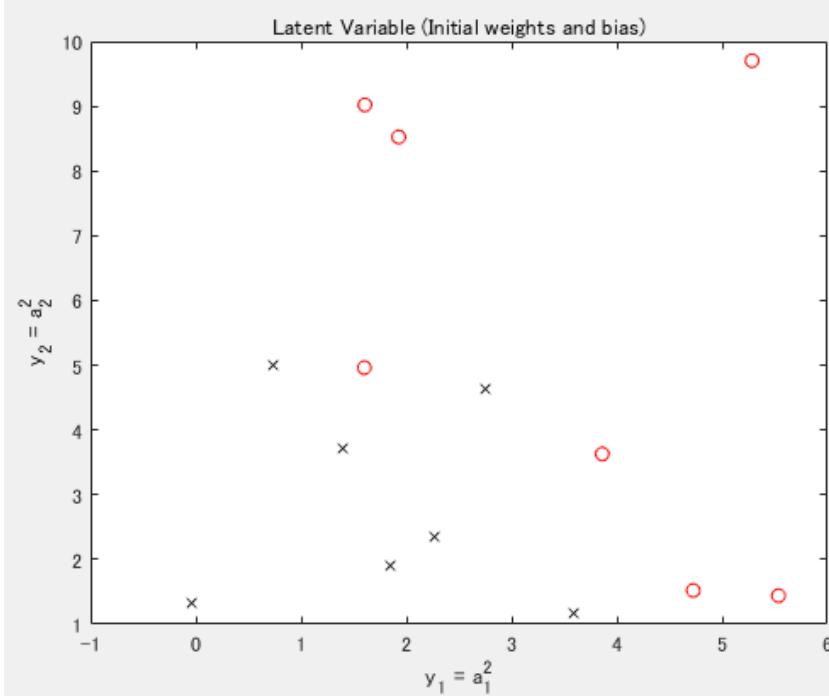
白:1
黒:0



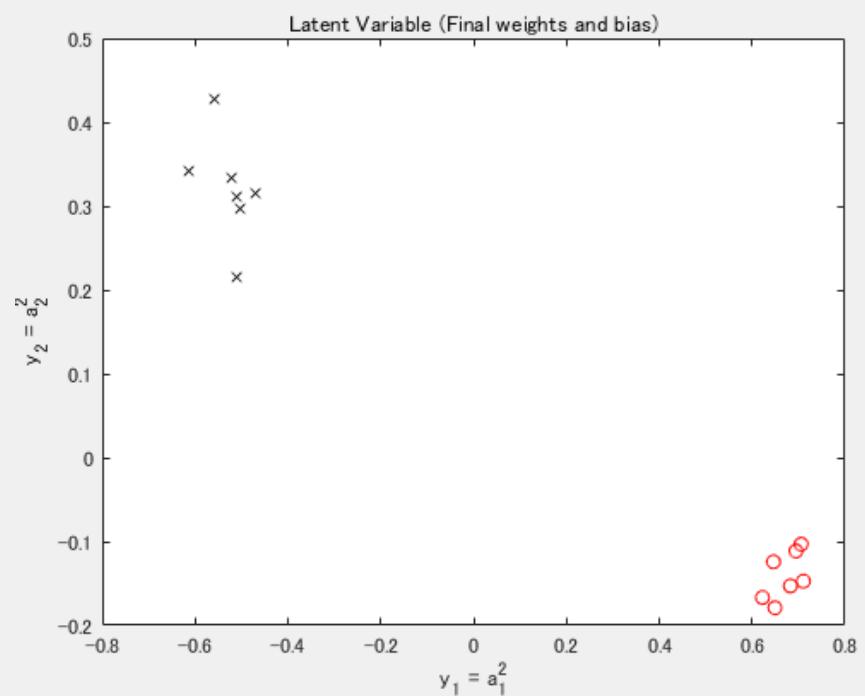


潜在変数 (1画素誤り6パターン)

Initial parameters



Final parameters





応用実習1:VAE実習

- 作業フォルダ: ./VAE
- rng(2024)を変更し、どのように変化するか確認せよ。
- 2ビット誤りがあった場合、どうなるか確認せよ。
- zをそれぞれ定義し、画像を出力せよ。

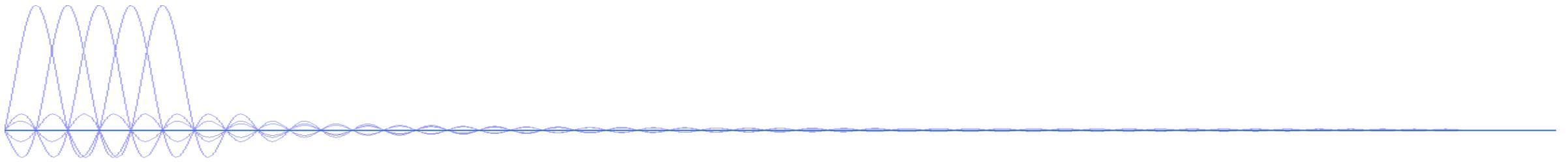


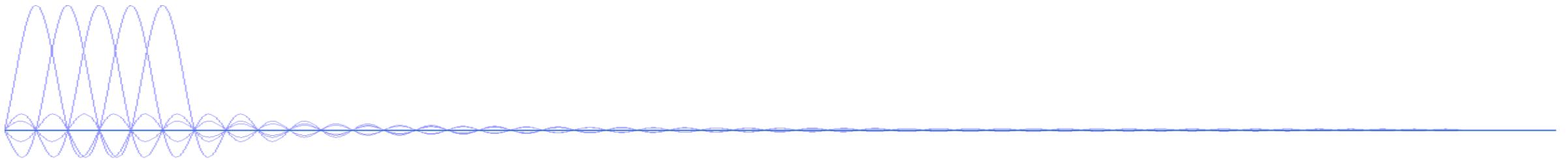
応用実習2:AE/VAE実習

■ 作業フォルダ: .\AE_MNIST

■ 手書き文字を学習させてみよう

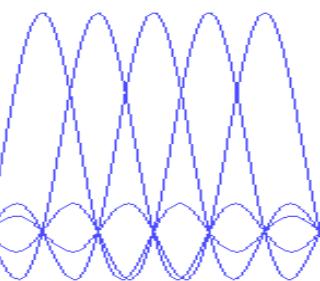
□MNIST_AEやMNIST_VAEを動作させて、手書き文字の学習を行い、AEとVAEとの結果の違いについて考察して、レポートのまとめよう。



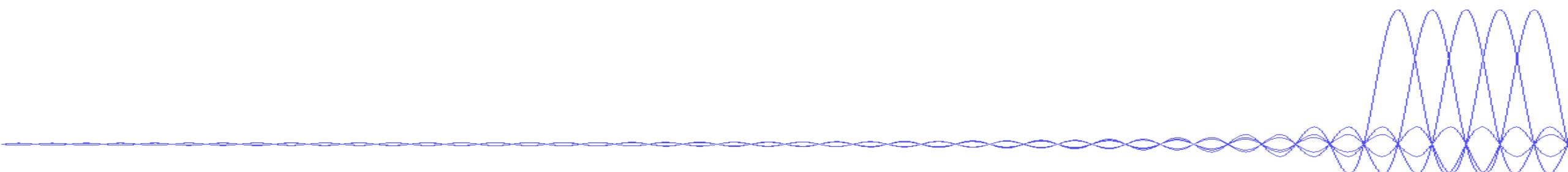


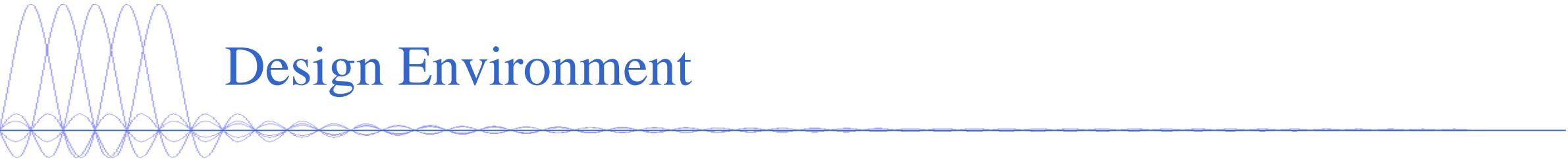
基本的なCPUシステムの作成と実証

作業フォルダ: .\common_sys



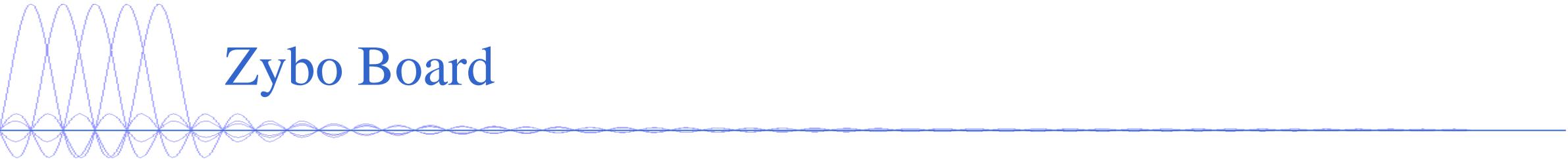
FPGAボード(ZyboZ7-10)の説明





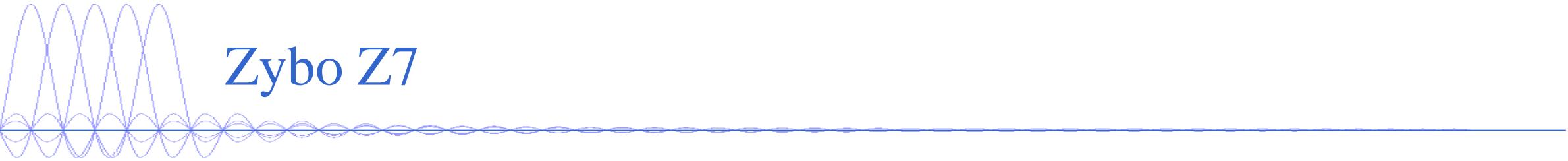
Design Environment

- Board :Zybo / Zybo Z7 (Z7-020)
- FPGA
 - Zynq-7000 XC7Z010-1CLG400C (Zybo)
 - Zynq-7000 XC7Z020-1CLG400C (Zybo Z7)
- 512MB DDR3
- MicroSDカード
- Port for display
 - 1080p HDMI
 - 8-bit VGA (only Zybo)
- Software
 - OS : Windows 10
 - MATLAB/Simulink R2017b
 - VIVADO 2018.3/System Generator 2018.3



Zybo Board

- 28,000 logic cells
- 240 KB Block RAM
- 80 DSP slices
- On-chip dual channel, 12-bit, 1 MSPS analog-to-digital converter (XADC)
- 650 MHz dual-core Cortex™-A9 processor
- On-board JTAG programming and UART to USB converter
- DDR3 memory controller with 8 DMA channels
- 512 MB x32 DDR3 w/ 1050Mbps bandwidth
- 128 Mb Serial Flash w/ QSPI interface
- microSD slot (supports Linux file system)
- High-bandwidth peripheral controllers: 1G Ethernet, USB 2.0, SDIO
- Low-bandwidth peripheral controller: SPI, UART, I2C
- Dual-role (Source/Sink) HDMI port
- 16-bits per pixel VGA output port
- Trimode (1Gbit/100Mbit/10Mbit) Ethernet PHY
- OTG USB 2.0 PHY (supports host and device)
- External EEPROM (programmed with 48-bit globally unique EUI-48/64™ compatible identifier)
- Audio codec with headphone out, microphone and line in jacks
- GPIO: 6 pushbuttons, 4 slide switches, 5 LEDs
- Six Pmod ports (1 processor-dedicated, 1 dual analog/digital)



Zybo Z7

•ZYNQ Processor

- 667 MHz dual-core Cortex-A9 processor
- DDR3L memory controller with 8 DMA channels and 4 High Performance AXI3 Slave ports
- High-bandwidth peripheral controllers: 1G Ethernet, USB 2.0, SDIO
- Low-bandwidth peripheral controllers: SPI, UART, CAN, I2C
- Programmable from JTAG, Quad-SPI flash, and microSD card
- Programmable logic equivalent to Artix-7 FPGA

•Memory

- 1 GB DDR3L with 32-bit bus @ 1066 MHz
- 16 MB Quad-SPI Flash with factory programmed 128-bit random number and 48-bit globally unique EUI-48/64™ compatible identifier
- microSD slot

•Power

- Powered from USB or any 5V external power source

•USB and Ethernet

- Gigabit Ethernet PHY
- USB-JTAG Programming circuitry
- USB-UART bridge
- USB 2.0 OTG PHY with host and device support

•Audio and Video

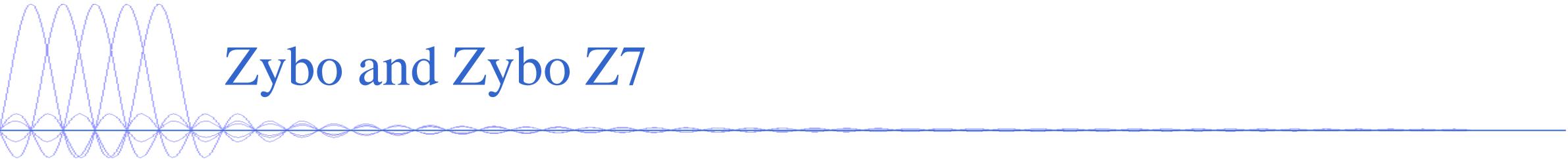
- Pcam camera connector with MIPI CSI-2 support
- HDMI sink port (input) with/without* CEC
- HDMI source port (output) with CEC
- Audio codec with stereo headphone, stereo line-in, and microphone jacks

•Switches, Push-buttons, and LEDs

- 6 push-buttons (2 processor connected)
- 4 slide switches
- 5 LEDs (1 processor connected)
- 2 RGB LEDs (1*)

•Expansion Connectors

- 6 Pmod ports (5*)
 - 8 Total Processor I/O
 - 40 Total FPGA I/O (32*)
 - 4 Analog capable 0-1.0V differential pairs to XADC



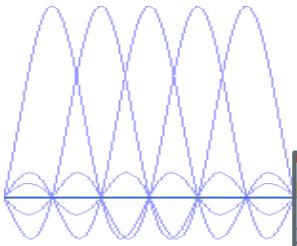
Zybo and Zybo Z7

- Zybo

- <https://reference.digilentinc.com/reference/programmable-logic/zybo/start>

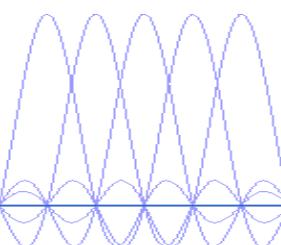
- Zybo Z7

- <https://reference.digilentinc.com/reference/programmable-logic/zybo-z7/start>



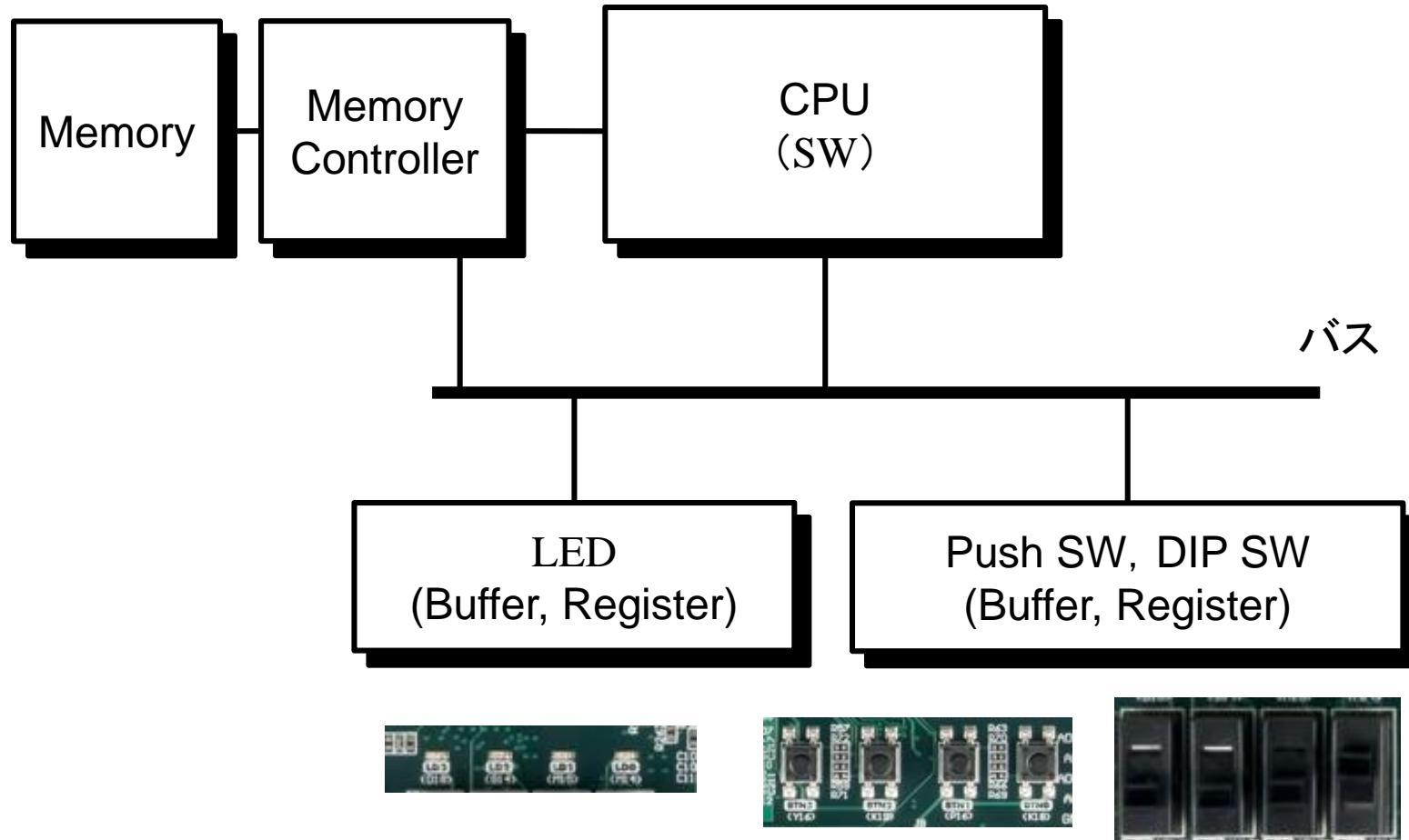
Zybo Z7-10 and Zybo Z7-20

Product Variant	Zybo Z7-10	Zybo Z7-20
Zynq Part	XC7Z010-1CLG400C	XC7Z020-1CLG400C
1 MSPS On-chip ADC	Yes	Yes
Look-up Tables (LUTs)	17,600	53,200
Flip-Flops	35,200	106,400
Block RAM	270 KB	630 KB
Clock Management Tiles	2	4
Total Pmod ports	5	6
Fan connector	No	Yes
Zynq heat sink	No	Yes
HDMI CEC Support	TX port only	TX and RX ports
RGB LED	1	2

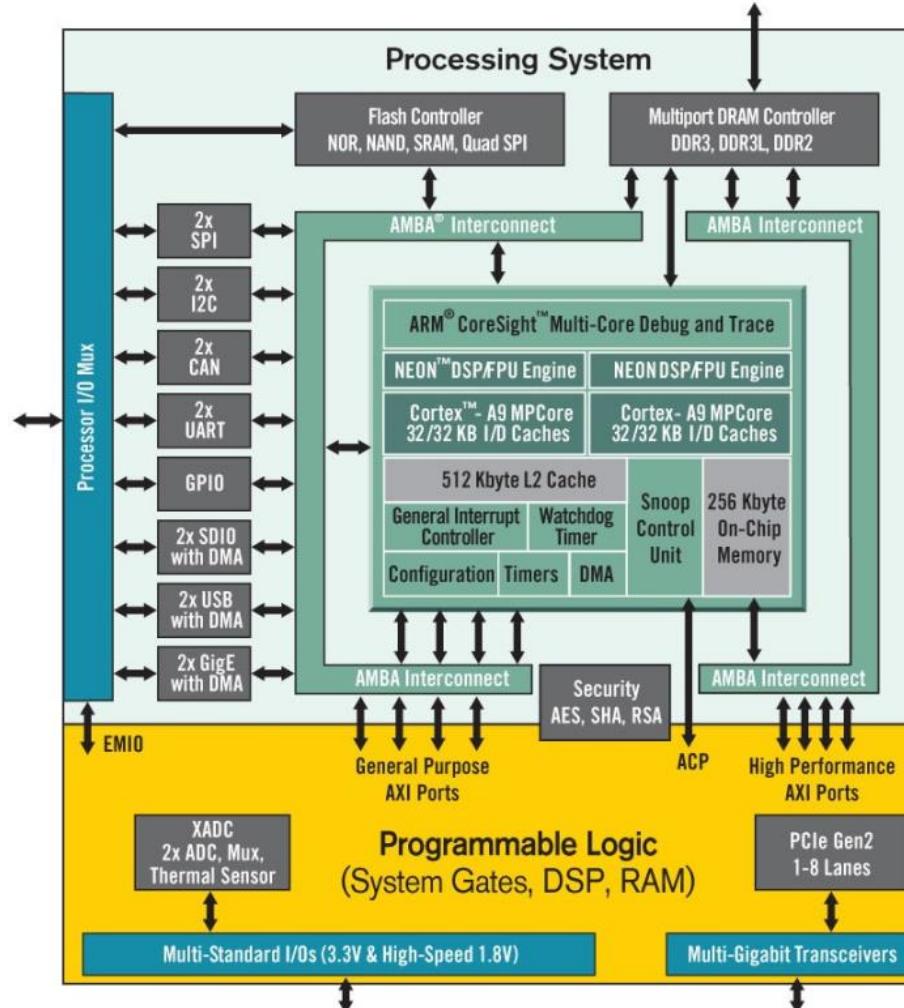


HW/SW Co-design Basic Structure

■ CPU(Software) and Peripheral (Hardware)

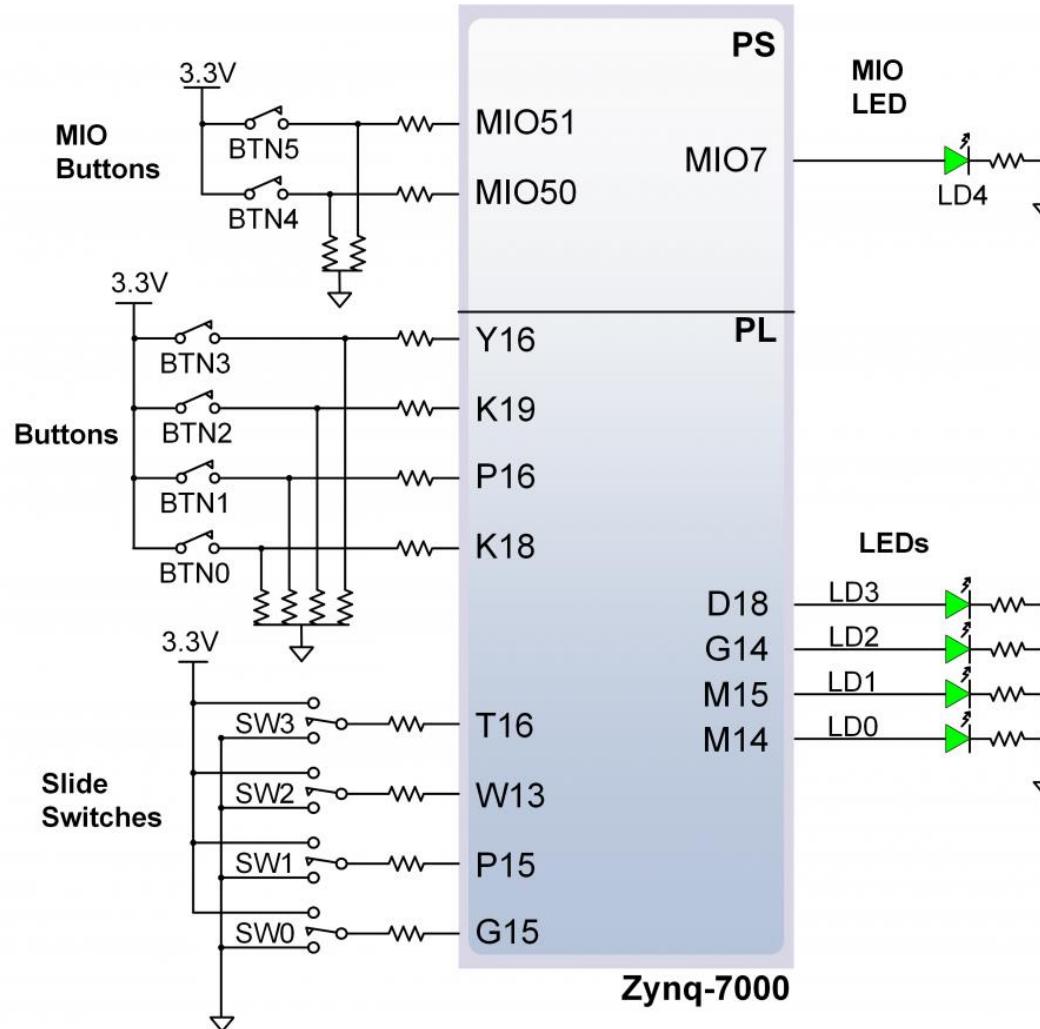


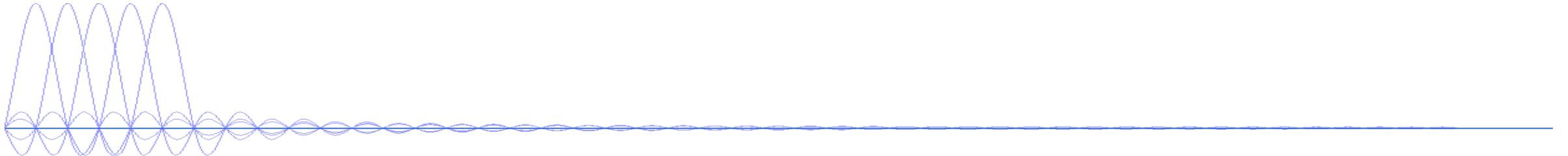
Processing System and Programmable Logic



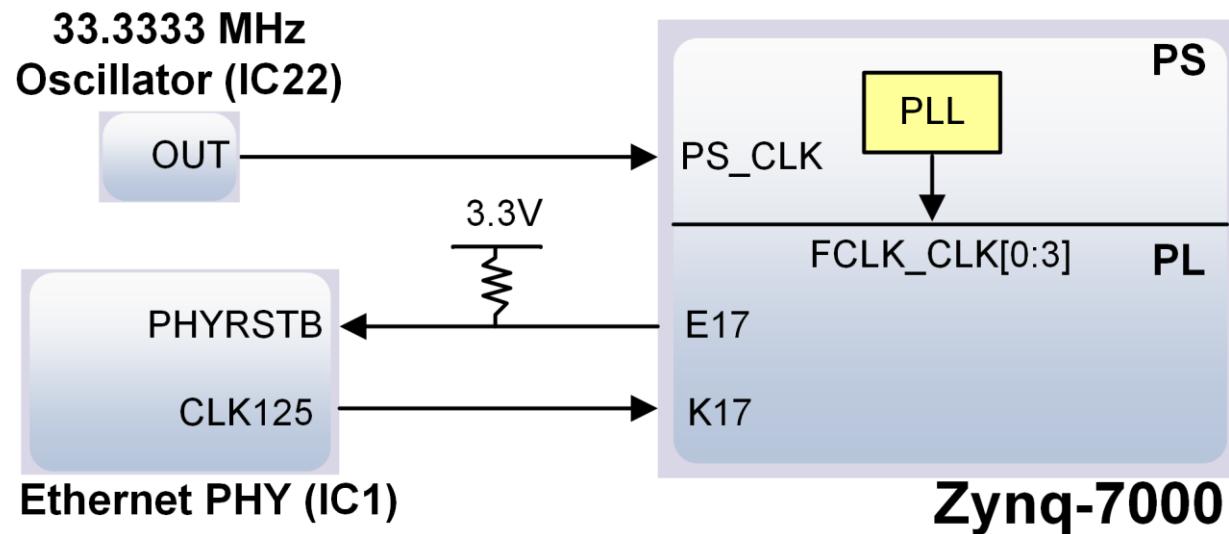
Zynq-7000 All Programmable SoC : <http://japan.xilinx.com/content/xilinx/ja/products/silicon-devices/soc/zynq-7000.html>

HW/SW Co-design Basic Structure (Buttons, Switches, and LEDs)





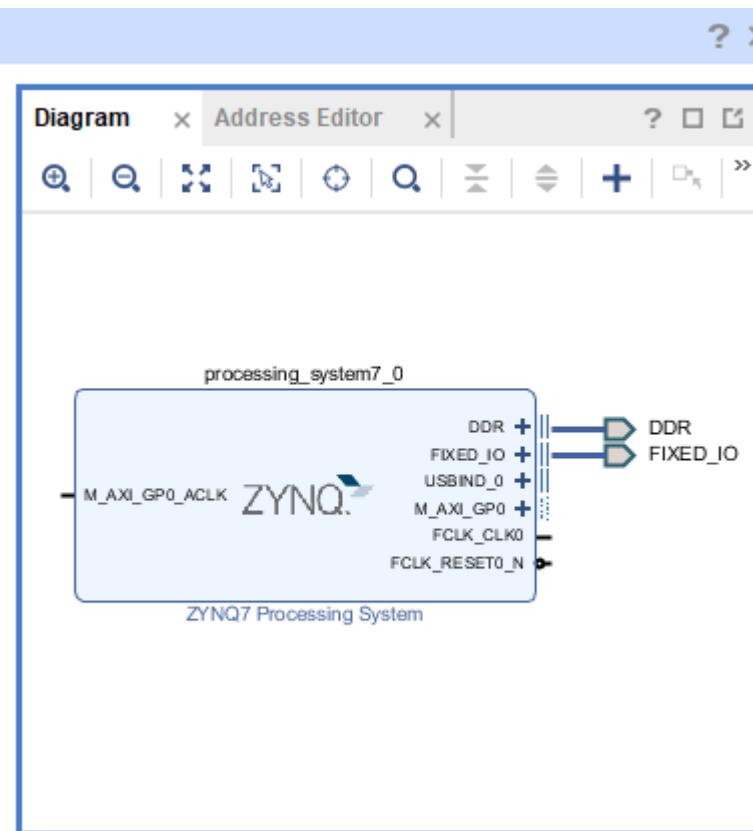
- 125 MHz clock (K17) が直接利用可能
 - E17をLowにするとクロック共有が停止することに注意





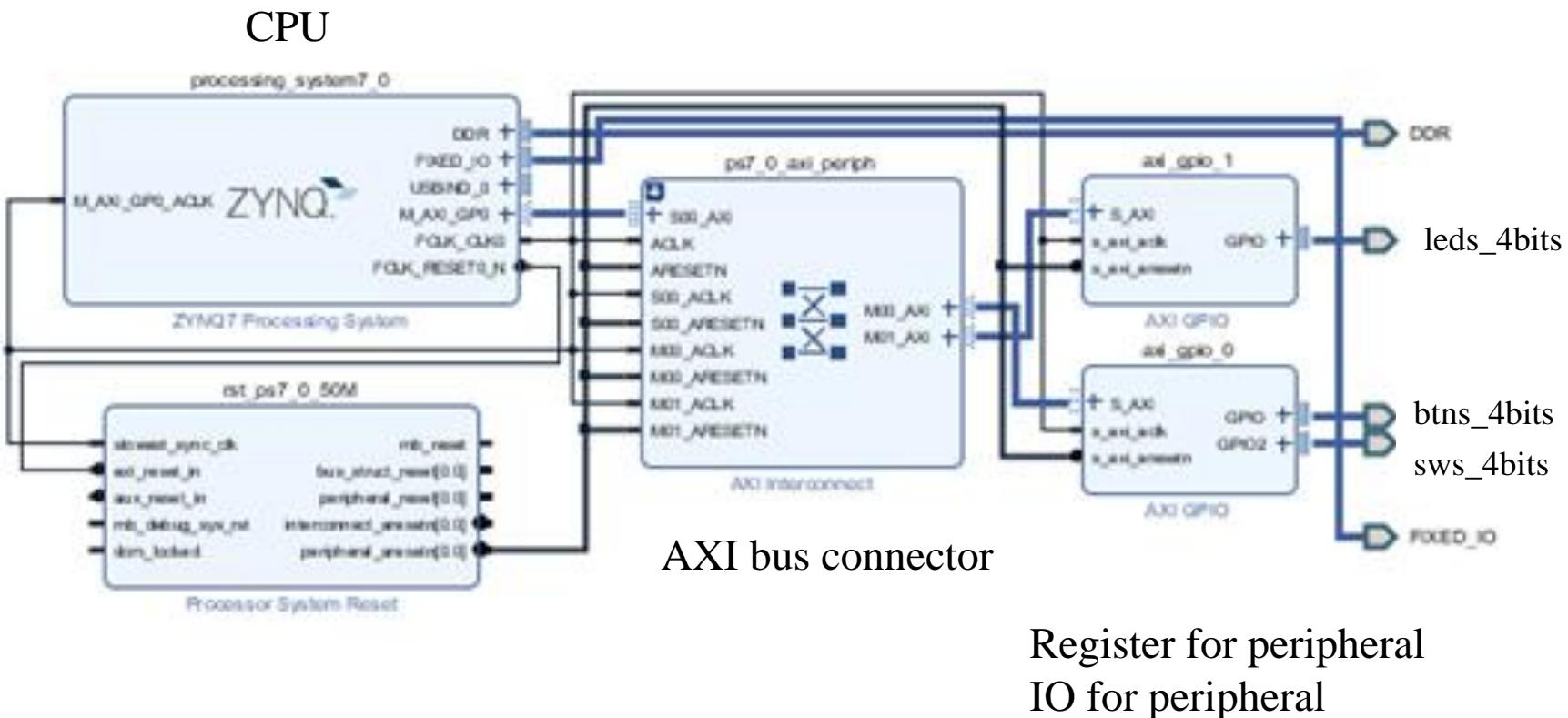
基本システムの構築 First Step

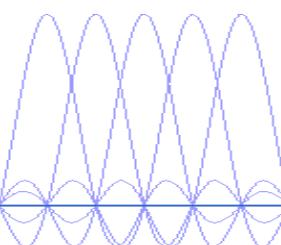
■ CPUを配置



基本システムの構築 Second Step

■ バスや周辺回路を配置





基本システムの構築 Third Step

■ SWやボタンを読み込んだり、LEDを光らせるプログラムの作成

```
XGpio_Initialize(&input, XPAR_AXI_GPIO_0_DEVICE_ID); //initialize input XGpio variable  
XGpio_SetDataDirection(&input, 1, 0xF); //set first channel tristate buffer to input  
XGpio_SetDataDirection(&input, 2, 0xF); //set second channel tristate buffer to input  
  
button_data = XGpio_DiscreteRead(&input, 1); //get button data  
switch_data = XGpio_DiscreteRead(&input, 2); //get switch data
```

xparameters.h

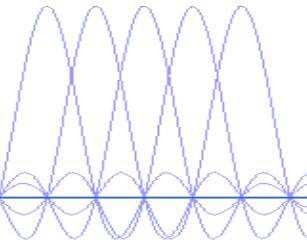
0x4120_0000



Button data

```
/* Definitions for peripheral AXI_GPIO_0 */  
#define XPAR_AXI_GPIO_0_BASEADDR 0x41200000  
#define XPAR_AXI_GPIO_0_HIGHADDR 0x4120FFFF  
#define XPAR_AXI_GPIO_0_DEVICE_ID 0  
#define XPAR_AXI_GPIO_0_INTERRUPT_PRESENT 0  
#define XPAR_AXI_GPIO_0_IS_DUAL 1
```

Button0

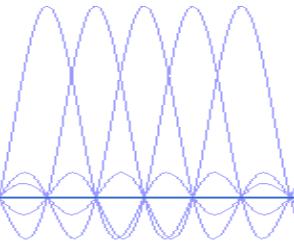


基本システムの構築

■ ボタンからのデータ読み込み

```
XGpio_Initialize(&input, XPAR_AXI_GPIO_0_DEVICE_ID);      //initialize input XGpio variable  
  
XGpio_SetDataDirection(&input, 1, 0xF);                  //set first channel tristate buffer to input  
XGpio_SetDataDirection(&input, 2, 0xF);                  //set second channel tristate buffer to input  
  
switch_data = XGpio_DiscreteRead(&input, 2);           //get switch data  
button_data = XGpio_DiscreteRead(&input, 1);           //get button data
```

⇒ switch_data = *((volatile unsigned int*) (XPAR_GPIO_0_BASEADDR + 0x04));
button_data = *((volatile unsigned int*) (XPAR_GPIO_0_BASEADDR + 0x00));



基本システムの構築 Third Step

```
XGpio_Initialize(&output, XPAR_AXI_GPIO_1_DEVICE_ID); //initialize output XGpio variable  
XGpio_SetDataDirection(&output, 1, 0x0); //set first channel tristate buffer to output
```

0x4121_0000

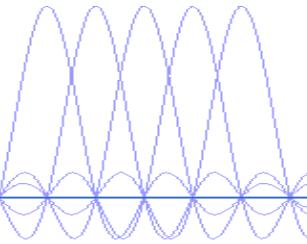


LED data

xparameters.h

LED0

```
/* Definitions for peripheral AXI_GPIO_1 */  
#define XPAR_AXI_GPIO_1_BASEADDR 0x41210000  
#define XPAR_AXI_GPIO_1_HIGHADDR 0x4121FFFF  
#define XPAR_AXI_GPIO_1_DEVICE_ID 1  
#define XPAR_AXI_GPIO_1_INTERRUPT_PRESENT 0  
#define XPAR_AXI_GPIO_1_IS_DUAL 0
```

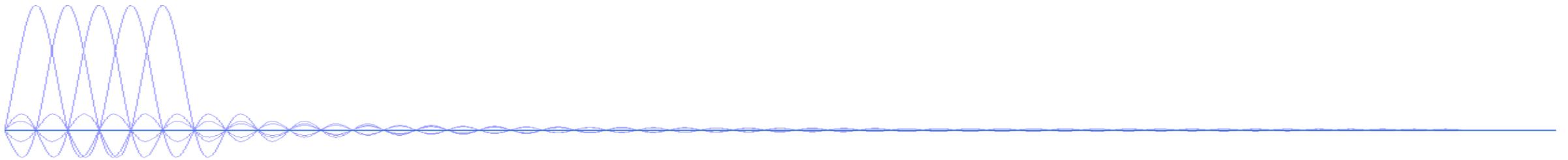


基本システムの構築

■ LED control (Ex.)

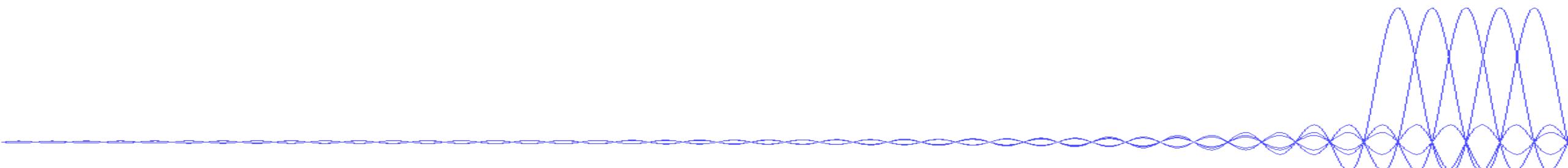
```
XGpio_Initialize(&output, XPAR_AXI_GPIO_1_DEVICE_ID); //initialize output XGpio variable  
  
XGpio_SetDataDirection(&output, 1, 0x0); //set first channel tristate buffer to output  
  
XGpio_DiscreteWrite(&output, 1, switch_data);
```

⇒ $\ast((\text{volatile unsigned int} \ast)(\text{XPAR_GPIO_1_BASEADDR} + 0x00)) = \text{switch_data};$



CPU, HW, SW実装実習1

Platform Tutorial



■ 簡単なCPUシステムを作成して、プログラミングになれる

作業フォルダ：./common_sys

■ 使用機器

Zybo Z7-10 / Zybo Z7-20

USB

■ ジャンパ

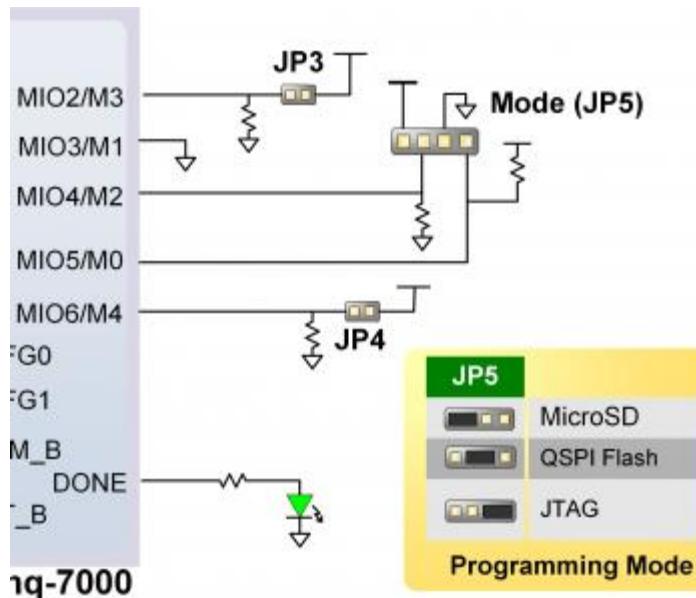
JTAG用に設定

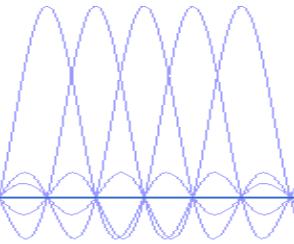


Zybo Boardのセットアップ

■ ジャンパーピンの設定

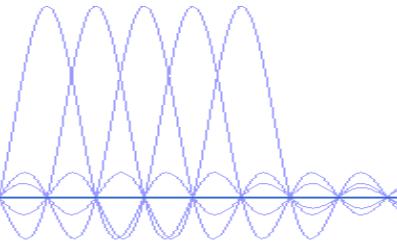
- JP5 : JTAG
- JP6 : USB (or WALL)





Platform Tutorial

- VIVADO ver 2022.02
- ZYBO Z7-010 / Zybo Z7-20
- USB cable (USB2.0 Type A - Micro B)
- Tera term



VIVADOのインストールとセットアップ

Install and Setup Vivado

■ VIVADOとDigilent Board Filesのインストール

- <https://reference.digilentinc.com/vivado/installing-vivado/start>

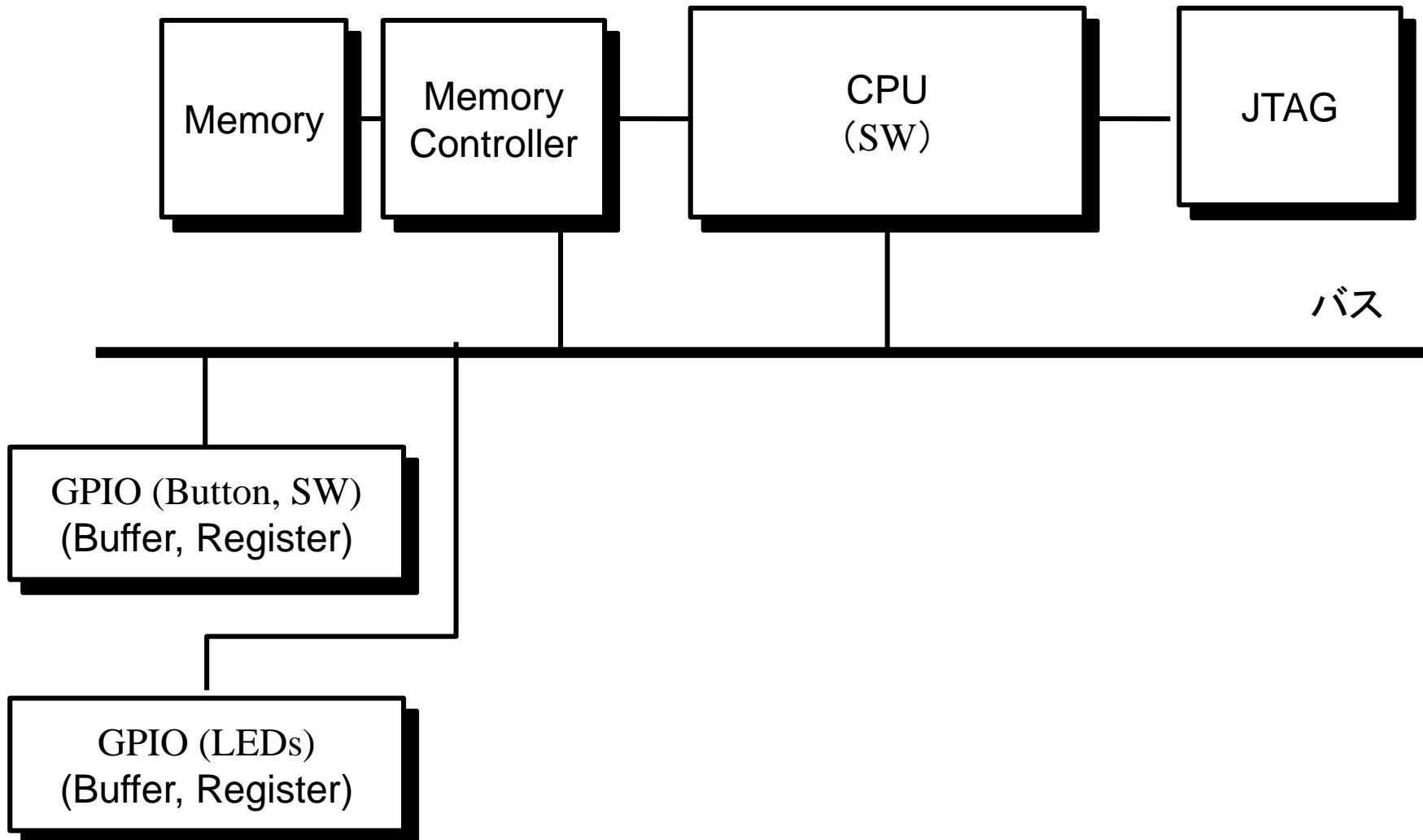
■ Adept USB deviceのインストール

- UARTに接続できない場合, 以下を確認

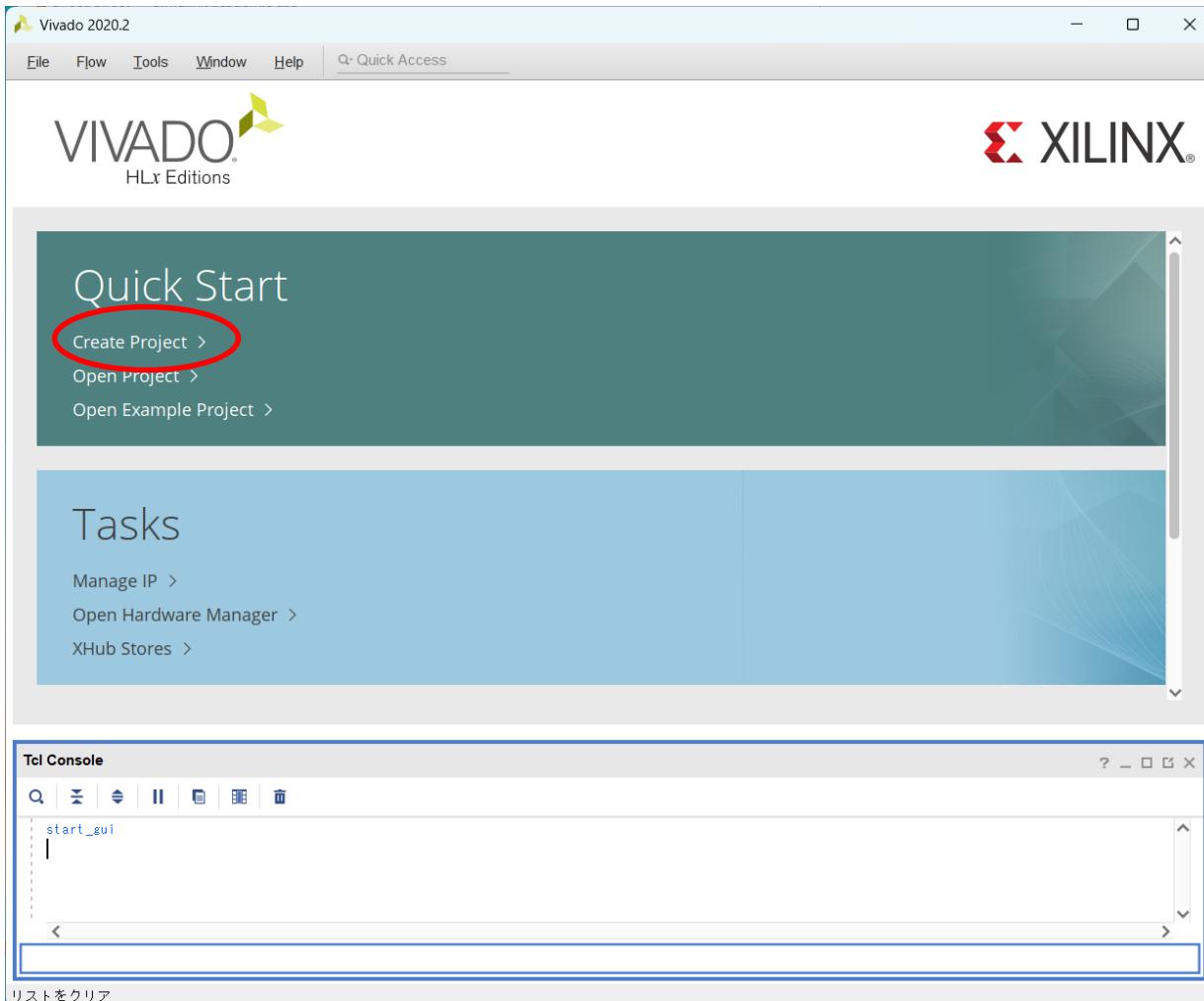
- Select D2XX Drivers in
<http://www.ftdichip.com/Products/ICs/FT2232H.htm>
- Select “setup executable” in Windows
- Get CDM v2.12.00....exe and run the program



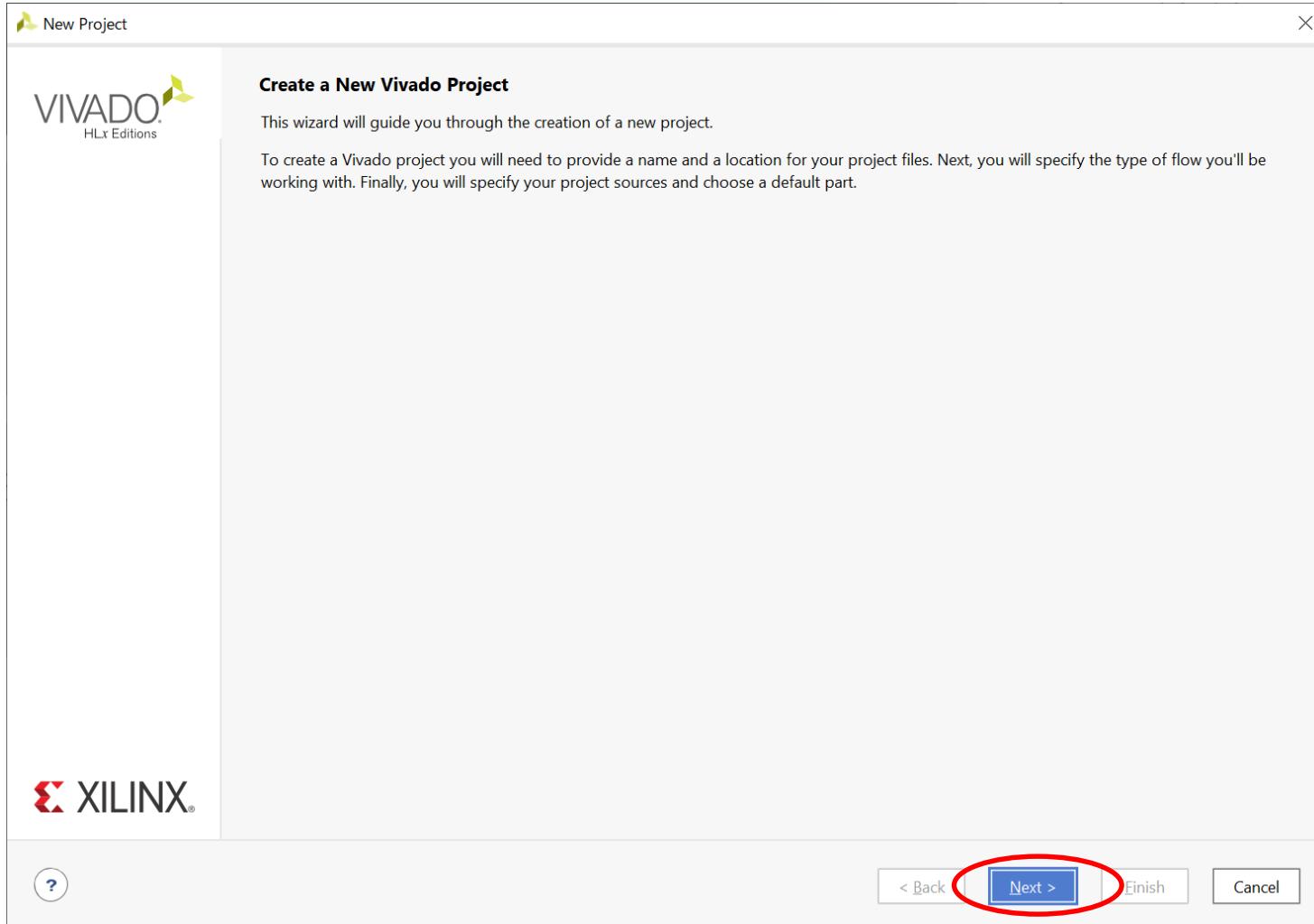
common_sysの実装と FPGA上での検証

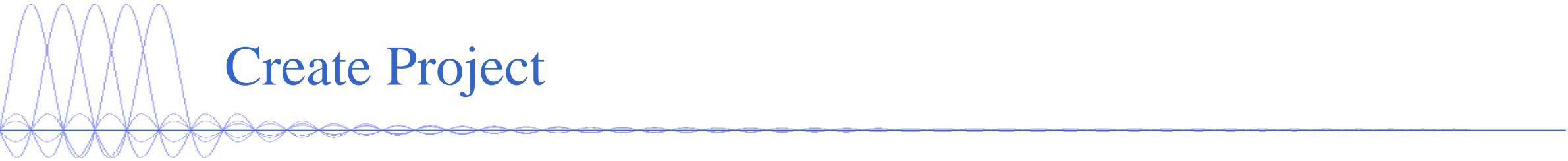


Create Project



Create Project





Create Project

New Project

Project Name
Enter a name for your project and specify a directory where the project data files will be stored.

Project name: common_sys

Project location: G:/LSI2025/common_sys/
 Create project

Project will be created at:

① Project locationを設定
② プロジェクト名を設定

? < Back Next > Finish Cancel

The 'Next >' button is highlighted with a red oval.

Create Project

New Project

Project Type
Specify the type of project to create.

RTL Project
You will be able to add sources, create block designs in IP Integrator, generate IP, run RTL analysis, synthesis, implementation, design planning and analysis.
 Do not specify sources at this time

Post-synthesis Project
You will be able to add sources, view device resources, run design analysis, planning and implementation.
 Do not specify sources at this time

I/O Planning Project
Do not specify design sources. You will be able to view part/package resources.

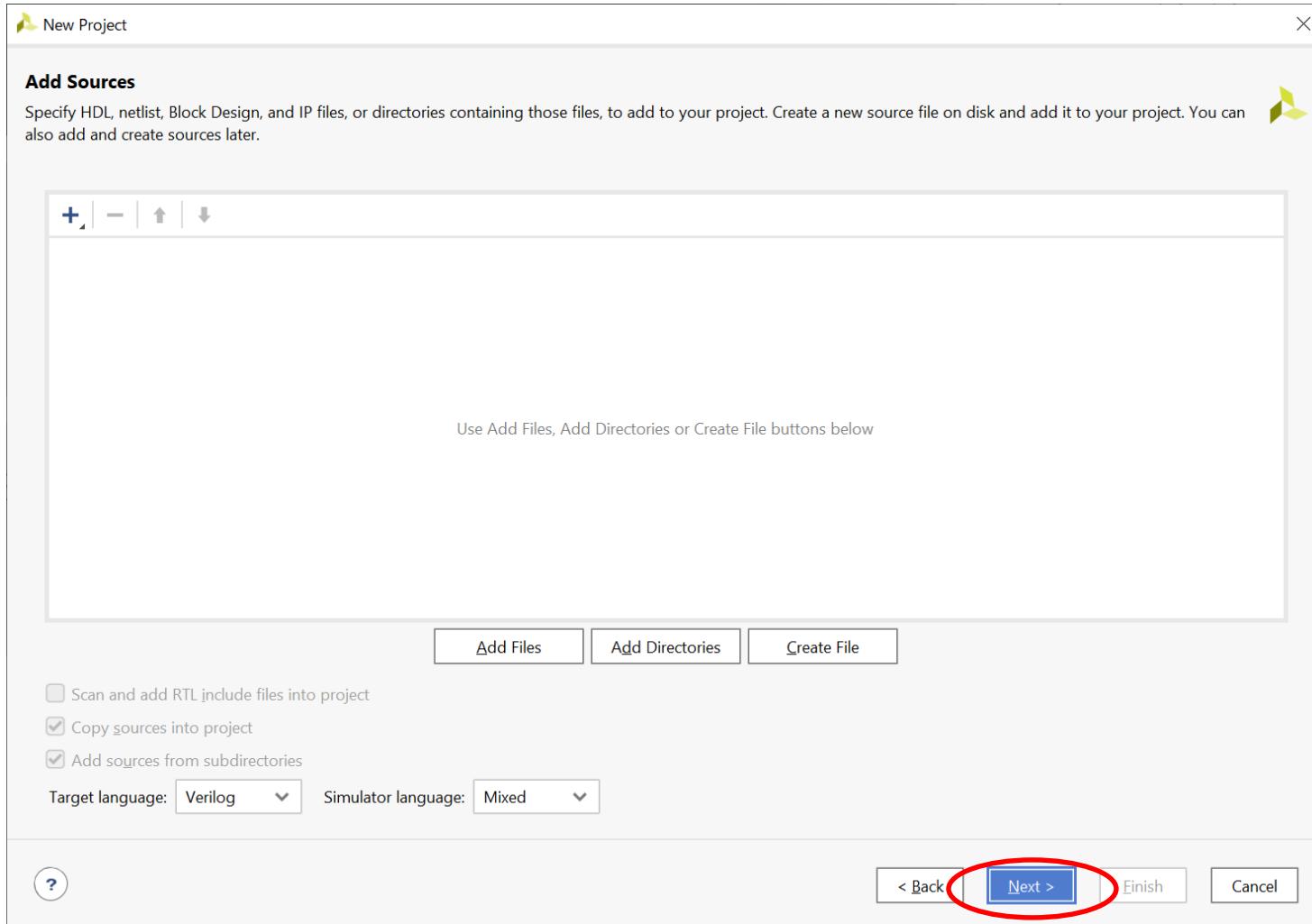
Imported Project
Create a Vivado project from a Synplify, XST or ISE Project File.

Example Project
Create a new Vivado project from a predefined template.

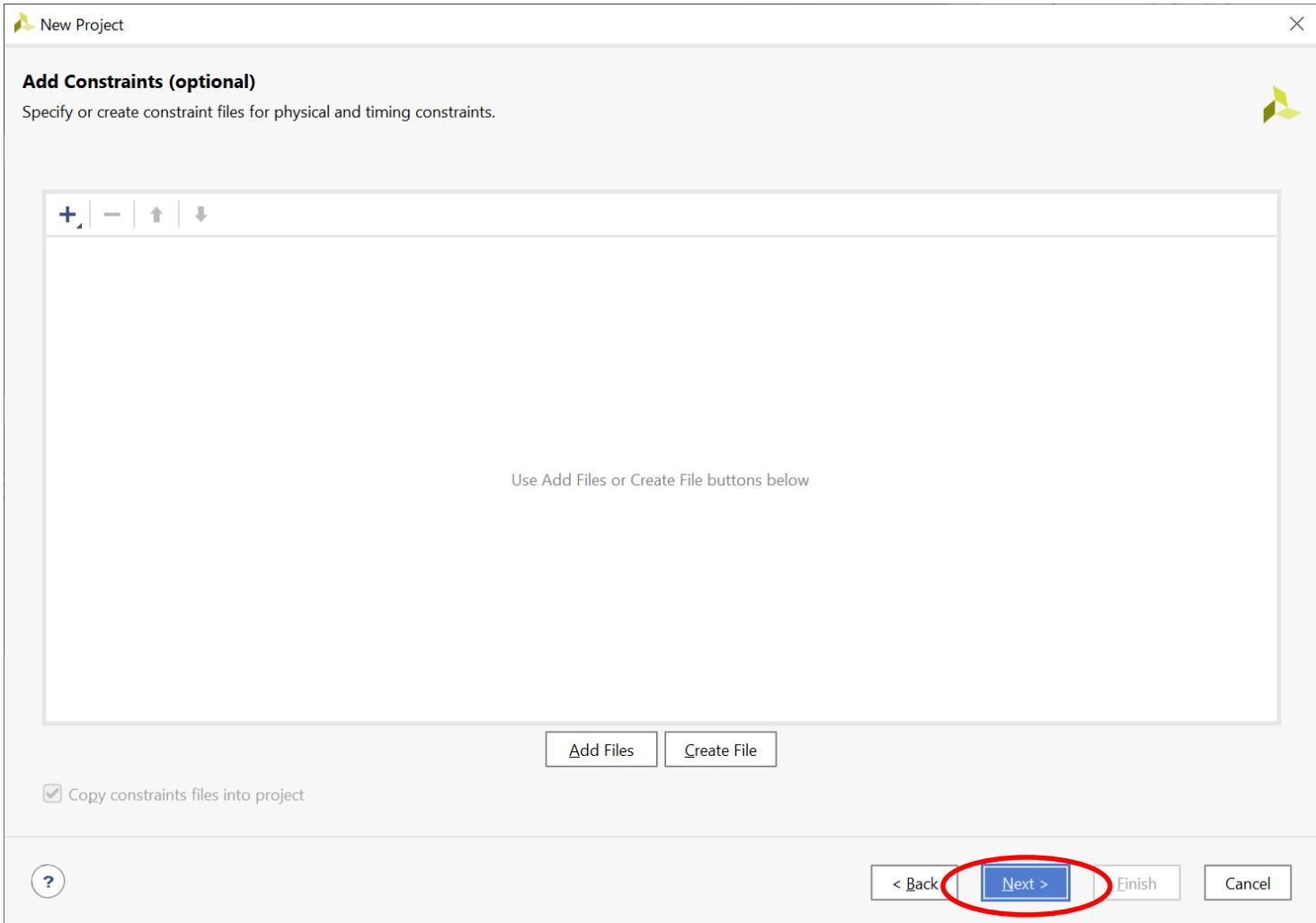
[?](#)

< Back Next > Finish Cancel

Create Project



Create Project



Create Project

New Project

Default Part
Choose a default Xilinx part or board for your project.

Part | **Boards**

Reset All Filters

Vendor: All Name: All Board Rev: Latest

Install/Update Boards

Preview Vendor File Version Part I/O Pin Count Board

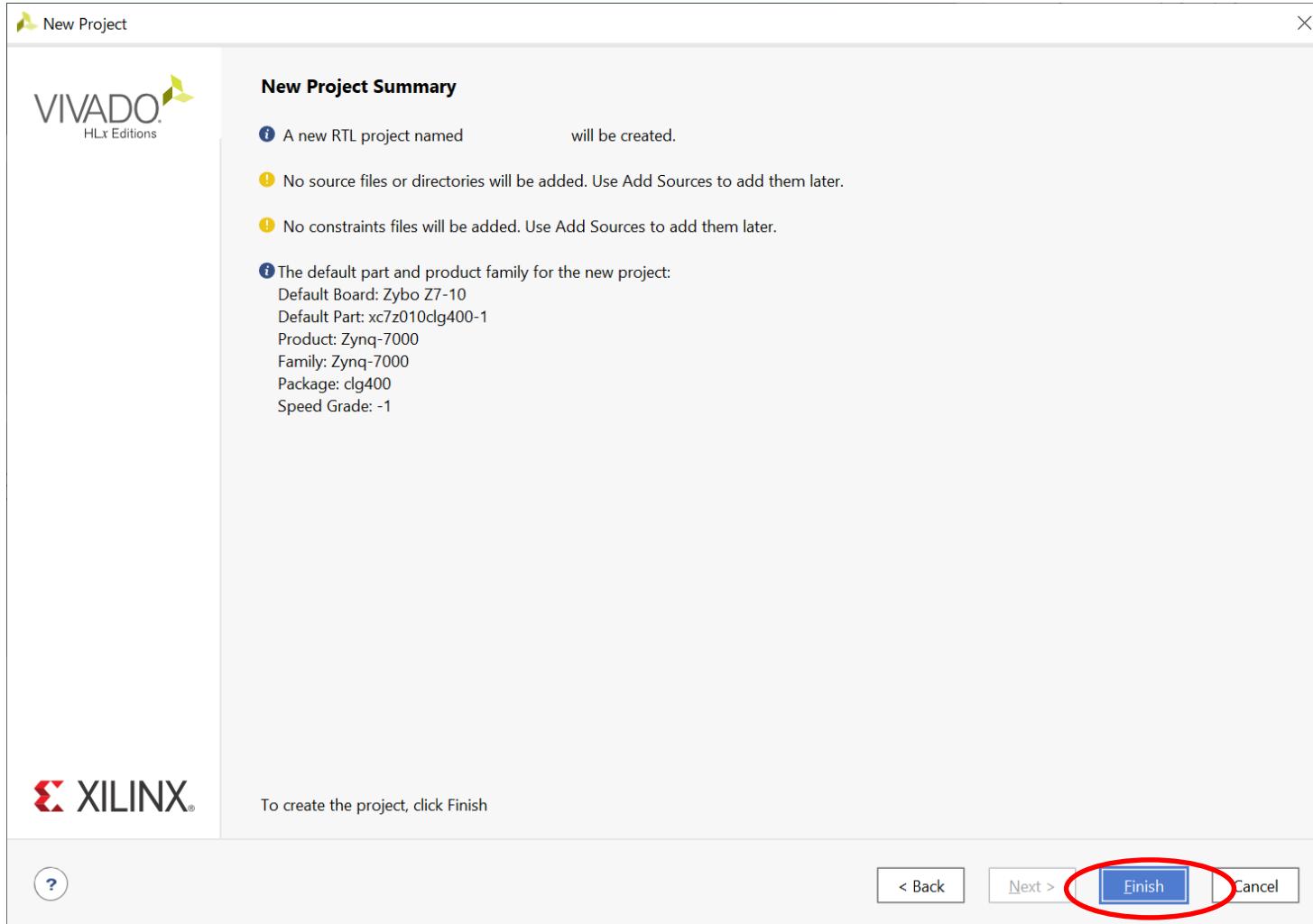
	alpha-data.com	1.1	xc7vx690tffg1157-2	1157	1.0
	alpha-data.com	1.0	xcku060-ffva1156-2-e	1156	1.0
	digilentinc.com	1.0	xc7z010clg400-1	400	B.2
	digilentinc.com	1.0	xc7z020clg400-1	400	B.2
	digilentinc.com	1.0	xc7z010clg400-1	400	B.3
	em.avnet.com	1.4	xc7z020clg484-1	484	d

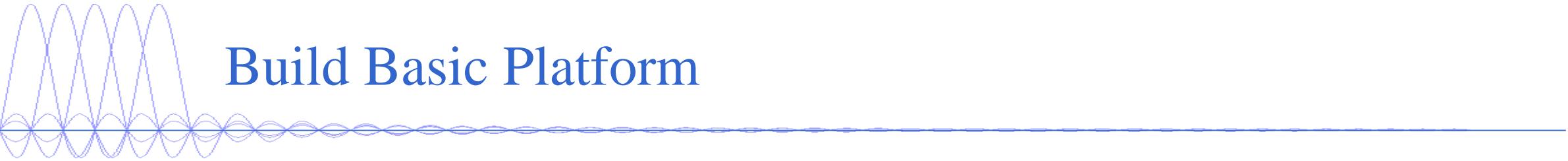
ZedBoard Zynq Evaluation and Development Kit
Add Daughter Card Connections

? < Back Next > Finish Cancel

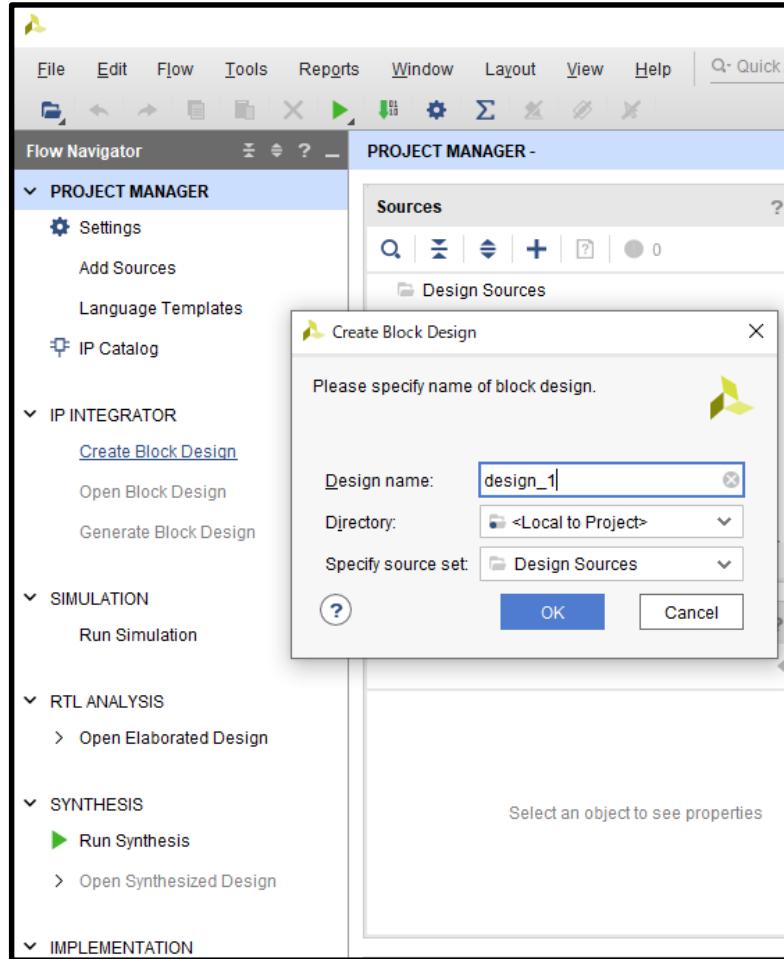
使用するボードにあわせて設定
Zybo z7-10
もしくは
Zybo z7-20
使うボードに合わせて選択すること

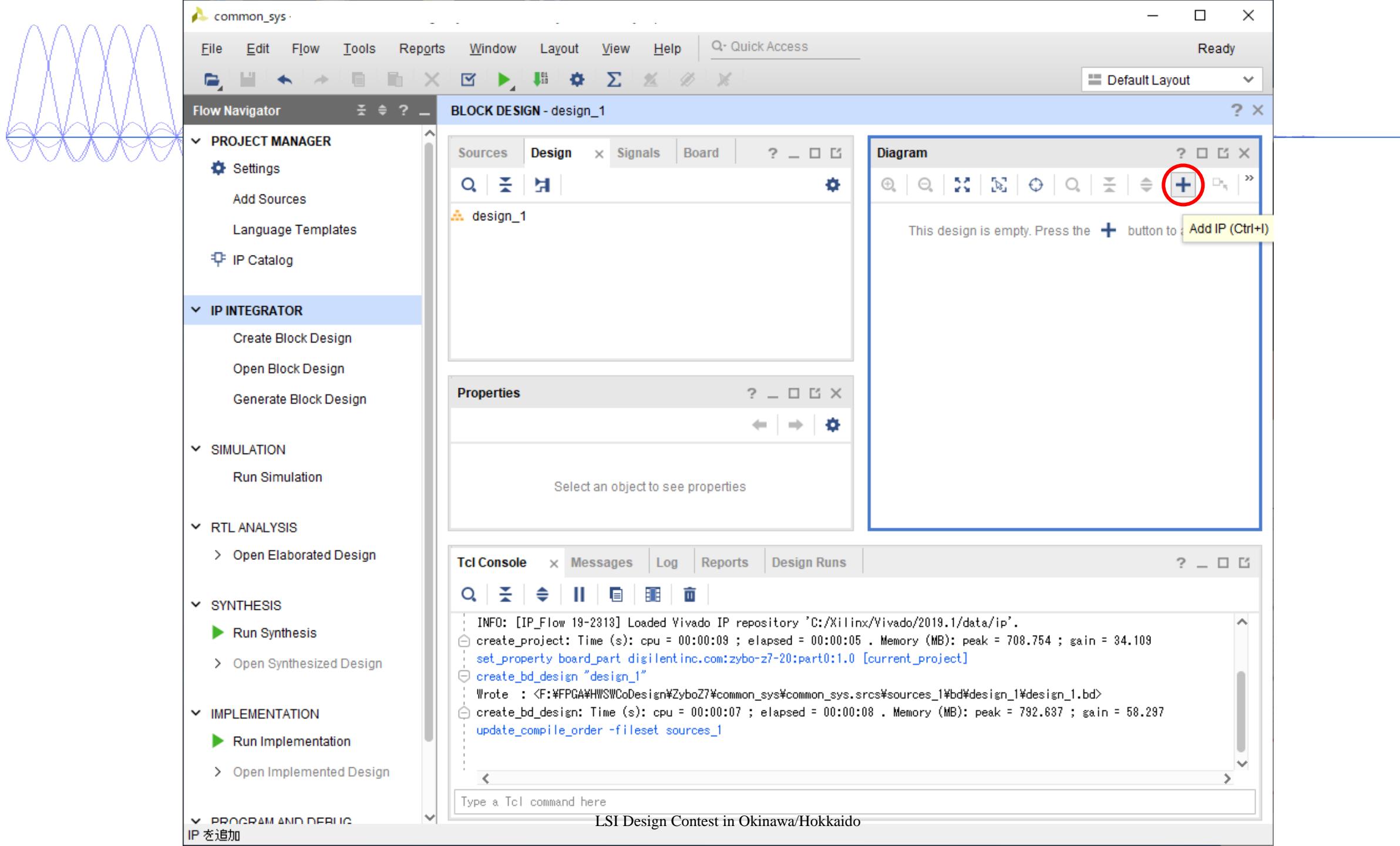
Create Project



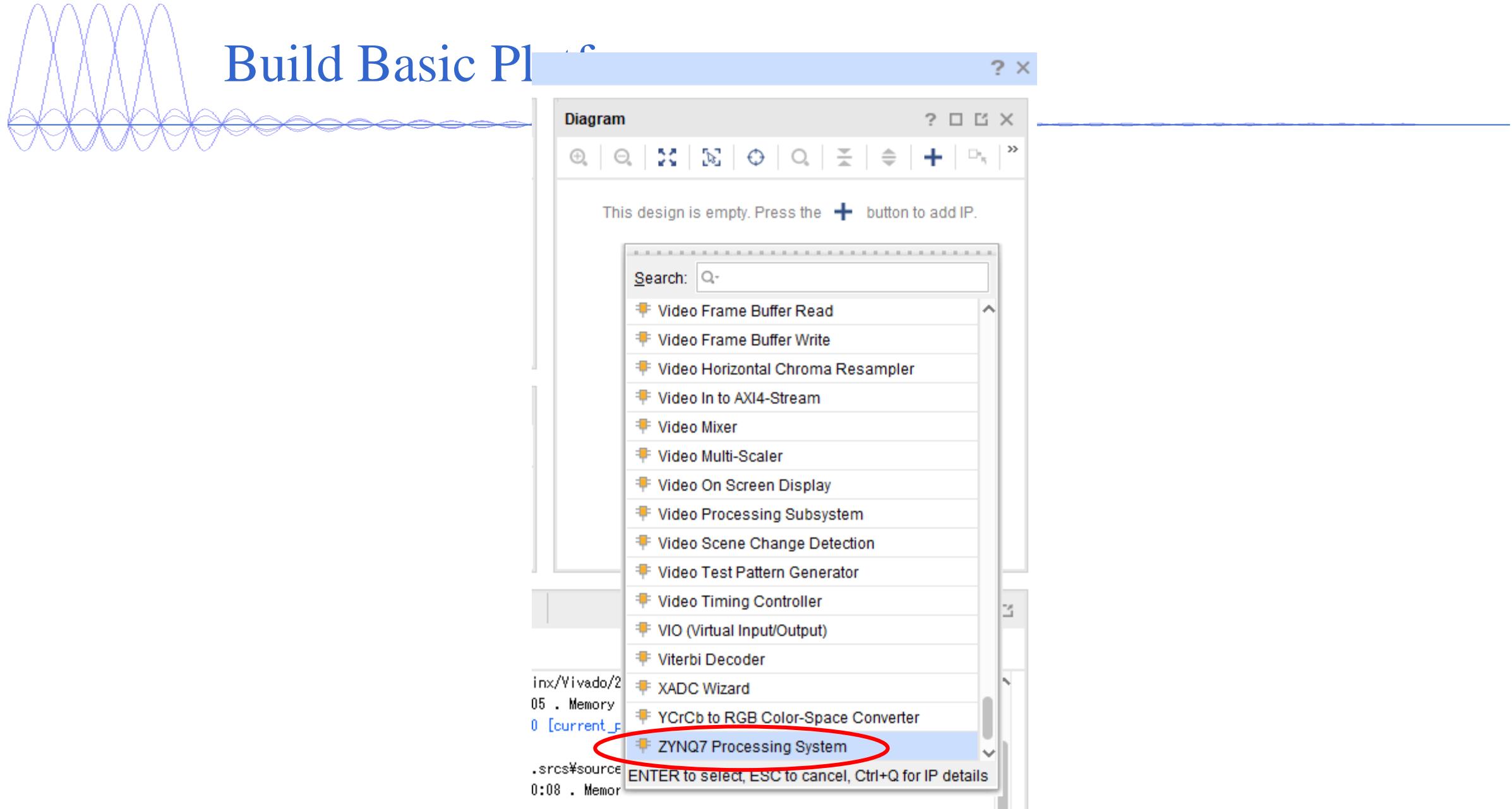


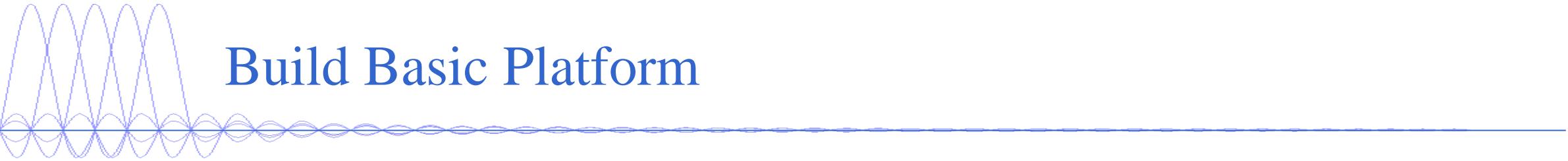
Build Basic Platform



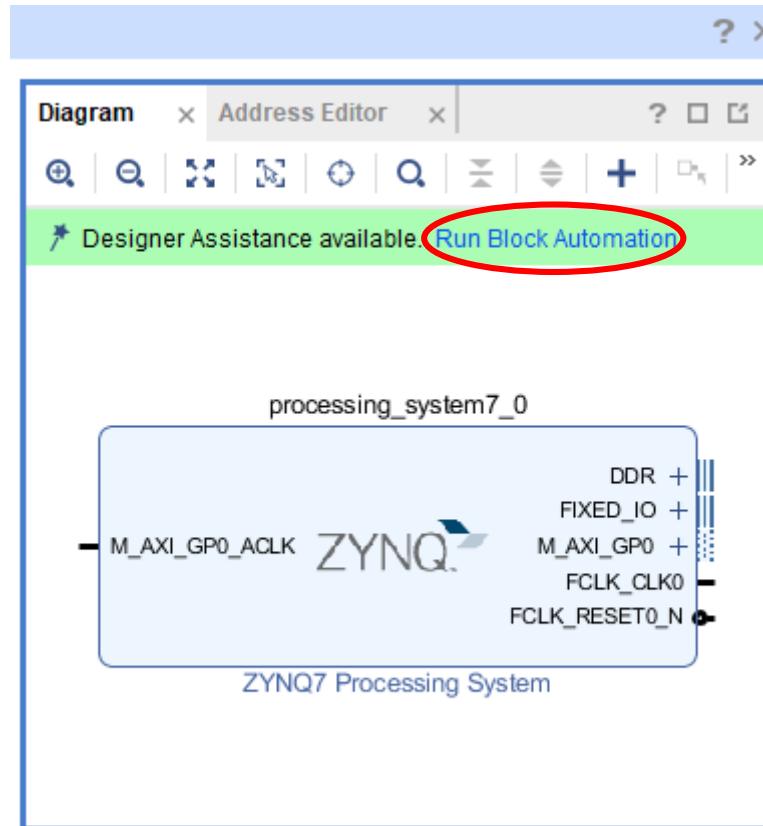


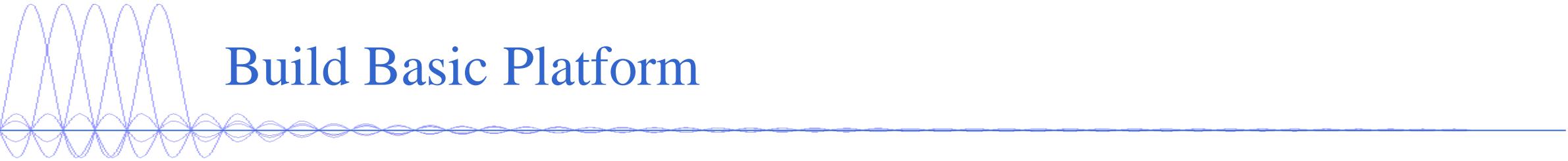
Build Basic PL



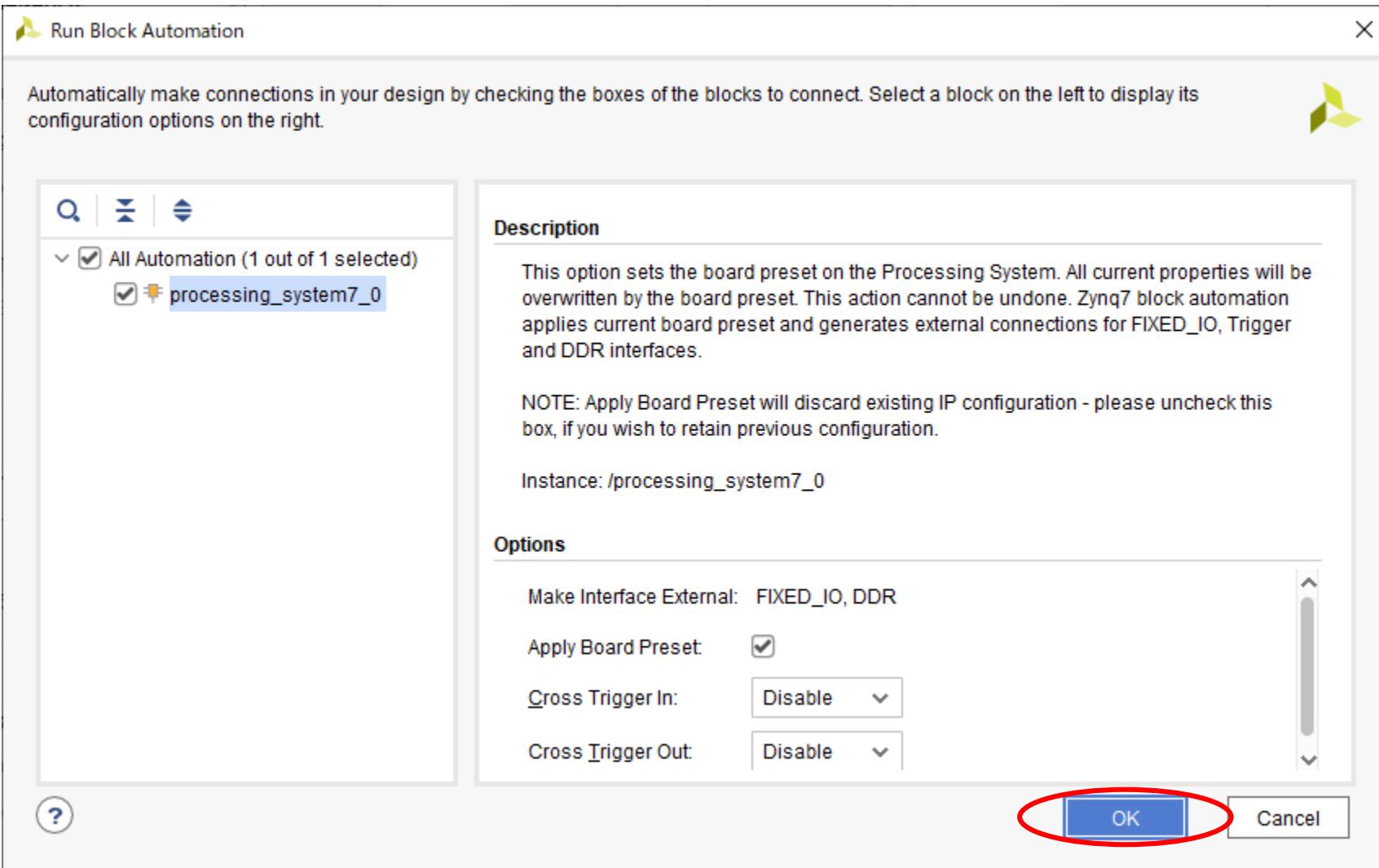


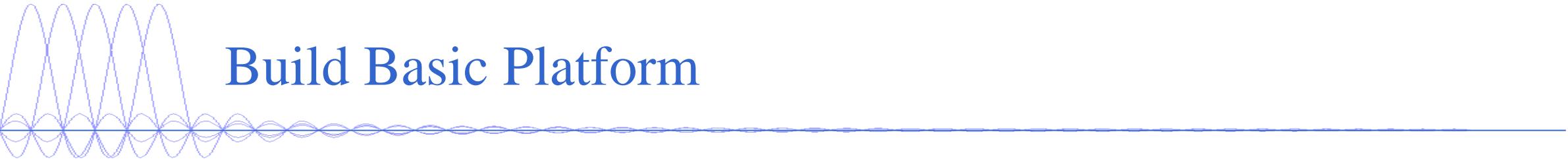
Build Basic Platform



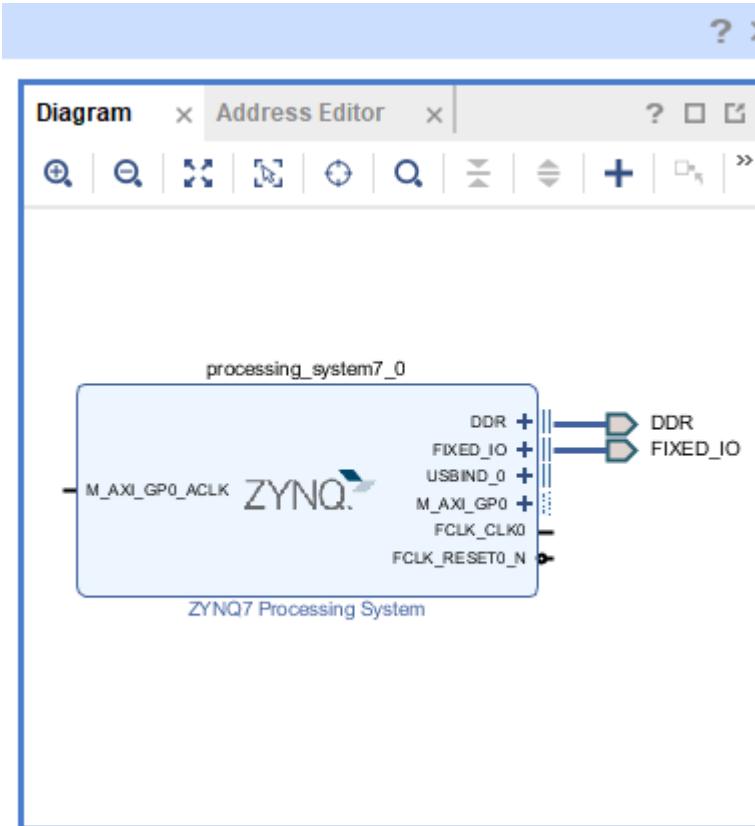


Build Basic Platform

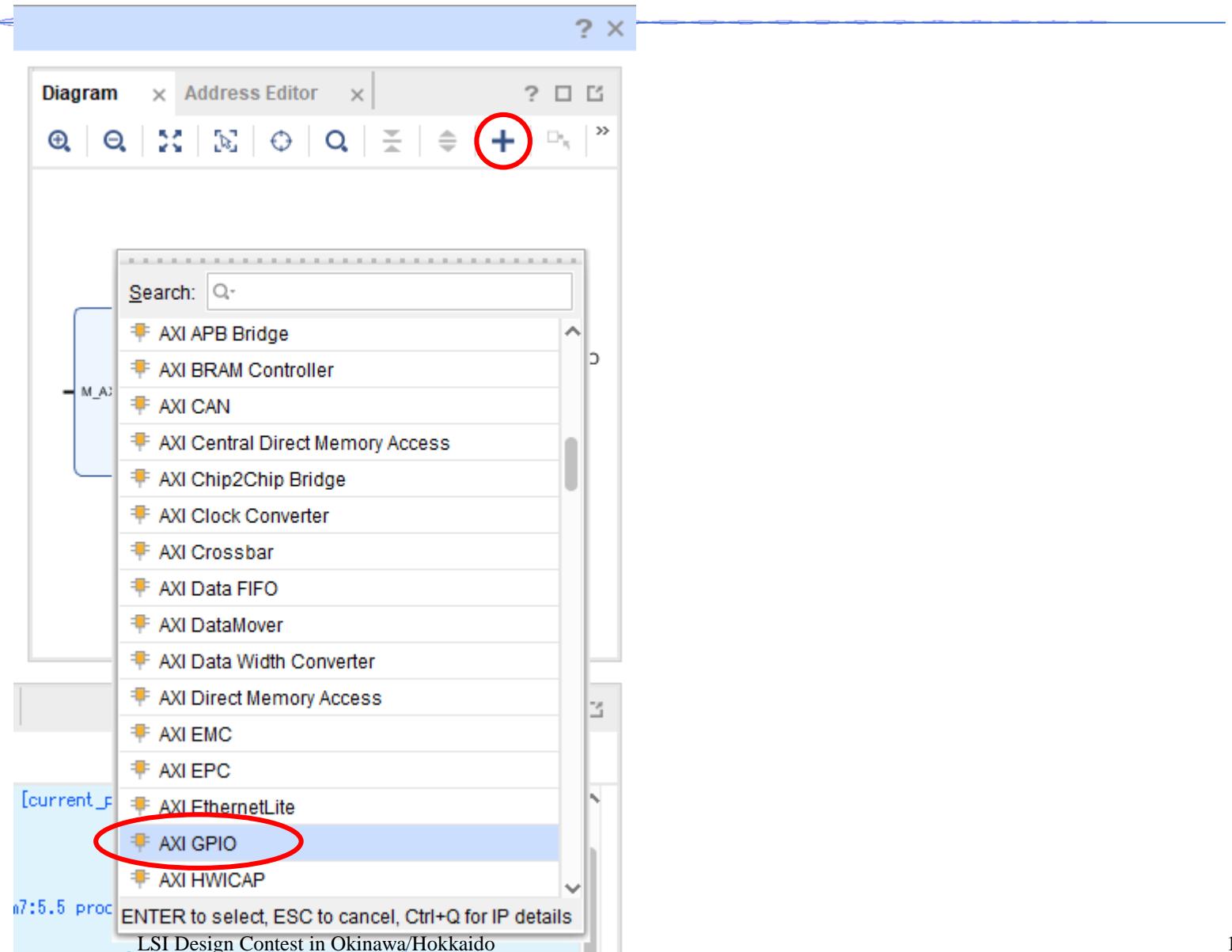


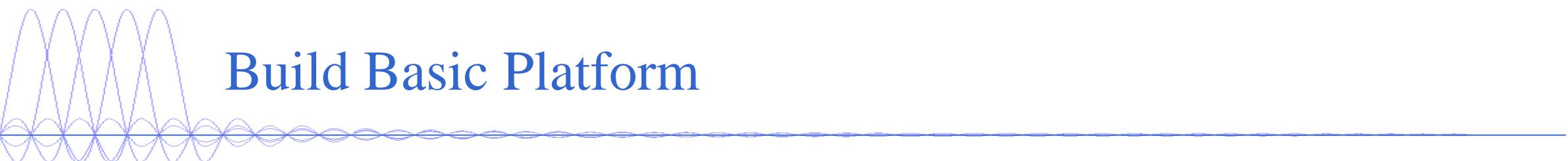


Build Basic Platform

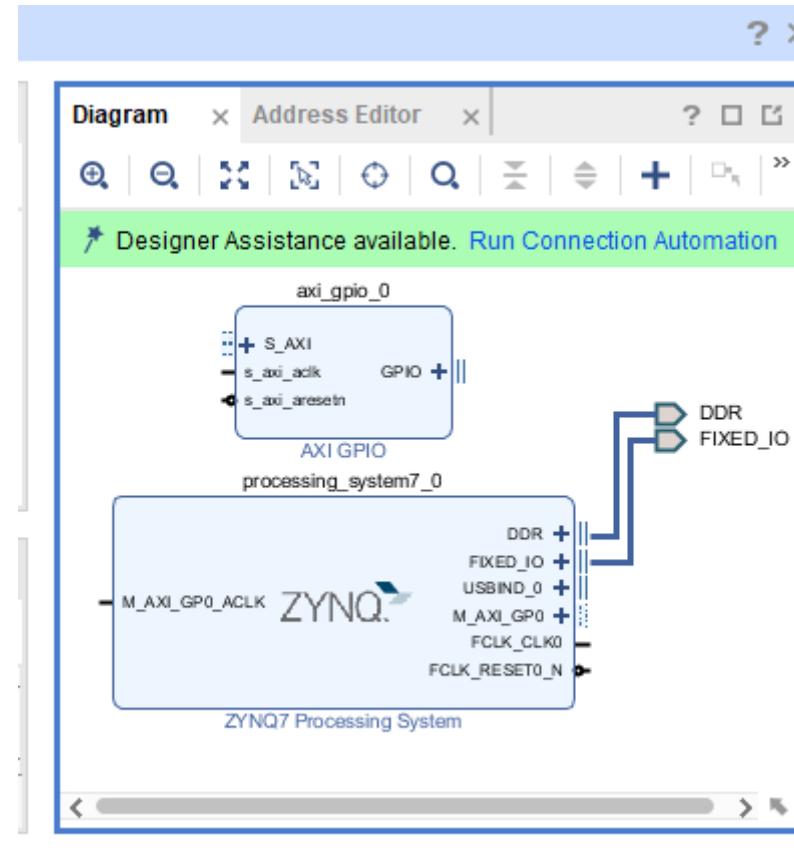


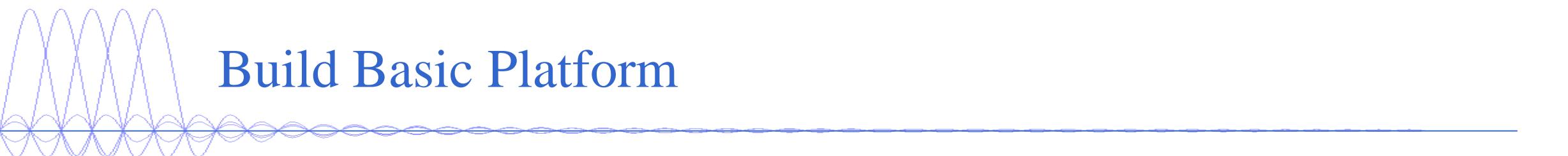
Build Basic Platform



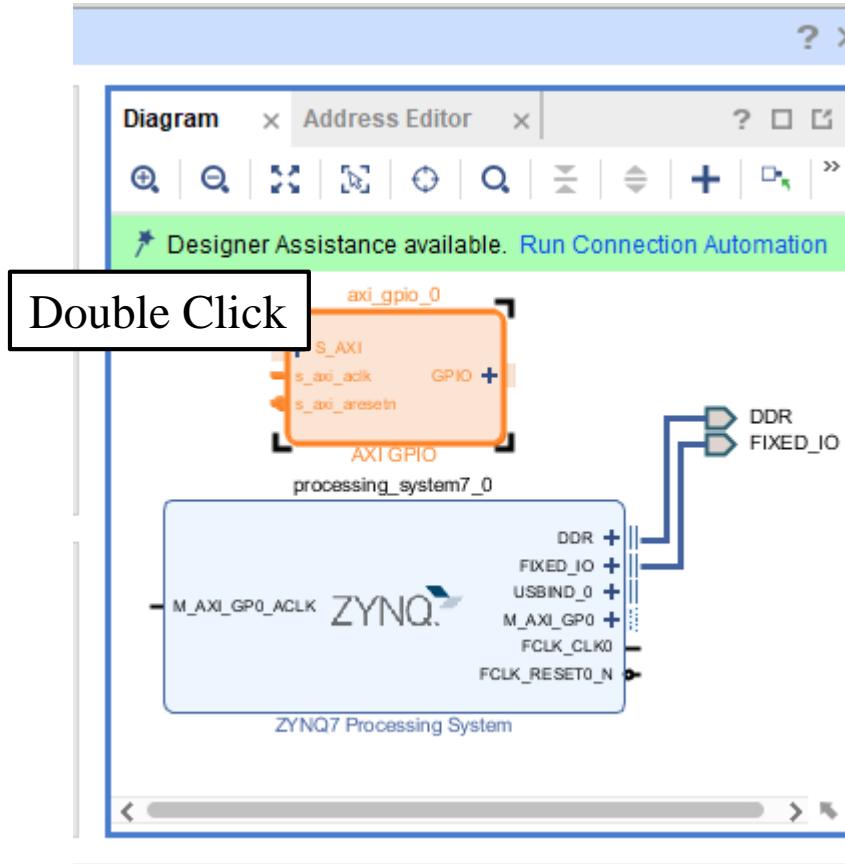


Build Basic Platform





Build Basic Platform



AXI GPIO (2.0)

[Documentation](#) [IP Location](#) Show disabled portsComponent Name [Board](#)[IP Configuration](#)

Associate IP interface with board interface

IP Interface	Board Interface
GPIO	Custom
GPIO2	Custom

[Clear Board Parameters](#) Enable Interrupt[OK](#)[Cancel](#)

Re-customize IP

AXI GPIO (2.0)

Documentation IP Location

Show disabled ports

Component Name: axi_gpio_0

Board IP Configuration

All Inputs

All Outputs

GPIO Width: 32 [1 - 32]

Default Output Value: 0x00000000 [0x00000000, 0xFFFFFFFF]

Default Tri State Value: 0xFFFFFFFF [0x00000000, 0xFFFFFFFF]

Enable Dual Channel

GPIO 2 Set this checkbox if dual channel GPIO is required

All Inputs

All Outputs

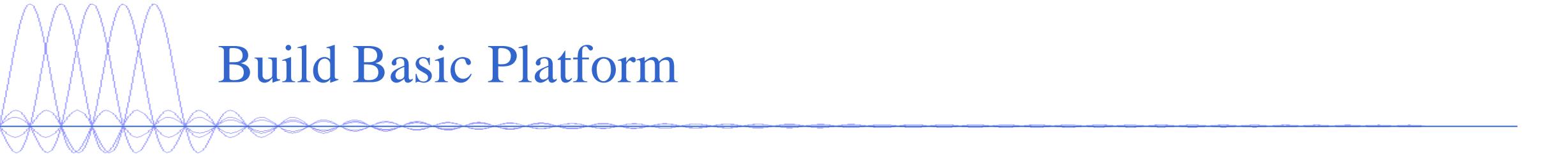
GPIO Width: 32 [1 - 32]

Default Output Value: 0x00000000 [0x00000000, 0xFFFFFFFF]

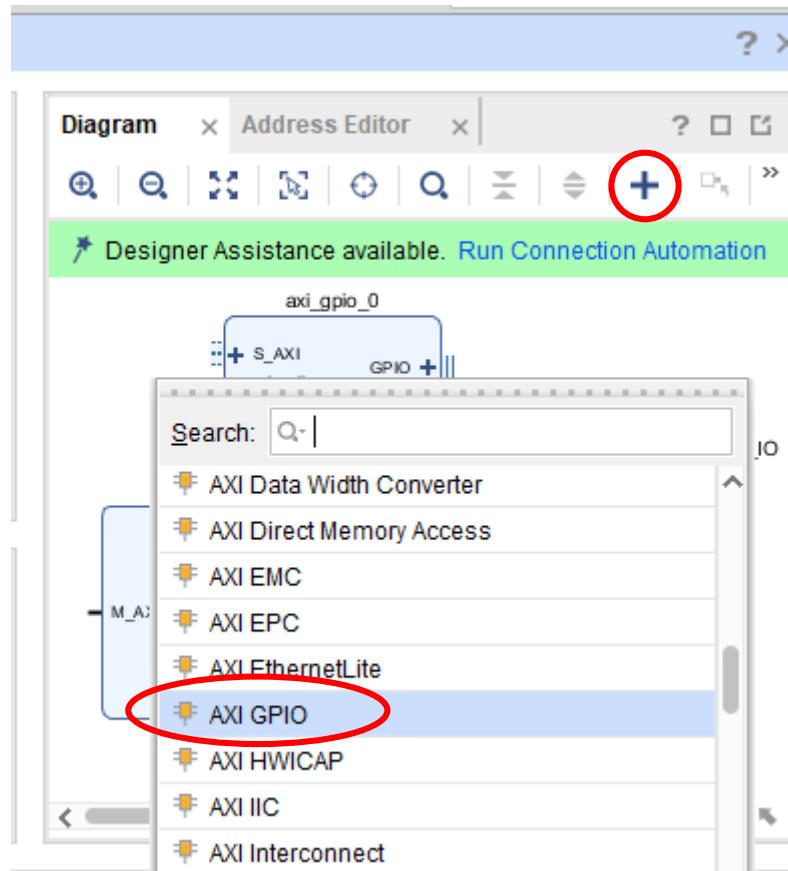
Enable Interrupt

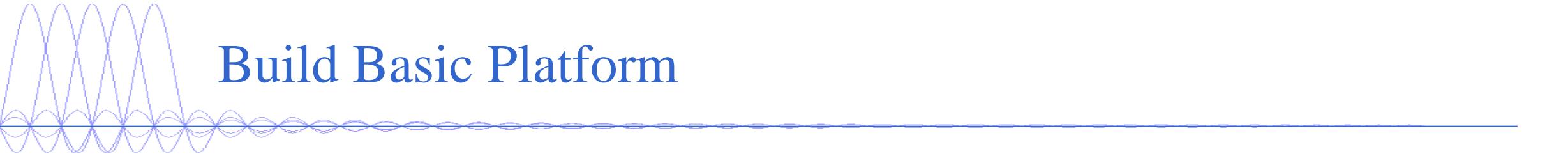
OK Cancel

LSI Design Contest in Okinawa/Hokkaido

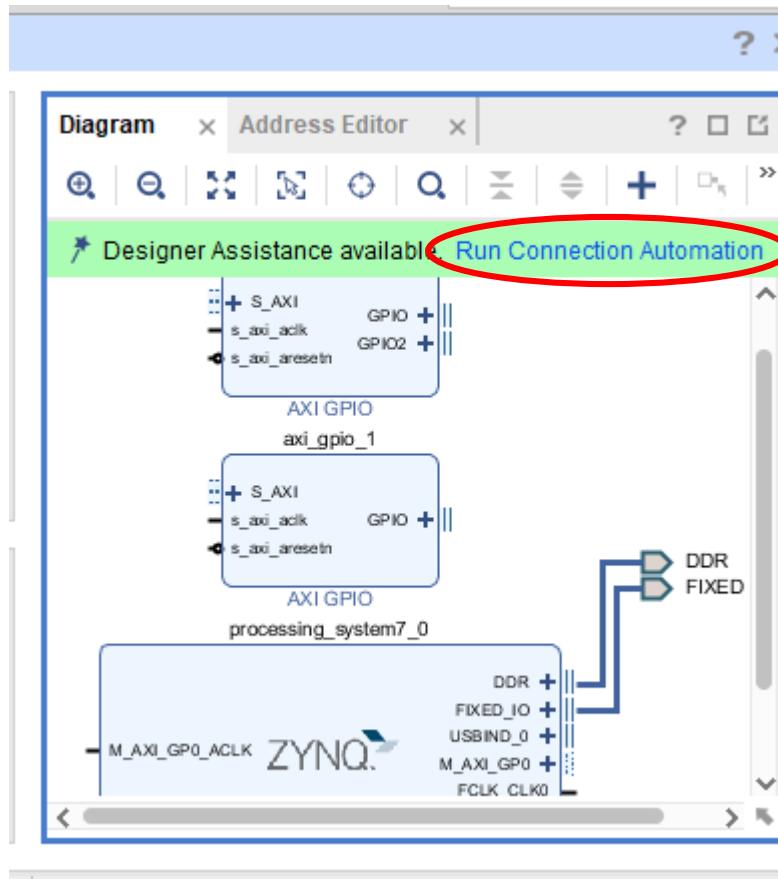


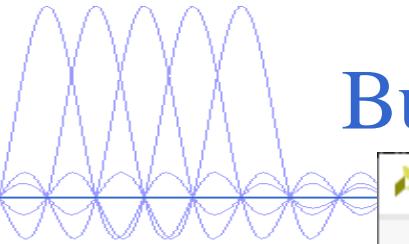
Build Basic Platform



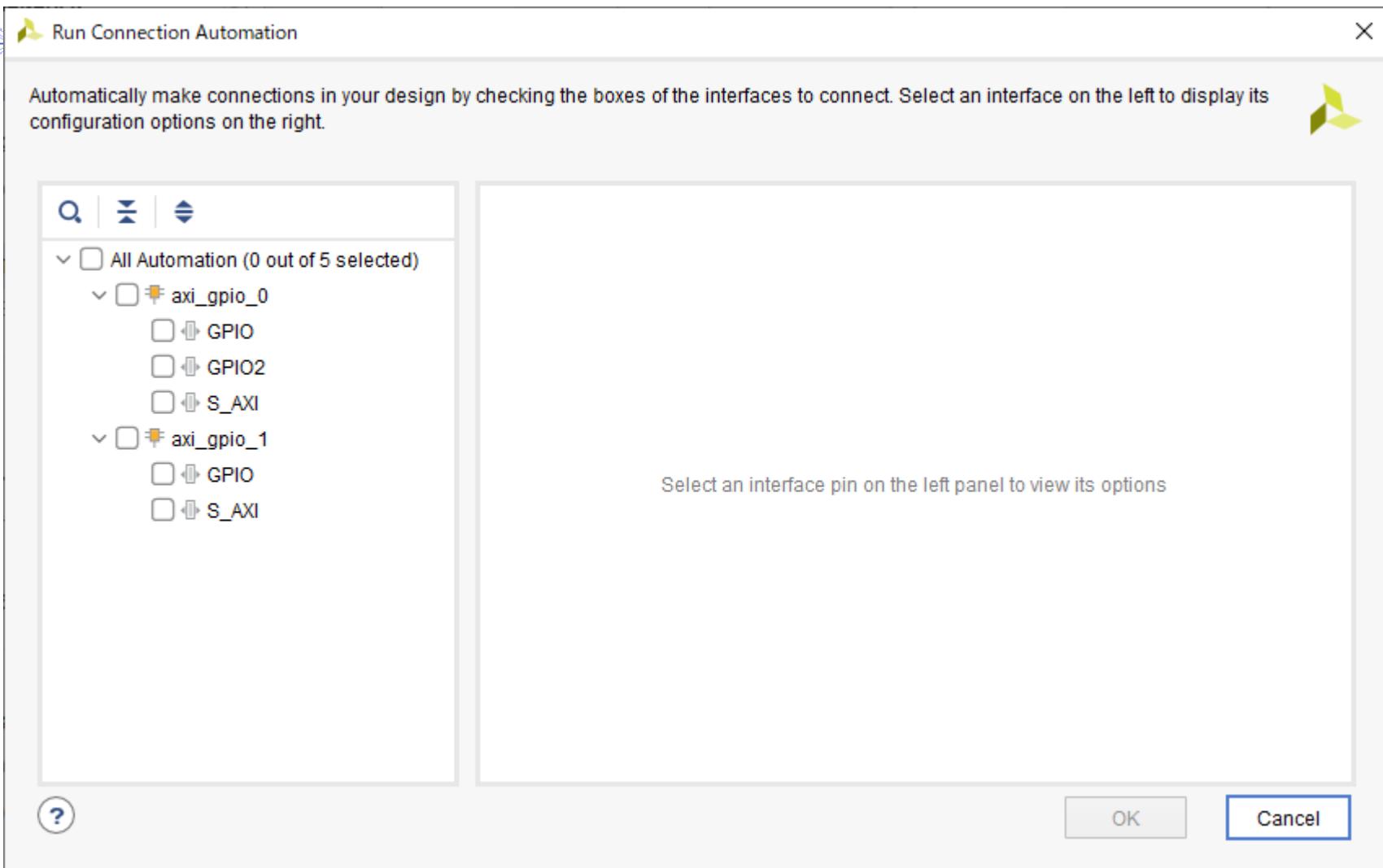


Build Basic Platform

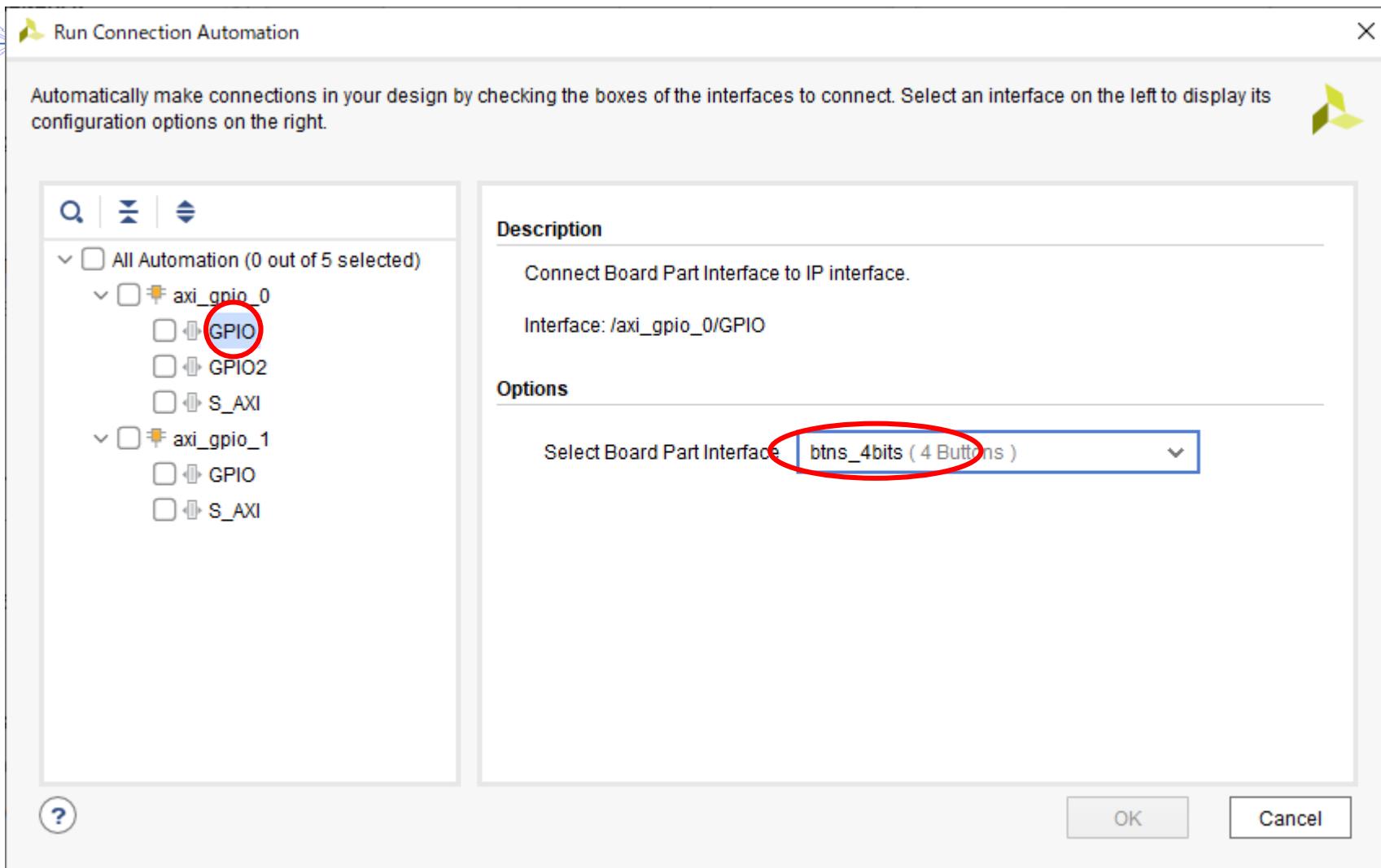




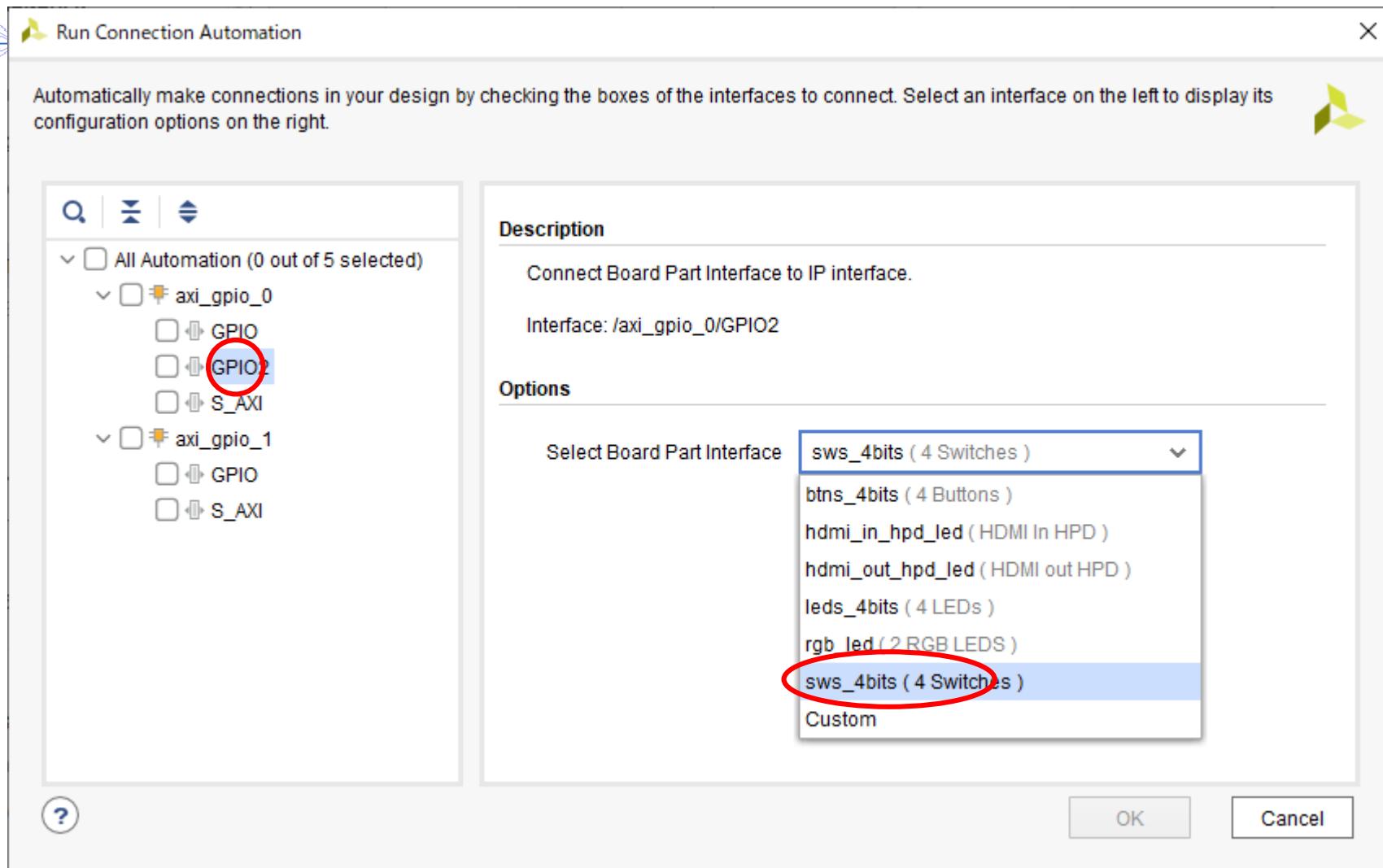
Build Basic Platform



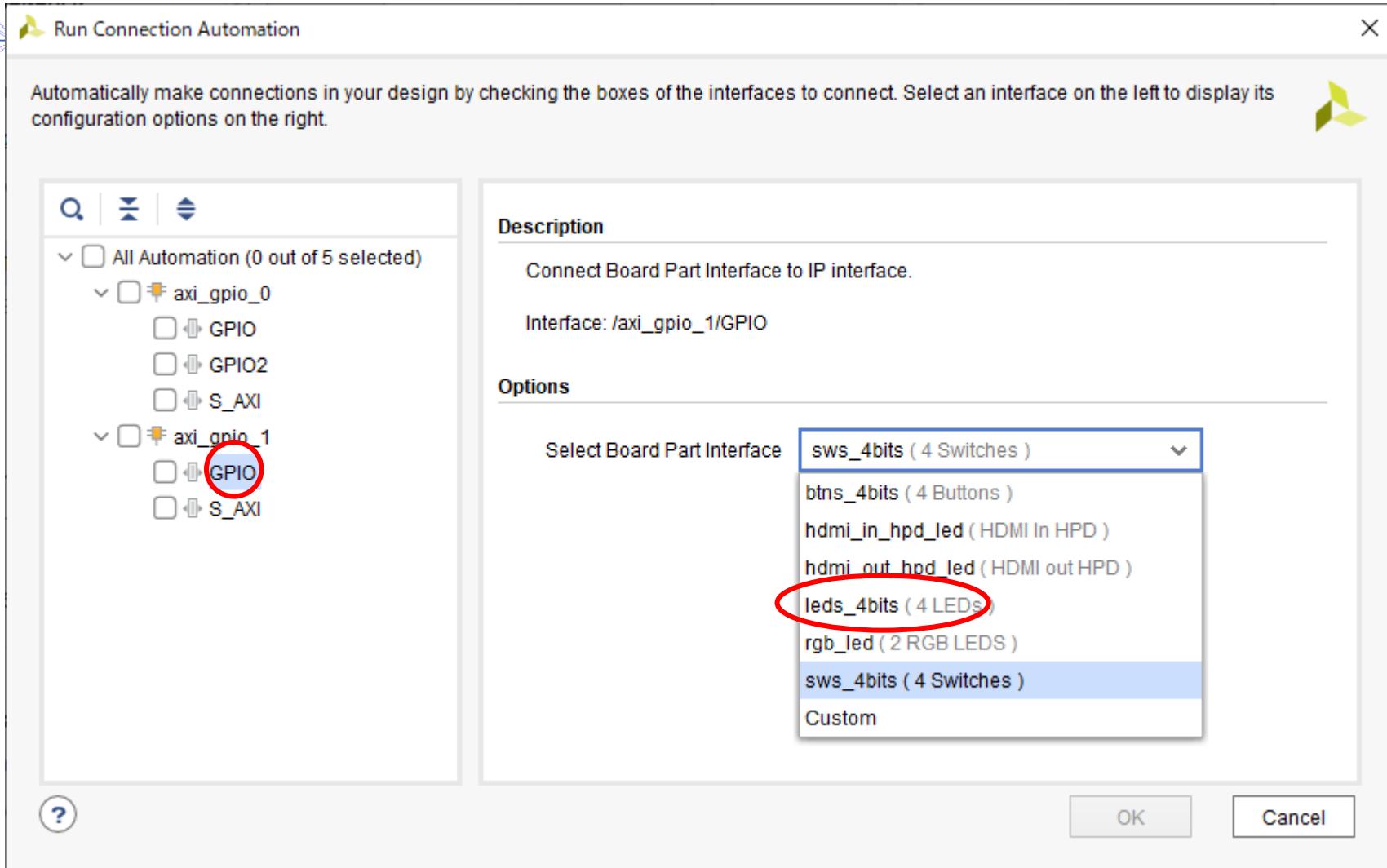
Build Basic Platform



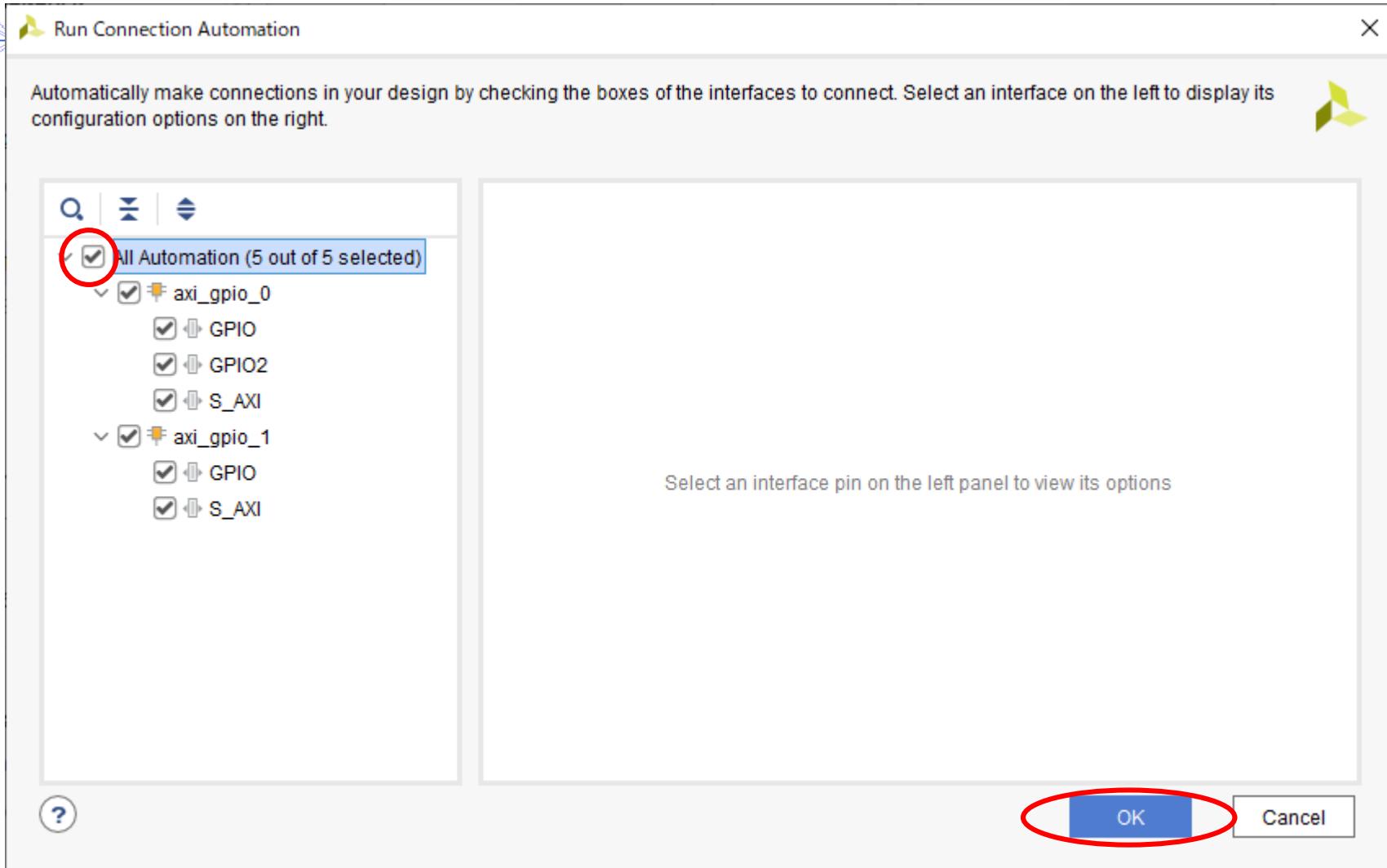
Build Basic Platform

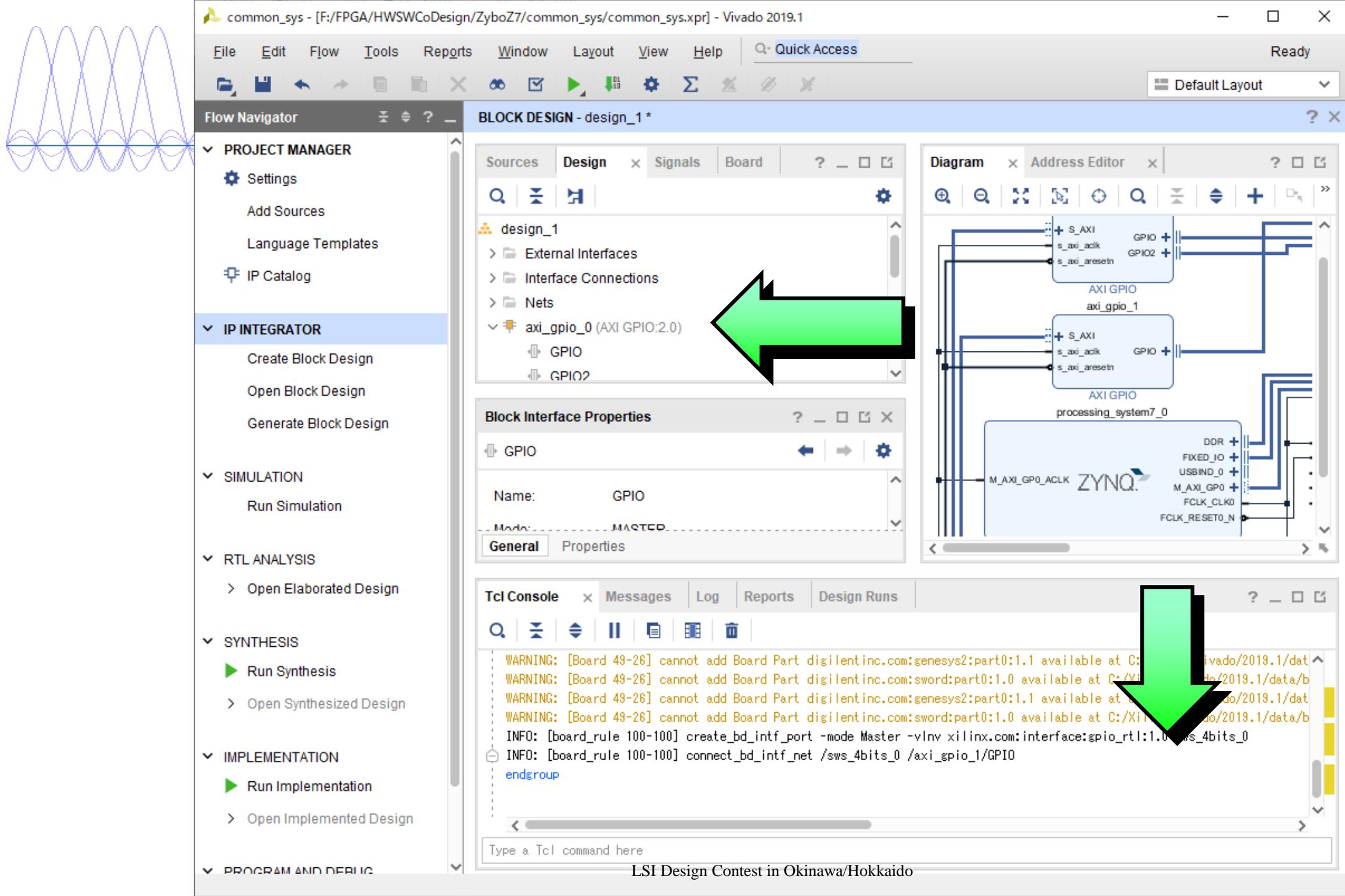


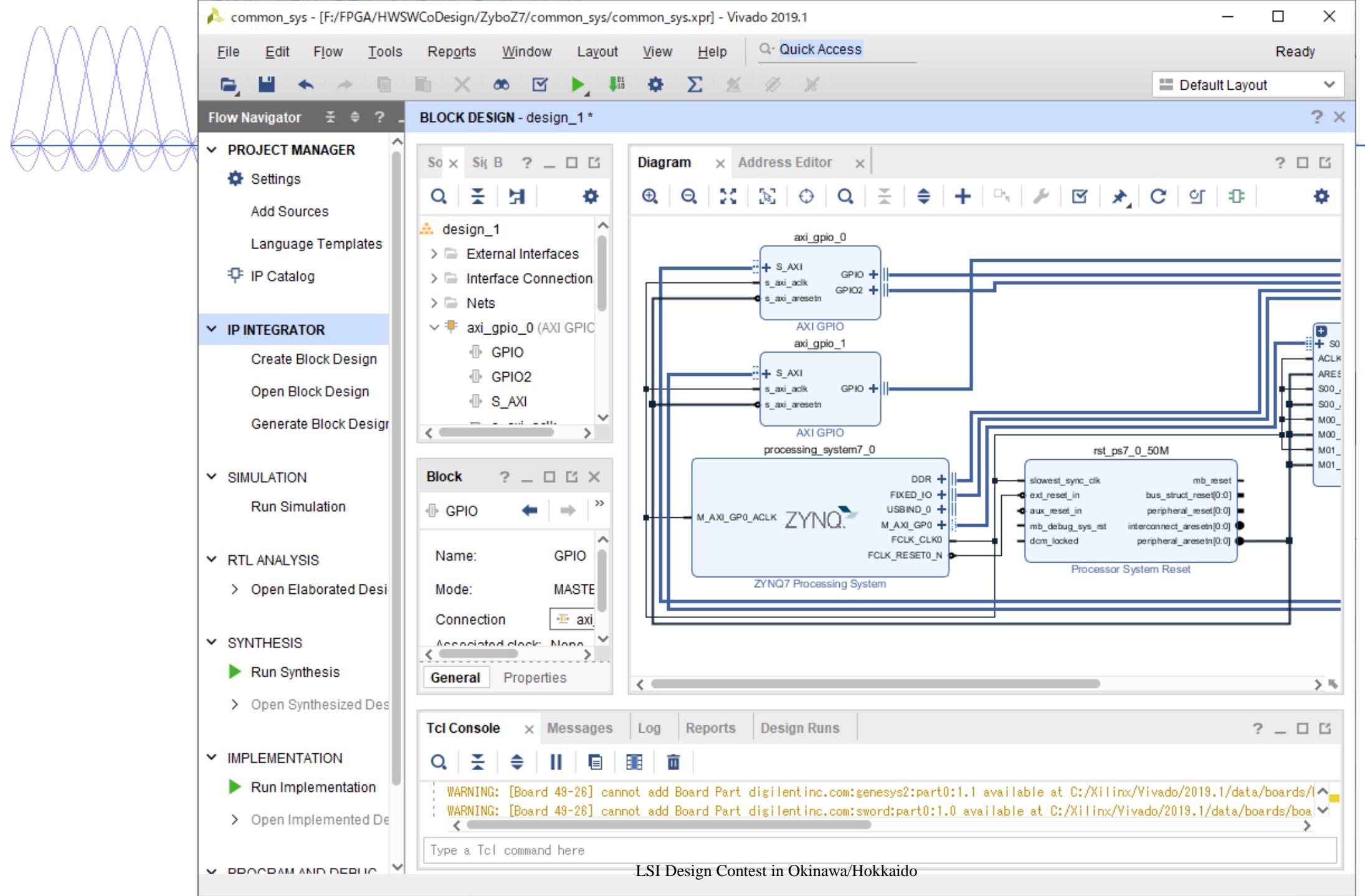
Build Basic Platform

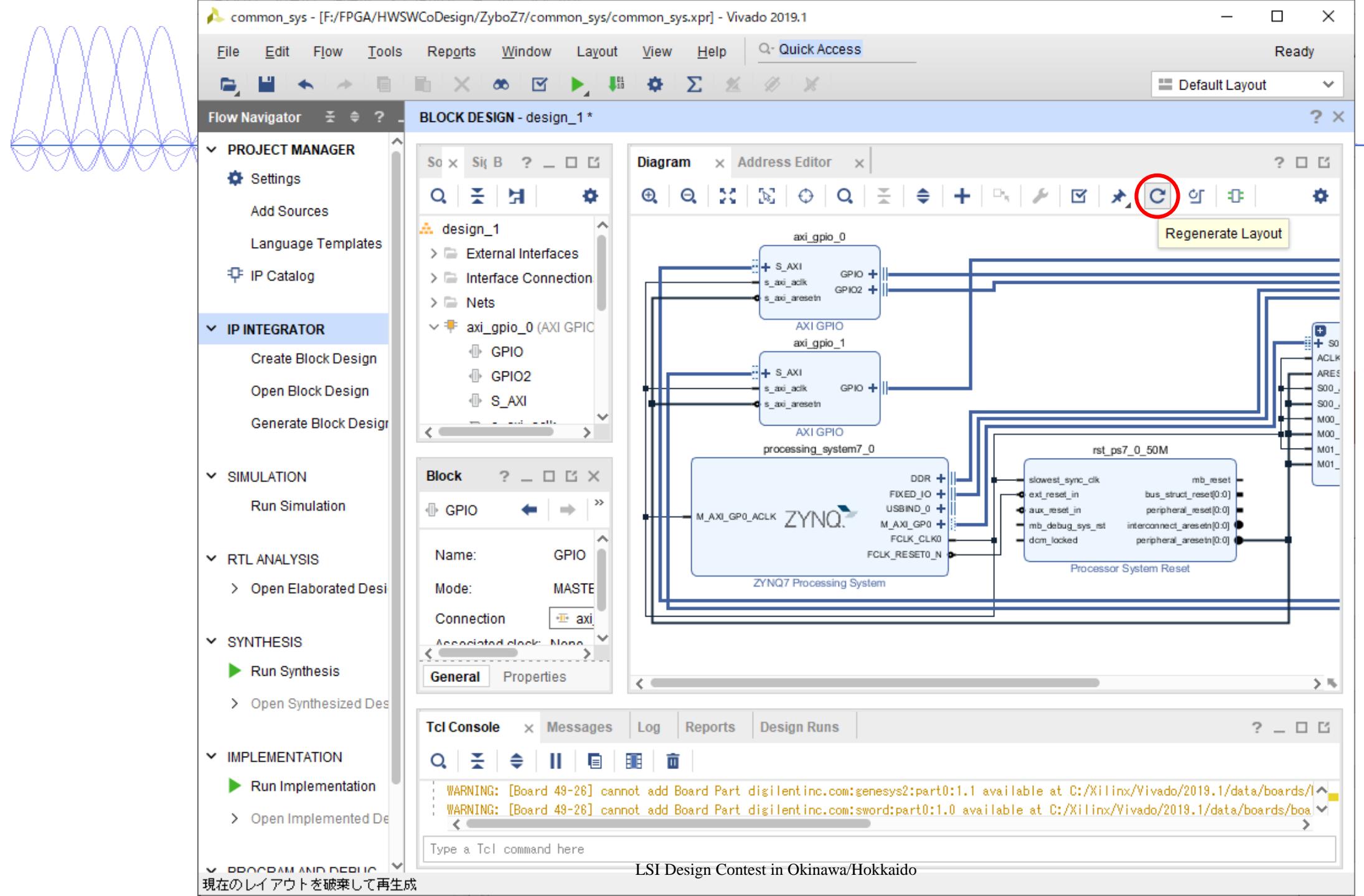


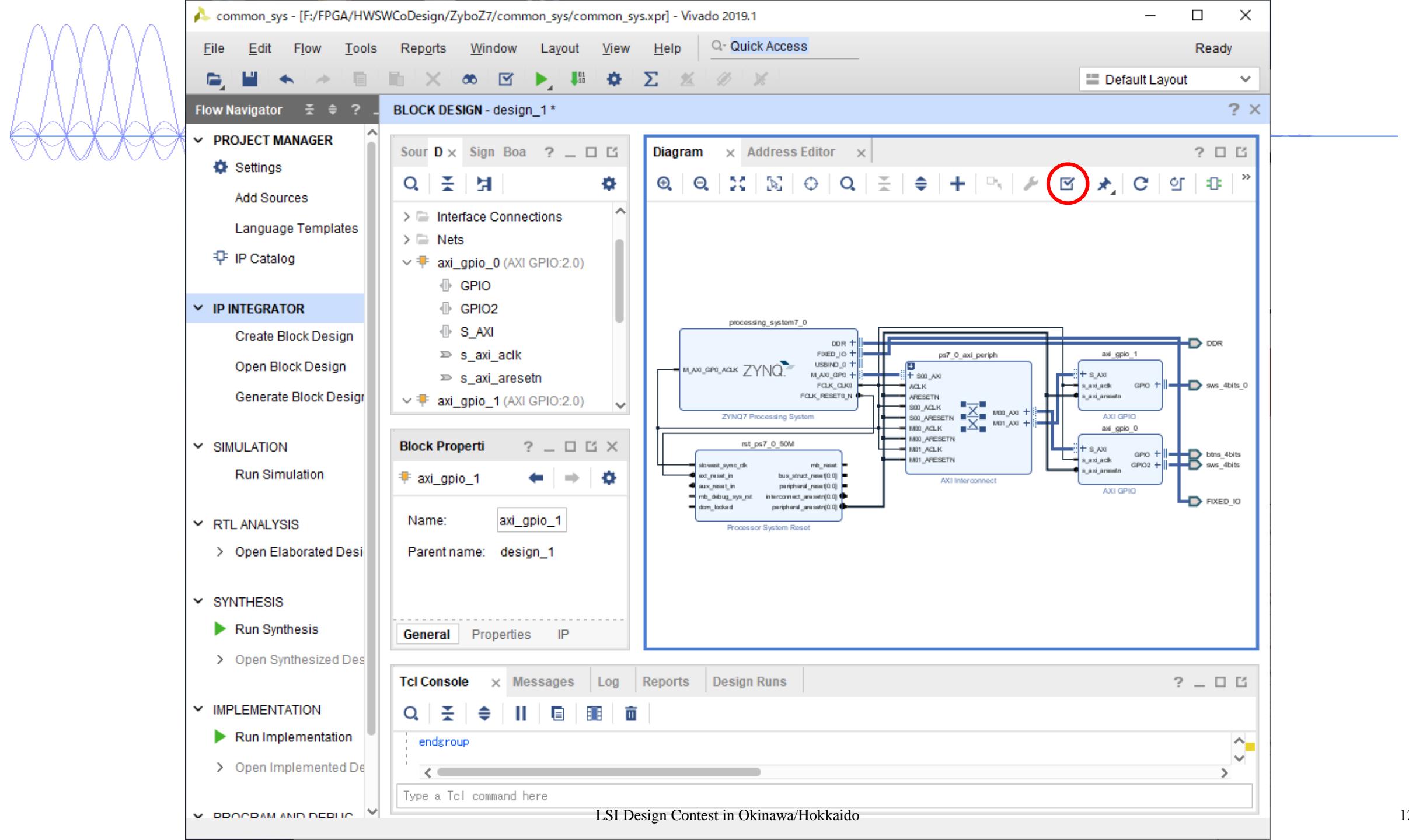
Build Basic Platform

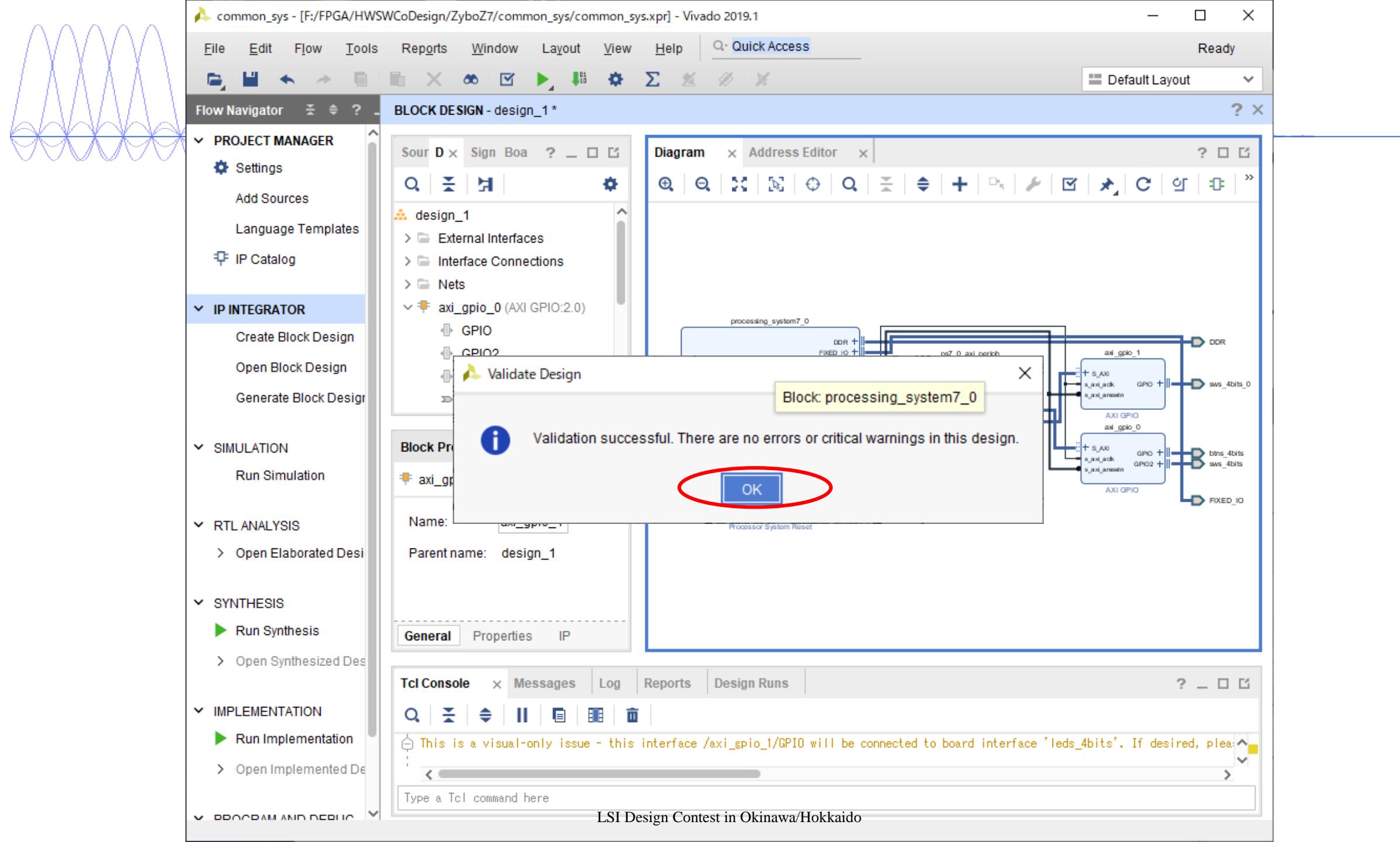


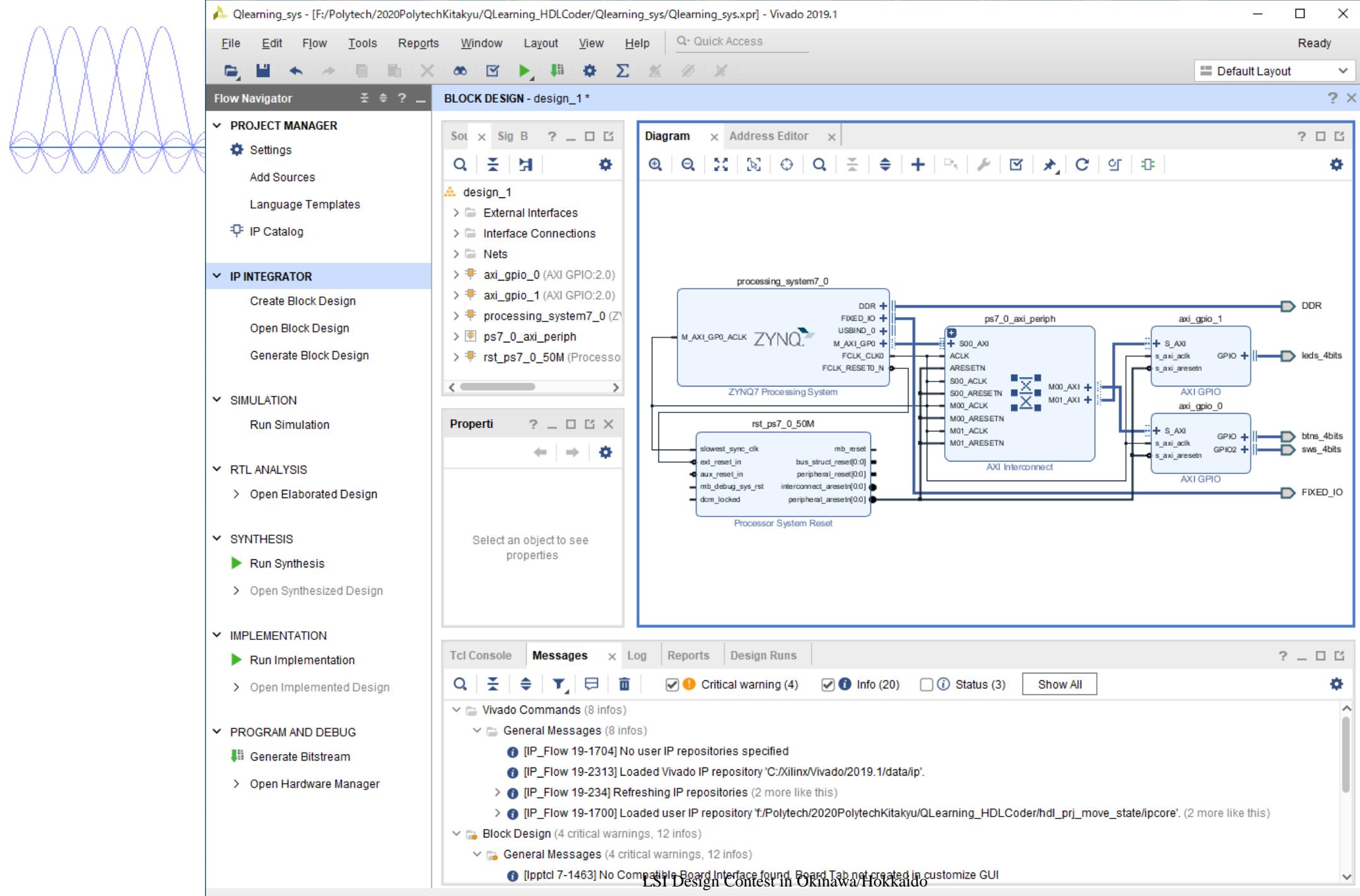


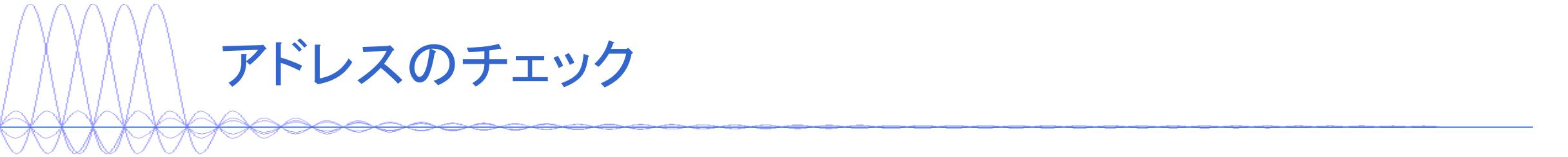




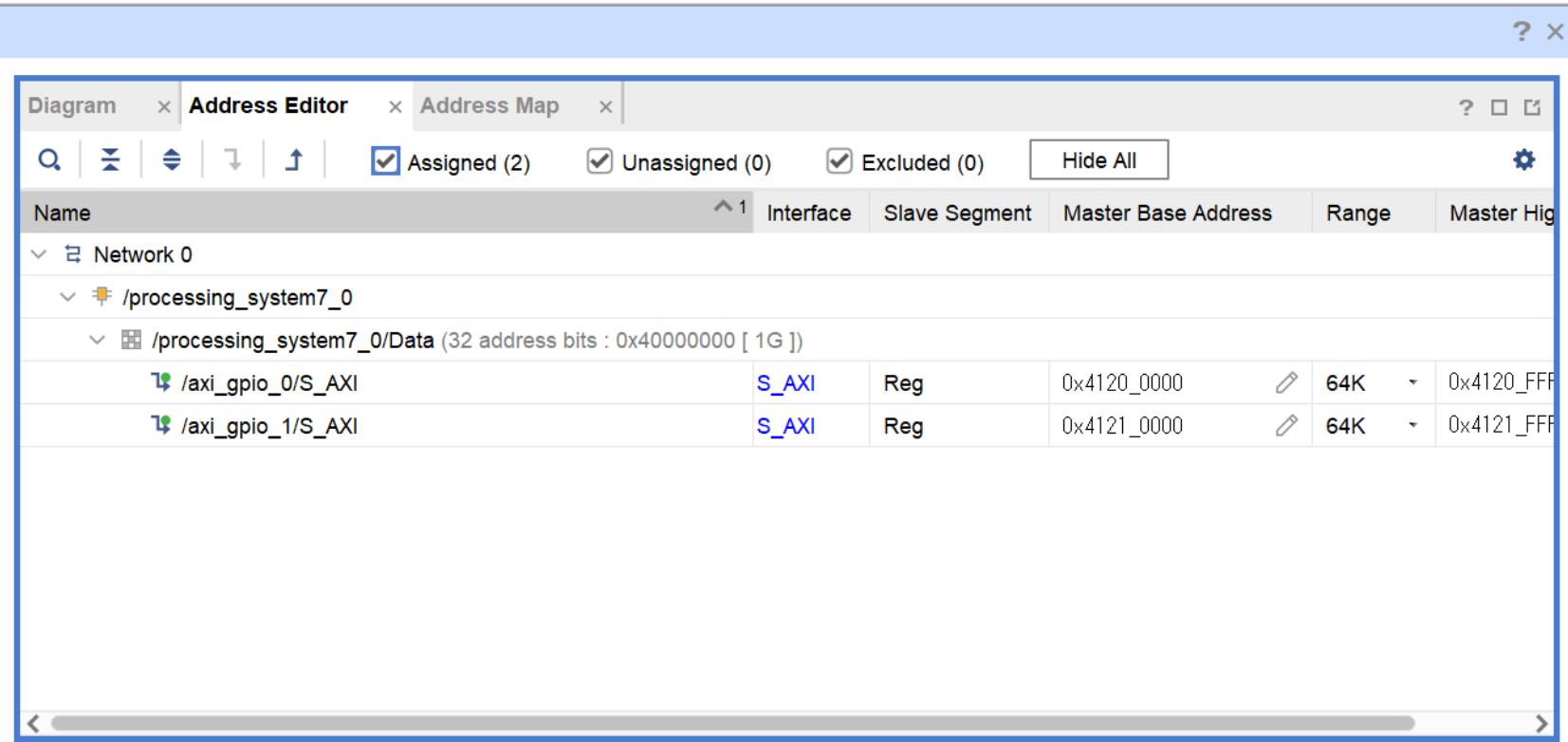








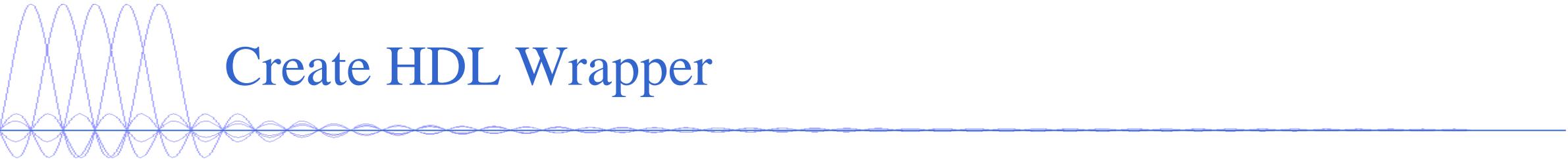
アドレスのチェック



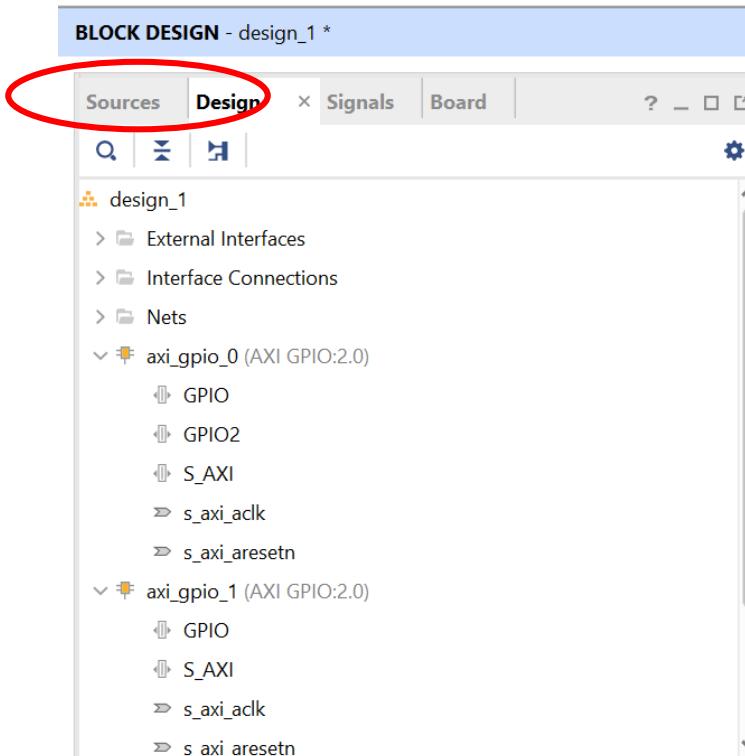
The screenshot shows the Address Map editor interface. The top navigation bar includes tabs for Diagram, Address Editor (selected), and Address Map. Below the tabs are search and filter tools, and checkboxes for Assigned (2), Unassigned (0), and Excluded (0). A Hide All button and a settings gear icon are also present.

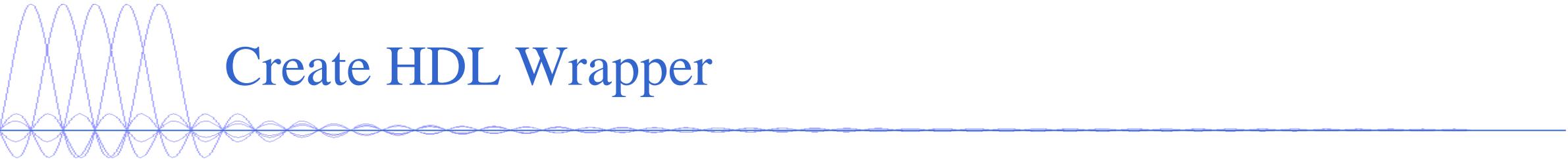
The main table displays memory assignments under the Network 0 section. The table columns are Name, Interface, Slave Segment, Master Base Address, Range, and Master High. Two entries are listed:

Name	Interface	Slave Segment	Master Base Address	Range	Master High
/processing_system7_0					
/processing_system7_0/Data (32 address bits : 0x40000000 [1G])					
/axi_gpio_0/S_AXI	S_AXI	Reg	0x4120_0000	64K	0x4120_FFF
/axi_gpio_1/S_AXI	S_AXI	Reg	0x4121_0000	64K	0x4121_FFF

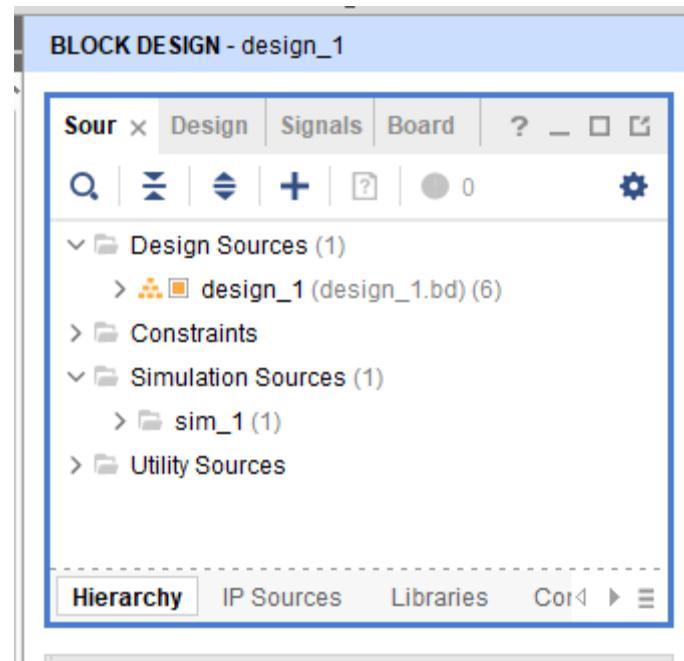


Create HDL Wrapper



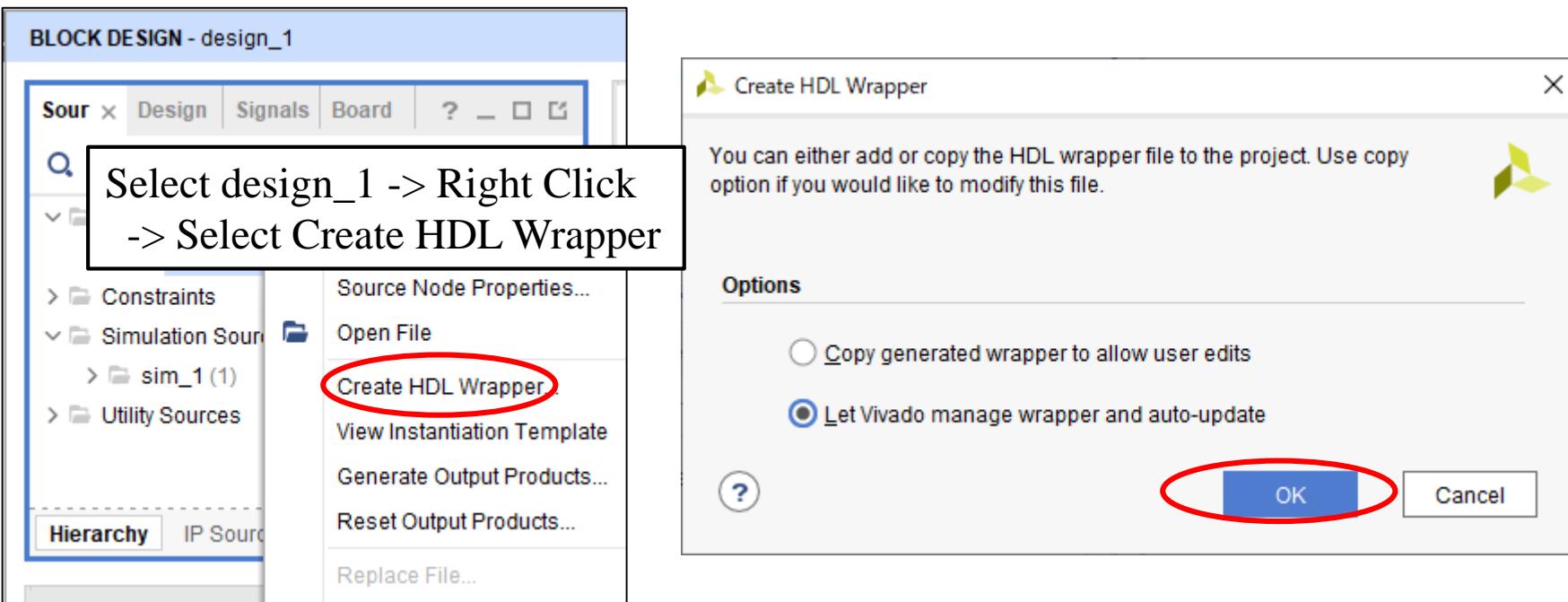


Create HDL Wrapper

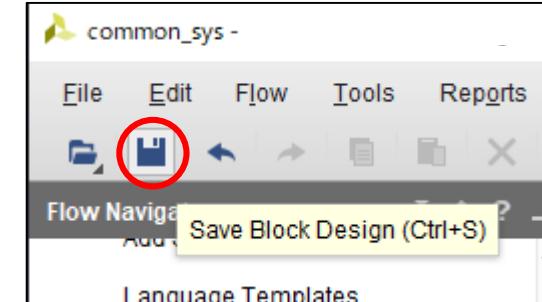
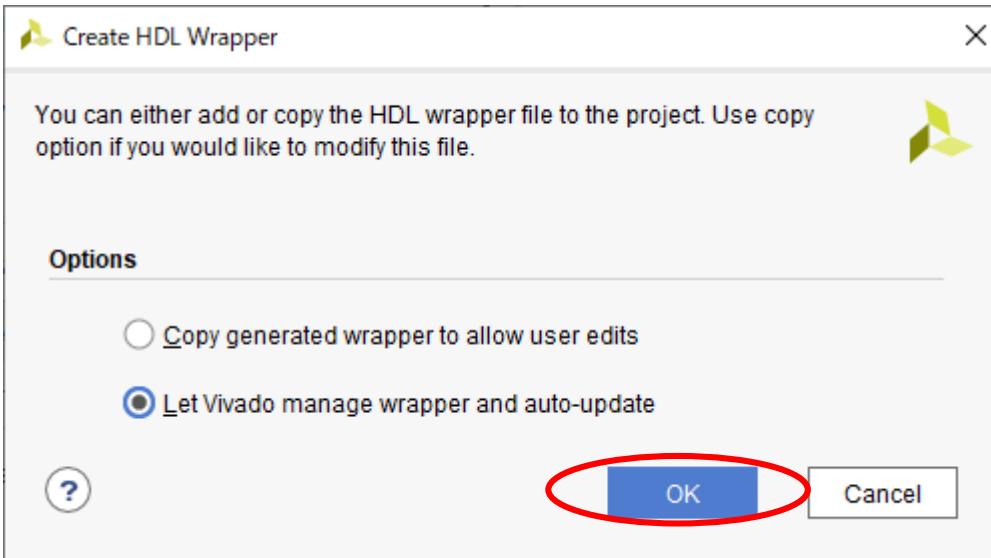


Create HDL Wrapper

design_1を一度左でクリックして、
その後右クリック

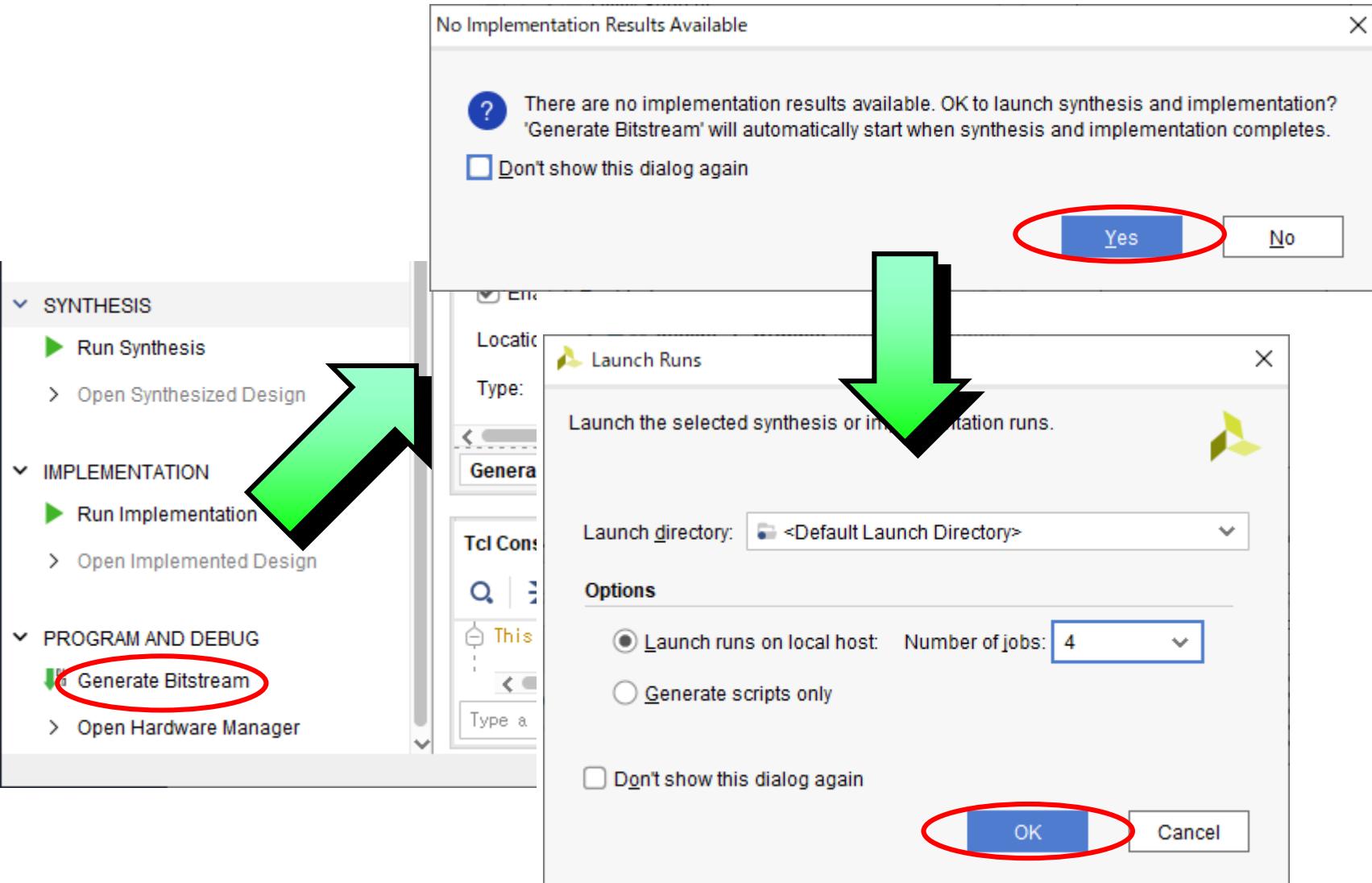


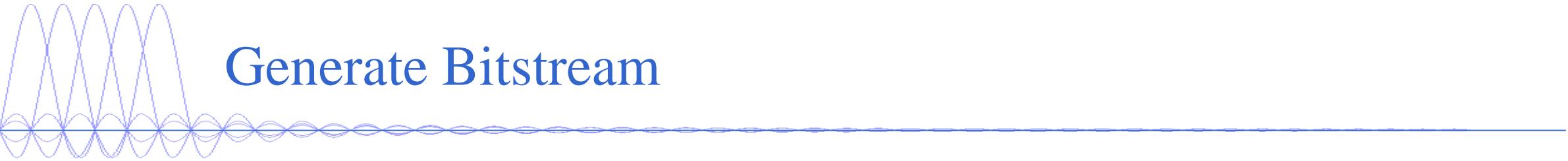
Create HDL Wrapper



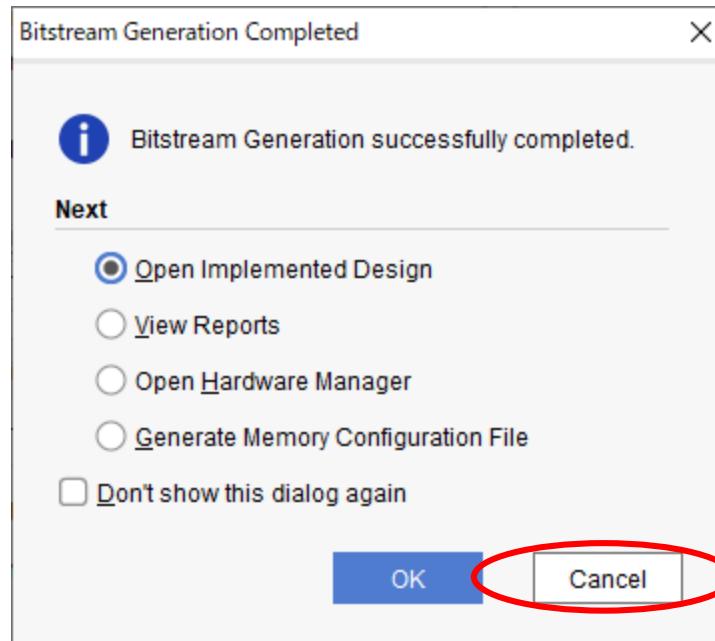
再びWrapperを作るとこのような
ダイアログが表示される

Generate Bitstream

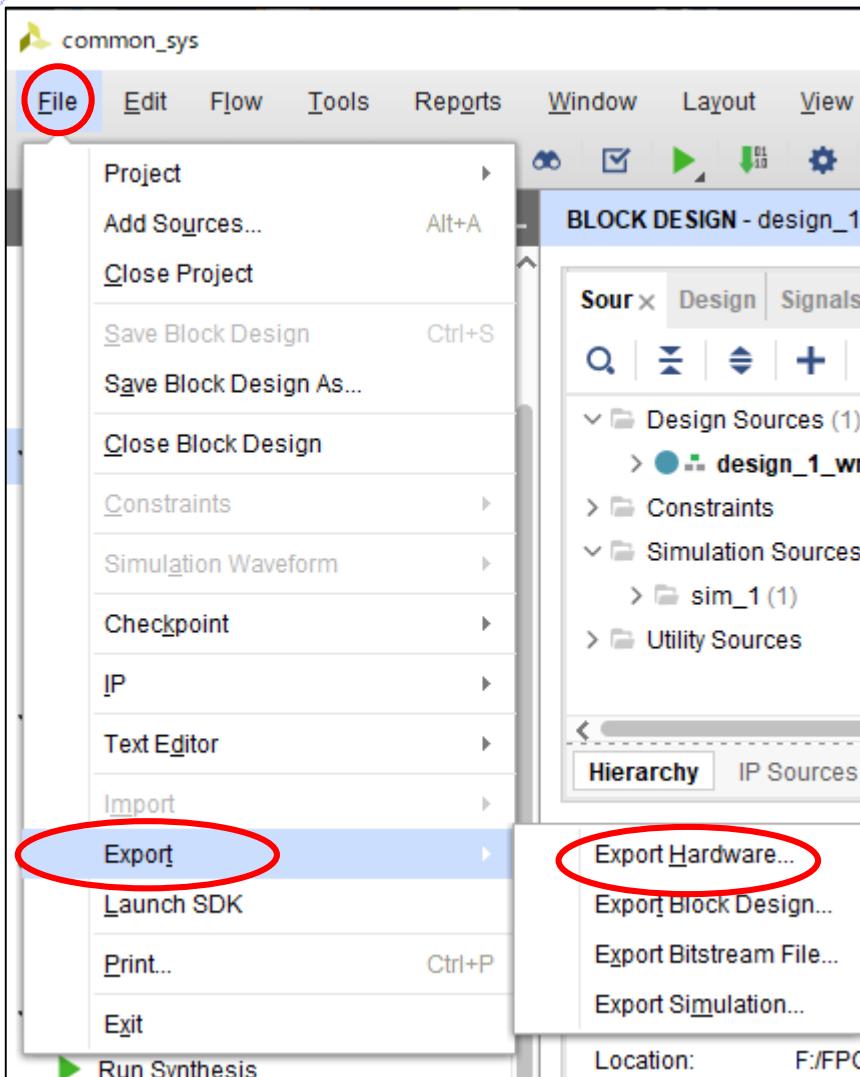




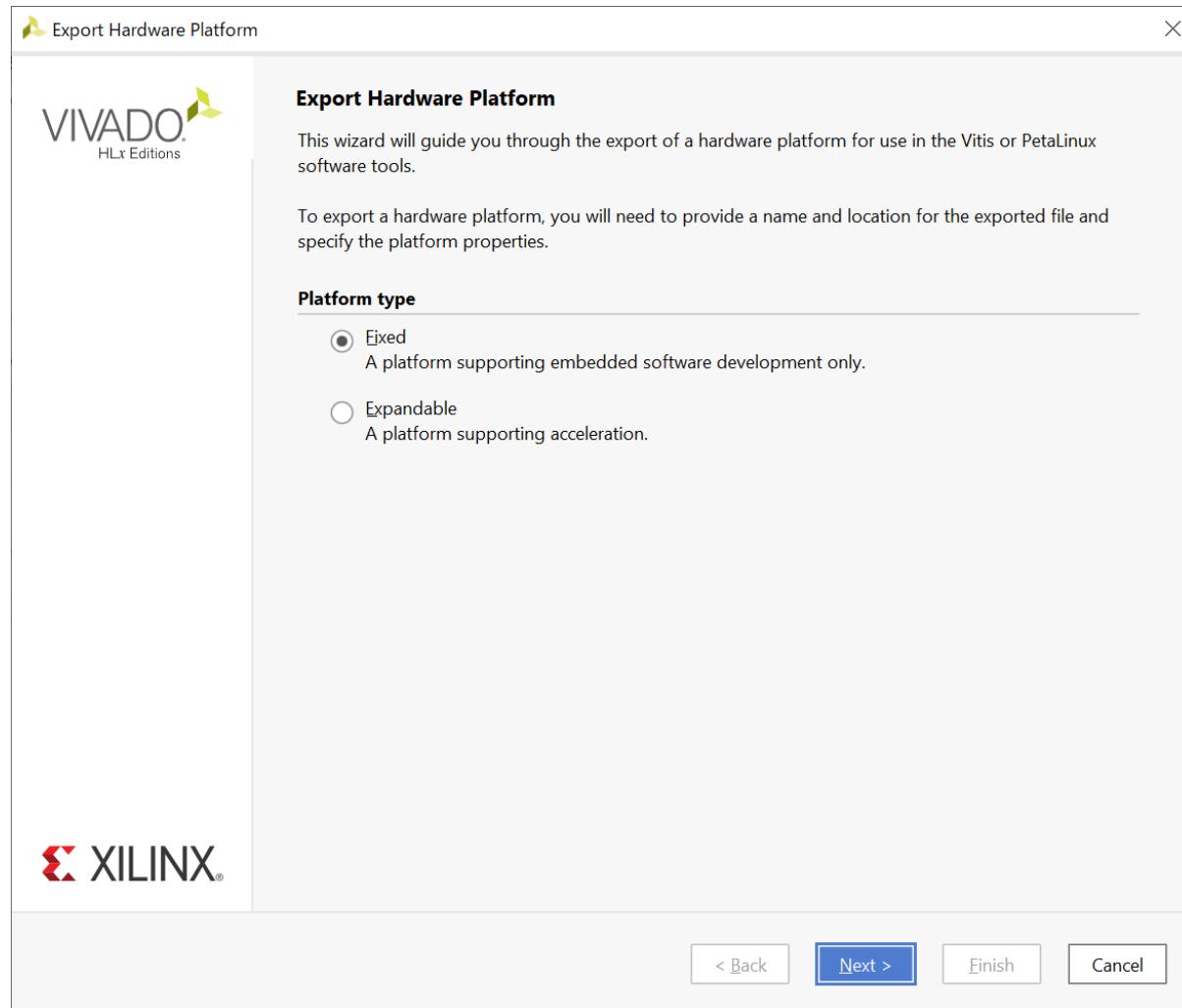
Generate Bitstream



Export Hardware

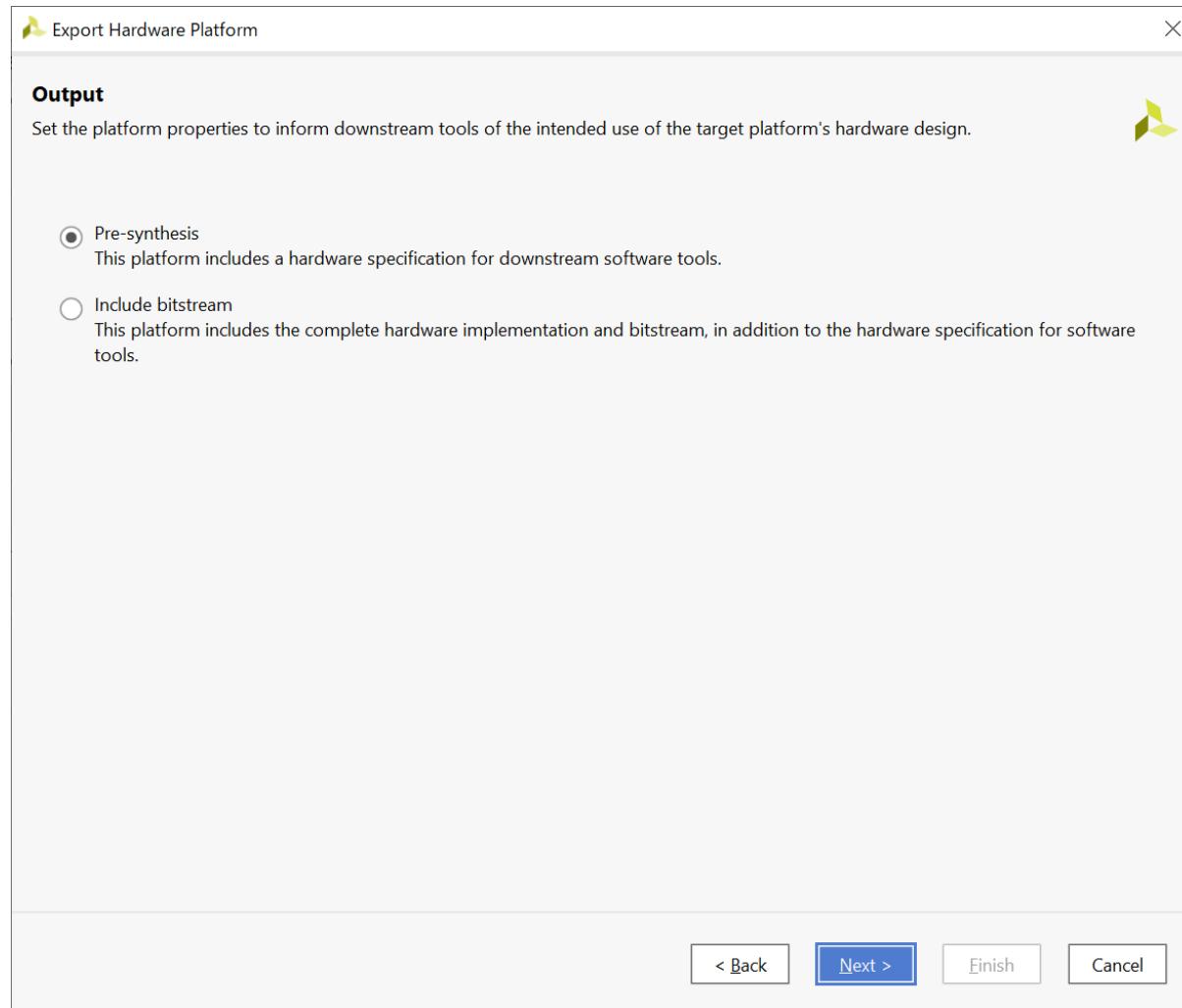


Export Hardwareの設定



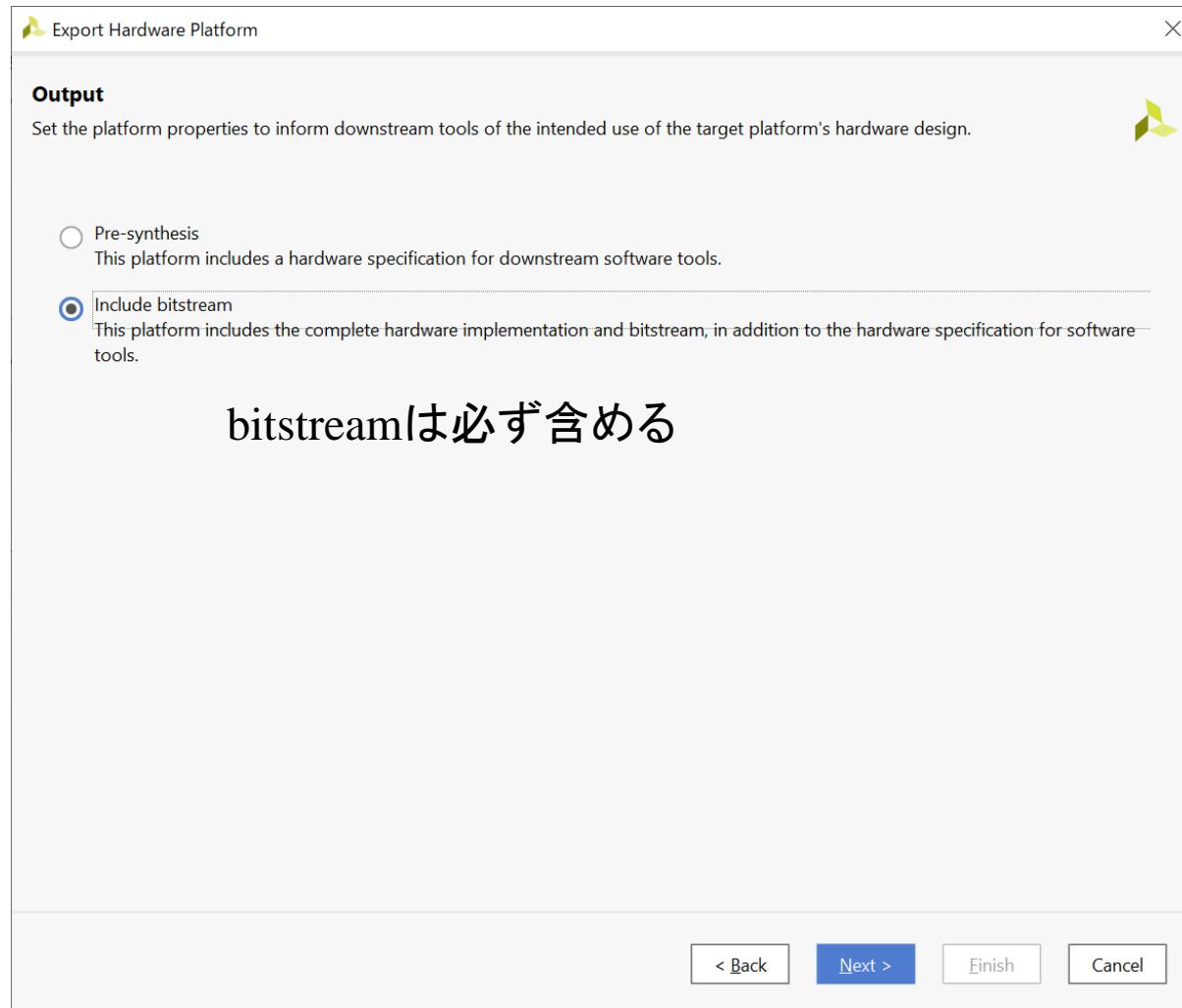


Export Hardwareの設定



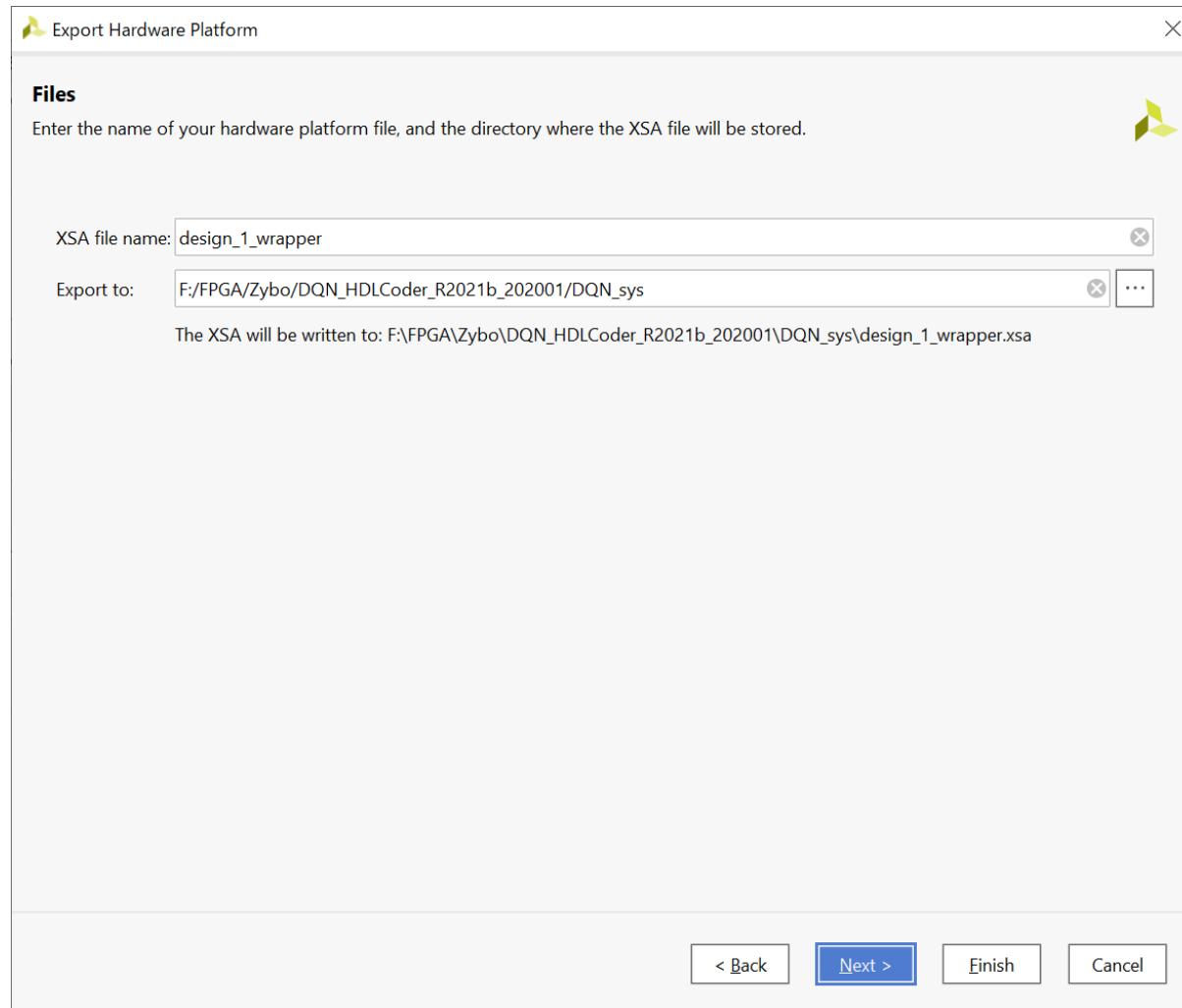


Export Hardwareの設定



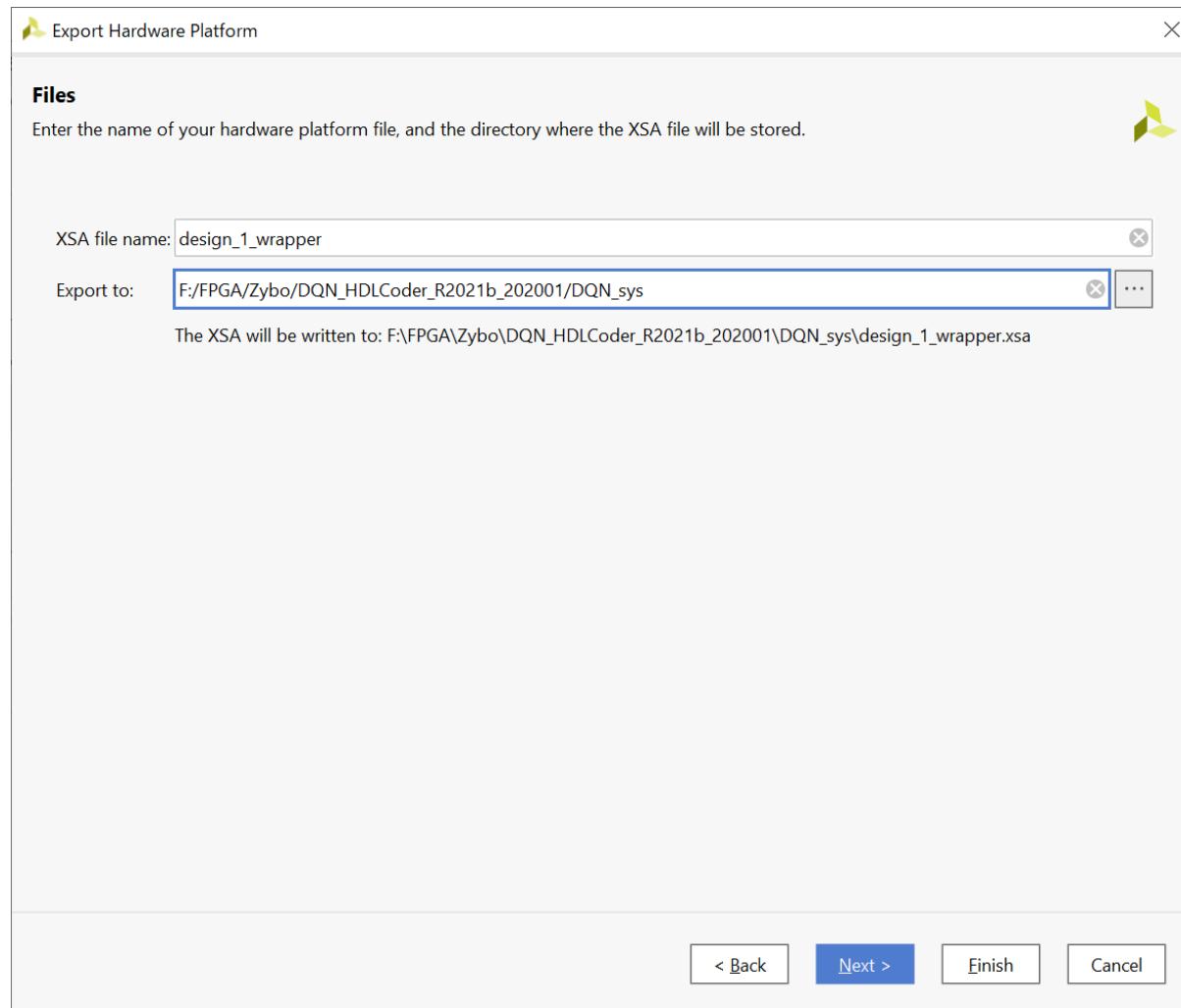


Export Hardwareの設定

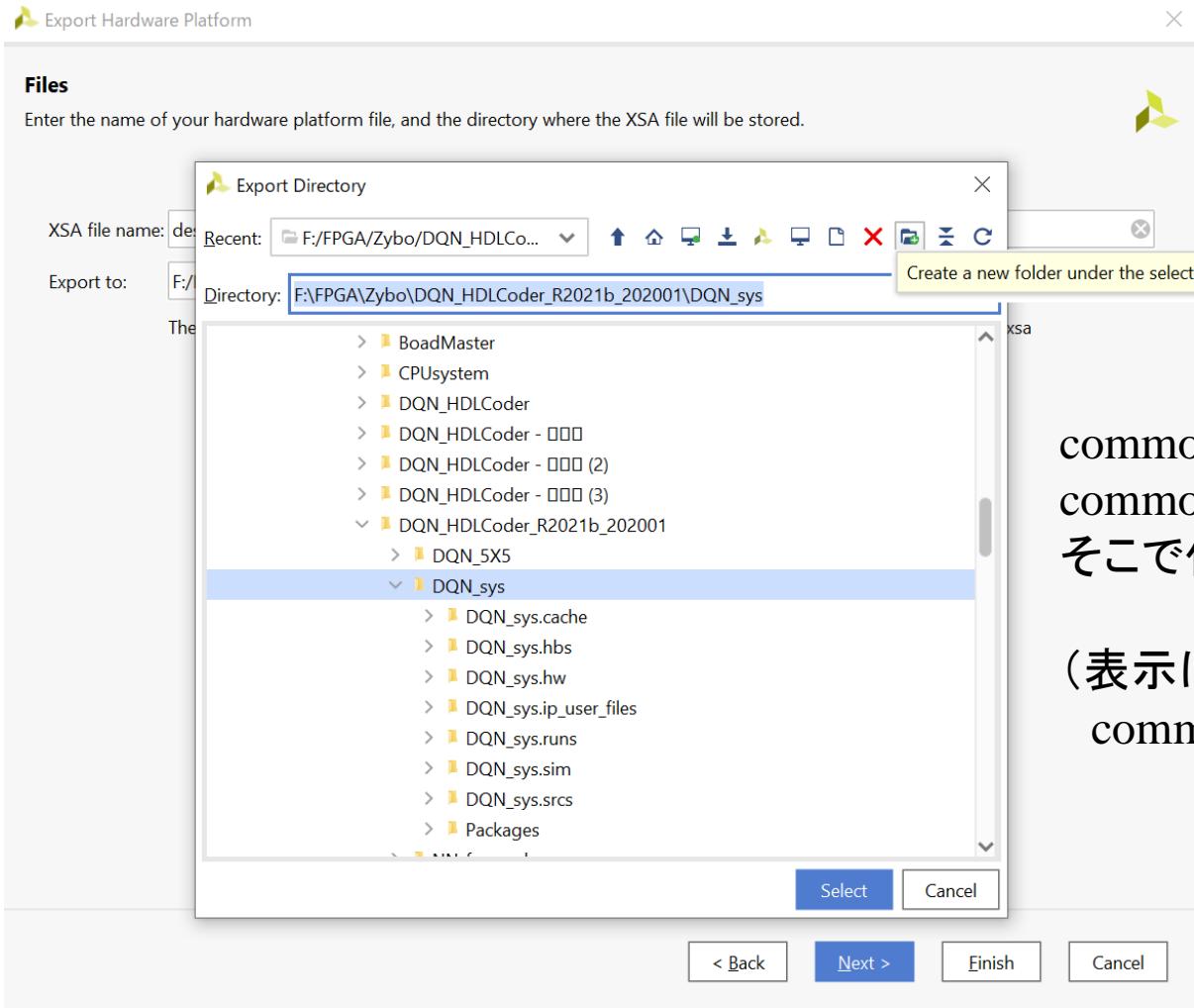




Export Hardwareの設定



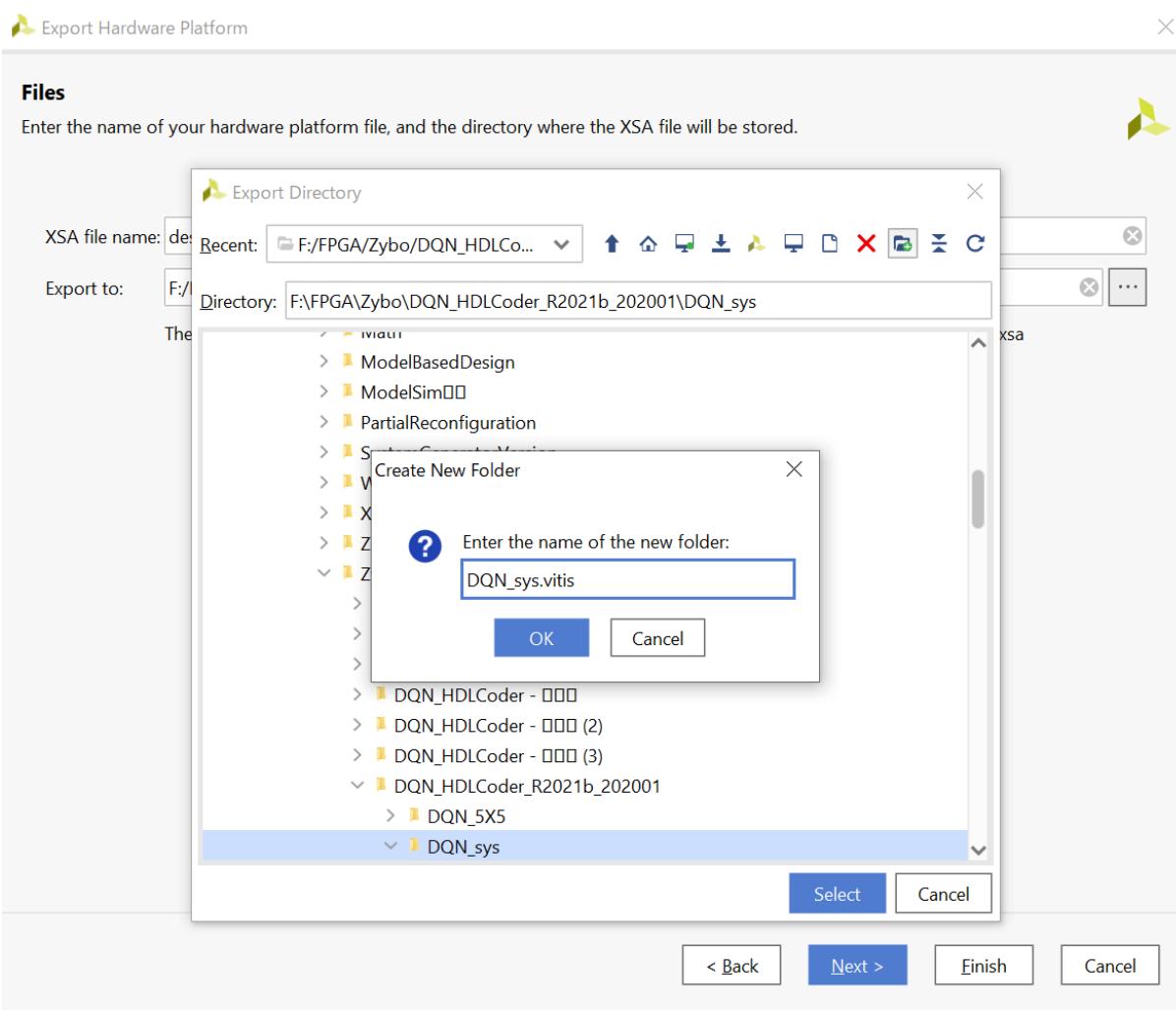
Export Hardwareの設定 - フォルダの作成 -



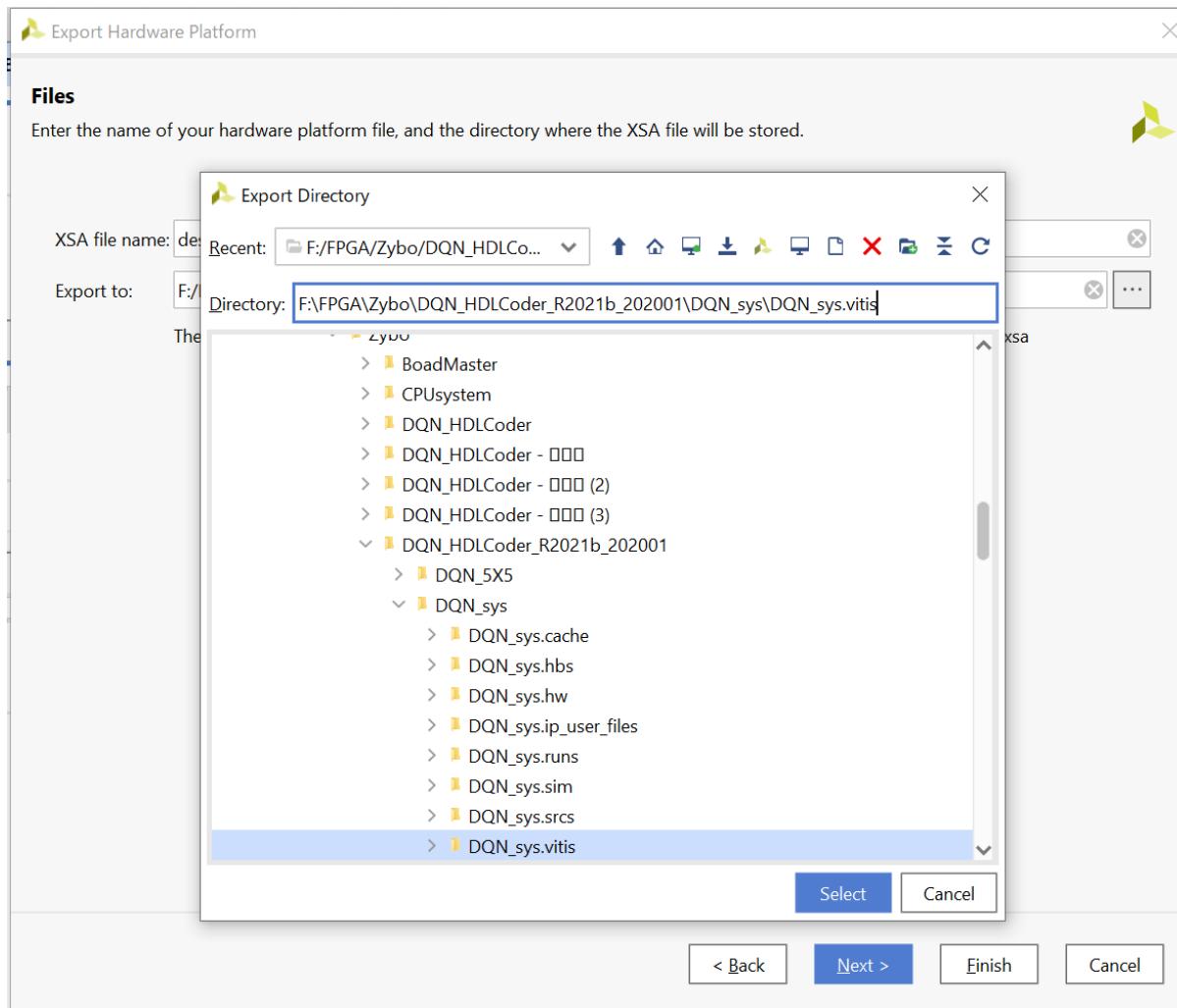
common_sysの下に
common_sys.vitisを作成し、
そこで作業を行う

(表示はDQNになっていますが、
common_sysでお願いします)

Export Hardwareの設定 - フォルダの作成 -

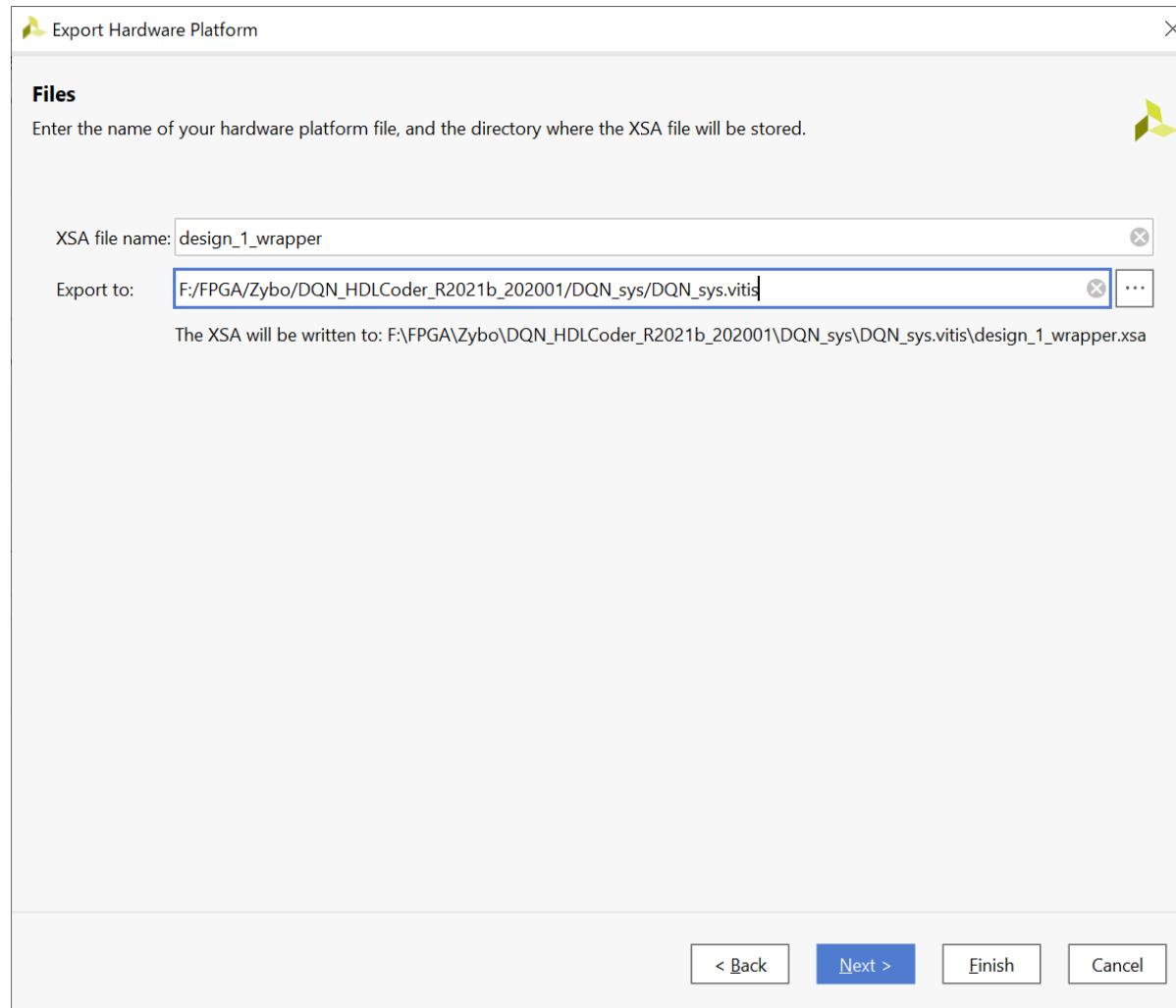


Export Hardwareの設定 - フォルダの設定 -

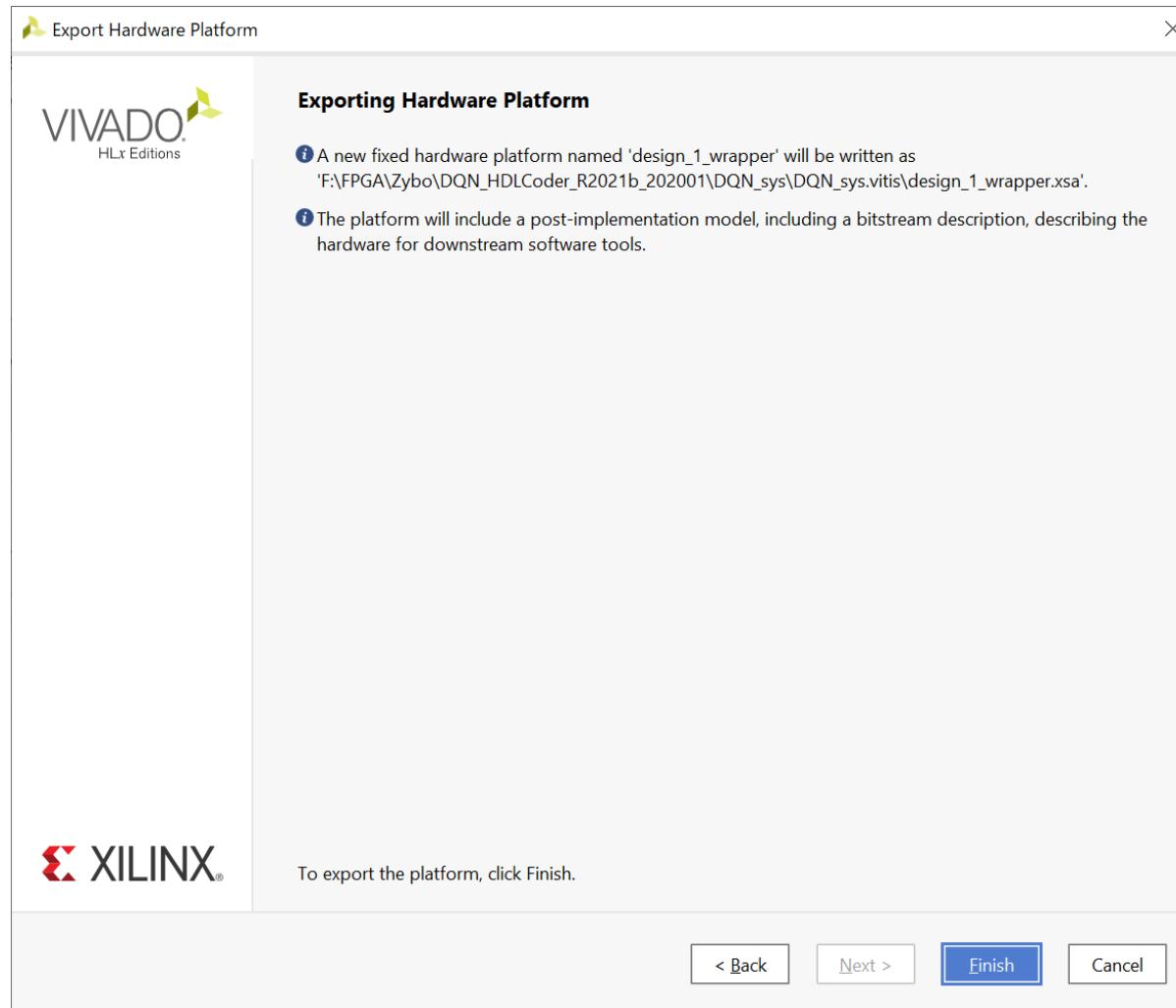




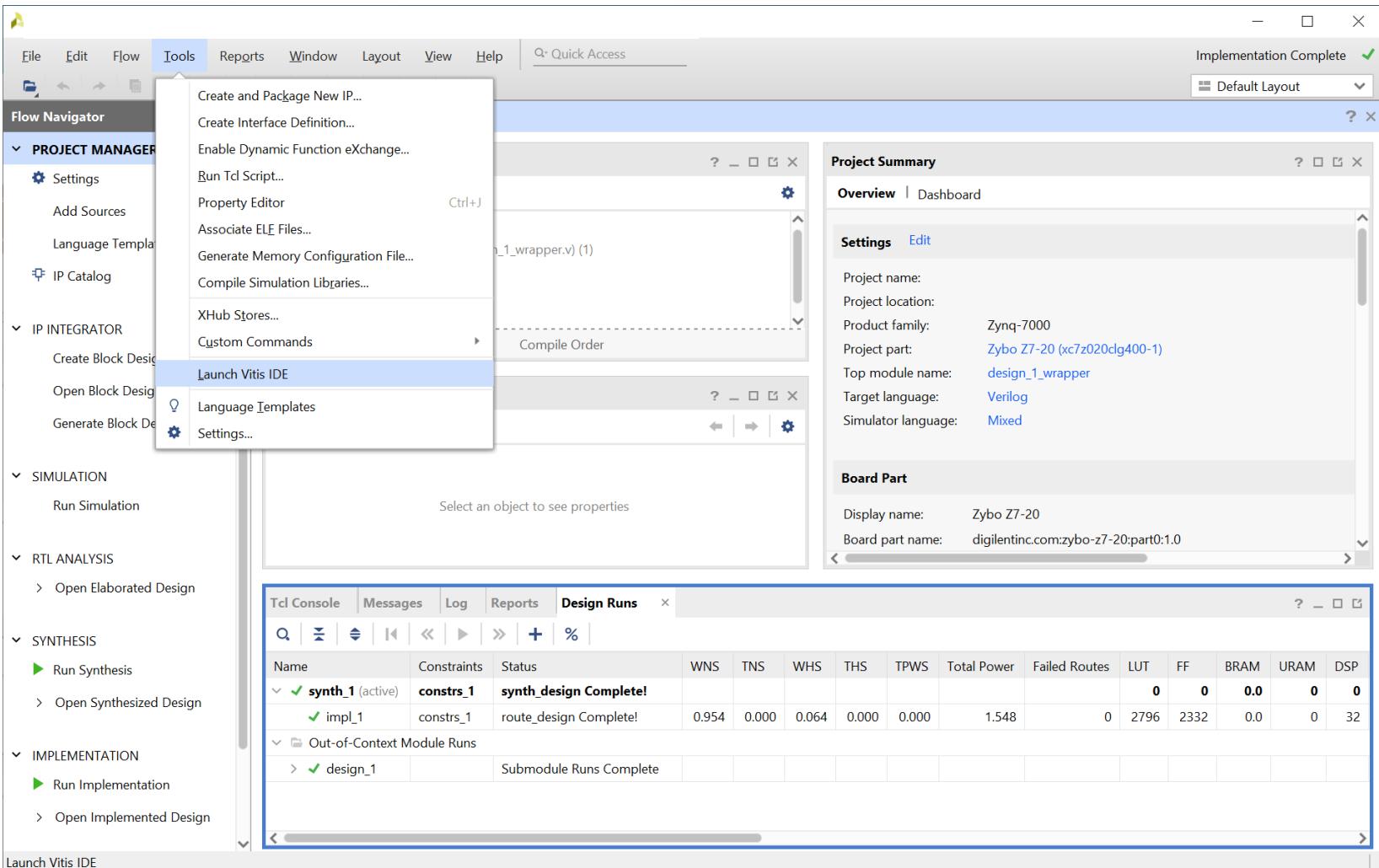
Export Hardwareの設定

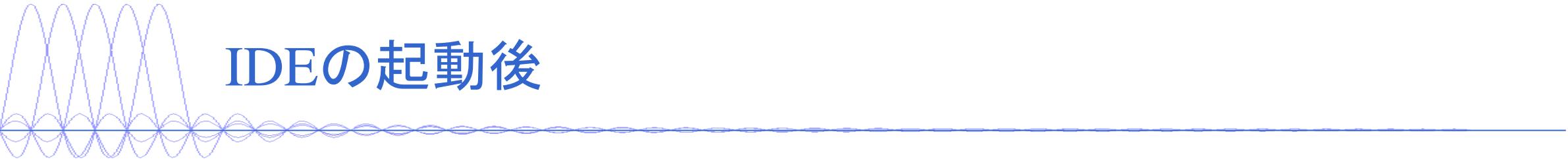


Export Hardwareの設定

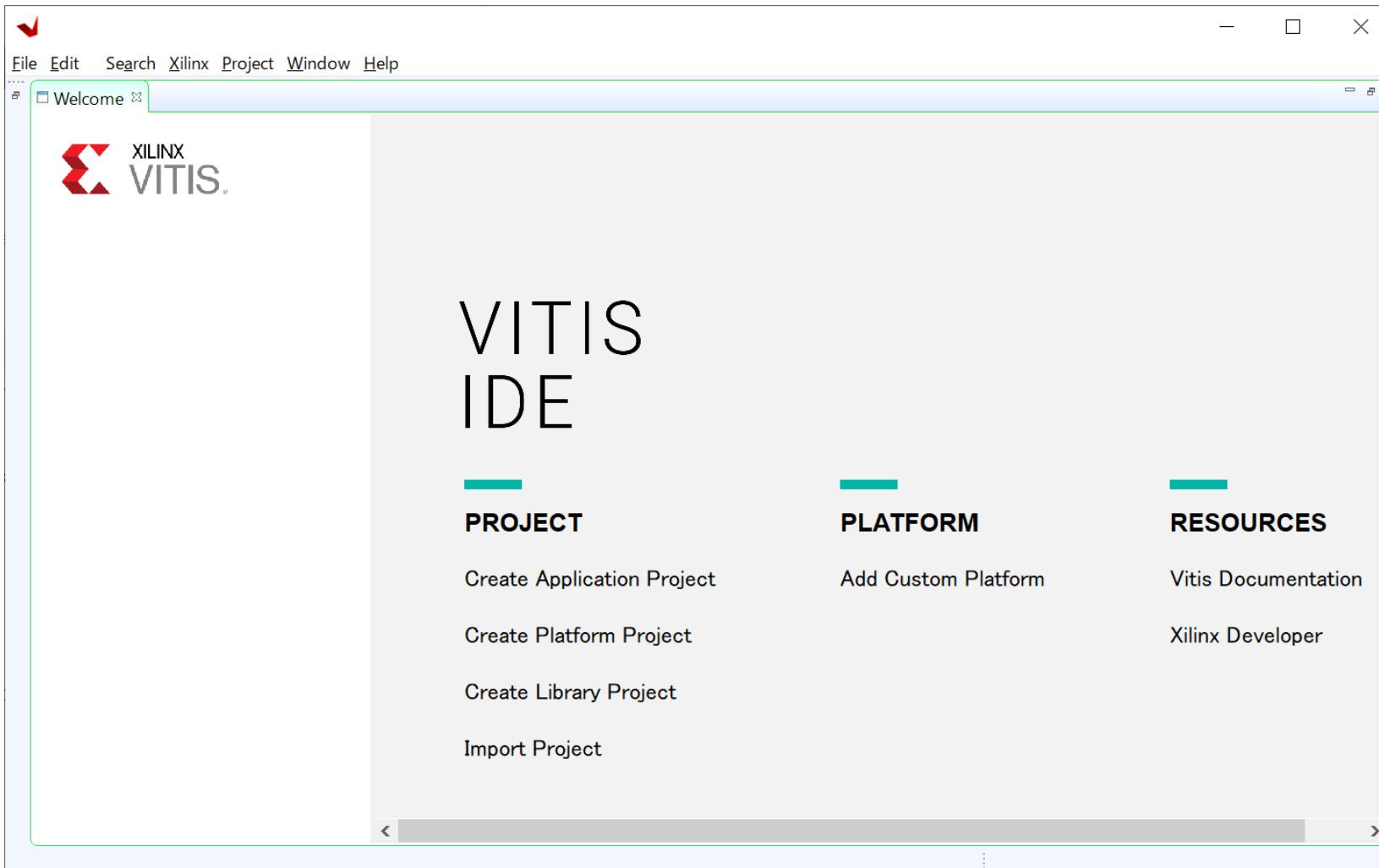


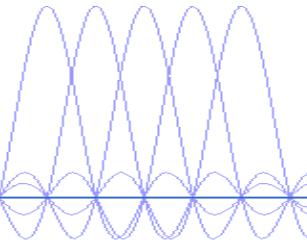
Vitis の起動



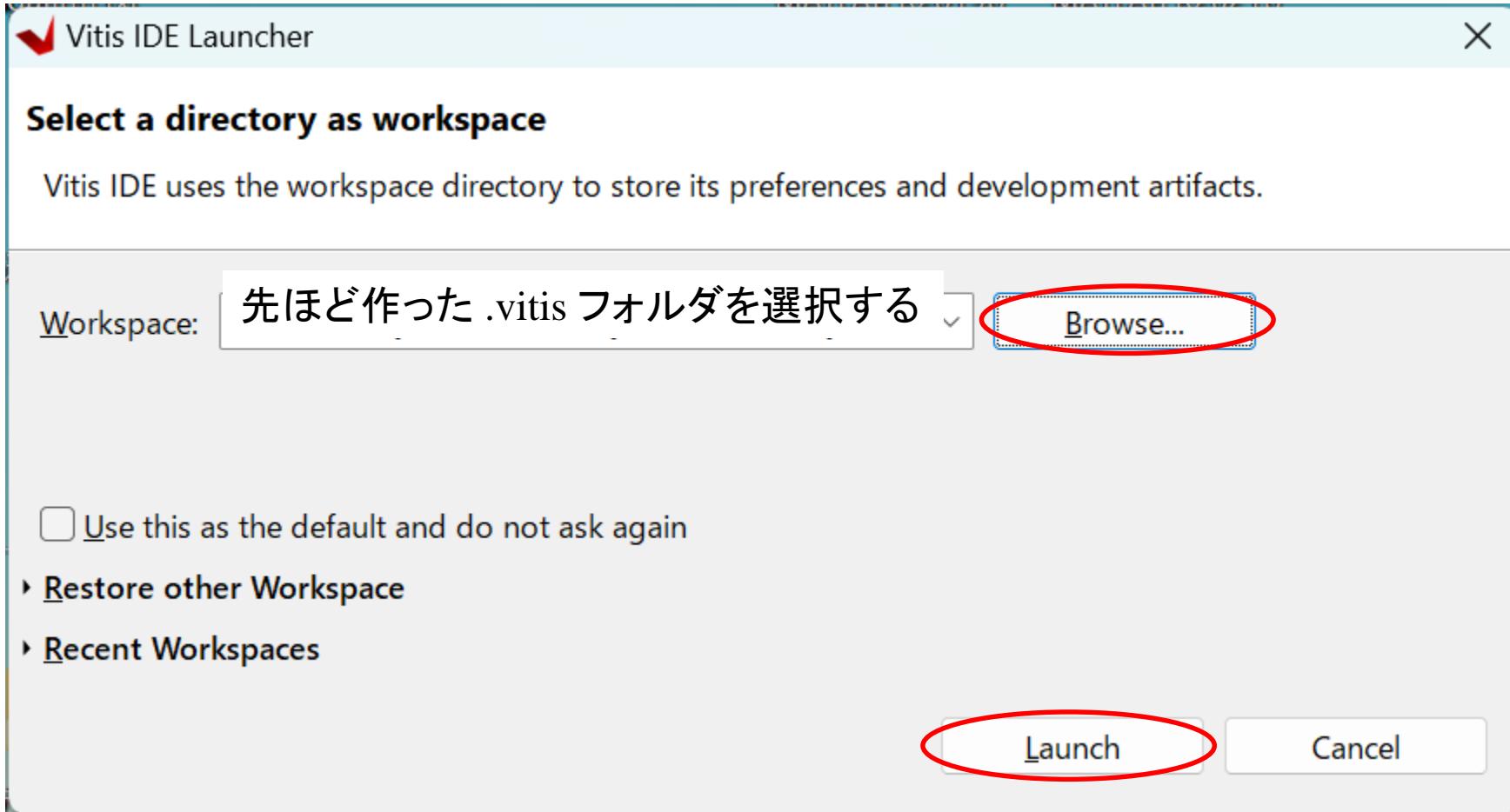


IDEの起動後

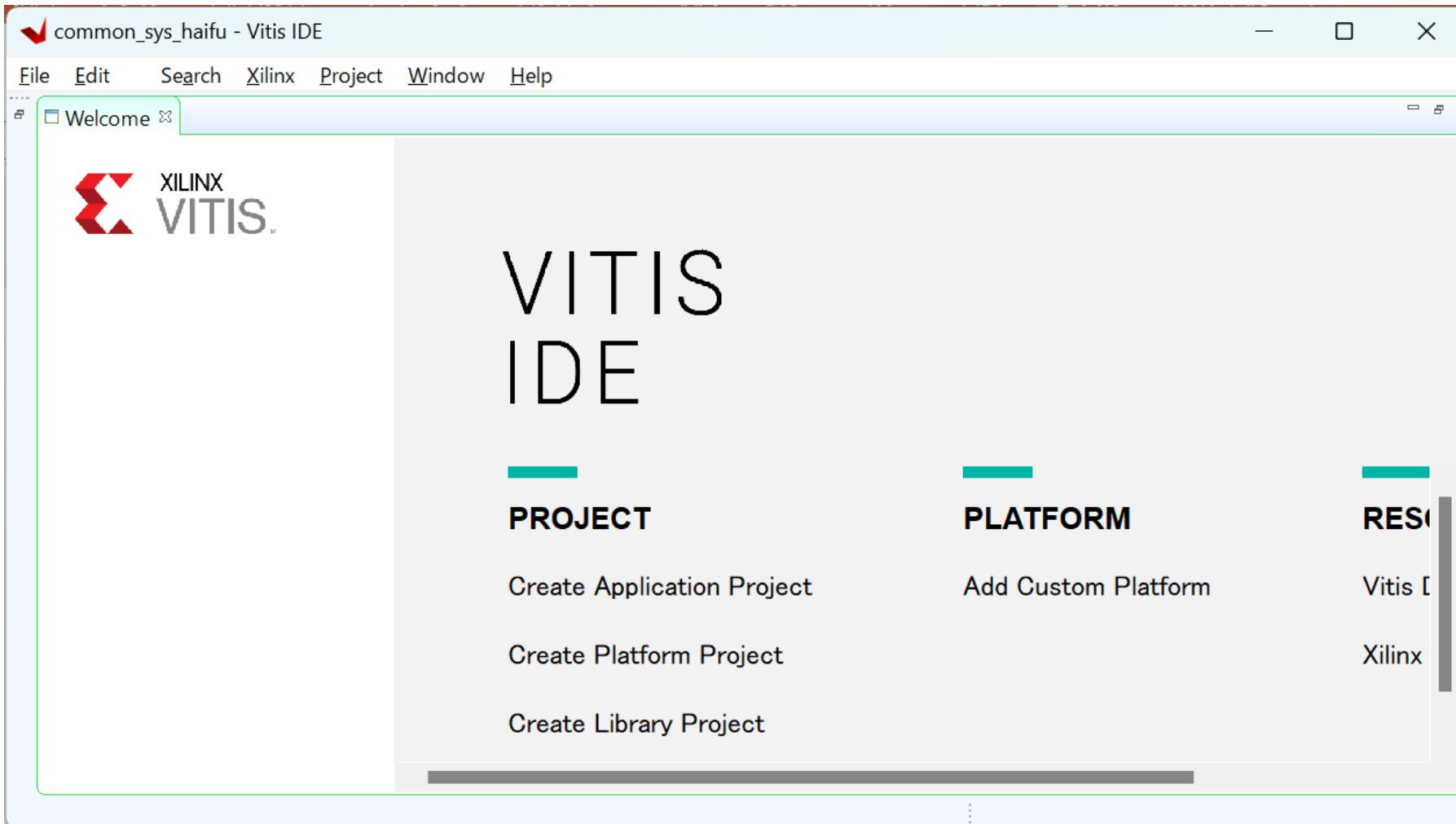




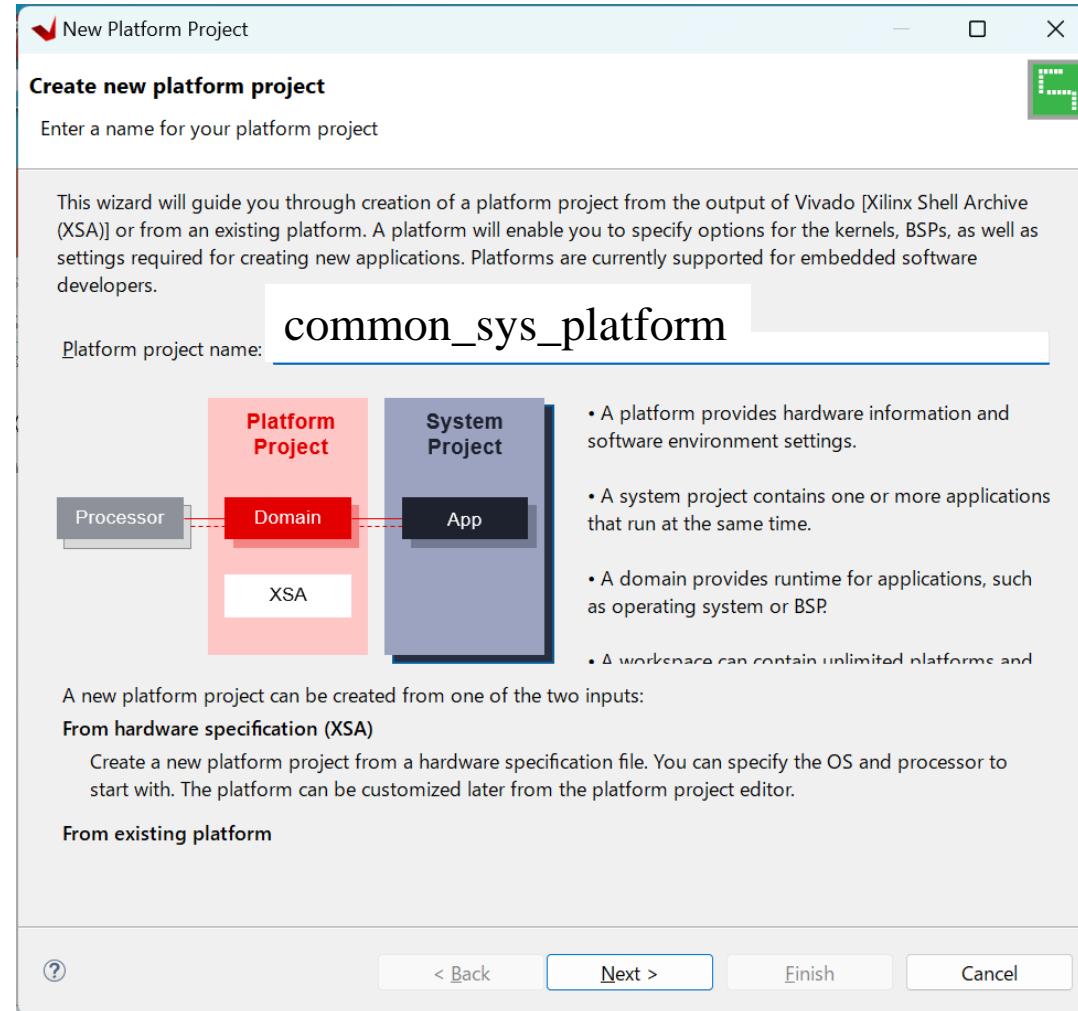
Vitisの起動: Workspaceの設定



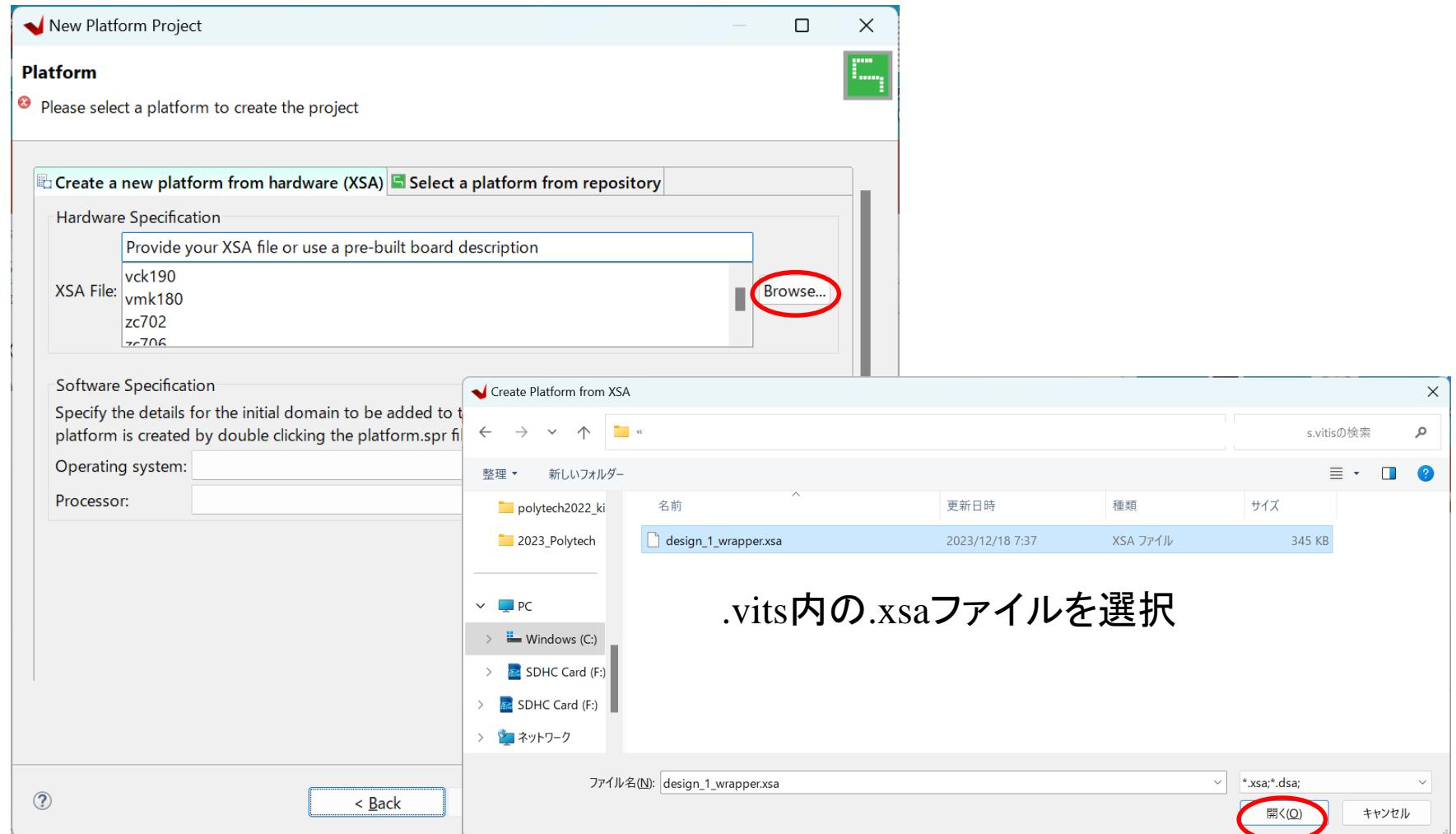
Create Platform Project



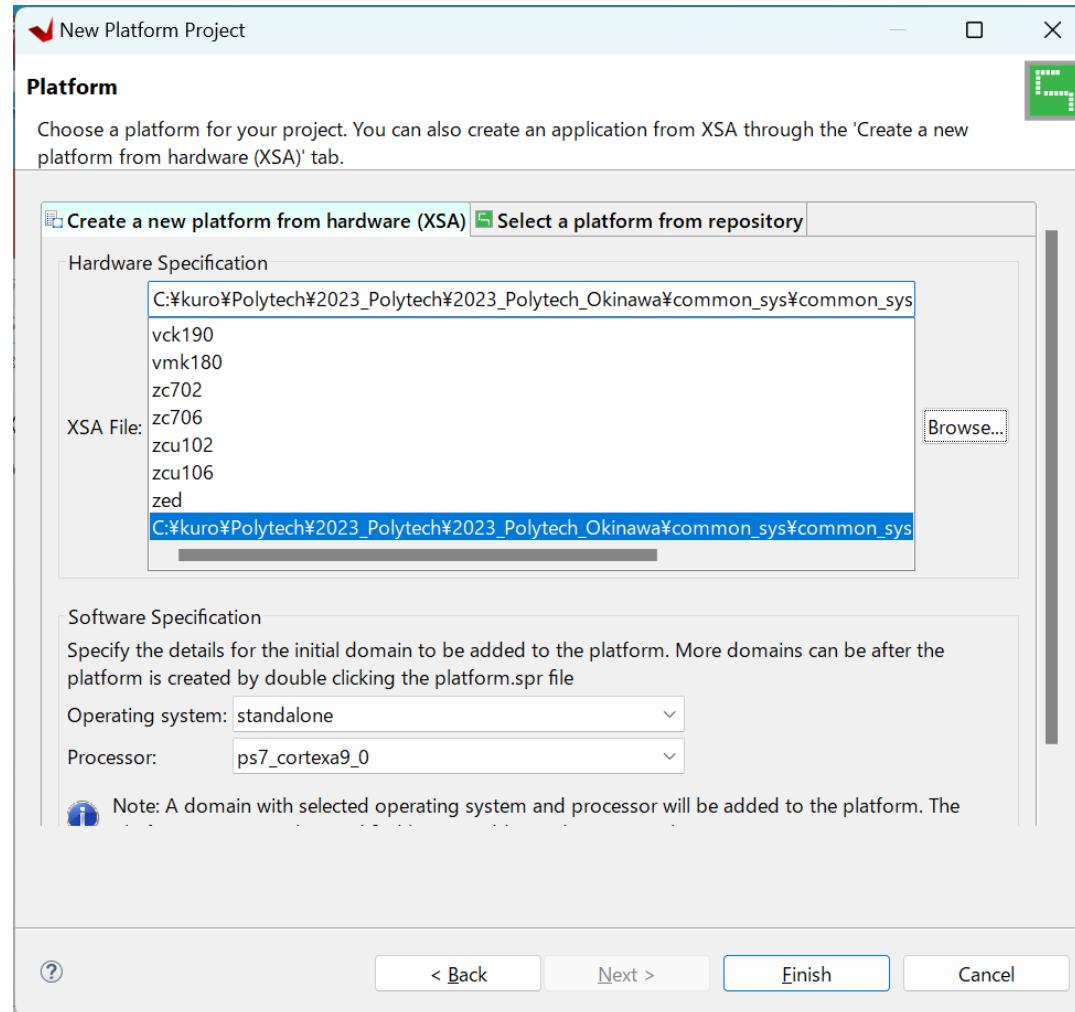
Create Platform Project



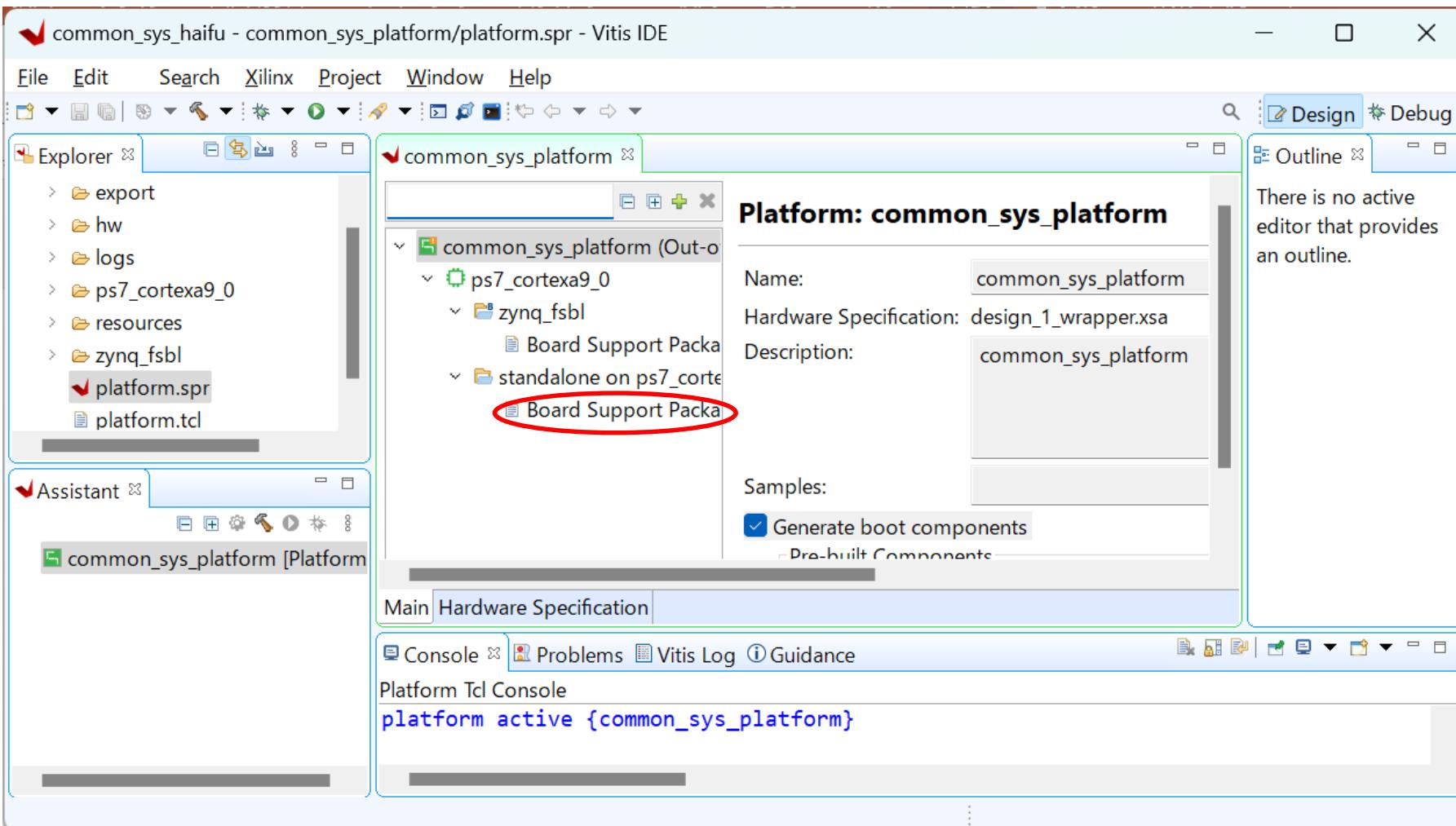
Create Platform Project



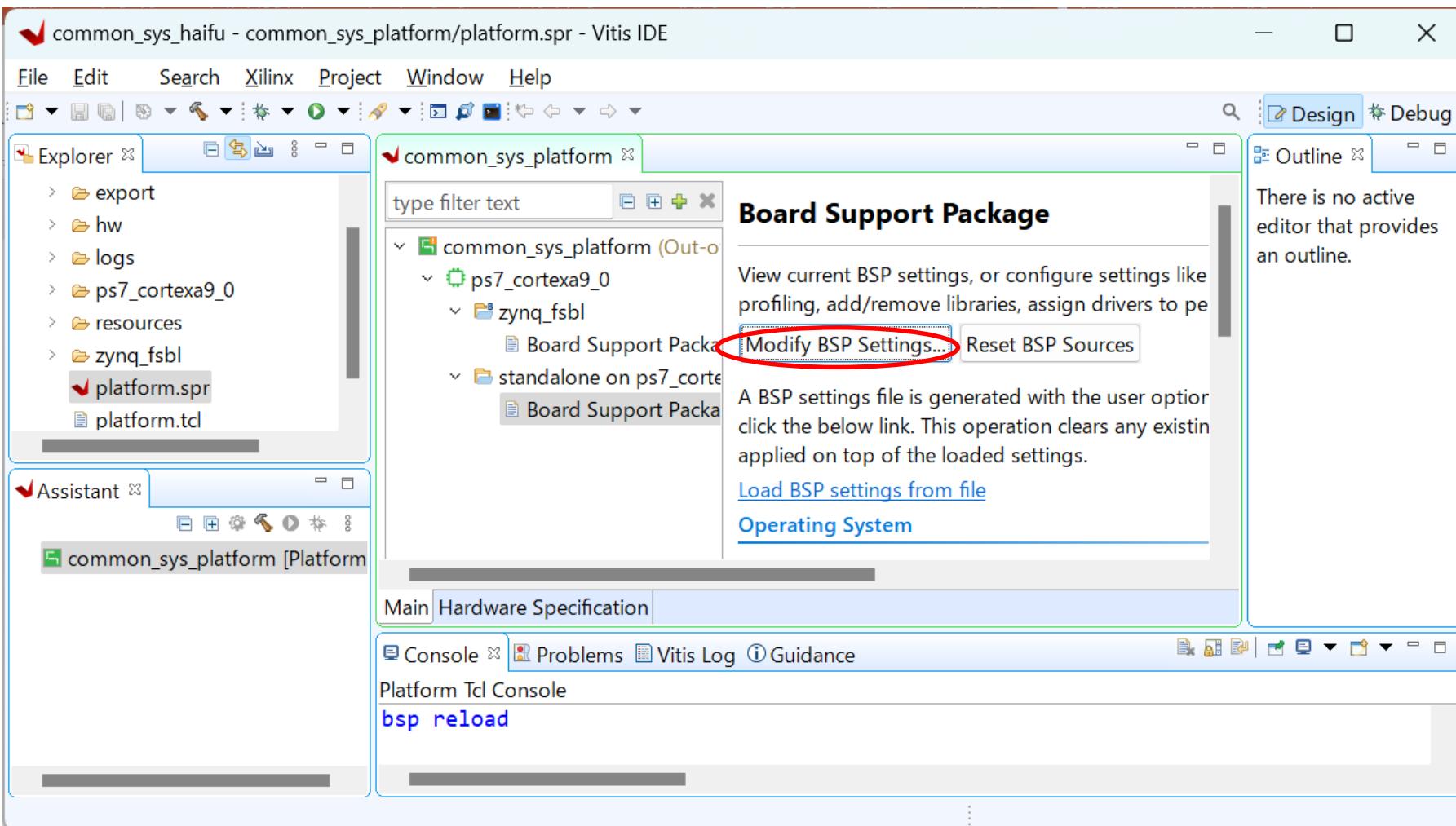
Create Platform Project



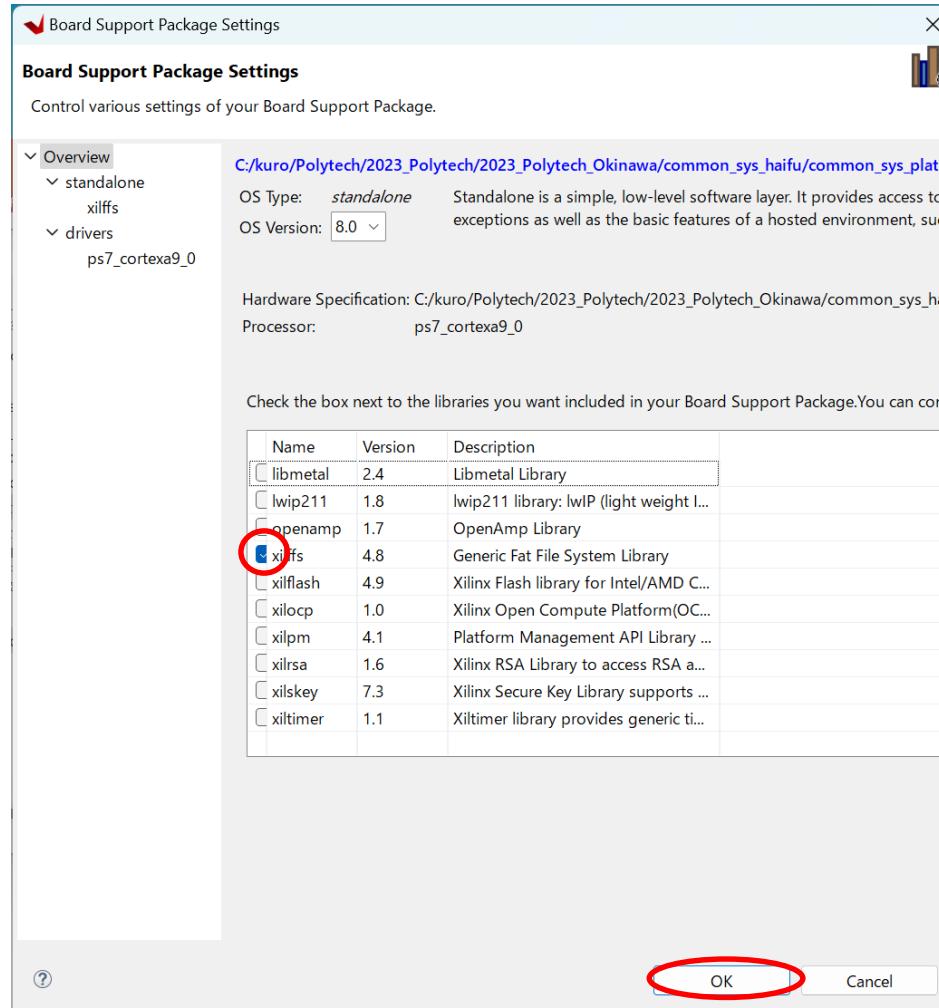
Modify BSP Settings



Modify BSP Settings

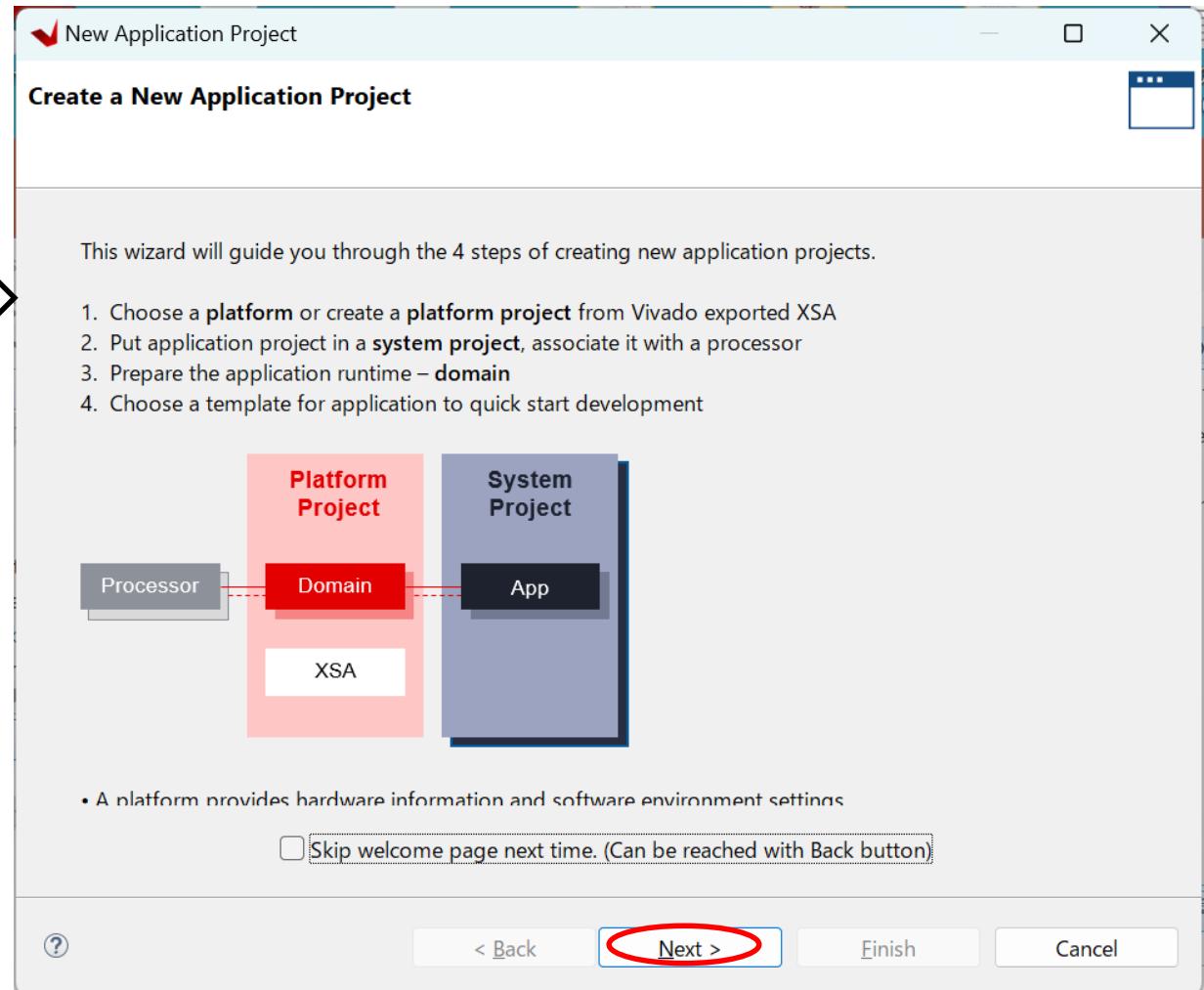
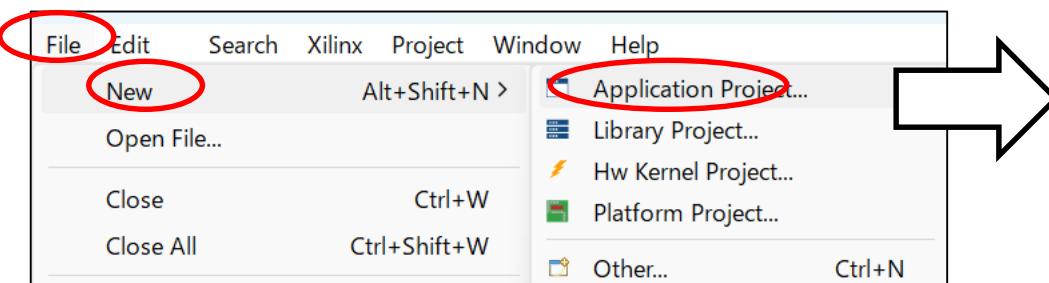


Modify BSP Settings

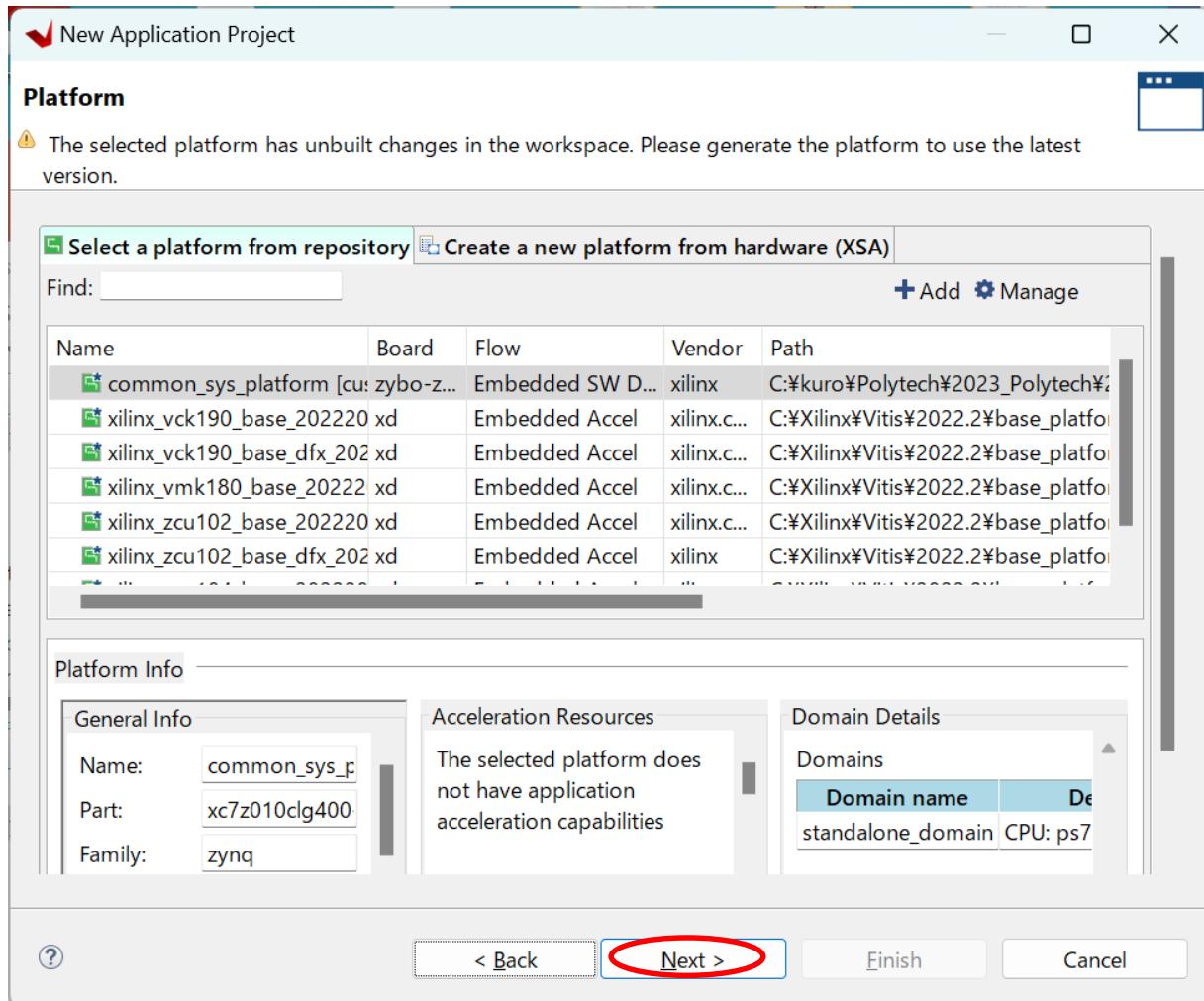


SDカードが使えるように
xilffsのライブラリを取り込む

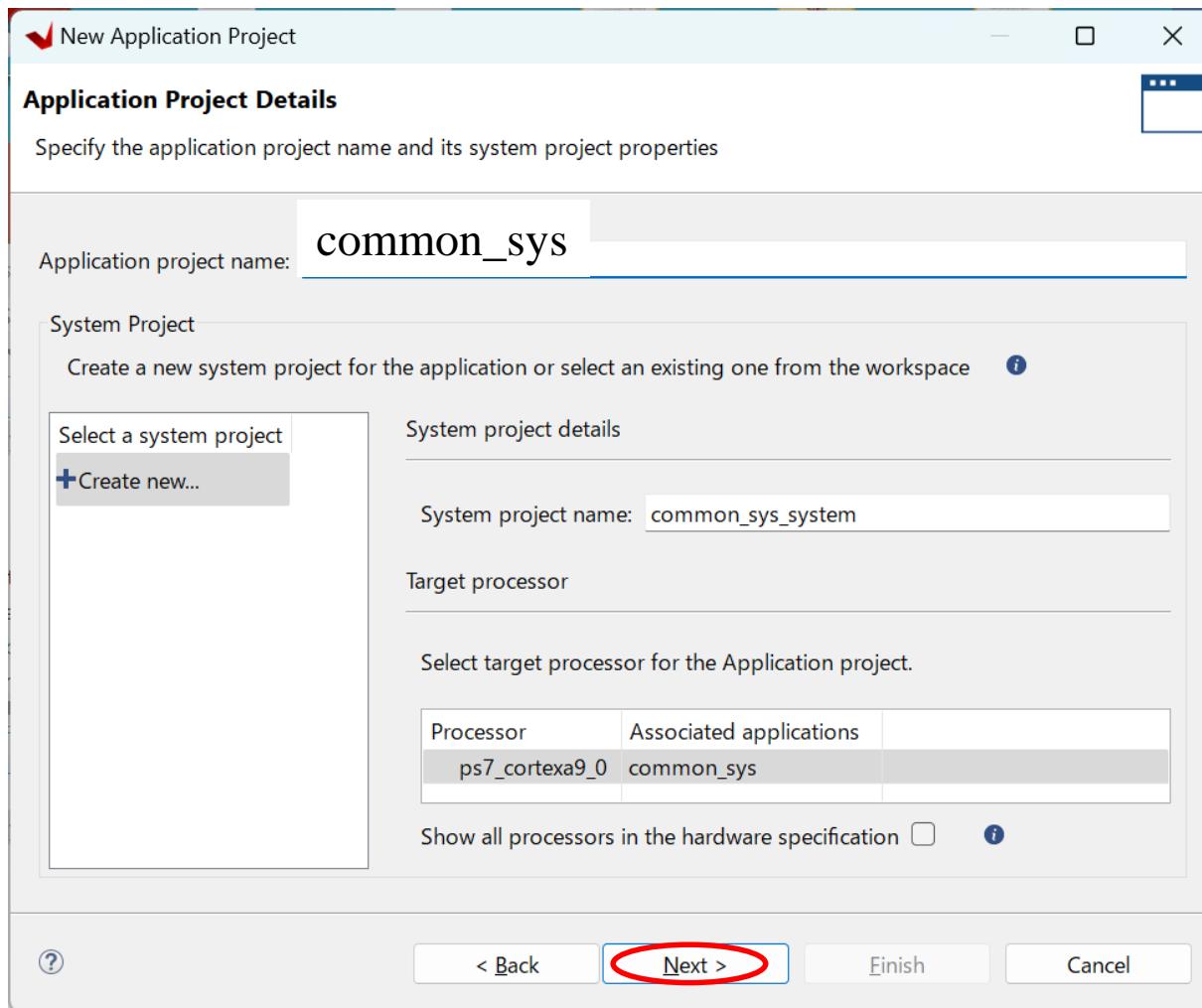
Create Application Project



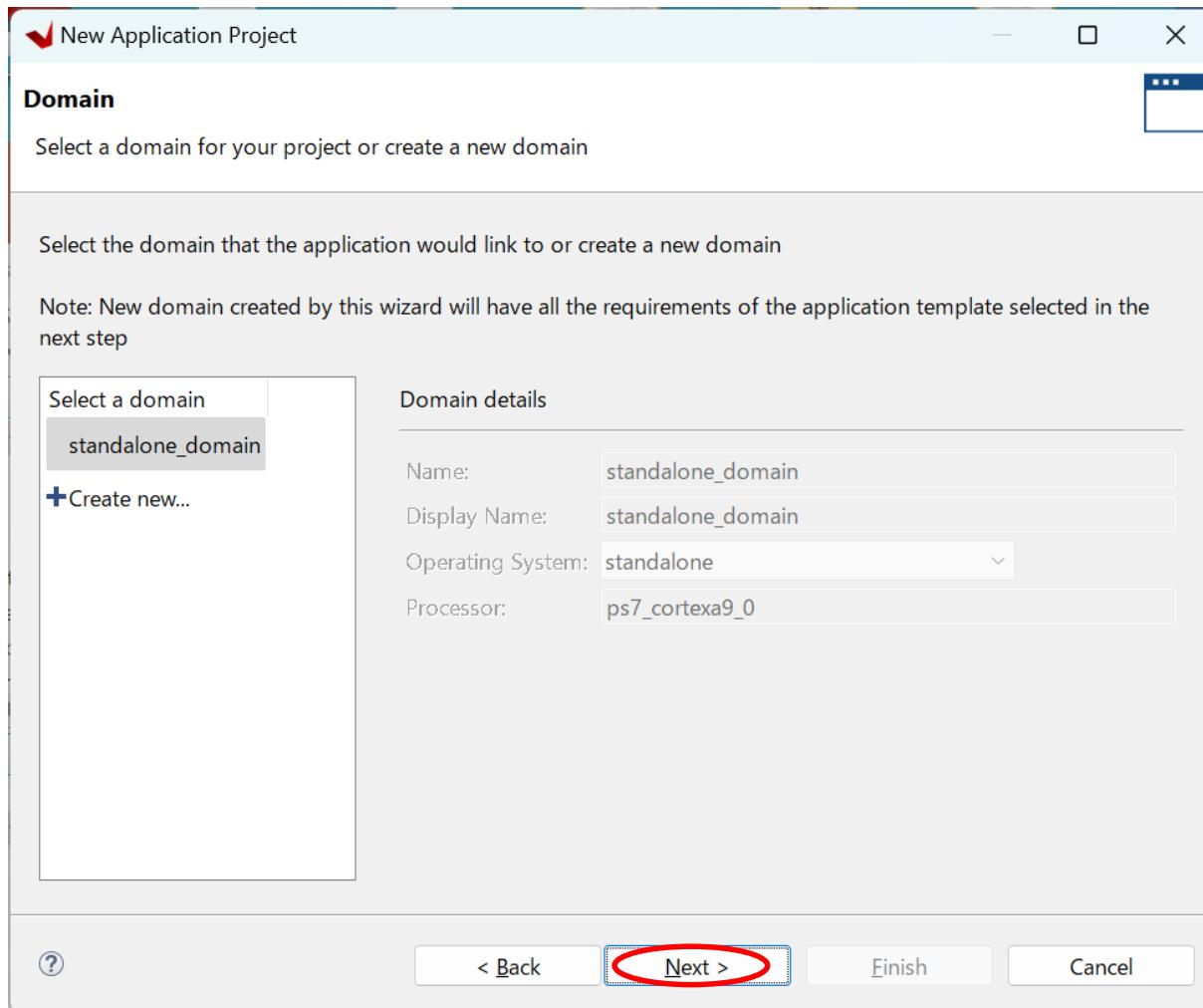
Create Application Project



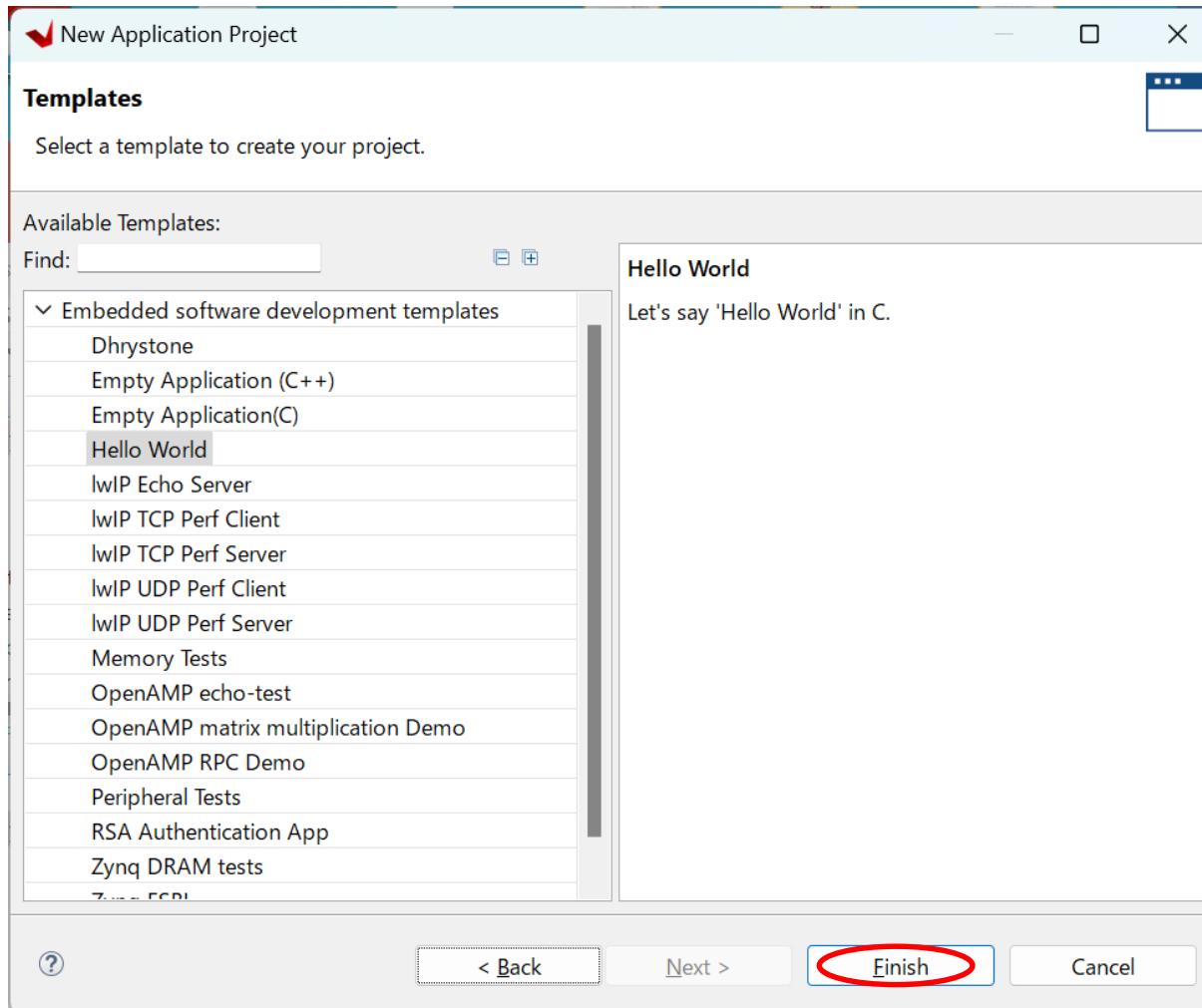
Create Application Project



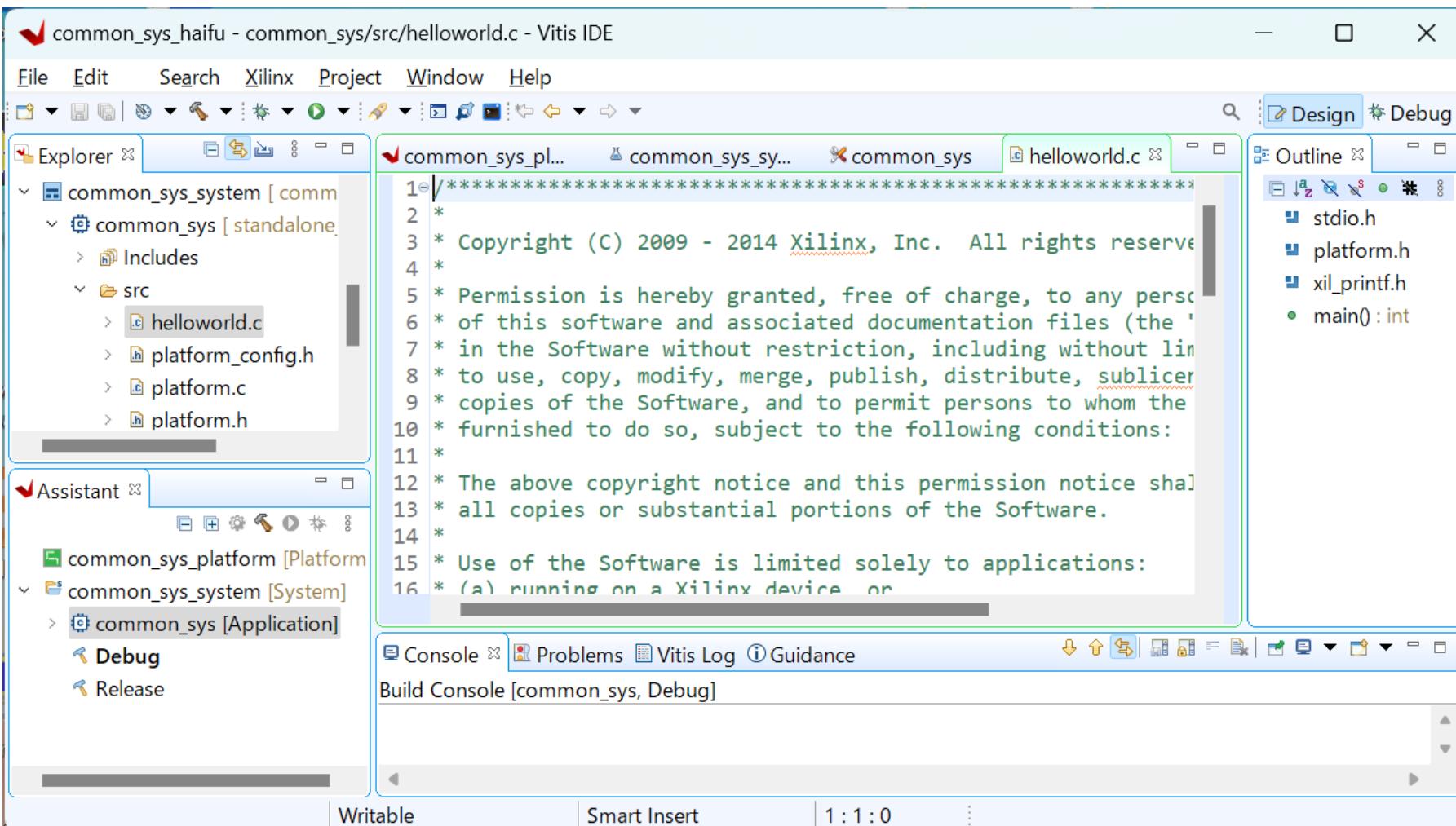
Create Application Project



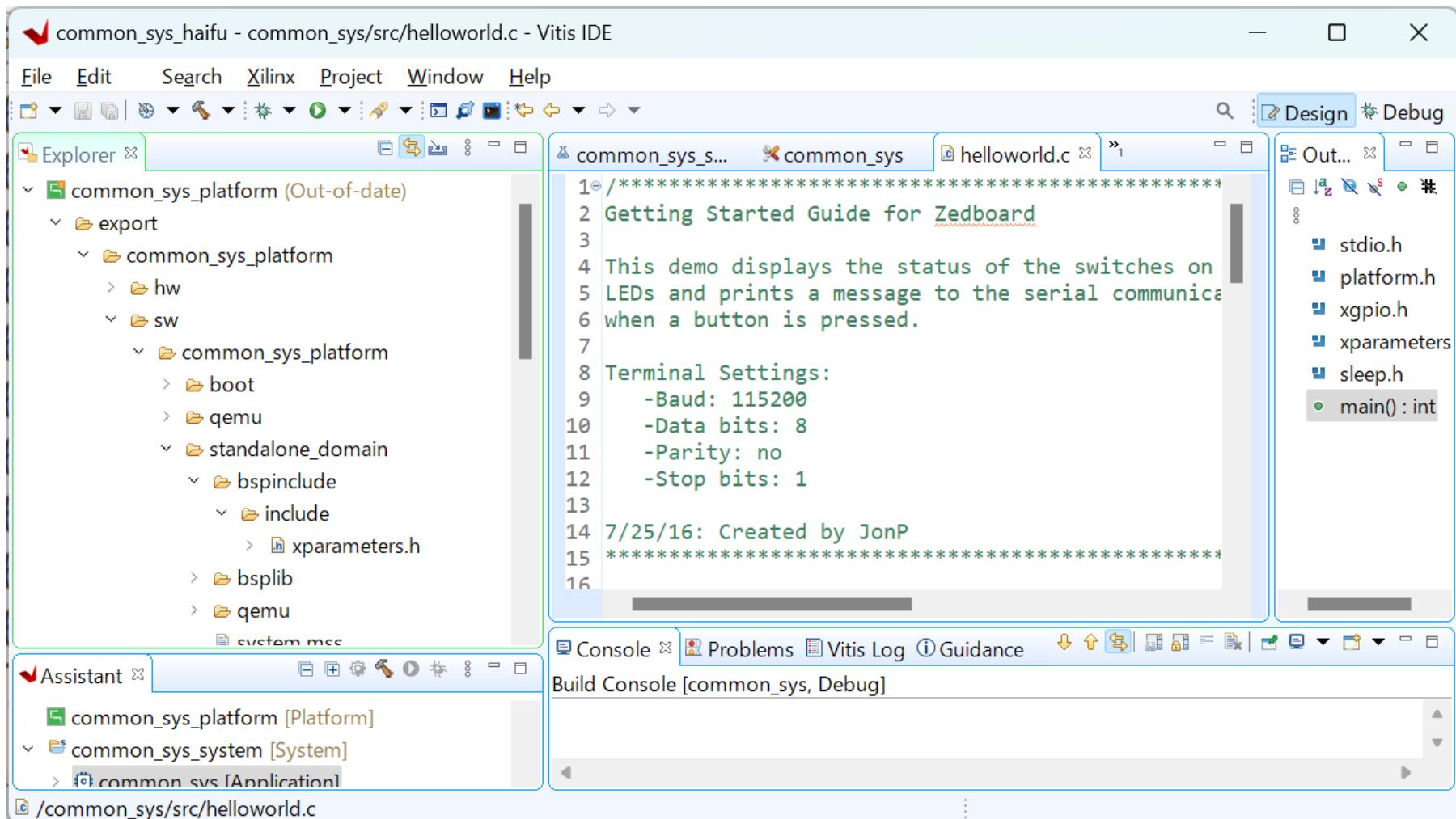
Create Application Project



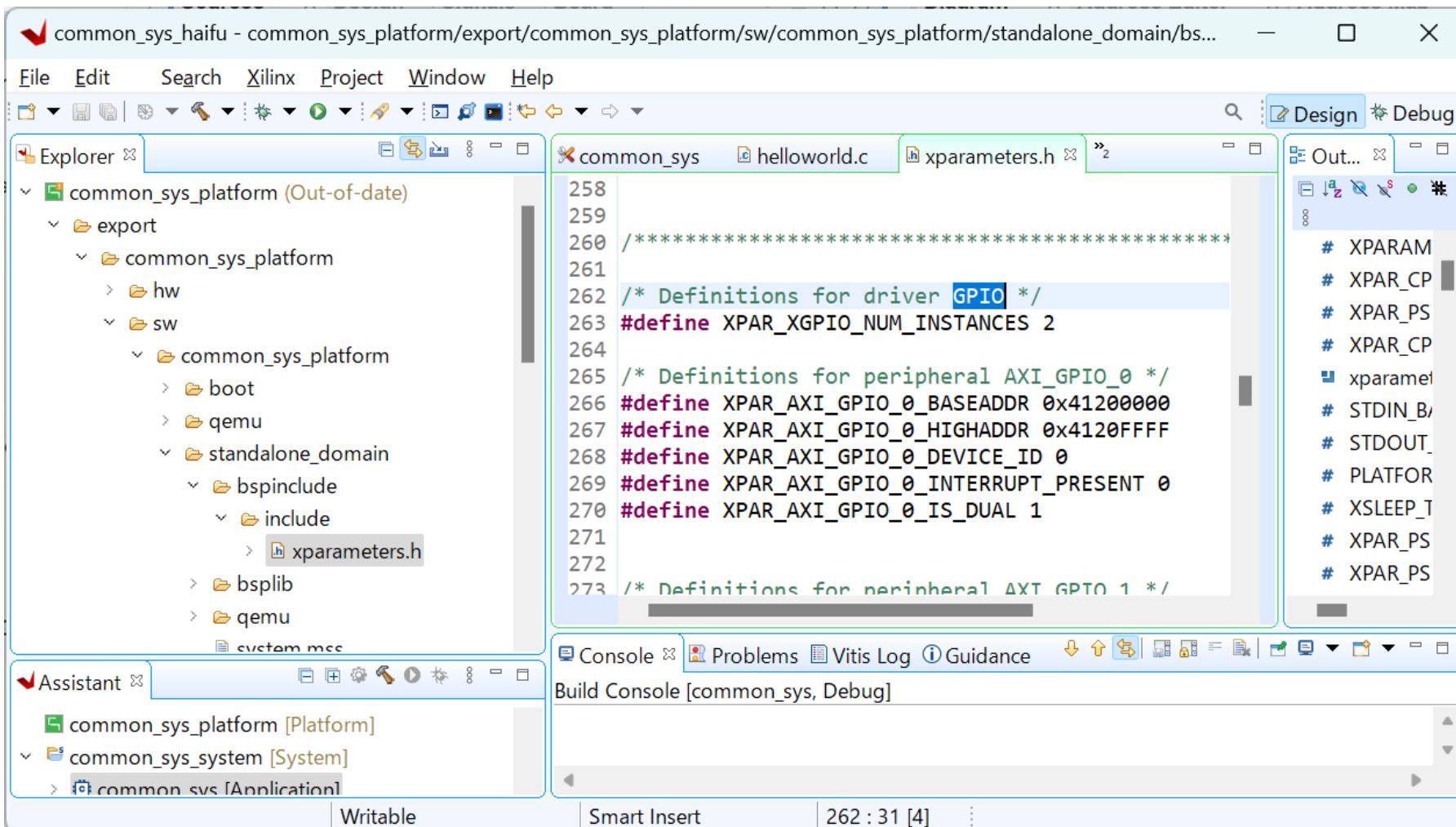
Create Application Project



アドレス設定されているファイル(xparameters.h)

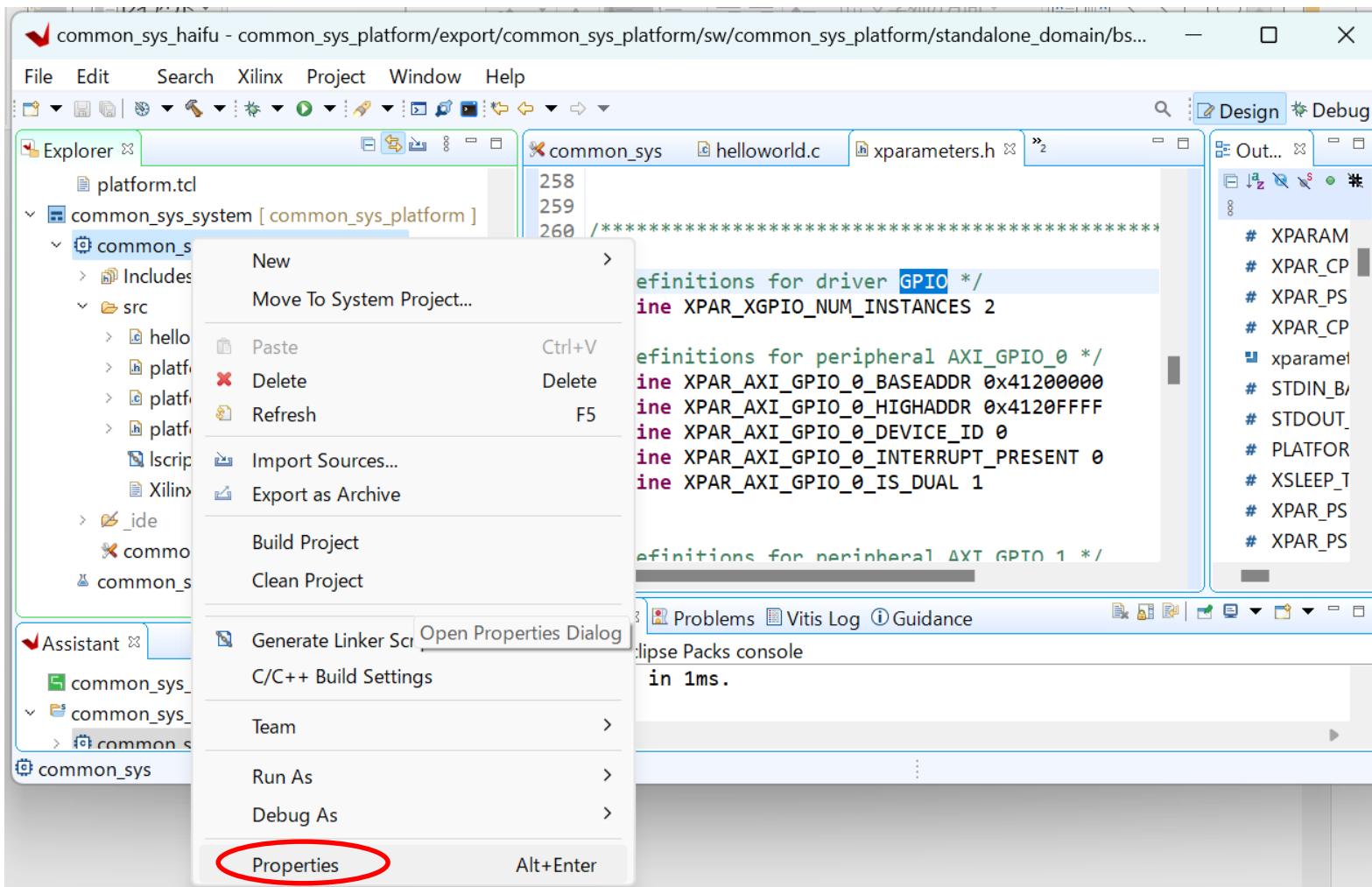


アドレスのチェック (GPIO_0:0x41200000がベースアドレス)

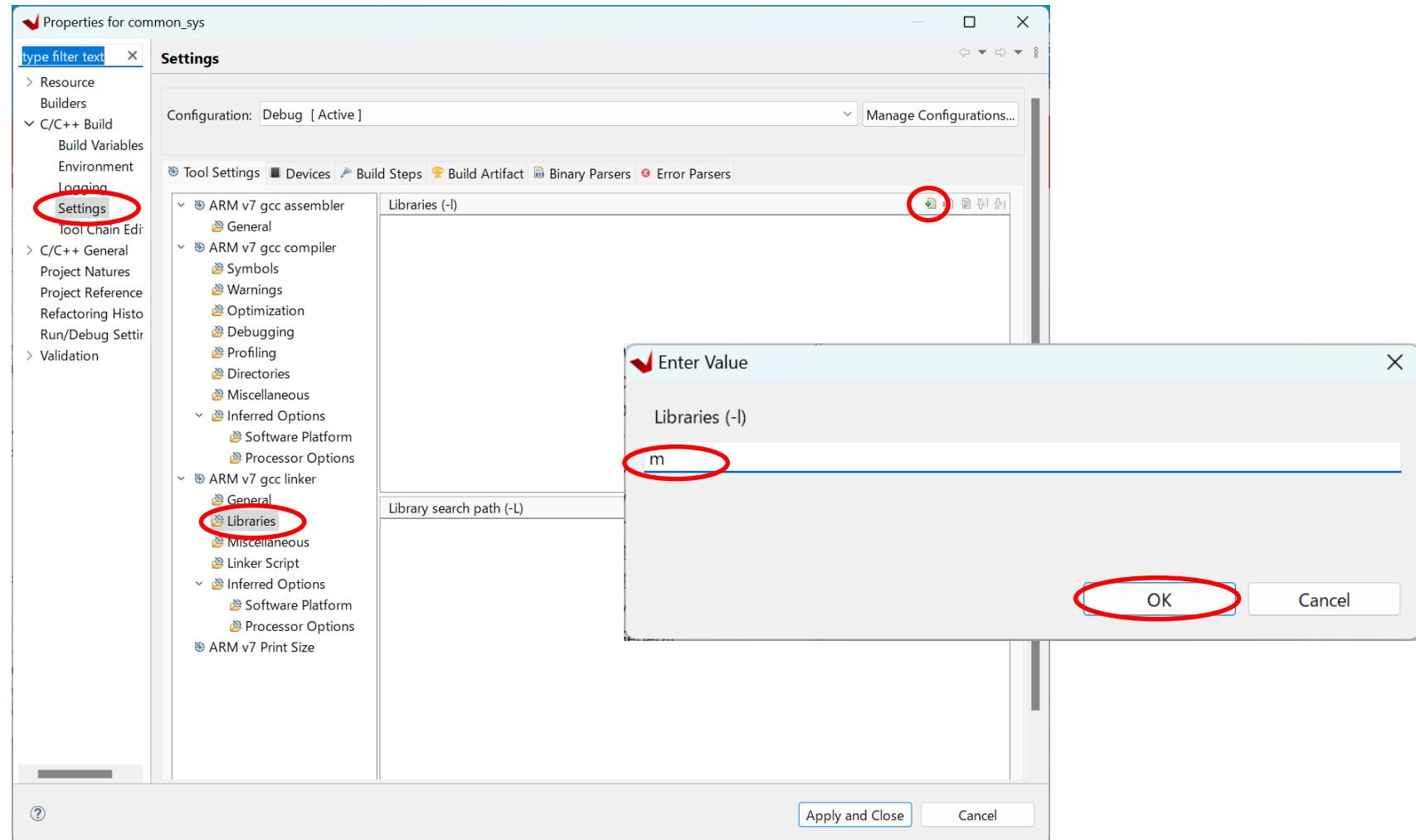


数学ライブラリの結合(math.hを使えるようにする)

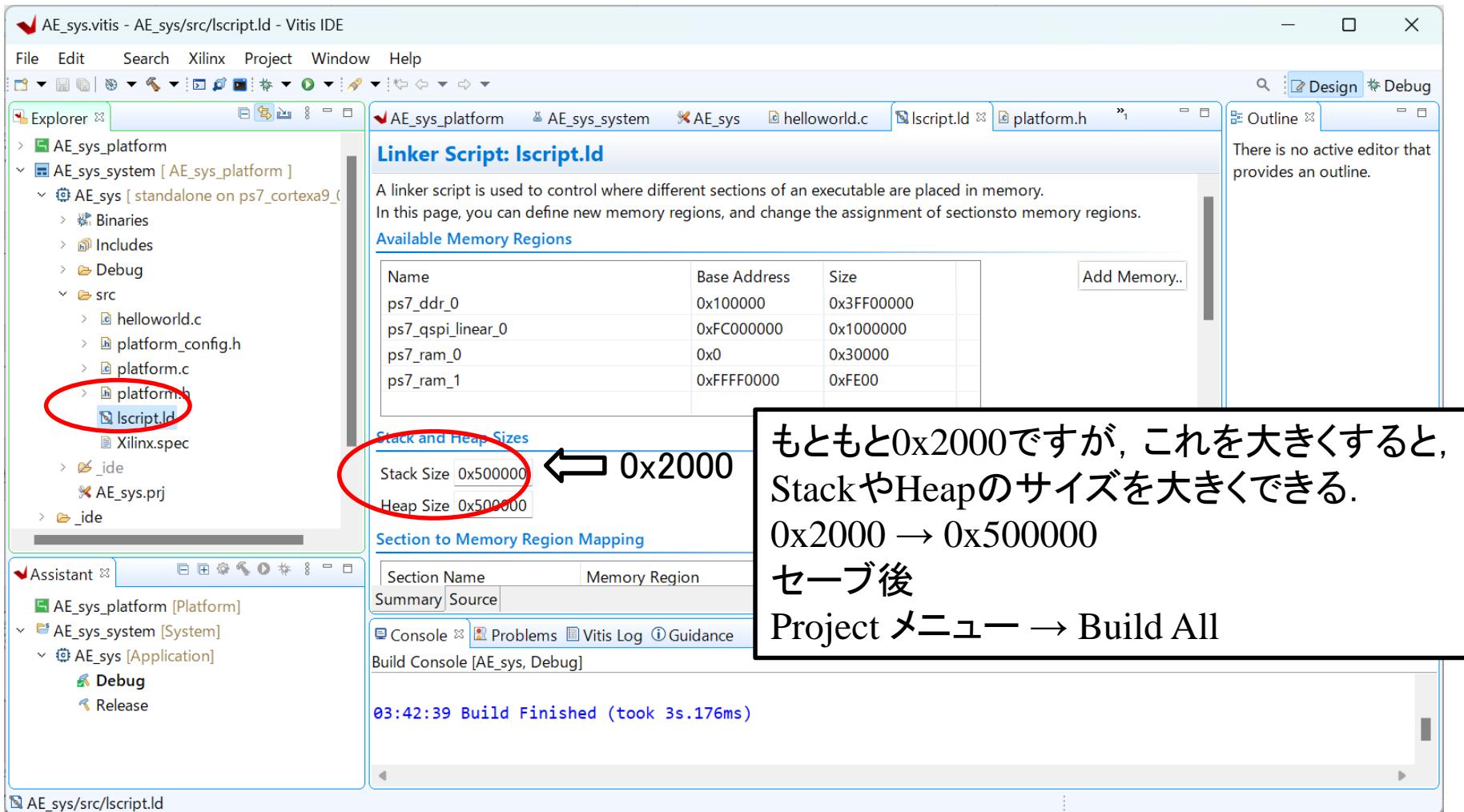
common_sys
で右クリック



数学ライブラリの結合

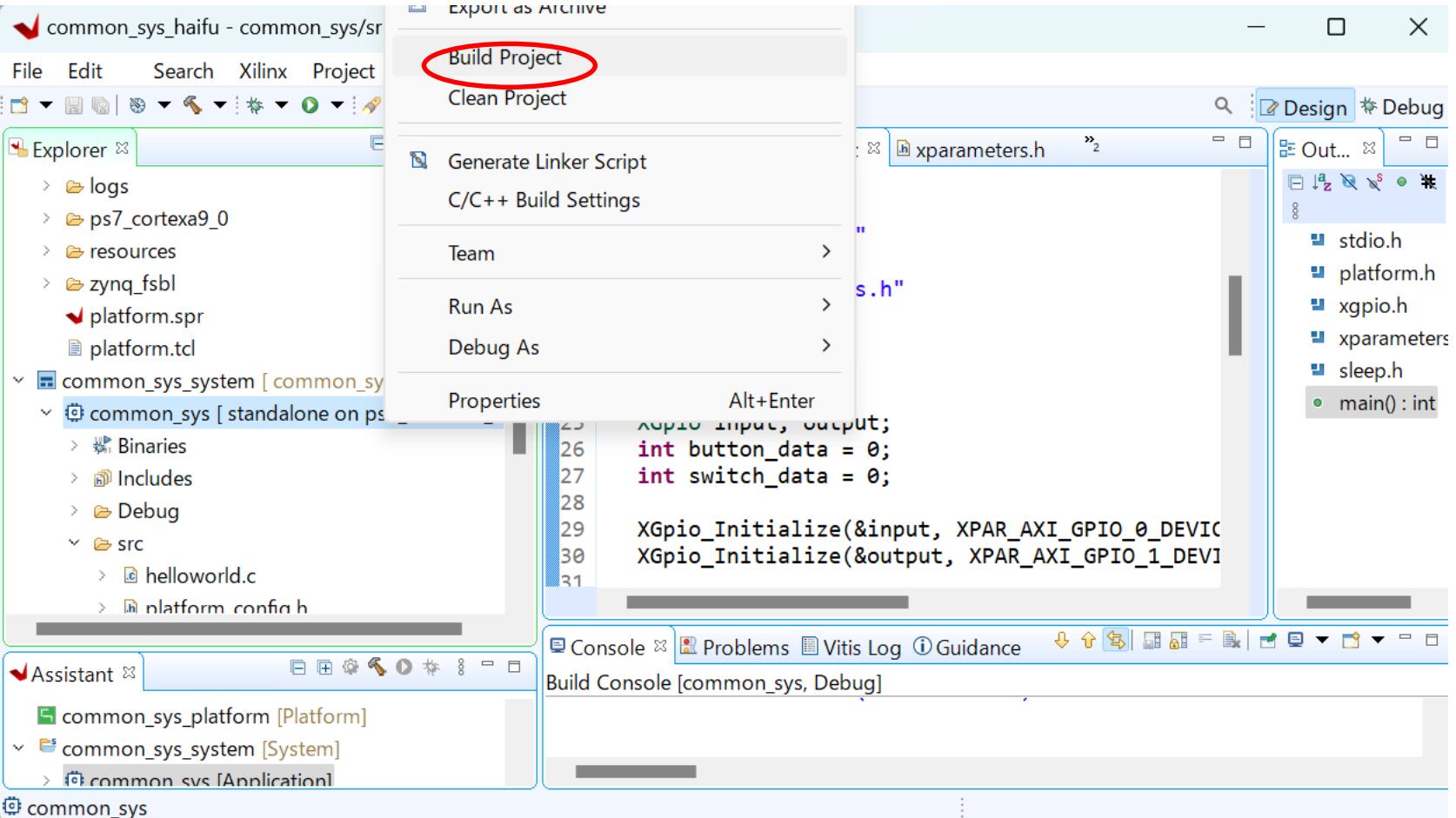


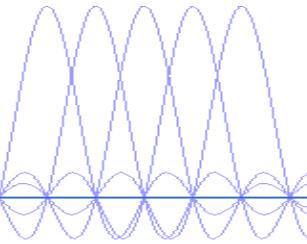
Stack SizeとHeap Sizeとの拡大



Build Project

common_sys
で右クリック





Zybo Z7-10の起動

- PCとZybo Z7-10をUSBケーブルで接続
- JP5をJTAGにジャンパ
- 電源ON

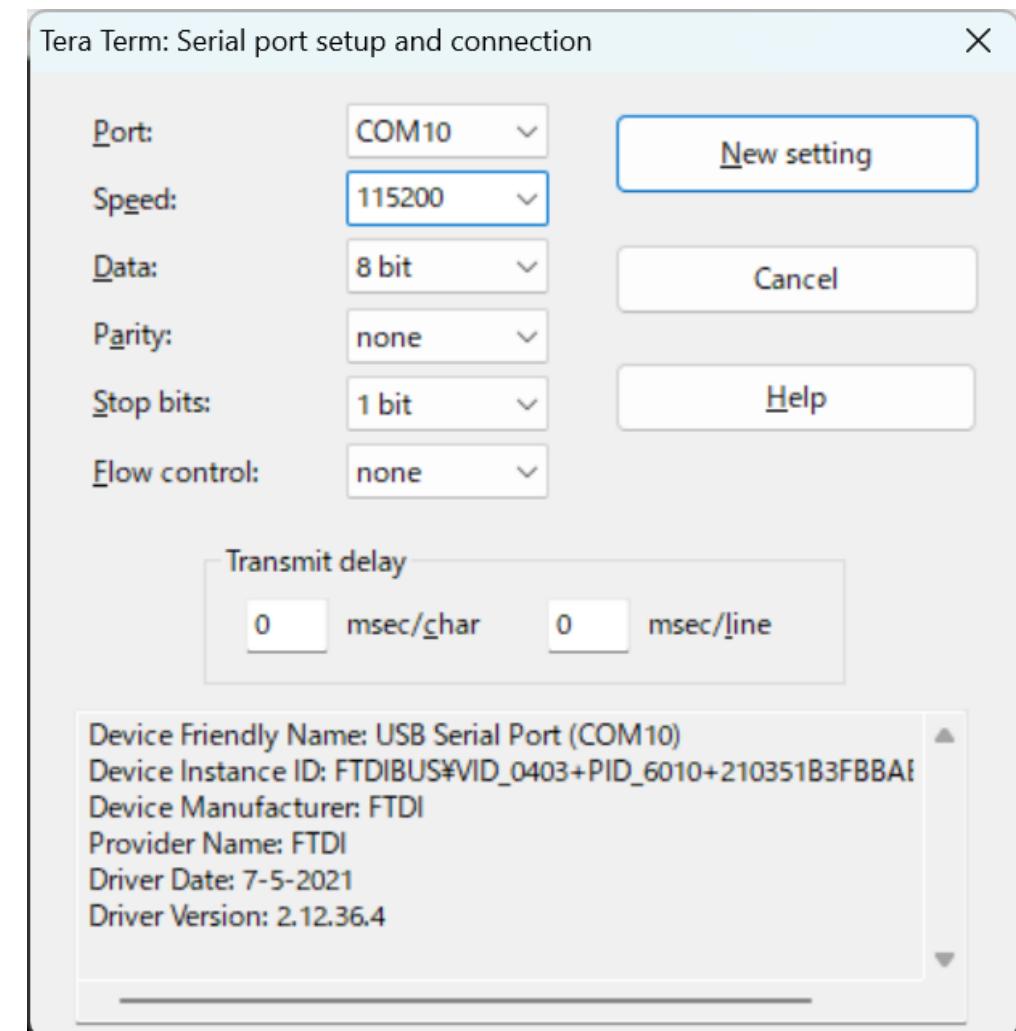
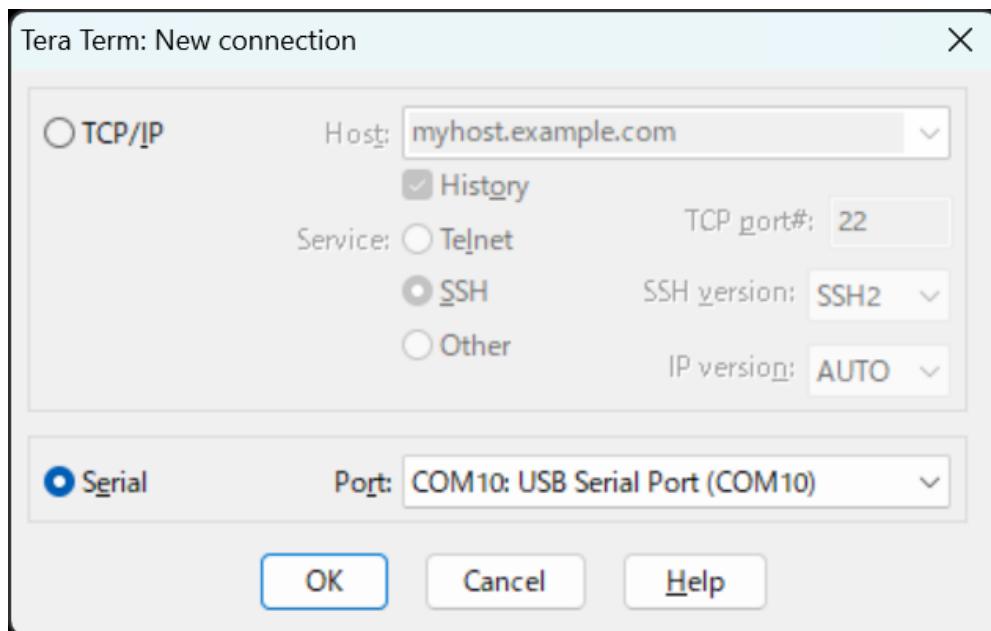
ターミナルの設定

■ Teratermの起動

□シリアルポートの設定

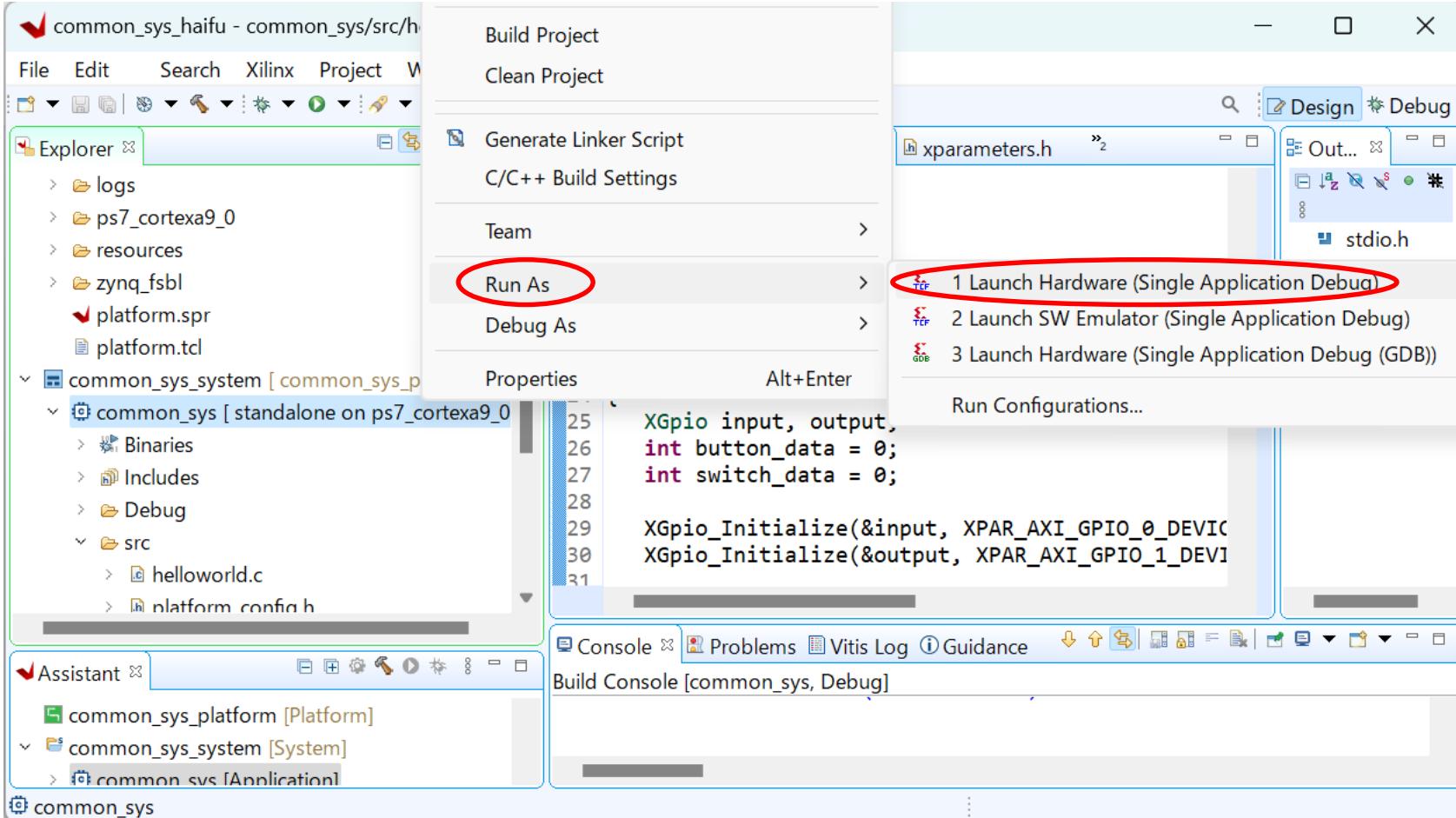
■ Setup → Serial port

□ Speed : 115200



ハードウェアのダウンロードとプログラムの起動

common_sys
で右クリック

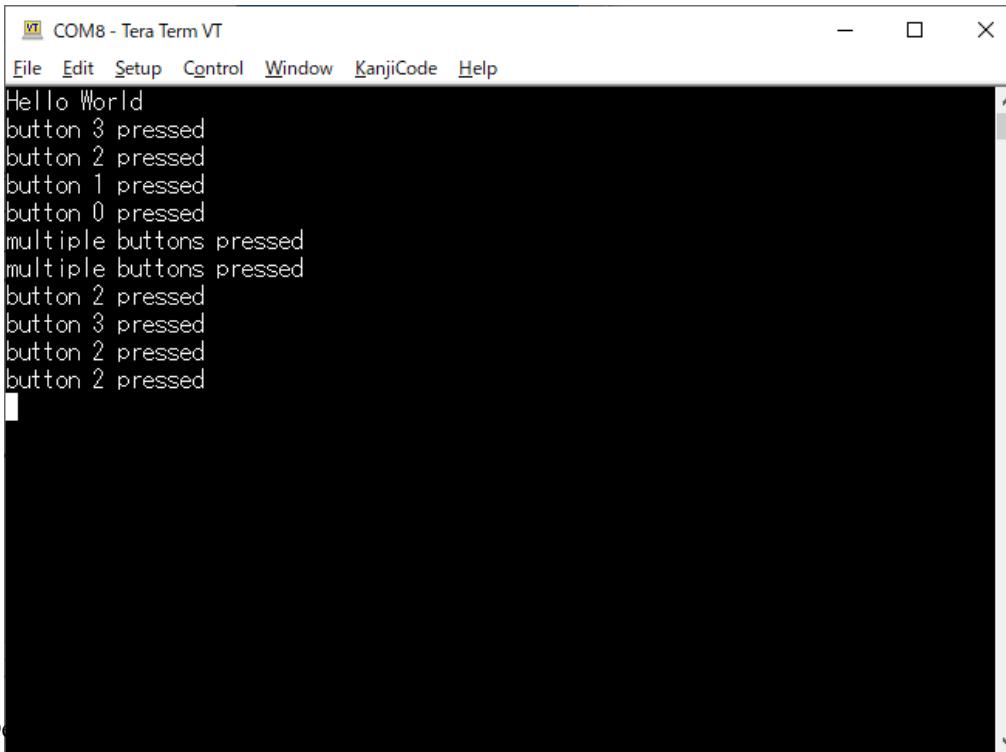


自動的に
ハードウェアの
ダウンロードも
行われます



Run Basic Platform

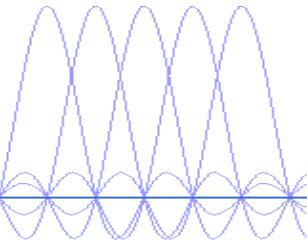
- BTN0 を押下 → “button 0 pressed” が表示
- BTN1+0 → “multiple buttons pressed” が表示
- SW0 をON
→ LED 0(LD0)
turns on.



COM8 - Tera Term VT

File Edit Setup Control Window KanjiCode Help

```
Hello World
button 3 pressed
button 2 pressed
button 1 pressed
button 0 pressed
multiple buttons pressed
multiple buttons pressed
button 2 pressed
button 3 pressed
button 2 pressed
button 2 pressed
```



Extra : Use tcl script

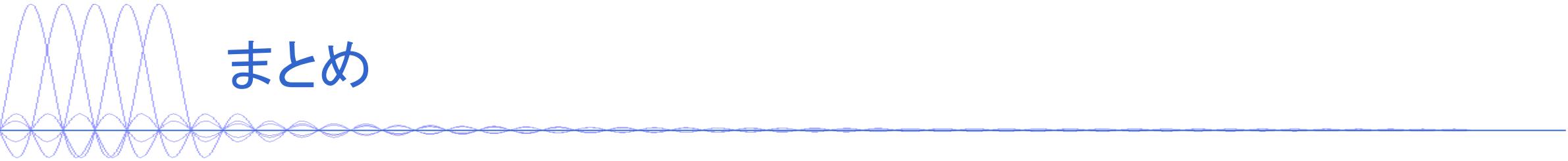
- Use Tcl Console
- Move HOME directory
 - Ex.) cd G:/LSI2025/common_sys/
- Read tcl scripts
 - source ./common_sys.tcl
- Continue from Export Hardware slide

■ 以下のようなアプリを作成

- 押されたボタンの数を示すプログラム

- Ex:

- | | |
|------------------|---------------------------------------|
| ■ Button 0 | → “1 button (Button 0) pressed.” |
| ■ button 0 + 3 | → “2 buttons (Button 0, 3) pressed.” |
| ■ button 0+1+3 | → “3 buttons (Button 0,1,3) pressed.” |
| ■ button 0+1+2+3 | → “4 buttons pressed.” |



まとめ

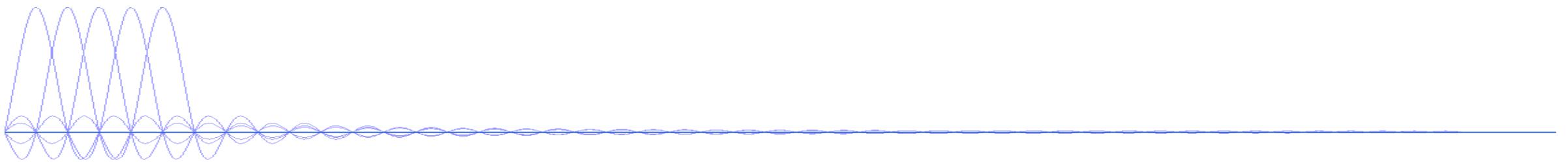
■ 基本システムの構築

- CPUやバス、周辺回路の配置、配線

■ HW/SW演習

- 基本システムの構築とLED制御

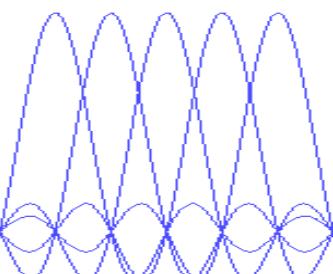
- ボタン情報の習得

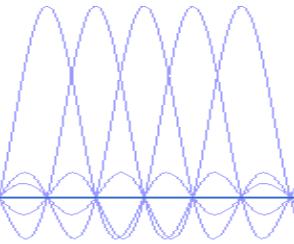


Zybo ボードのZynqでの利用できる メモリサイズを大きくするためには

1st February, 2024

Kyushu Institute of Technology
KUROSAKI, Masayuki





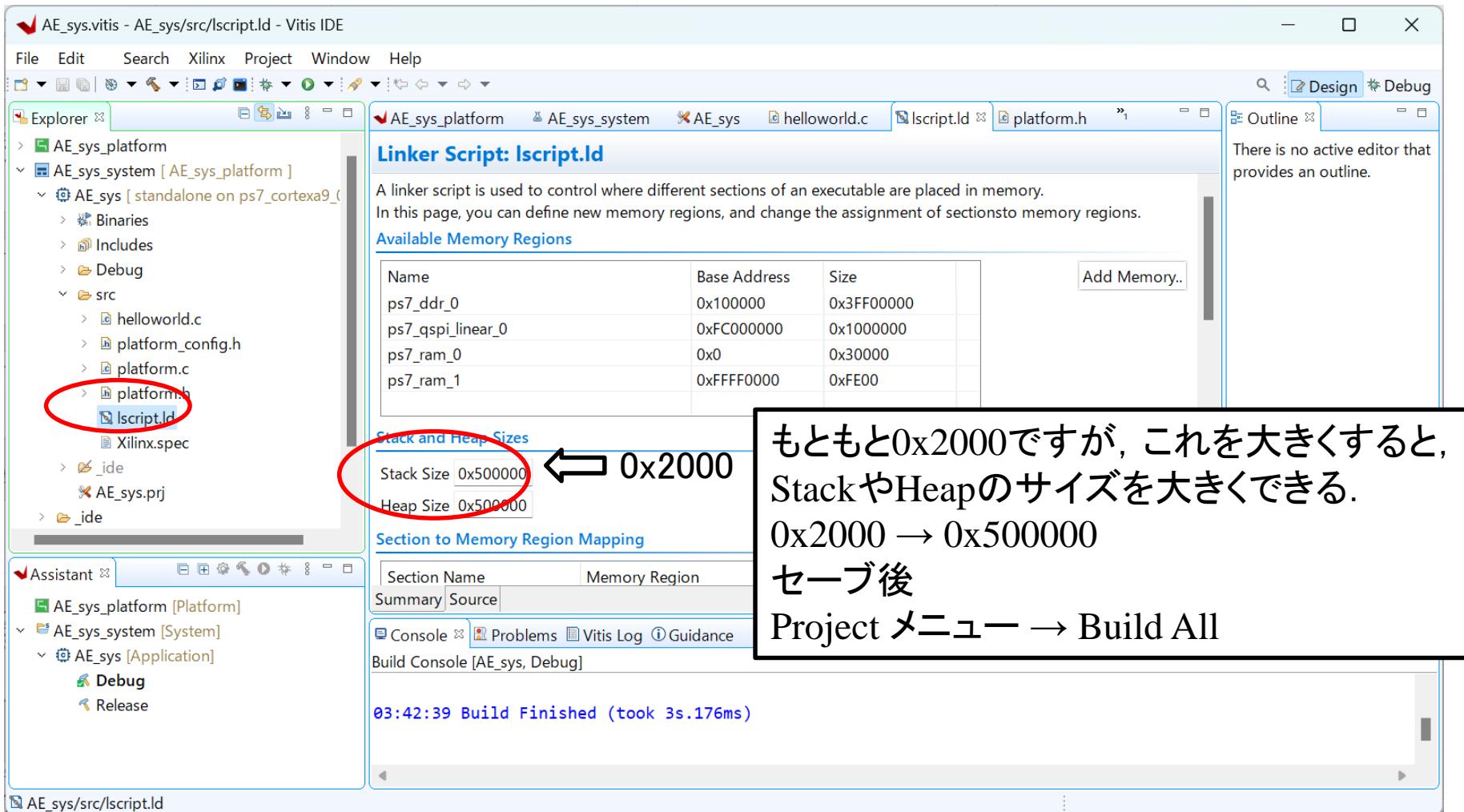
ZyboボードのZynqのメモリ設定

- Zynq-7000 All Programmable SoC Technical Reference Manual
- Table 29-1より
 - DDR-SDRAMが接続されるのは0010_0000 - 3FFF_FFFF

表 29-1: 初期 OCM/DDR アドレス マップ

アドレス範囲 (16 進数)	サイズ	CPU/ACP	その他のマスター
0000_0000 - 0000_FFFF	64KB	OCM	OCM
0001_0000 - 0001_FFFF	64KB	OCM	OCM
0002_0000 - 0002_FFFF	64KB	OCM	OCM
0003_0000 - 0003_FFFF	64KB	予約	予約
0004_0000 - 0007_FFFF	256KB	予約	予約
0008_0000 - 000B_FFFF	256KB	予約	DDR
000C_0000 - 000C_FFFF	64KB	予約	DDR
000D_0000 - 000D_FFFF	64KB	予約	DDR
000E_0000 - 000E_FFFF	64KB	予約	DDR
000F_0000 - 000F_FFFF	64KB	OCM3 (エイリアス)	DDR
0010_0000 - 3FFF_FFFF	1,023MB	DDR	DDR
<hr/>			
FFFC_0000 - FFFC_FFFF	64KB	予約	予約
FFFD_0000 - FFFD_FFFF	64KB	予約	予約
FFFE_0000 - FFFE_FFFF	64KB	予約	予約
FFFF_0000 - FFFF_FFFF	64KB	OCM3	OCM3

Stack SizeとHeap Sizeとの変更



確保されたメモリのアドレスのチェック

The screenshot shows the Vitis IDE interface with the project "AE_sys.vitis" open. The code editor displays a C file named "helloworld.c". The code initializes three arrays: "gc_buff" (310,000 elements), "uc_buff" (930,000 elements), and "buff" (930,000 dynamic elements). The "Outline" panel on the right shows the declared variables. The "Console" tab at the bottom shows build logs and a successful build message.

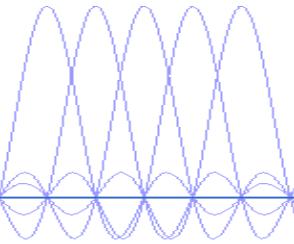
310,000個の配列

```
27  
28 unsigned char gc_buff[310000];  
29  
30 int main()  
31 {  
32     XGpio input, output;  
33     int button_data = 0;  
34     int switch_data = 0;  
35  
36     FIL fil;  
37     FATFS fatfs;  
38     FIL fil_i, fil_o;  
39     FATFS fatfs_i, fatfs_o;  
40     char FileName[32] = "test.txt";  
41     char ImageReadFileName[32] = "imagepr5.raw";  
42     char ImageWriteFileName[32] = "imagepr0.raw";  
43  
44     FRESULT Res;  
45     UINT NumBytesRead, NumBytesWrite;  
46     u32 FileSize = 640*480*3;  
47     TCHAR *Path = "0:/";  
48     unsigned char uc_buff[3*310000];  
49     int i;  
50     unsigned char *buff;  
51  
52     buff = (unsigned char *)malloc(sizeof(char)*3*310000);  
53  
54     XGpio_Initialize(&input, XPAR_AXI_GPIO_0_DEVICE_ID); //initialize input XGpi  
55     XGpio Initialize(&output, XPAR_AXI GPIO 1 DEVICE ID); //initialize output
```

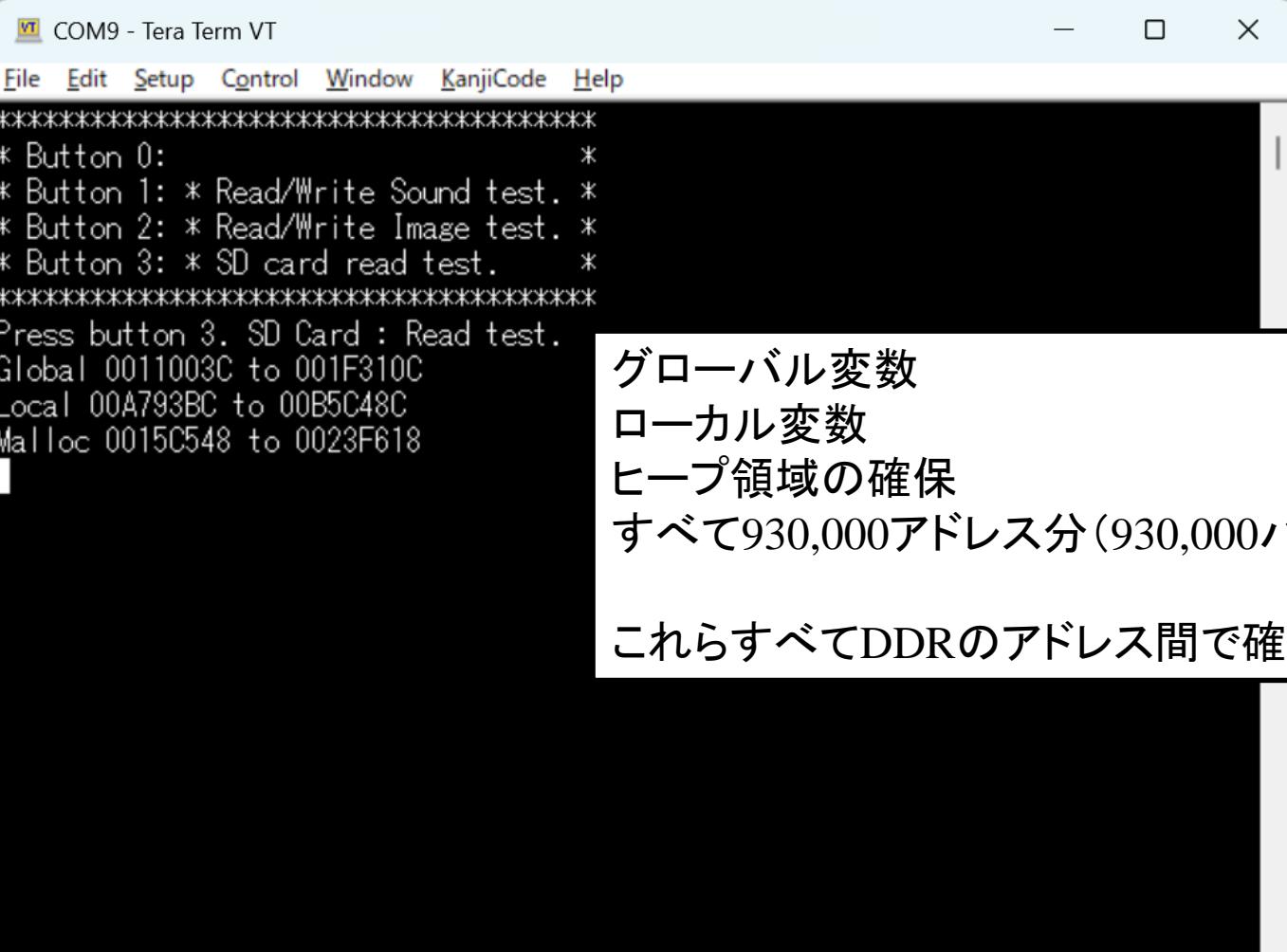
930,000個の配列

930,000個の動的配列の確保

Writable Smart Insert 33 : 21 : 883



確保されたメモリのアドレスのチェック



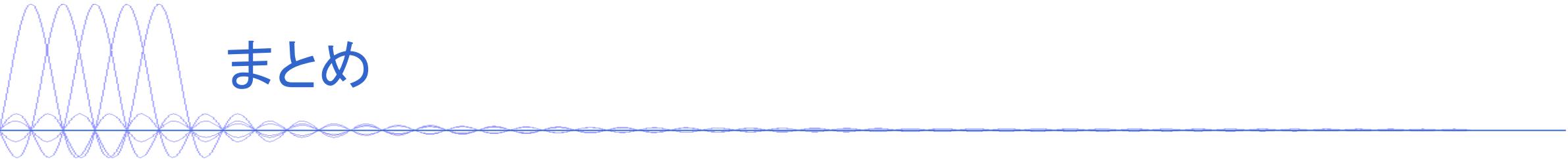
COM9 - Tera Term VT

File Edit Setup Control Window KanjiCode Help

```
*****
* Button 0: *
* Button 1: * Read/Write Sound test. *
* Button 2: * Read/Write Image test. *
* Button 3: * SD card read test. *
*****
Press button 3. SD Card : Read test.
Global 0011003C to 001F310C
Local 00A793BC to 00B5C48C
Malloc 0015C548 to 0023F618
```

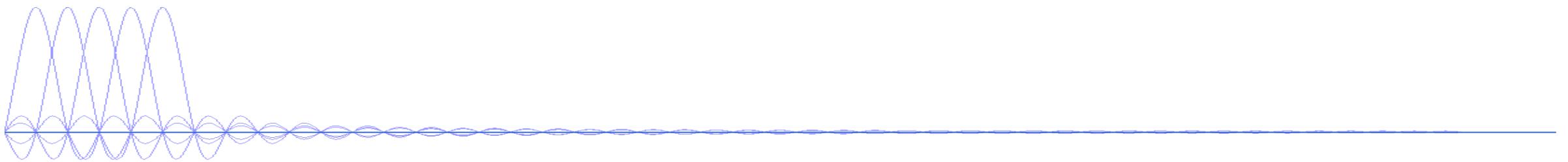
グローバル変数
ローカル変数
ヒープ領域の確保
すべて930,000アドレス分(930,000バイト)確保

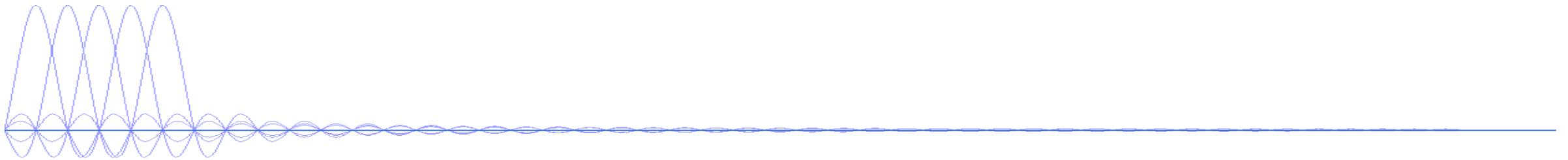
これらすべてDDRのアドレス間で確保されている。



まとめ

- Zybo ボードのZynqでの利用できるメモリサイズを大きくするためには
 - Linker Script : lscript.ld のサイズを変更
 - Stack Size
 - Heap Size



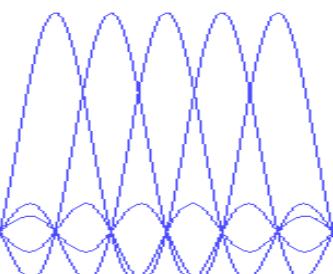


数学ライブラリのリンク方法について

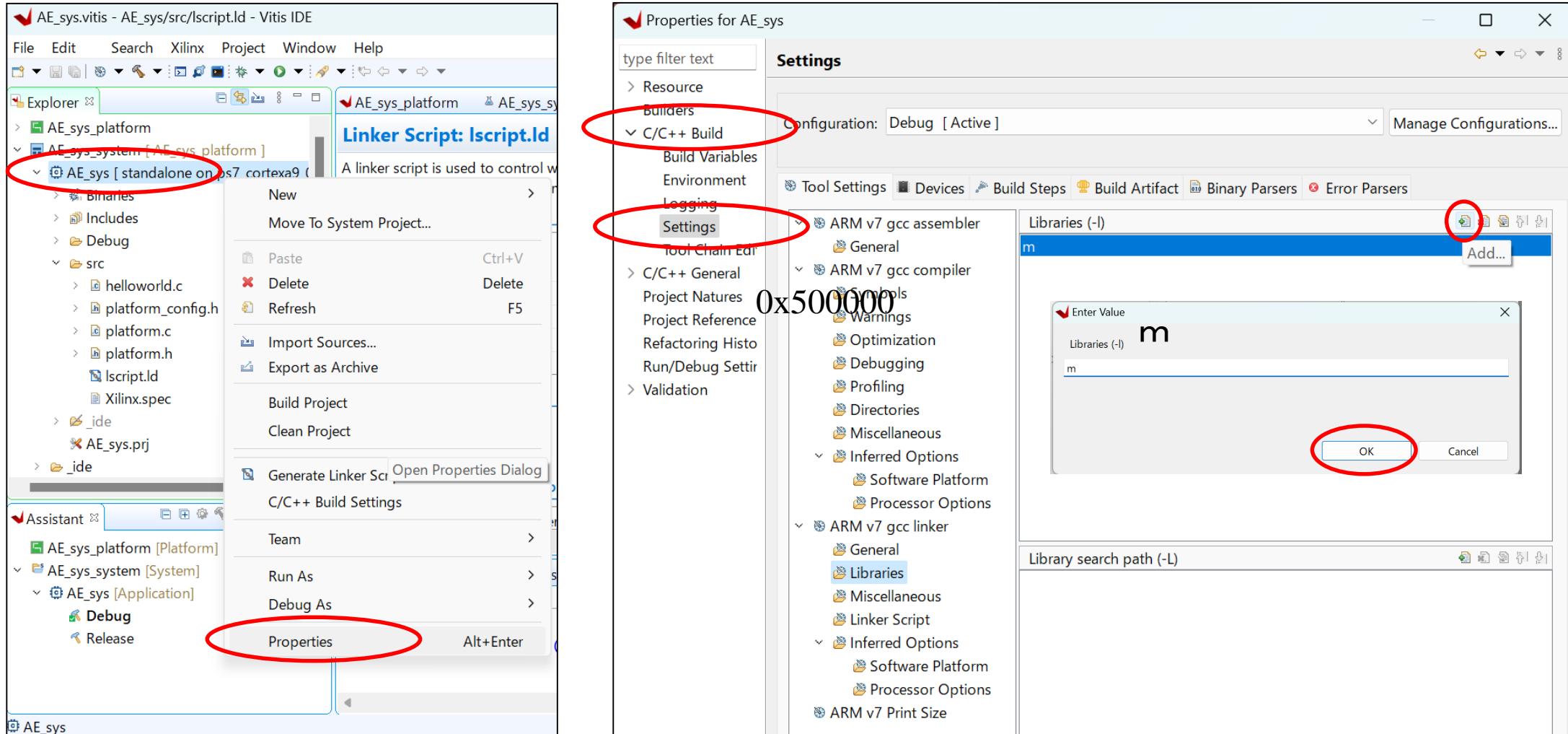
1st February, 2024

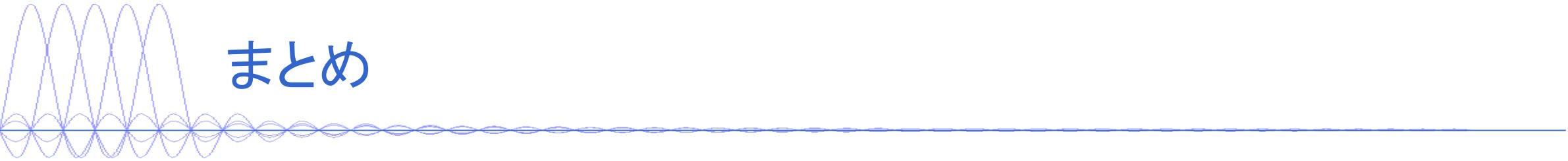
Kyushu Institute of Technology

KUROSAKI, Masayuki



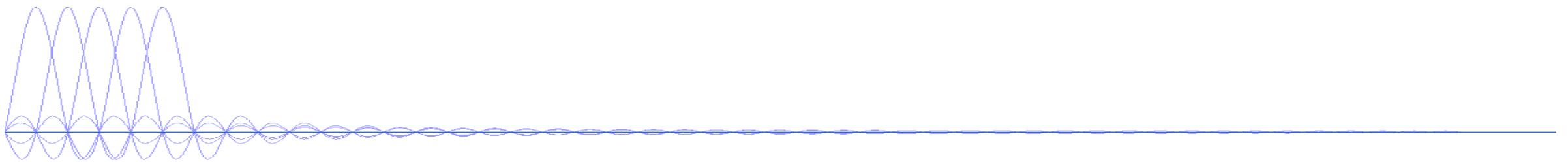
数学ライブラリの追加について これでmath.hが利用できる





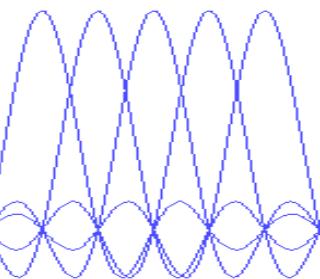
まとめ

- 数学ライブラリを利用する場合
 - 数学ライブラリを追加する必要がある

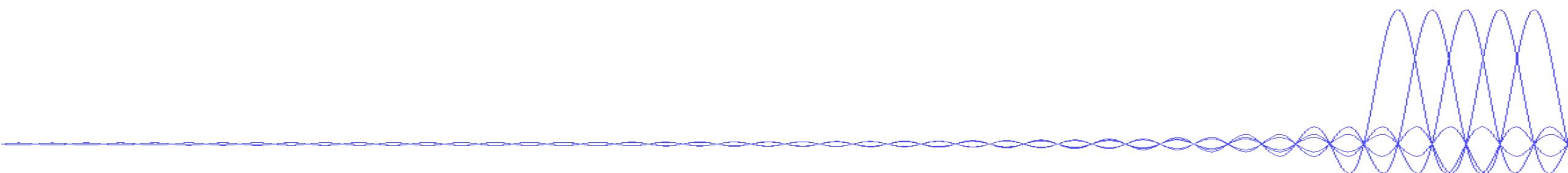


SDカードシステムの作成と実証

作業フォルダ: .\sdcard_sys



SDカードの利用



MicroSD

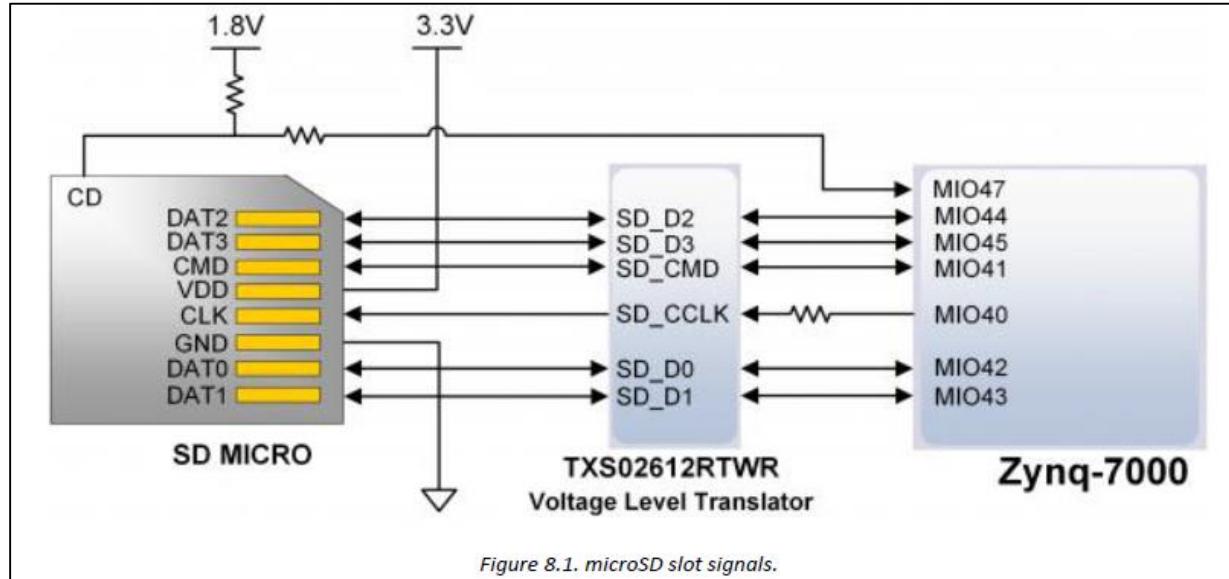
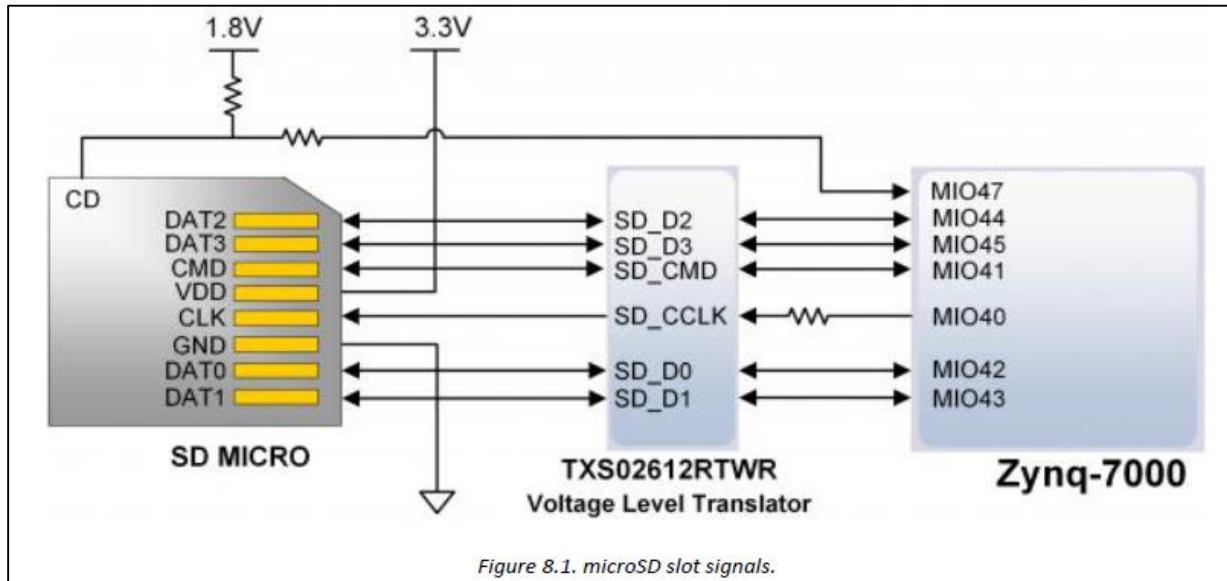


Figure 8.1. microSD slot signals.

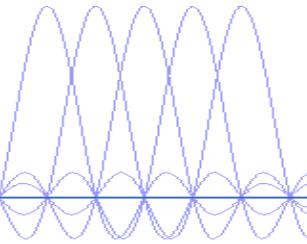
- MicroSD slot (J4) for non-volatile external memory storage
 - The slot is wired to Bank 1/501 MIO[40-45], and also includes a card detect signal attached to MIO 47.
 - The SD slot is powered from 3.3V, but is connected through MIO Bank 1/501 (1.8V). Therefore, a TI TXS02612 level shifter performs this translation.

MicroSD



Signal Name	Description	Zynq Pin	SD Slot Pin
SD_D0	Data[0]	MIO42	7
SD_D1	Data[1]	MIO43	8
SD_D2	Data[2]	MIO44	1
SD_D3	Data[3]	MIO45	2
SD_CCLK	Clock	MIO40	5
SD_CMD	Command	MIO41	3
SD_CD	Card Detect	MIO47	9

Table 8.1. MicroSD pinout.



Create Hardware

■ Platform tutorial (common_sys)と同じ手順

- Create Hardware (same as common_sys)
- Export Hardware
- Launch VITIS

Copy Program

The screenshot shows the Xilinx SDK IDE interface. The Project Explorer on the left displays the project structure, including files like design_1_wrapper_hw_platform.c, ps7_init_gpl.c, ps7_init.h, ps7_init.html, ps7_init.tcl, system.hdf, and source files under sdcard_sys. The central code editor window shows the helloworld.c file. A callout box highlights the code in the editor with the text "Copy program (from sdcard_sys.c)". The code itself is a C program that includes stdio.h, platform.h, xgpi.h, xparameters.h, and sleep.h, and defines a main() function that reads data from an SD card. The bottom right corner of the code editor shows the line number 140:1. The bottom right panel shows the SDK Log with several INFO messages related to the Xilinx command-line interface.

```
/* Copyright (C) 2009 - 2014 Xilinx, Inc. All rights reserved. */
/* project: sdcard_sys
 * SDカードのテスト
 * @Press button 3 then read data from SD card.
 * File name : test.txt (as Default)
 */

#include <stdio.h>
#include "platform.h"
#include <xgpi.h>
#include "xparameters.h"
#include "sleep.h"

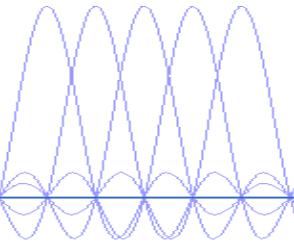
#include "xil_printf.h"
//#include "xsdps.h"
#include "xil_cache.h"
// Use ff.h to access SD card.
#include "ff.h"

int main()
{
    XGpio input, output;
    int button_data = 0;
    int switch_data = 0;

    FIL fil;
    FATFS fatfs;
```

SDK Log:

```
19:30:45 INFO  : Registering command handlers for application
19:30:45 INFO  : Launching XSCT server: xsct.bat
19:30:45 INFO  : XSCT server has started successfully
19:30:45 INFO  : Successfully done setting XSCT environment
19:30:46 INFO  : Successfully done setting SDK environment
19:30:46 INFO  : Processing command line option
```

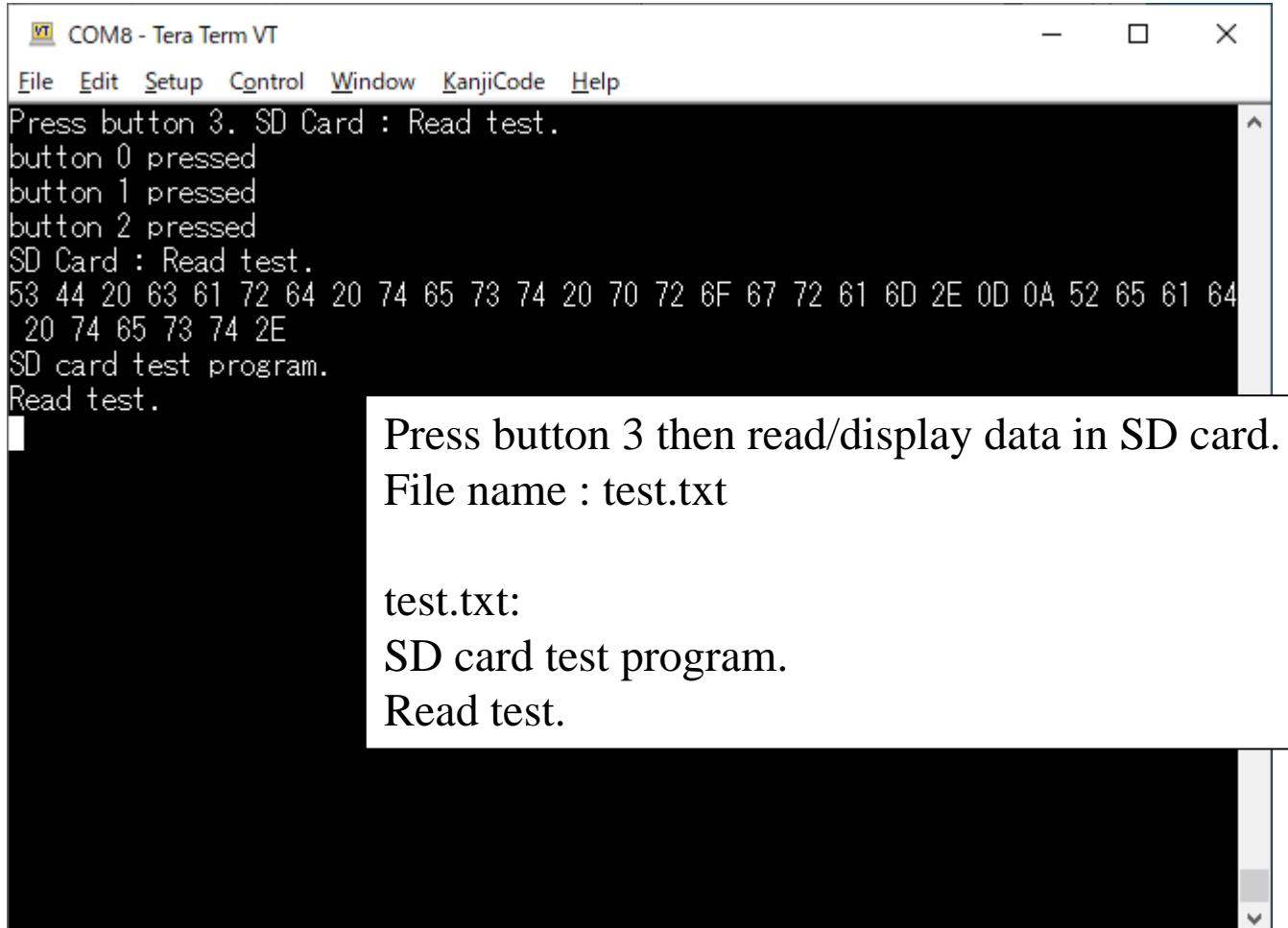


Program FPGA / Run Program

- Build Project
- Run As → Launch on Hardware



Result of SD Card Test



COM8 - Tera Term VT

File Edit Setup Control Window KanjiCode Help

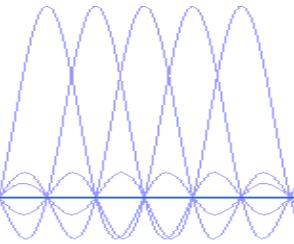
```
Press button 3. SD Card : Read test.
button 0 pressed
button 1 pressed
button 2 pressed
SD Card : Read test.
53 44 20 63 61 72 64 20 74 65 73 74 20 70 72 6F 67 72 61 6D 2E 0D 0A 52 65 61 64
 20 74 65 73 74 2E
SD card test program.
Read test.
```

Press button 3 then read/display data in SD card.
File name : test.txt

test.txt:
SD card test program.
Read test.

■ 濃淡画像(gray image)への変換

- image data の読み込み(imagepr0.raw)
- gray image への変換
- gray image dataの書き込み (gray.raw)



画像について Image

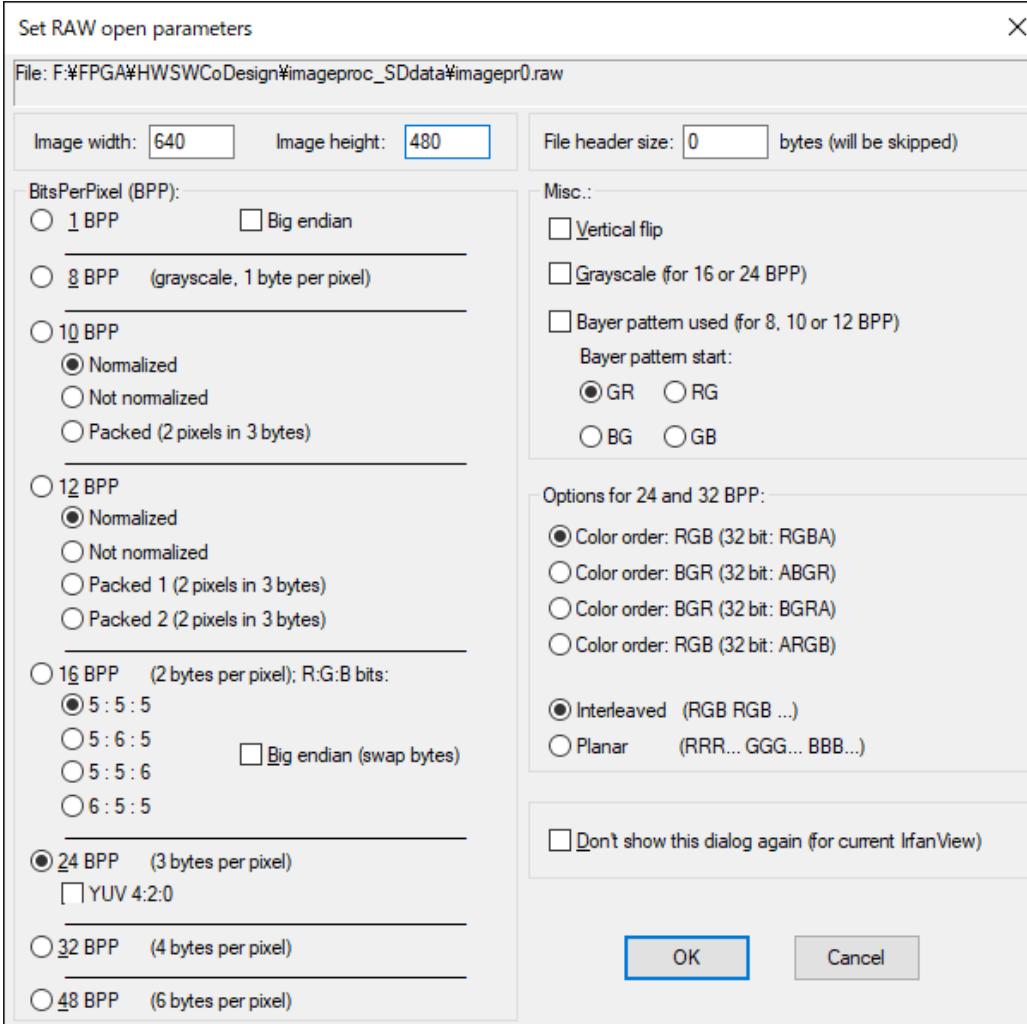
■ Image (imagepr0.raw)

- Image size : 640 x 480 pixels.
- Color data size : R 8bits, G 8bits, B 8bits
- Color data order: RGB RGB RGB (Interleaved)

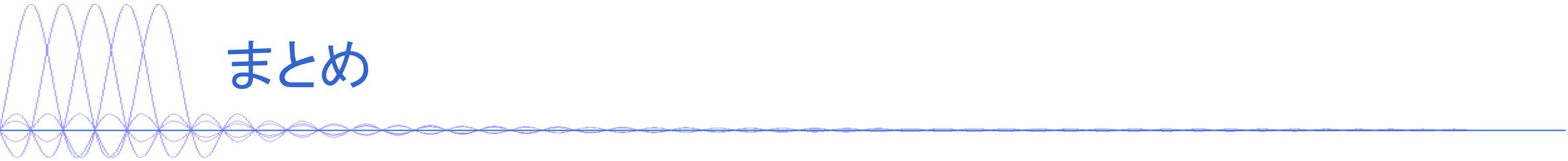
■ どのようにrawを読み込むのか？

- IrfanView (<https://www.irfanview.net/>)
- Drag and drop the raw file.
- RAW パラメータ設定 (See next slide)

Set RAW open parameters

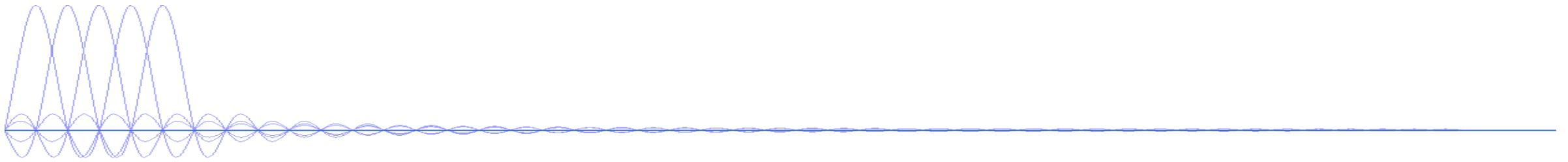


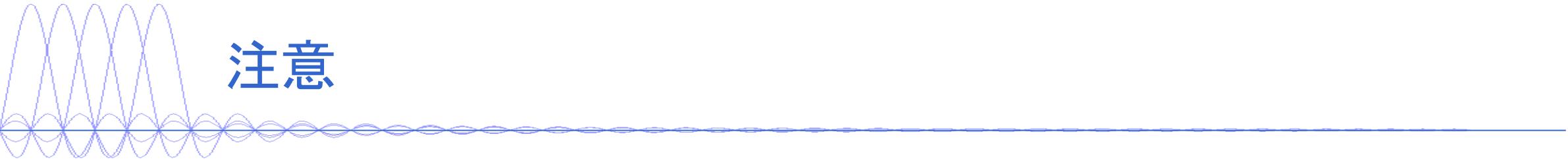
imagepr2.raw



まとめ

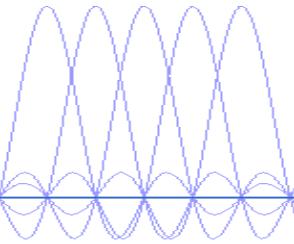
- SD カードプロジェクト
- HW/SW演習: Create new applications.





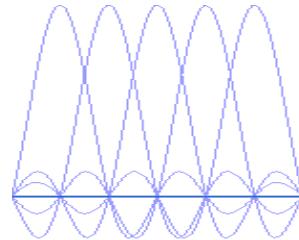
注意

- 実習については、つぎのページからのスライドではなく、応用実習集を確認のこと。



応用実習2: HW演習

- CommonSys演習: 以下のようなアプリを作成
 - 押されたボタンの数を示すプログラム
 - Ex:
 - Button 0 → “1 button (Button 0) pressed.”
 - button 0 + 3 → “2 buttons (Button 0, 3) pressed.”
 - button 0+1+3 → “3 buttons (Button 0,1,3) pressed.”
 - button 0+1+2+3 → “4 buttons pressed.”
- SDカード演習: 以下のようなアプリを作成 (応用的な課題: 最後までできなくても可)
 - imagepr0.raw を読み込み, データをコピーして imageprA.raw を出力するプログラム
- 締め切り: 2024/12/18、17:00まで



MATLAB演習／LSIコンテスト演習(再掲)

■ MATLAB演習

- チームで1通のレポートを作成しよう。
- 締め切り: 2024/12/17、17:00まで
- チーム名(仮)、役職、名前を記載してください。

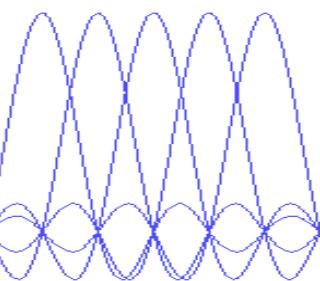
■ LSIコンテスト演習

- VAE(Variational AutoEncoder)の応用例を探しましょう。
 - 入力と出力がわかるように記載しましょう。
- LSIコンテストに向けて、どのようなシステムを作成すればいいかチームで考えてみよう。(2024/12/17 16:15までに1つ以上の案を出してみよう。).

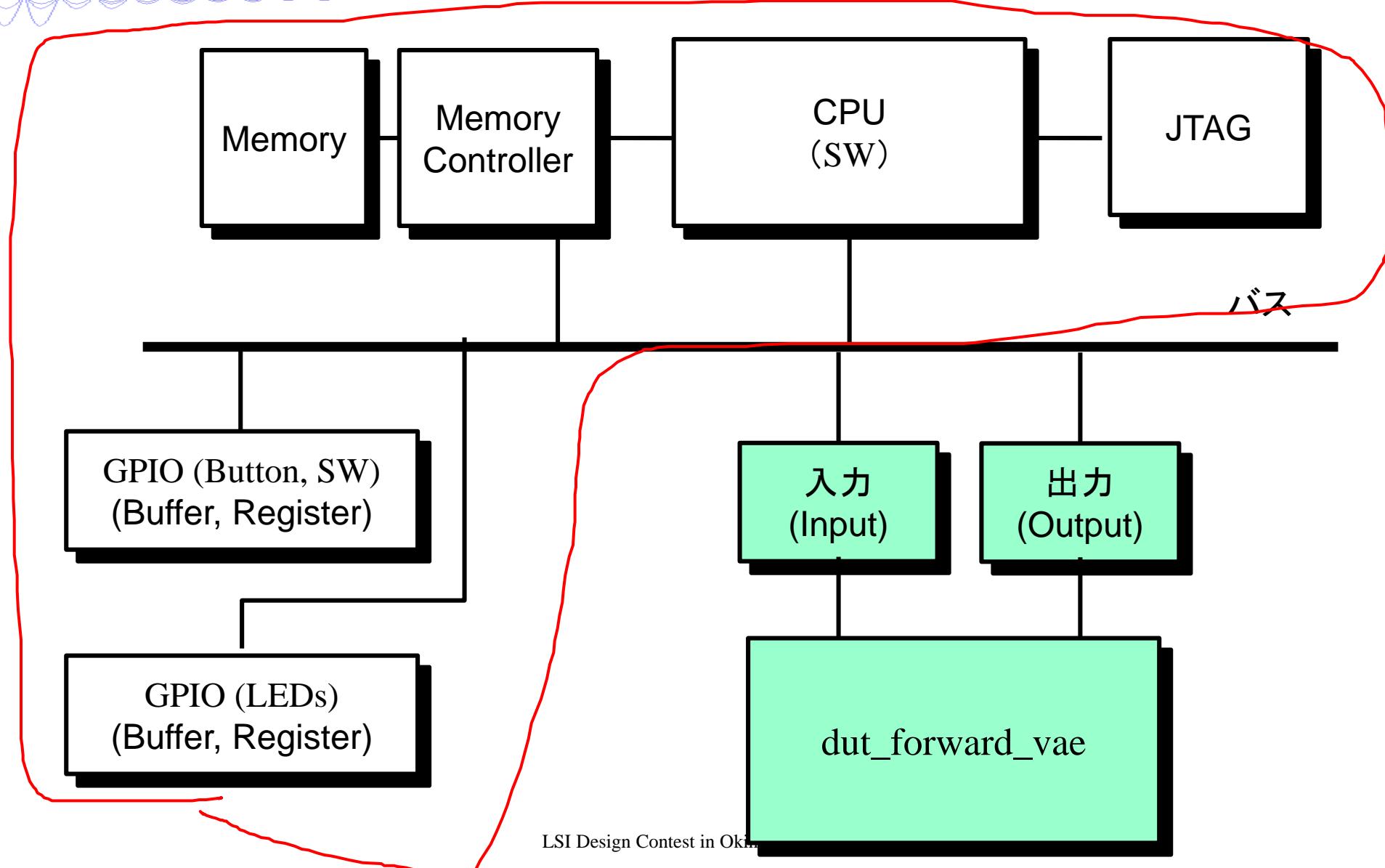
HW/SW協調設計演習

CPU + 周辺回路 + dut_forward_vae回路

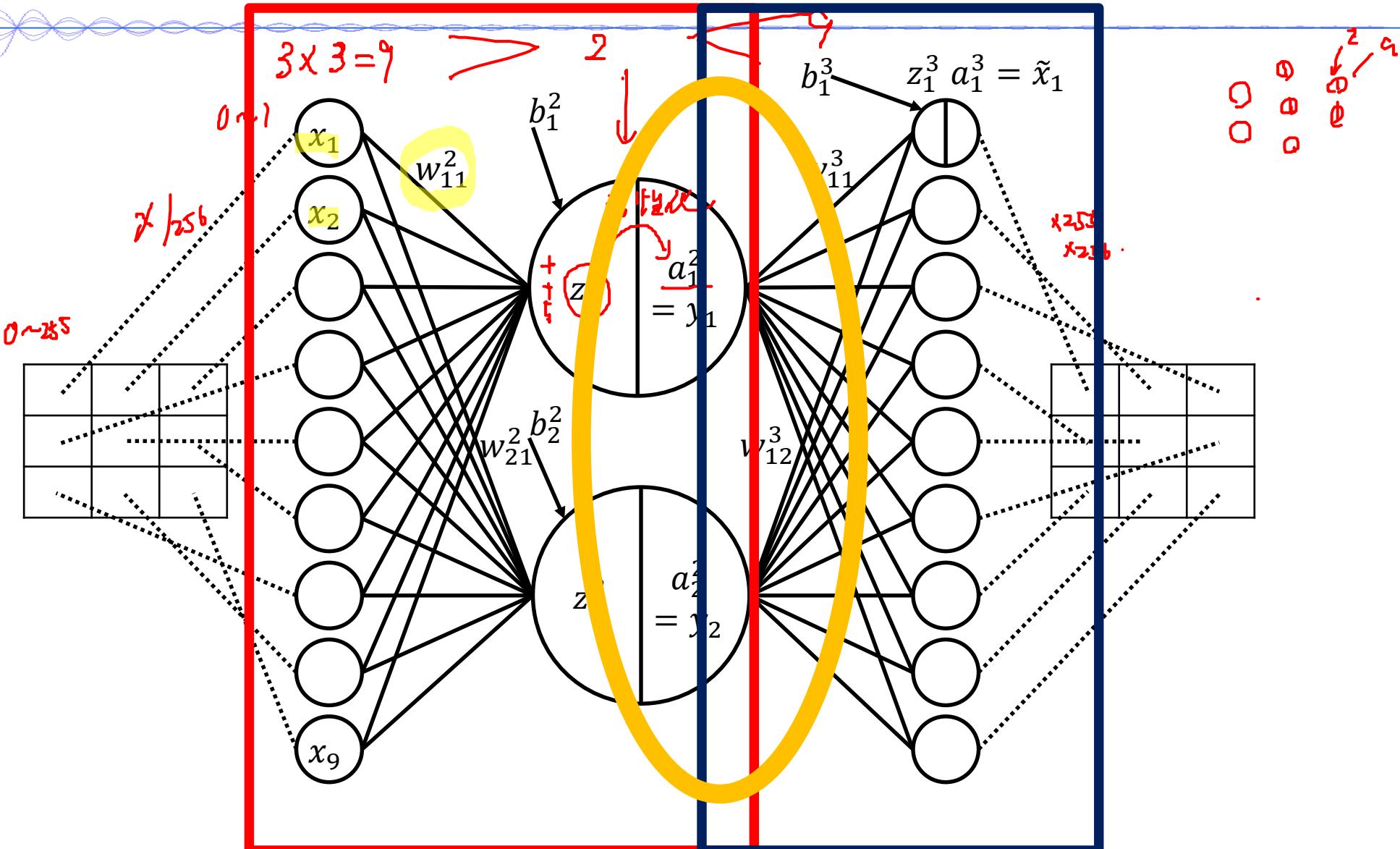
作業フォルダ: .\VAE_HW



dut_forward_vae回路の実装とFPGA上での検証

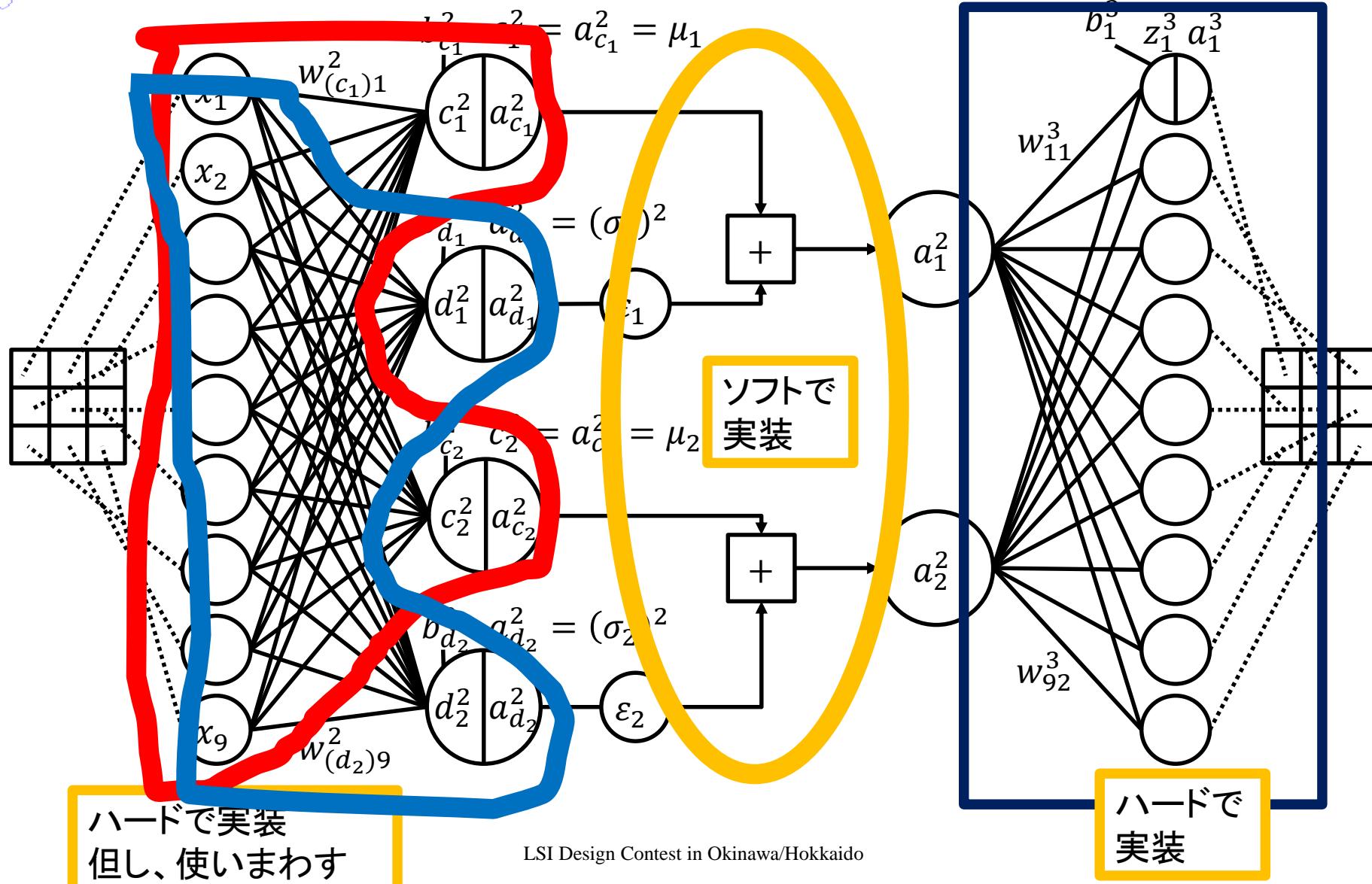


AEの例



VAEの例

- ・潜在変数 z^2 の平均を c^2 , 偏差を d^2 と表記している
- ・下付きの c_i^2, d_i^2 は c_i, d_i と表記している





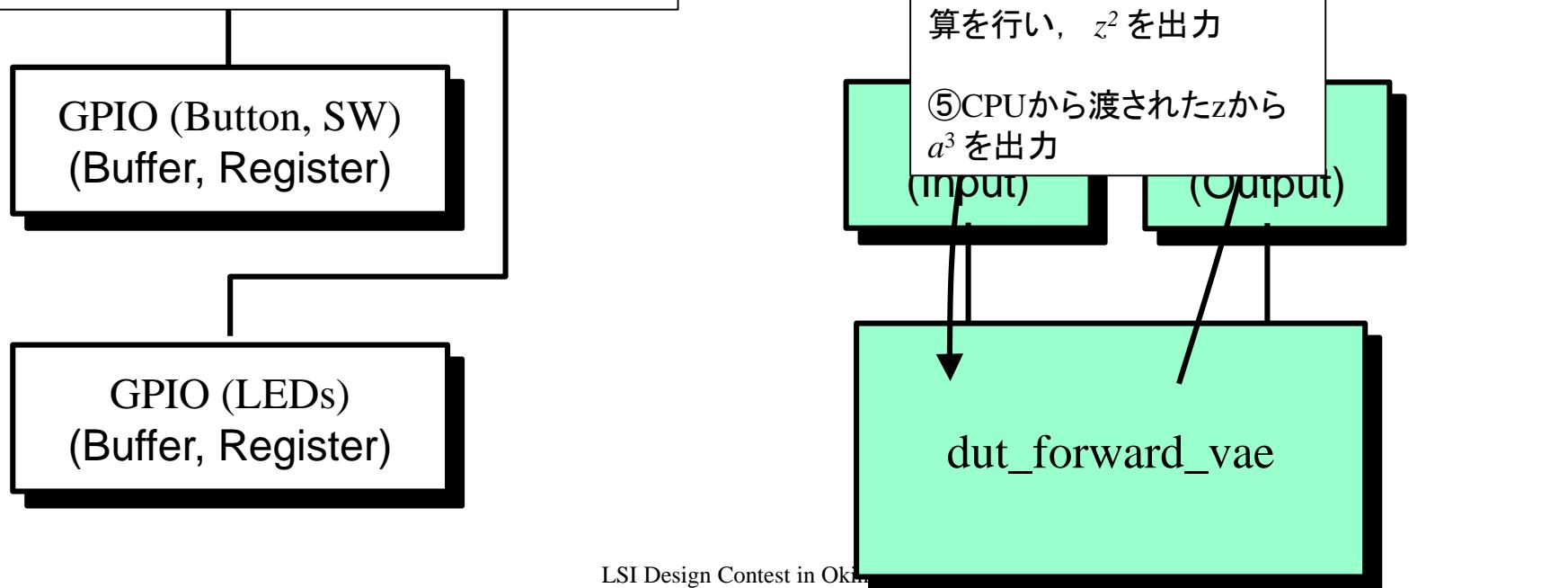
AEとVAEの構造についての気づき

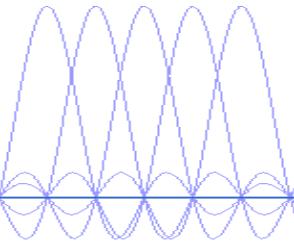
- 中間層から出力(復元層)までの構造がAEとVAEと同じ
- VAEの中間層(左)がAEに比べて増えている
- AE、VAEの入力は同じである
 - AE、VAEの入手力はすべて同じ
- 入力層から中間層に向けて、AEもVAEもすべてつながっている

dut_forward_vae回路とCPUとの関係

CPU: ウェイトやバイアスの管理
バックプロパゲーションの計算

- ①CPUはウェイトやバイアス、入力信号を回路に渡す
- ③ z^2 から a^2_mean や a^2_var を求める。
(②、③を2回繰り返す。1回目はmean、22回目はvarを計算する。)
- ④ a^2_mean や a^2_var から z を計算し回路に渡す。
- ⑥CPUは得られた計算結果を使って、バックプロパゲーションの計算を行い、ウェイトとバイアスをアップデートする。





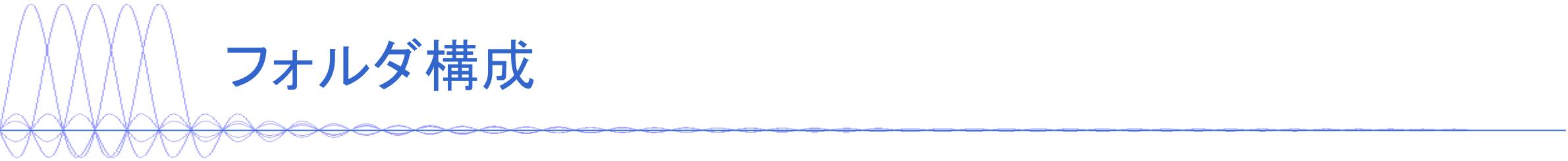
dut_forward_vae回路(ブロック)の概要

■ 入力

- ウェイトw2: 入力層から中間層へのウェイト(w2_12-w2_19, w2_21-w2_29)
 - バイアスb2: 入力層から中間層へのバイアス(b2_1, b2_2)
 - ウェイトw3: 中間層から出力層へのウェイト(w3_11, w3_12, w3_21, w3_22, …, w3_92)
 - バイアスb3: 中間層から出力層へのバイアス(b3_1, b3_2, …, b3_9)
 - 入力信号X: (X_1, X_2, …, X_9)
 - 中間の潜在変数z(z_1, z_2)
- ビット幅: (24,16) や (32,24)

■ 出力

- 中間層のz2: (z2_1, z2_2)
- 中間層の出力a2: (a2_1, a2_2)
- 出力層のz3: (z3_1, z3_2, …, z3_9)
- 出力層の出力a3: (a3_1, a3_2, …, a3_9)

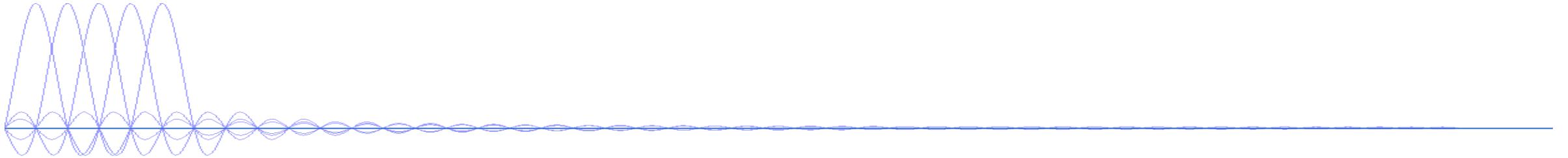


フォルダ構成

■ .\VAE_HW

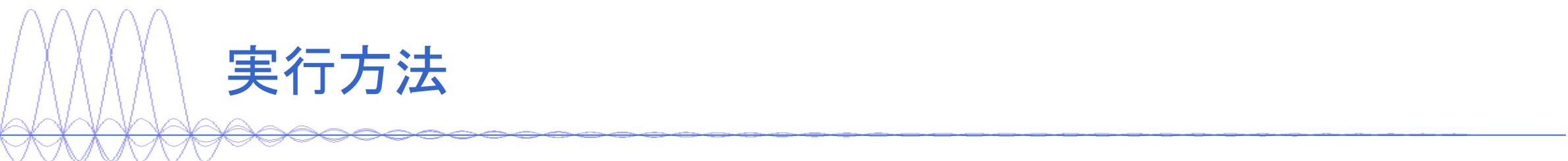
- vae_sys.c
- forward_VAE.slx
- Neuralnetwork_forward_VAE.m
- tb_forward_VAE_929.m

■ フォルダ名が長くなるとErrorするので、ある程度の長さにおさめること



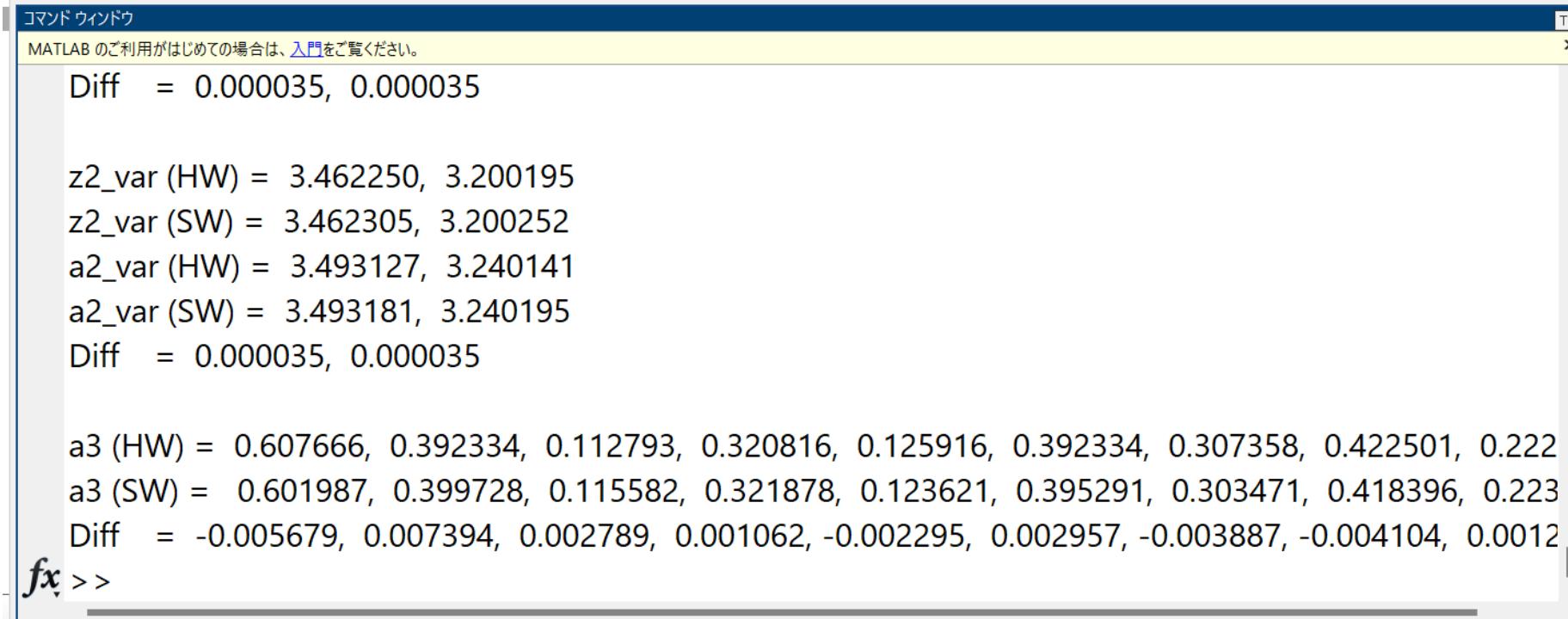
■ 以下の行数でHWを呼び出し実行している

- 123行目
- 140行目
- 168行目



実行方法

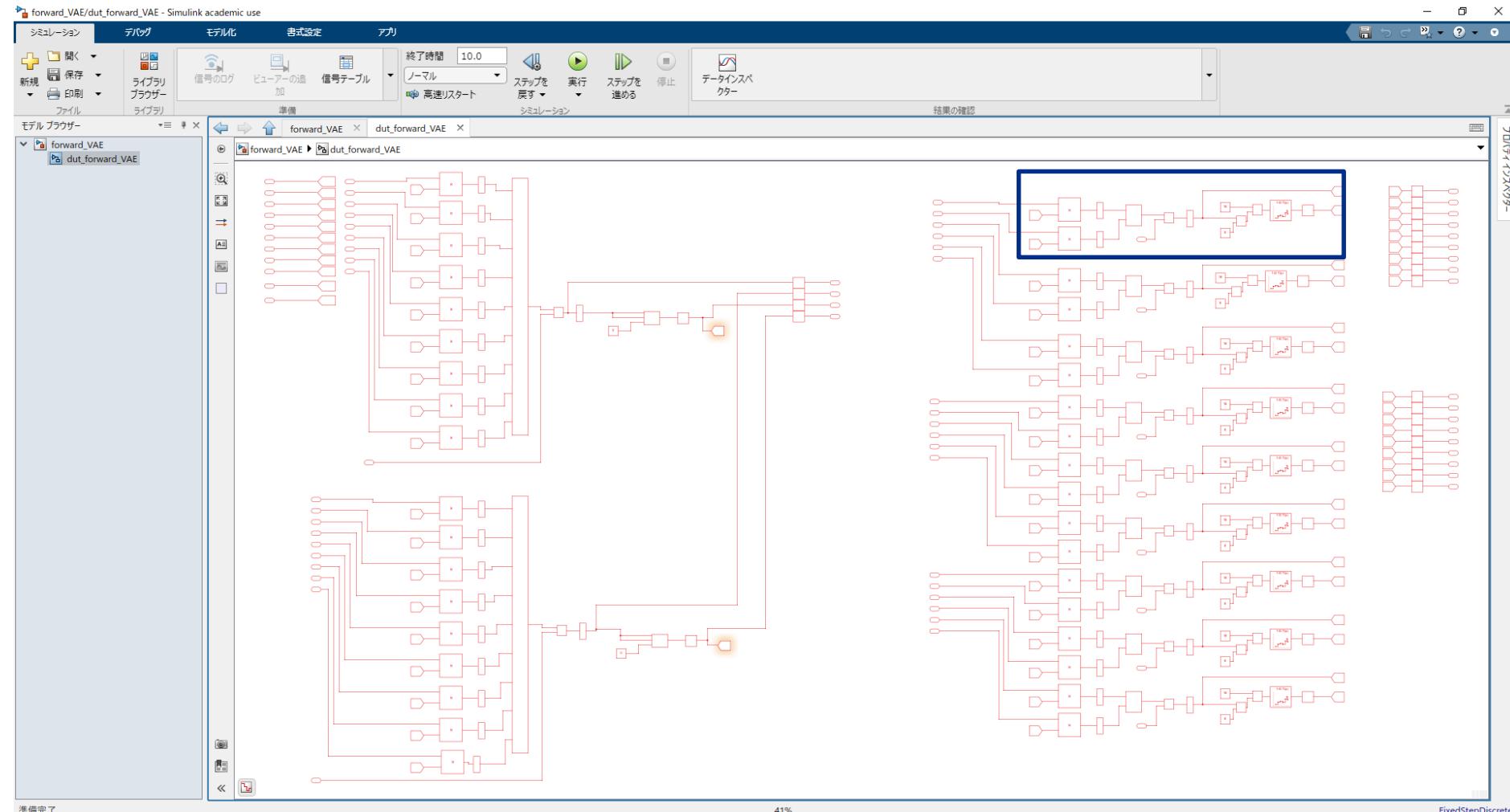
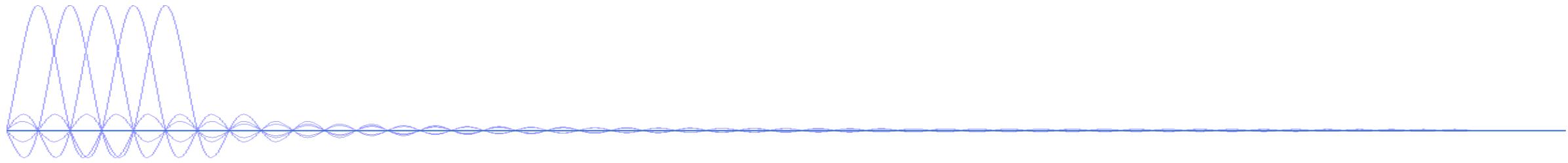
- tb_forward_VAE_929.m を起動
 - フォワードの部分の計算が実行される

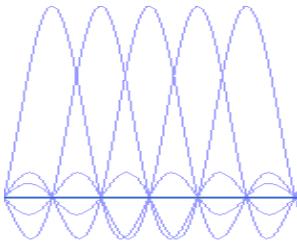


```
コマンド ウィンドウ
MATLAB のご利用がはじめての場合は、入門をご覧ください。
Diff = 0.000035, 0.000035

z2_var (HW) = 3.462250, 3.200195
z2_var (SW) = 3.462305, 3.200252
a2_var (HW) = 3.493127, 3.240141
a2_var (SW) = 3.493181, 3.240195
Diff = 0.000035, 0.000035

a3 (HW) = 0.607666, 0.392334, 0.112793, 0.320816, 0.125916, 0.392334, 0.307358, 0.422501, 0.222
a3 (SW) = 0.601987, 0.399728, 0.115582, 0.321878, 0.123621, 0.395291, 0.303471, 0.418396, 0.223
Diff = -0.005679, 0.007394, 0.002789, 0.001062, -0.002295, 0.002957, -0.003887, -0.004104, 0.0012
fx >>
```

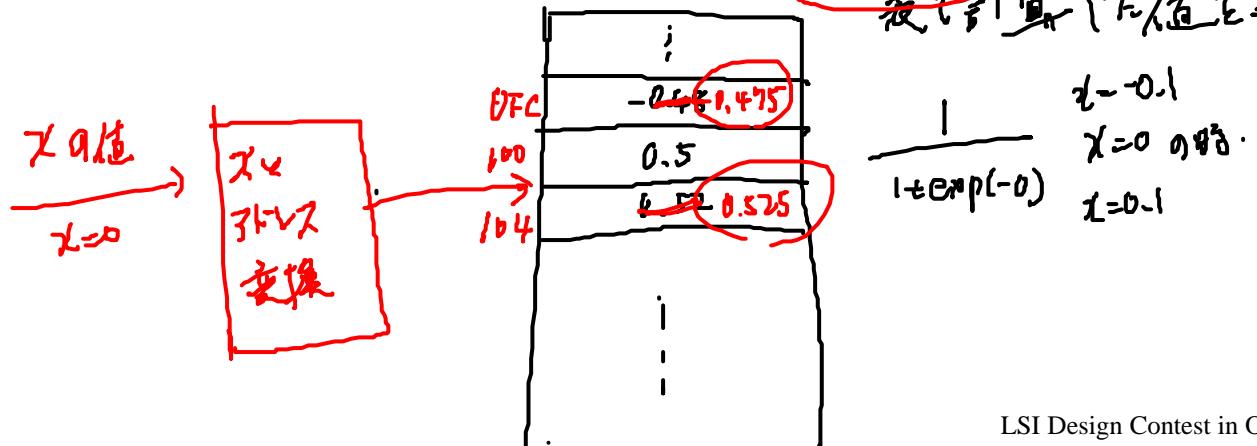




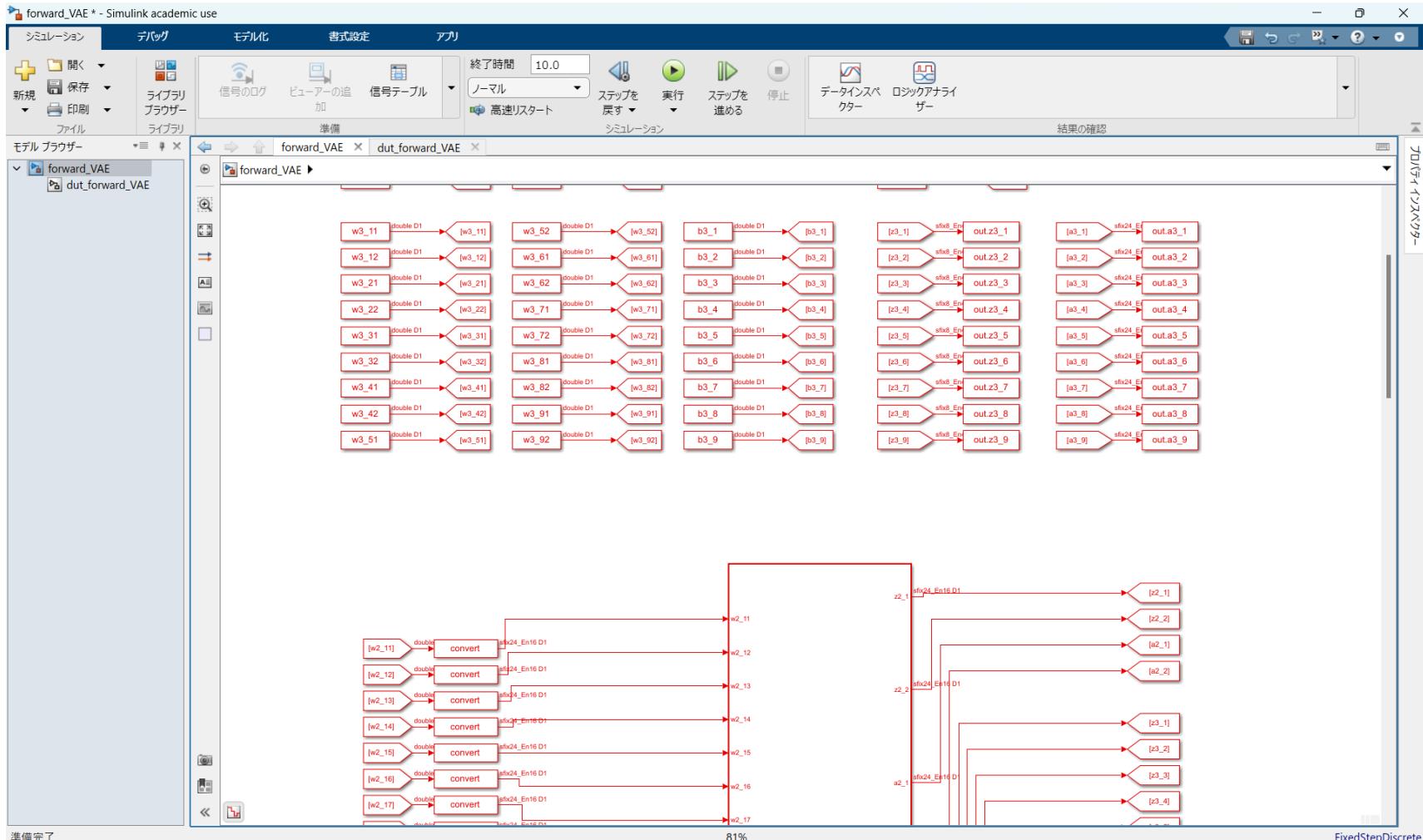
Sigmoid (非線形の演算について)

- $a_i^2 = \frac{1}{1+\exp(-z_i^2)}$
 - HWで計算することは難しい。
 - 割り算がある
 - e^{-x} の計算がある
 - 最初にいくつかの x で $\frac{1}{1+\exp(-x)}$ を計算しておく

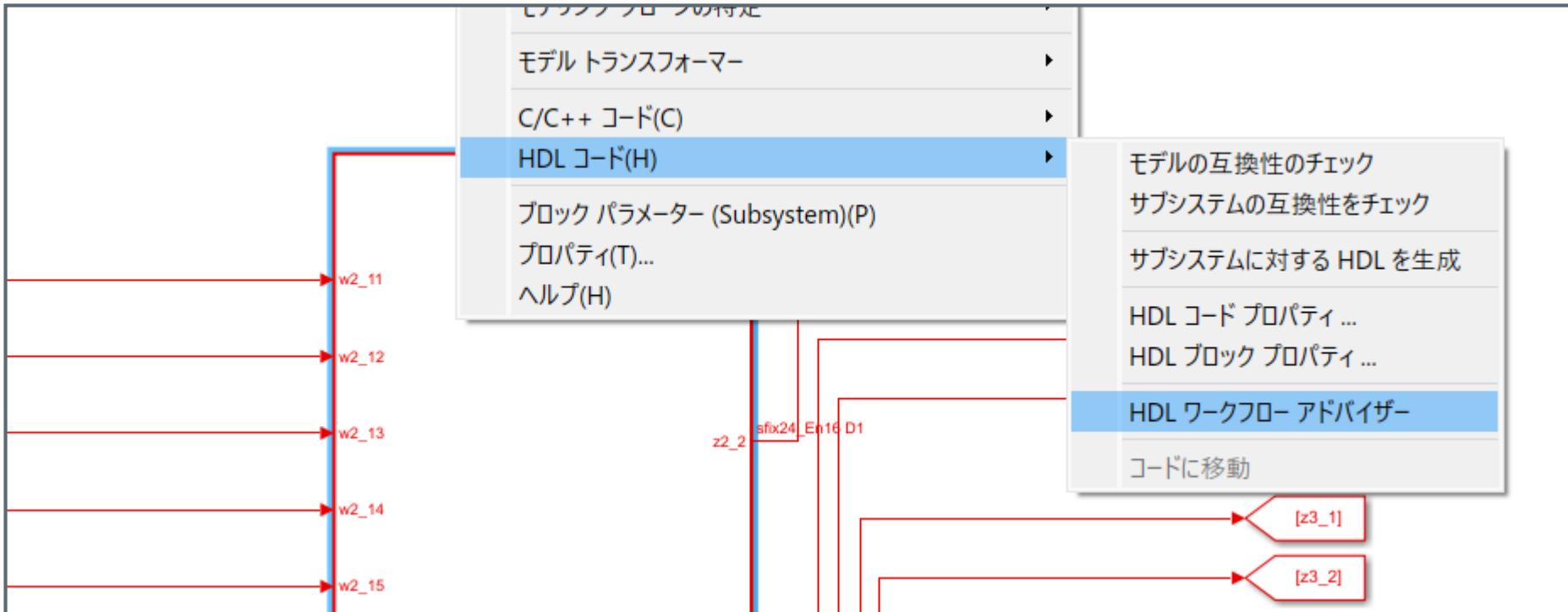
~~表計算した値をもとめる~~



forwar_VAE.slx



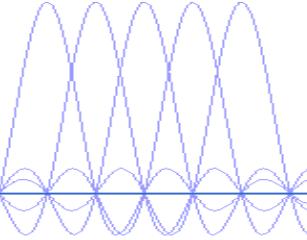
ハードウェアの生成(R2020b) HDL ワークフローアドバイザーの起動



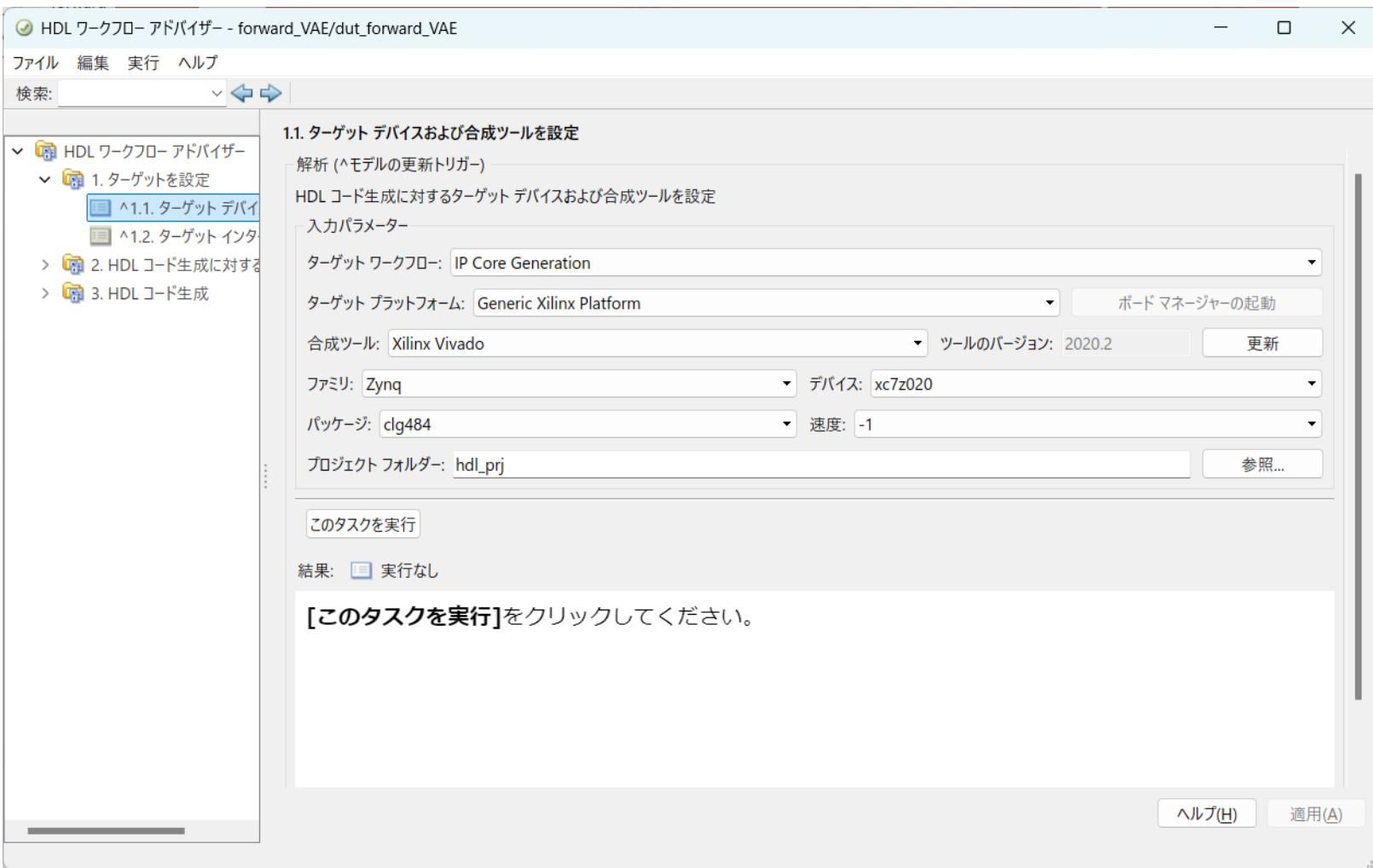
HDL化したいブロックを選択後
→ 右クリック
→ HDLコード → HDL ワークフローアドバイザ を選択

HDL ワークフロー アドバイザー起動時

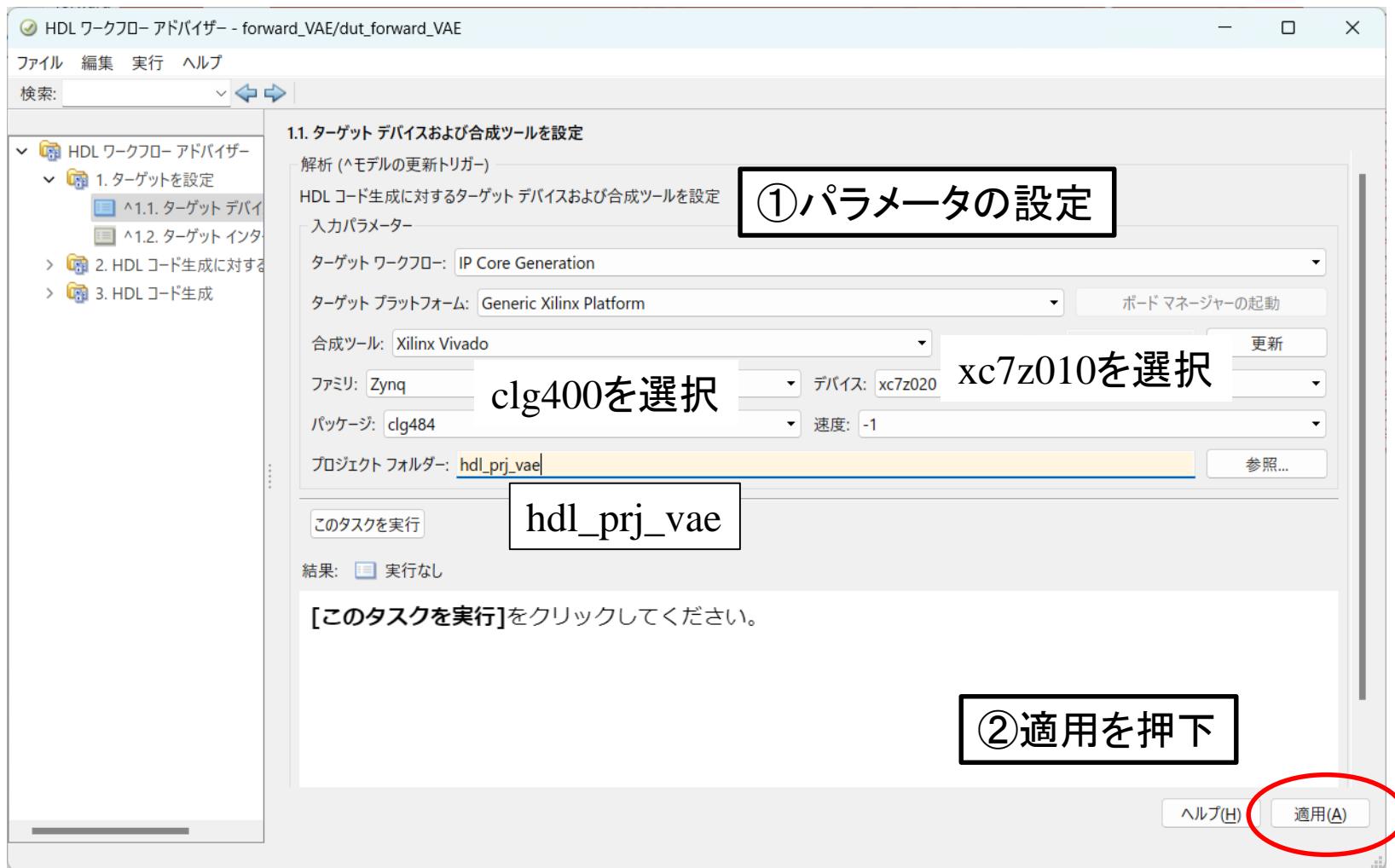


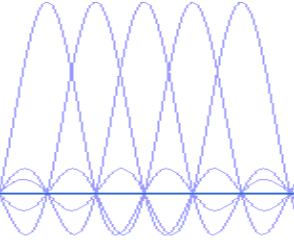


HDL ワークフローアドバイザー ターゲットの設定

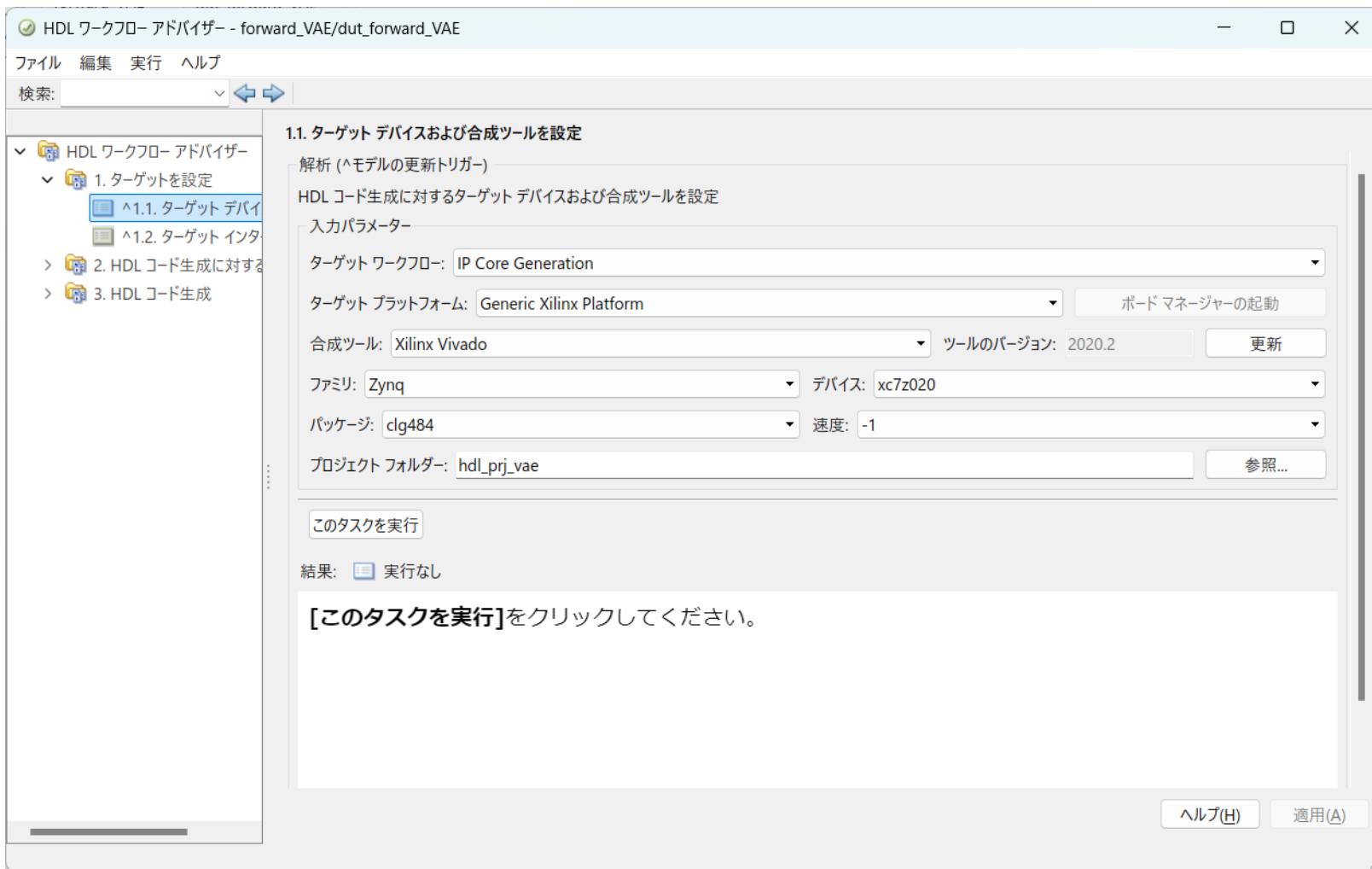


HDL ワークフローアドバイザー ターゲットの設定

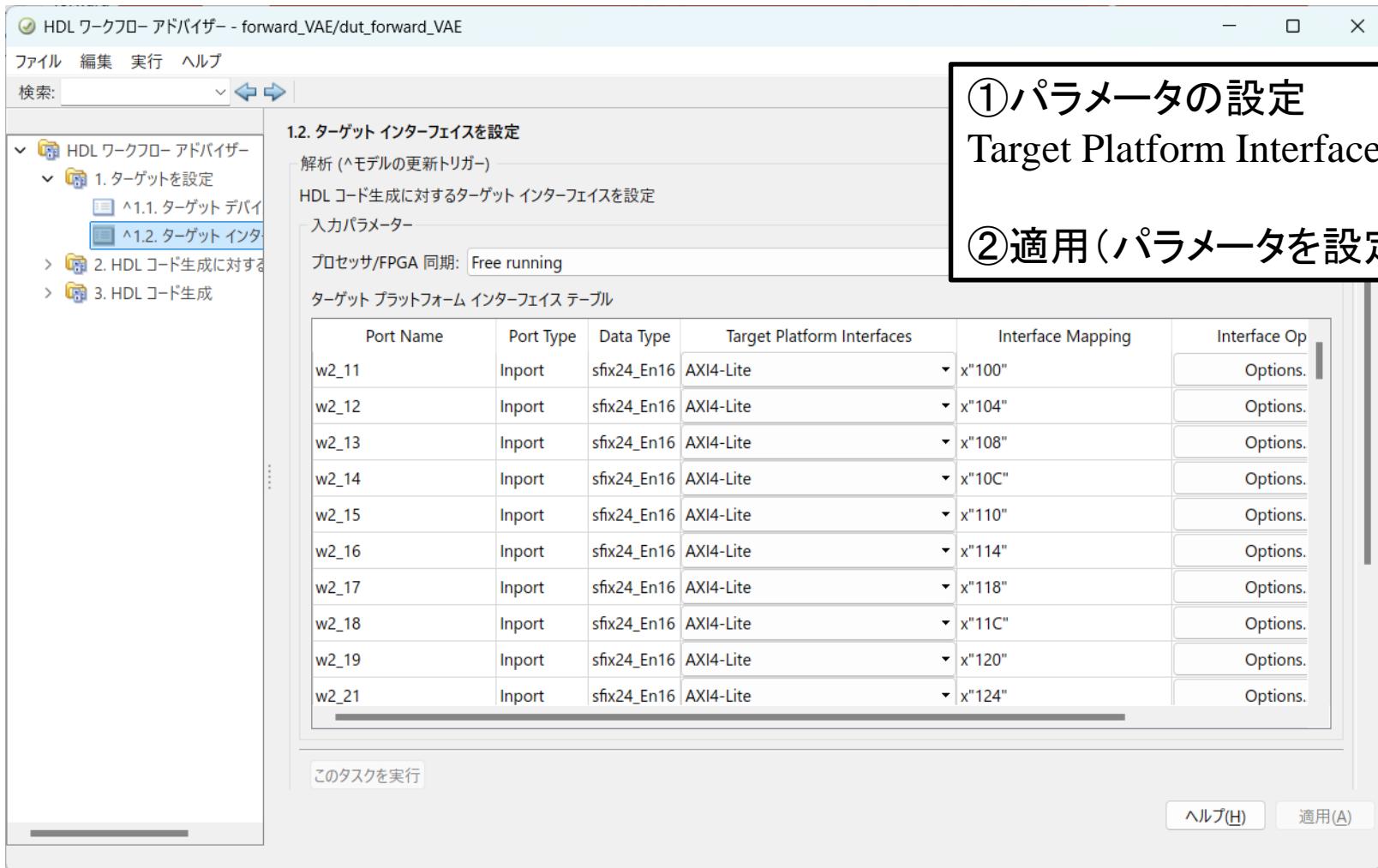




HDL ワークフローアドバイザー ターゲットの設定



HDL ワークフローアドバイザー ターゲットの設定



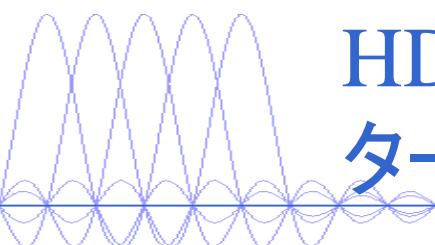
①パラメータの設定

Target Platform Interfaces : AXI4-Lite

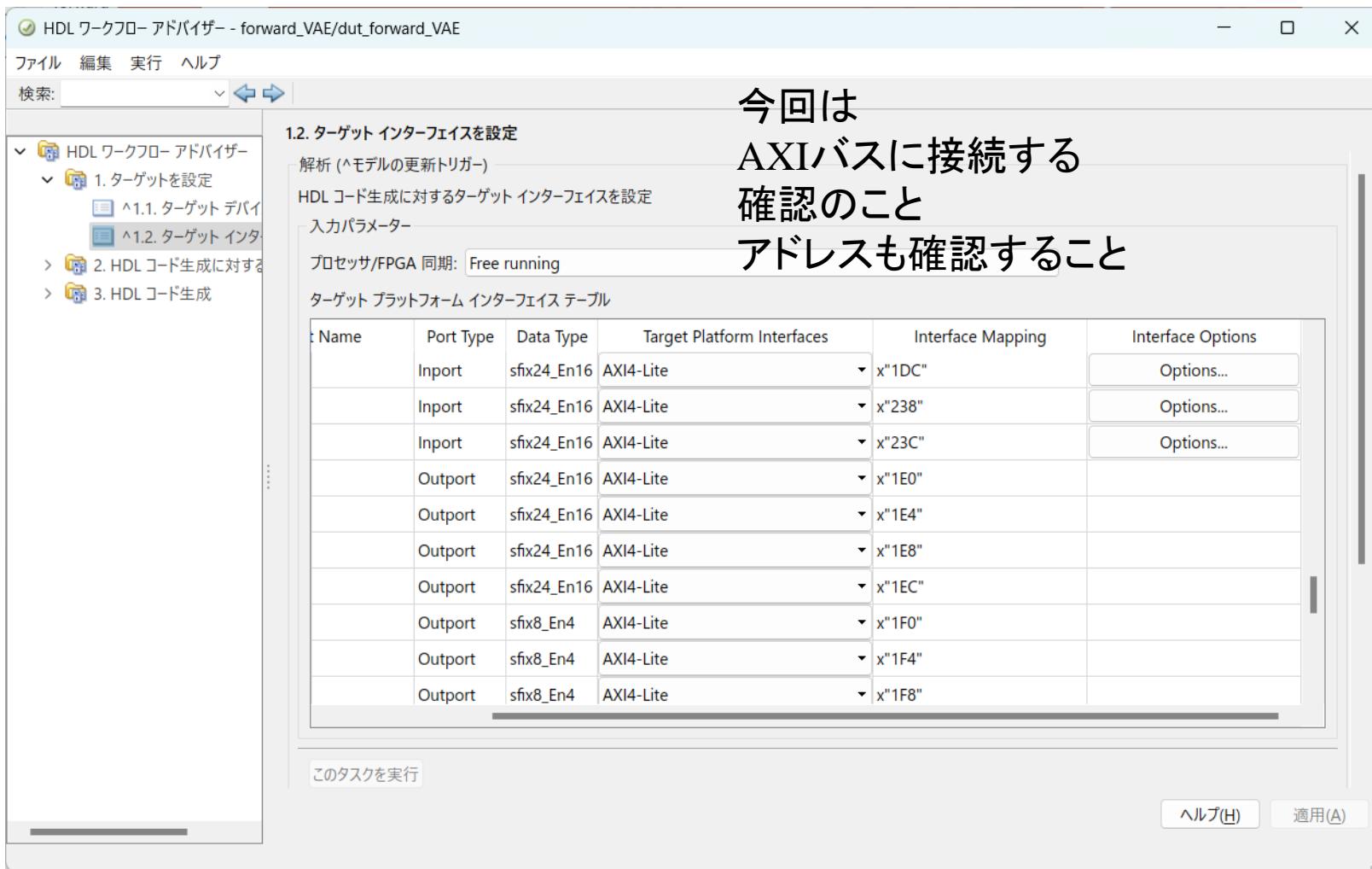
②適用(パラメータを設定した場合)

今回は
AXIバスに接続

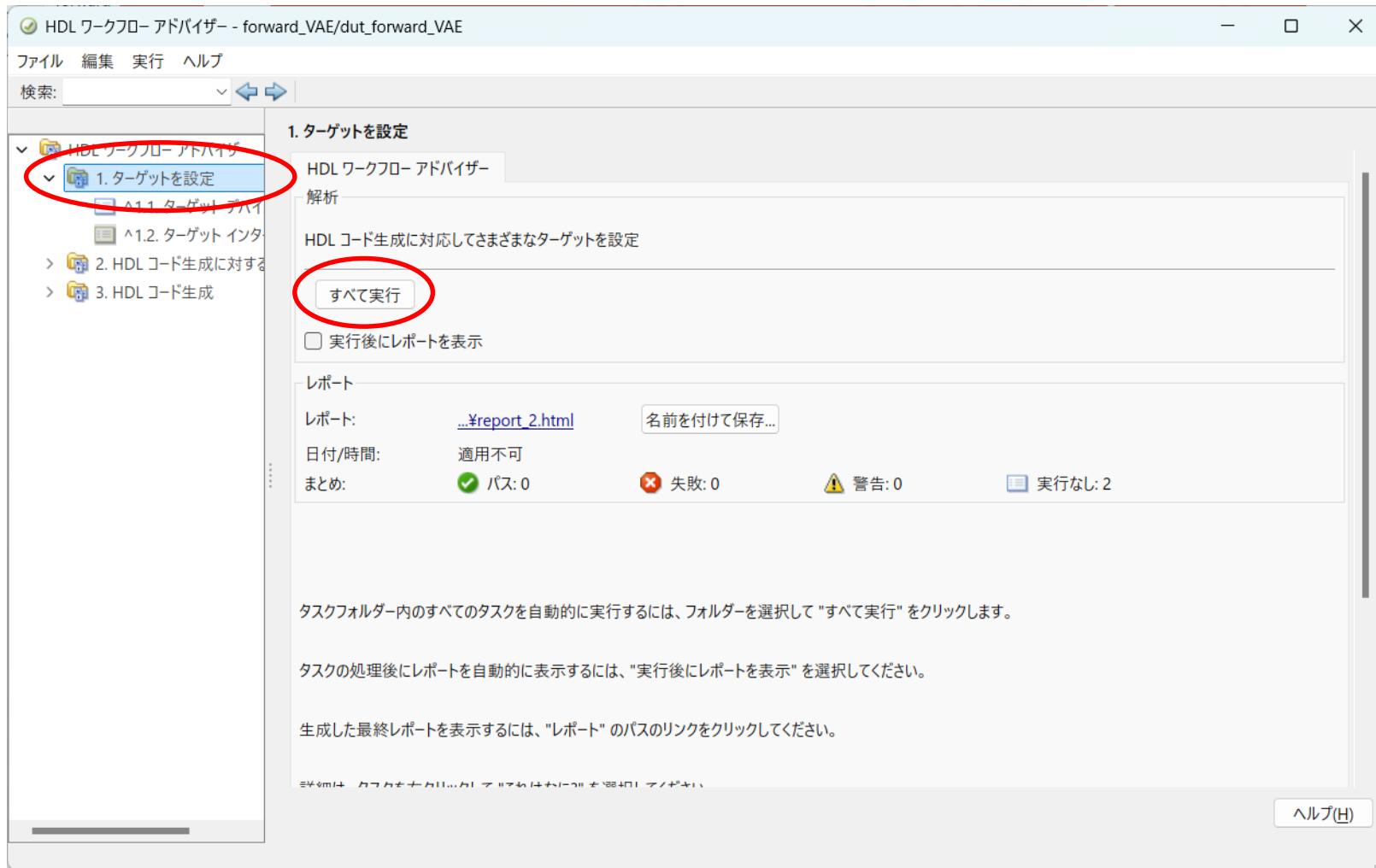
確認のこと
アドレスも確認



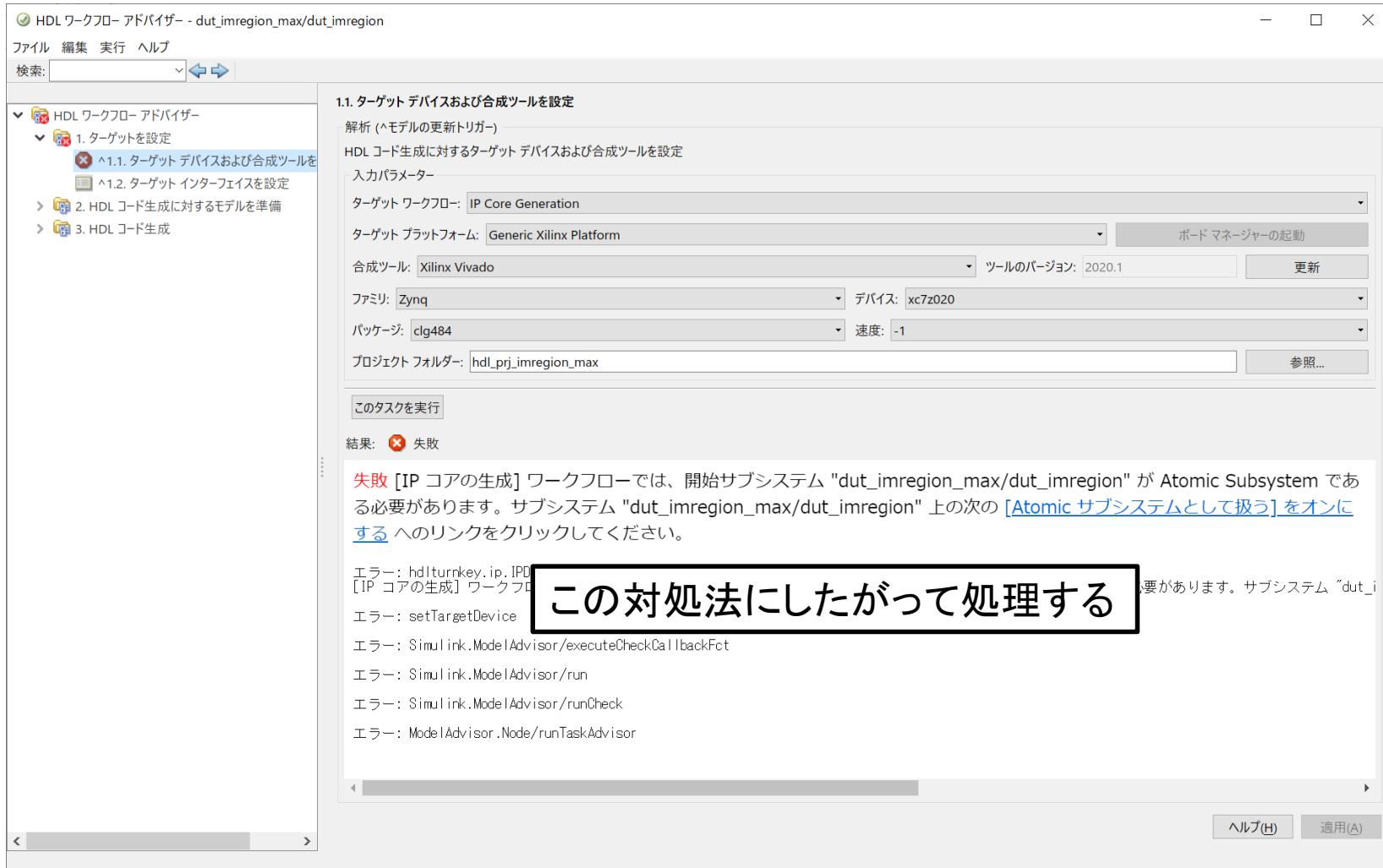
HDL ワークフローアドバイザー ターゲットの設定



HDL ワークフローアドバイザー ターゲットの設定

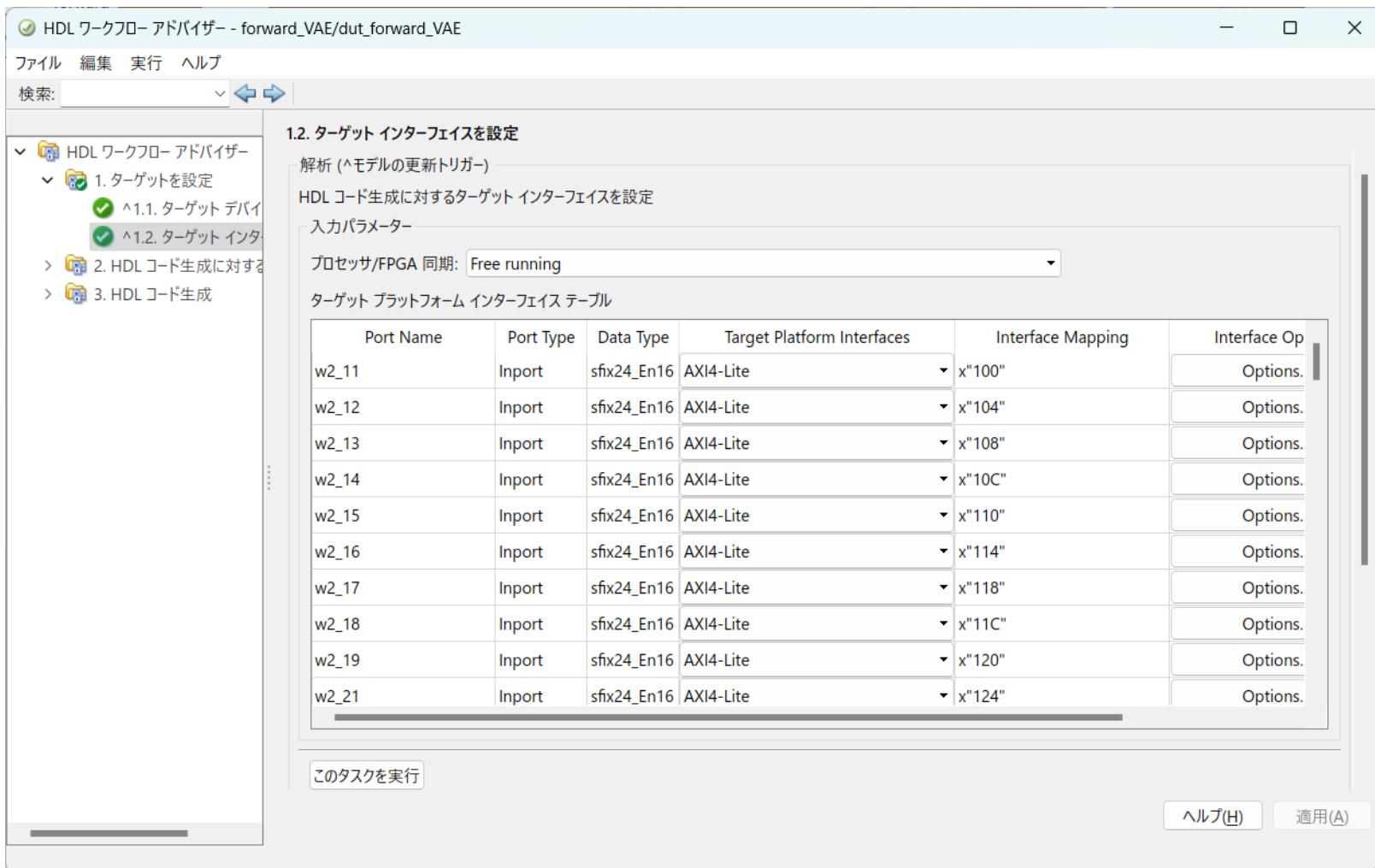


エラーがでたら

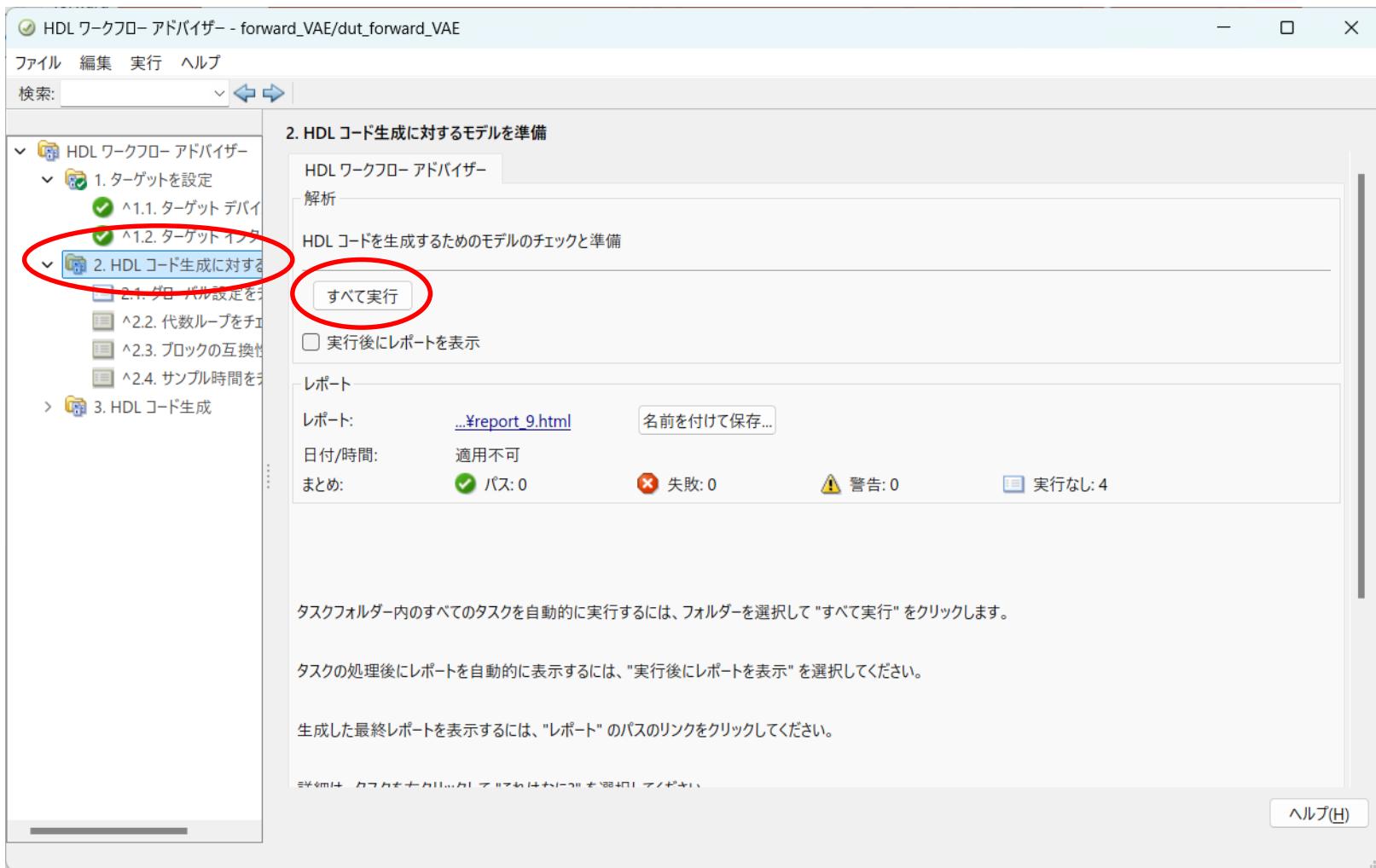




HDL ワークフローアドバイザー ターゲットの設定



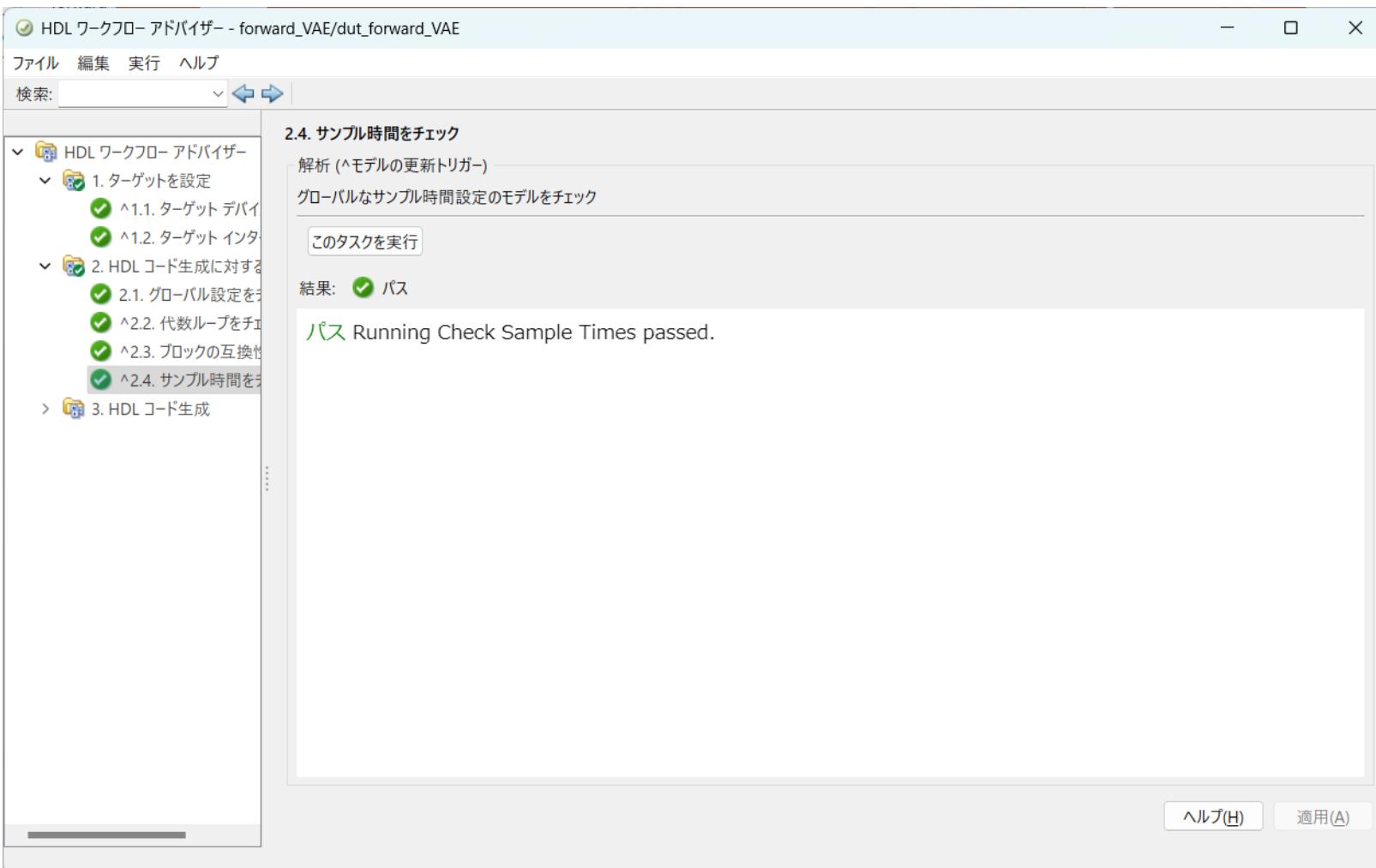
HDL ワークフローアドバイザー HDLコード生成に対する設定



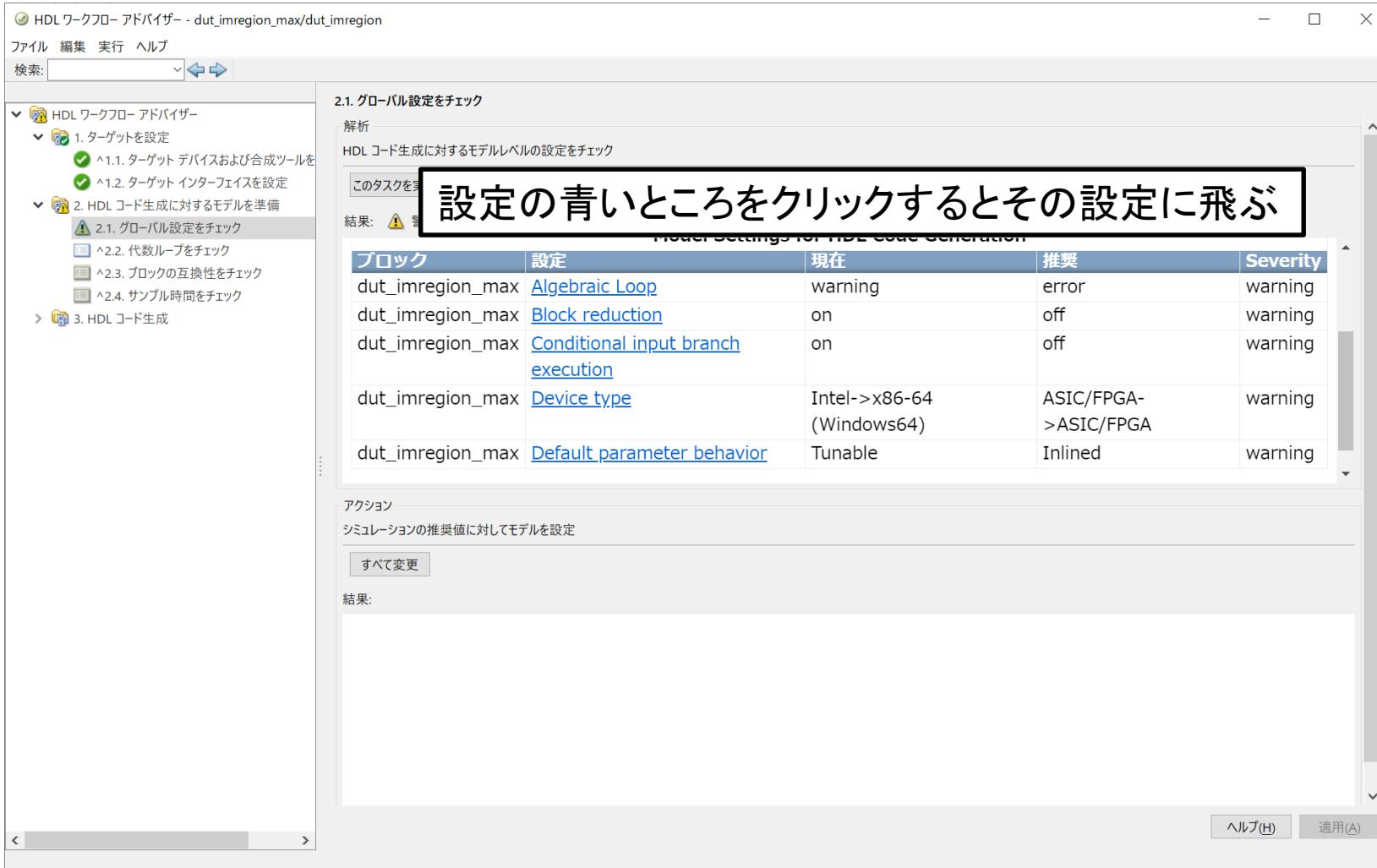


HDL ワークフローアドバイザー

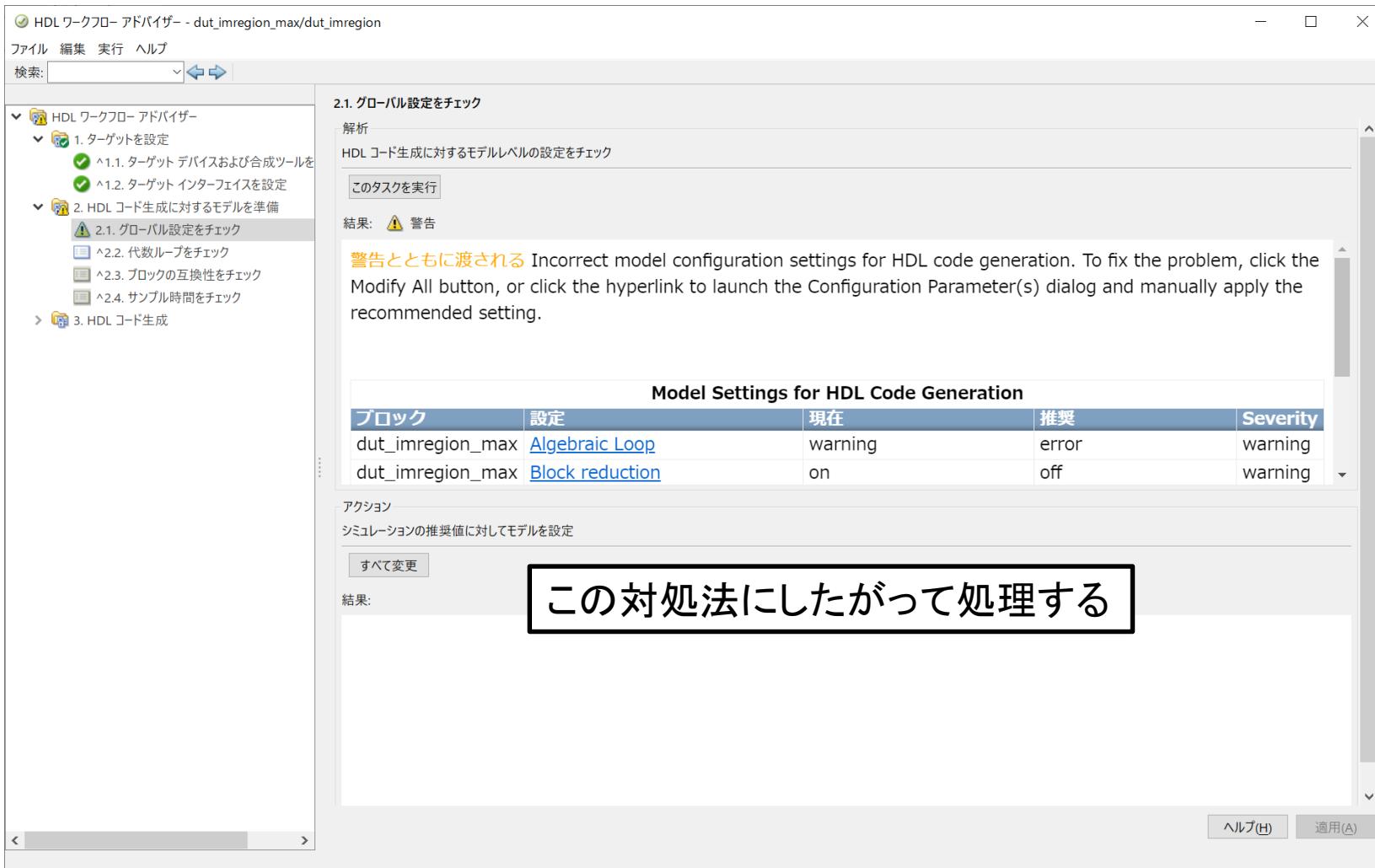
HDLコード生成に対する設定



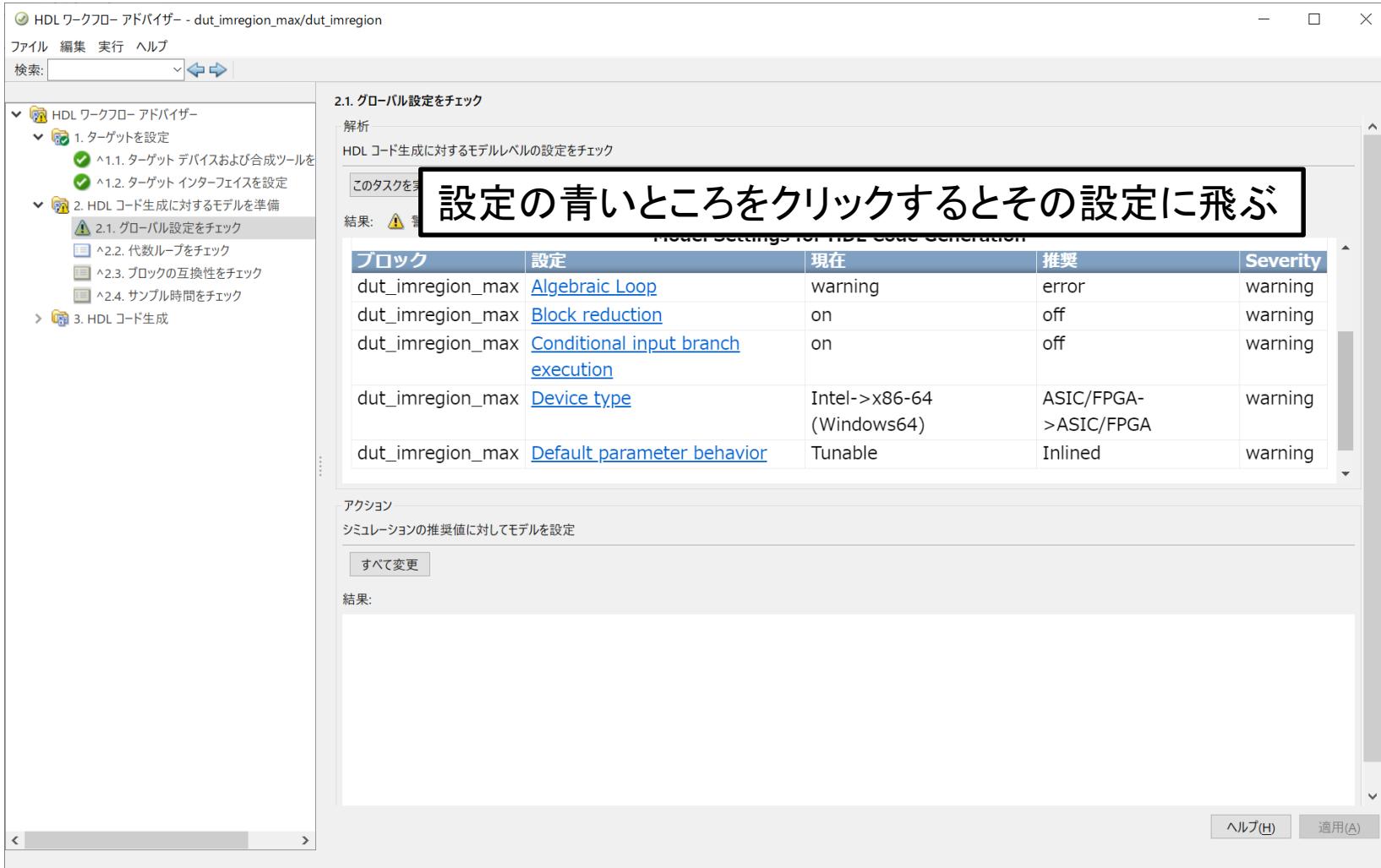
警告が出てきた場合は

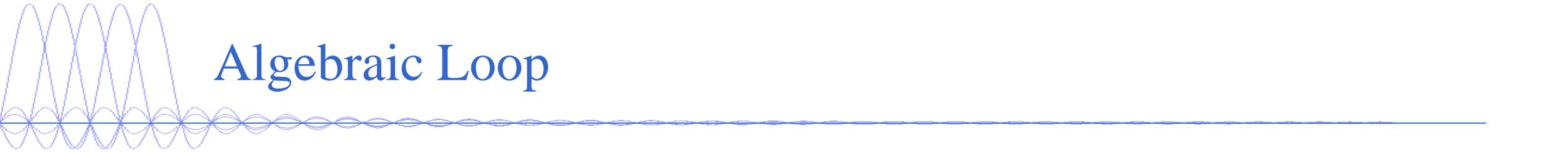


警告が出てきた場合は

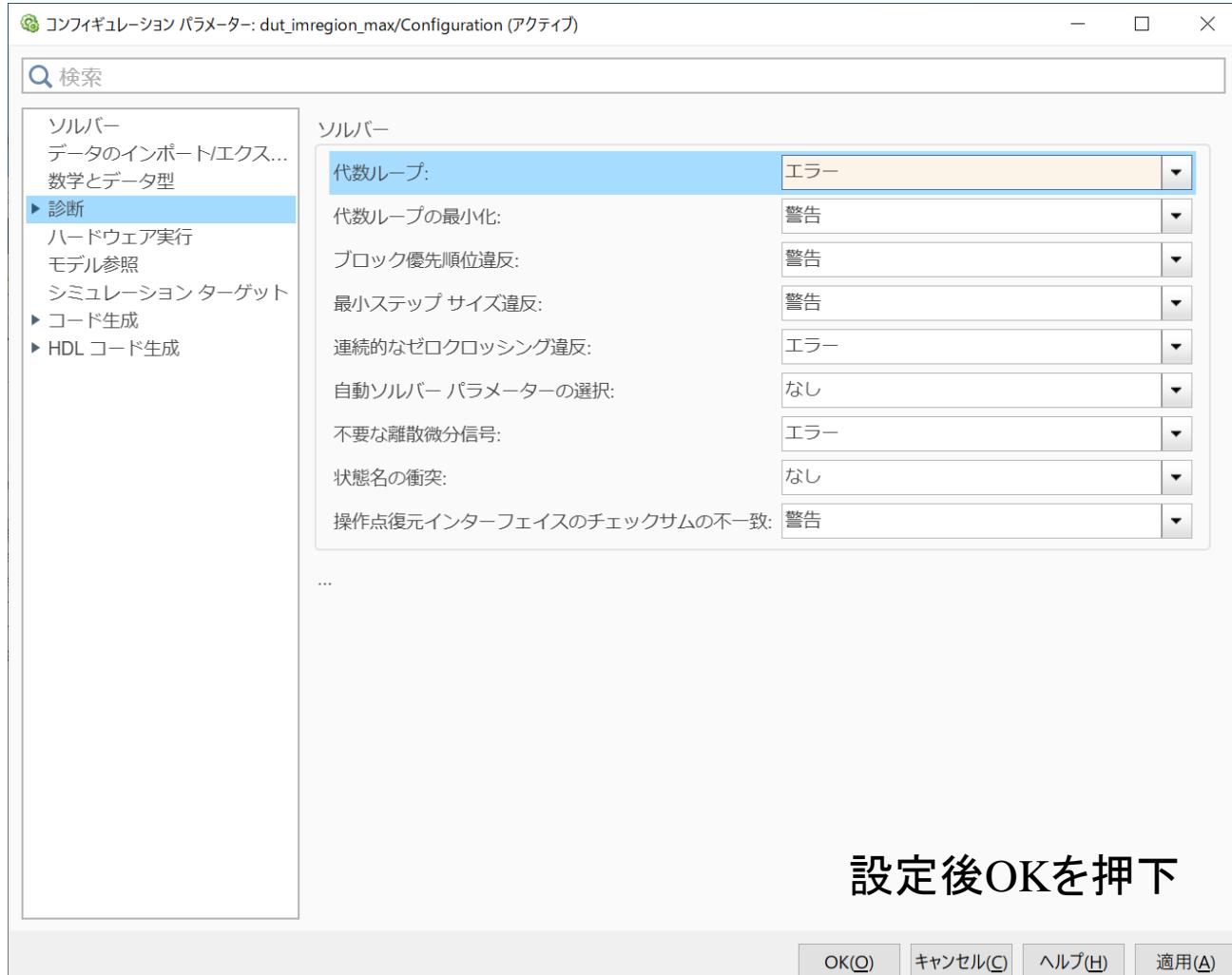


警告が出てきた場合は

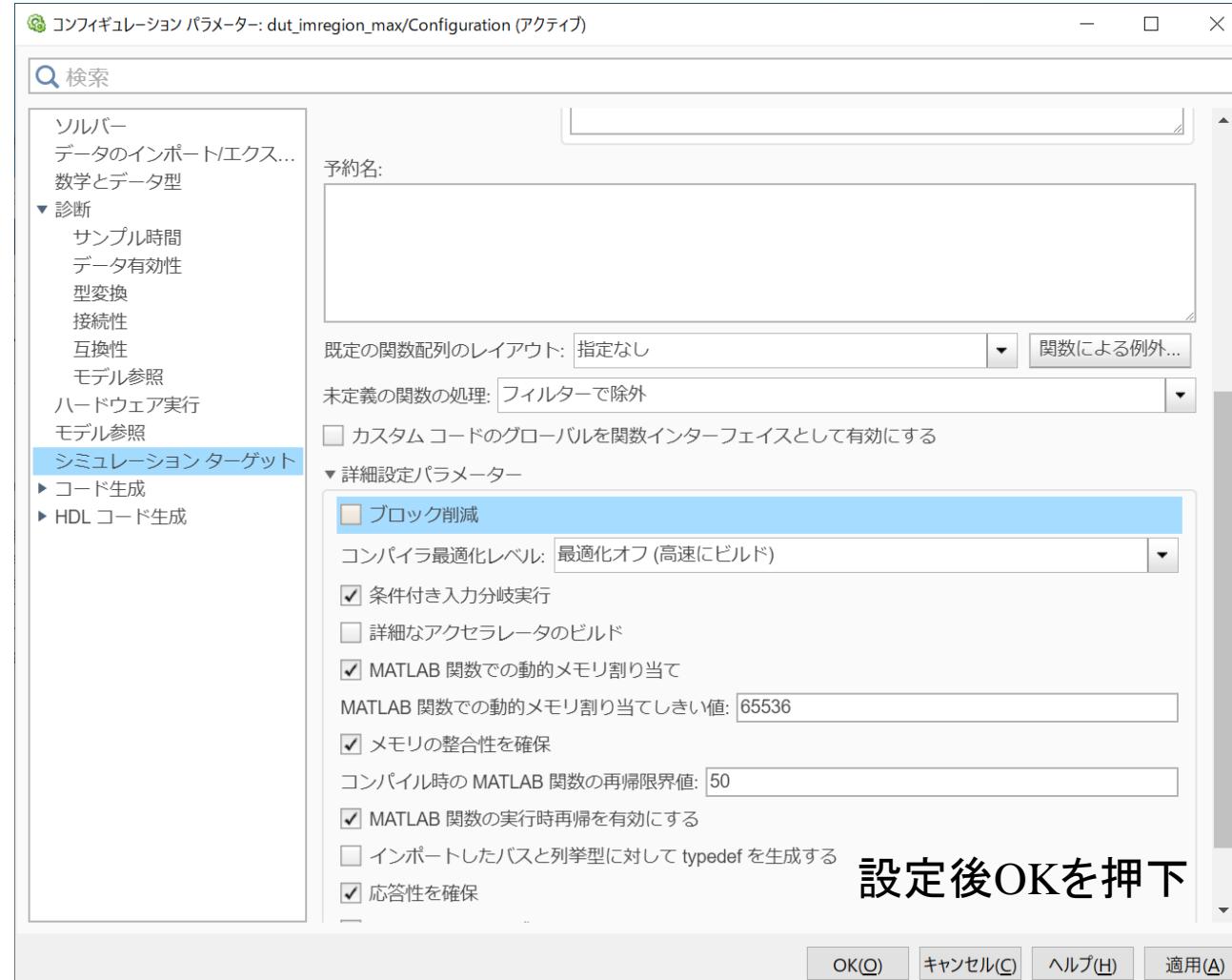




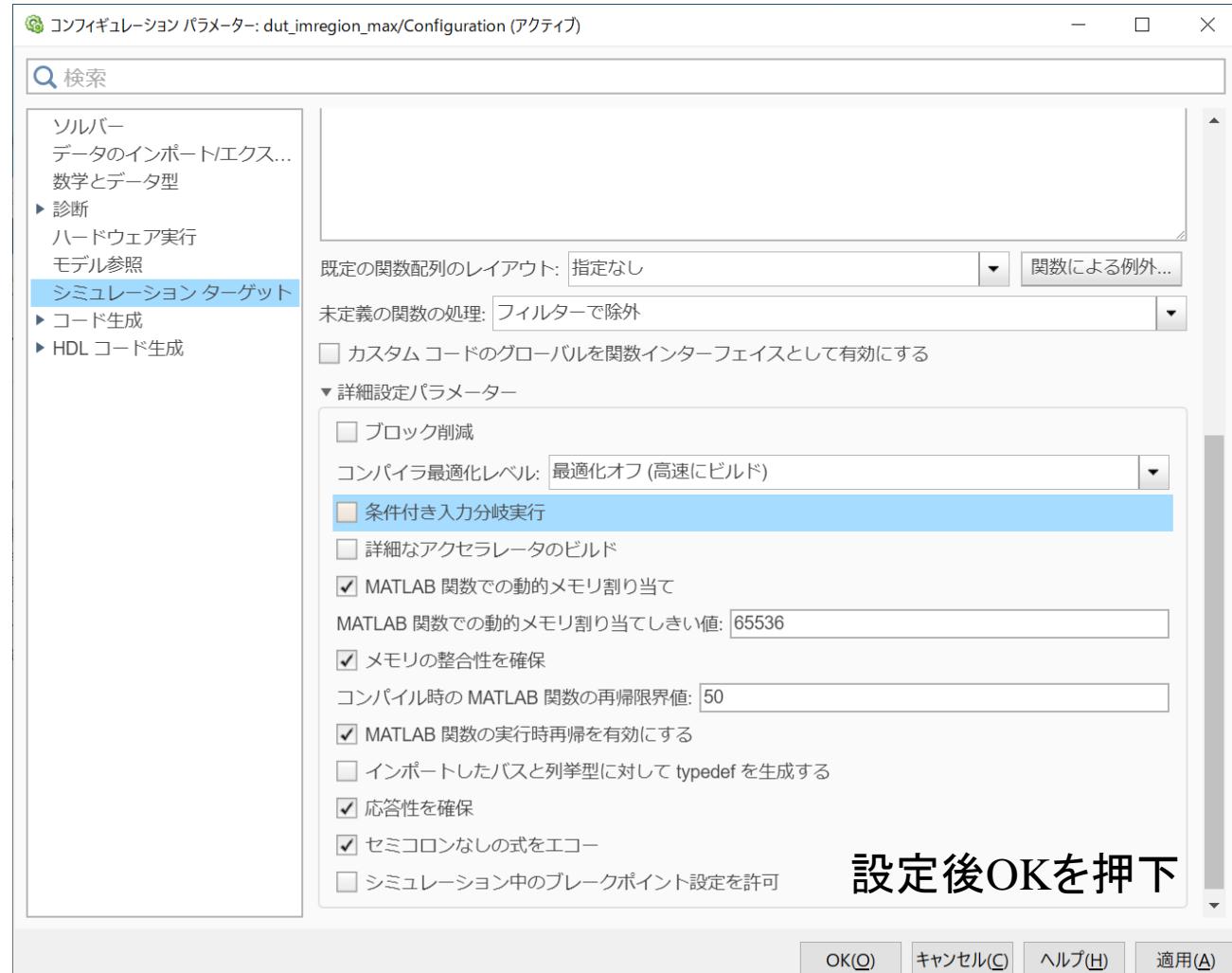
Algebraic Loop

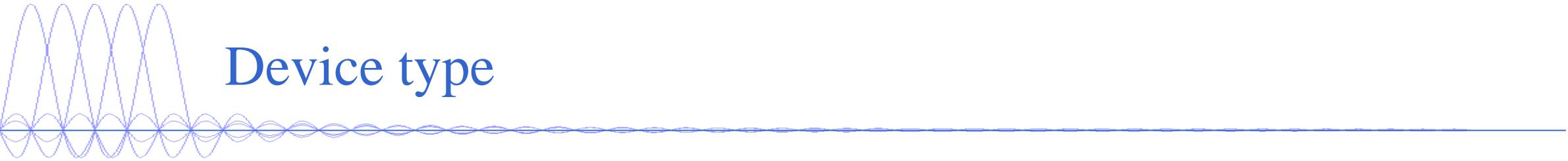


Block reduction

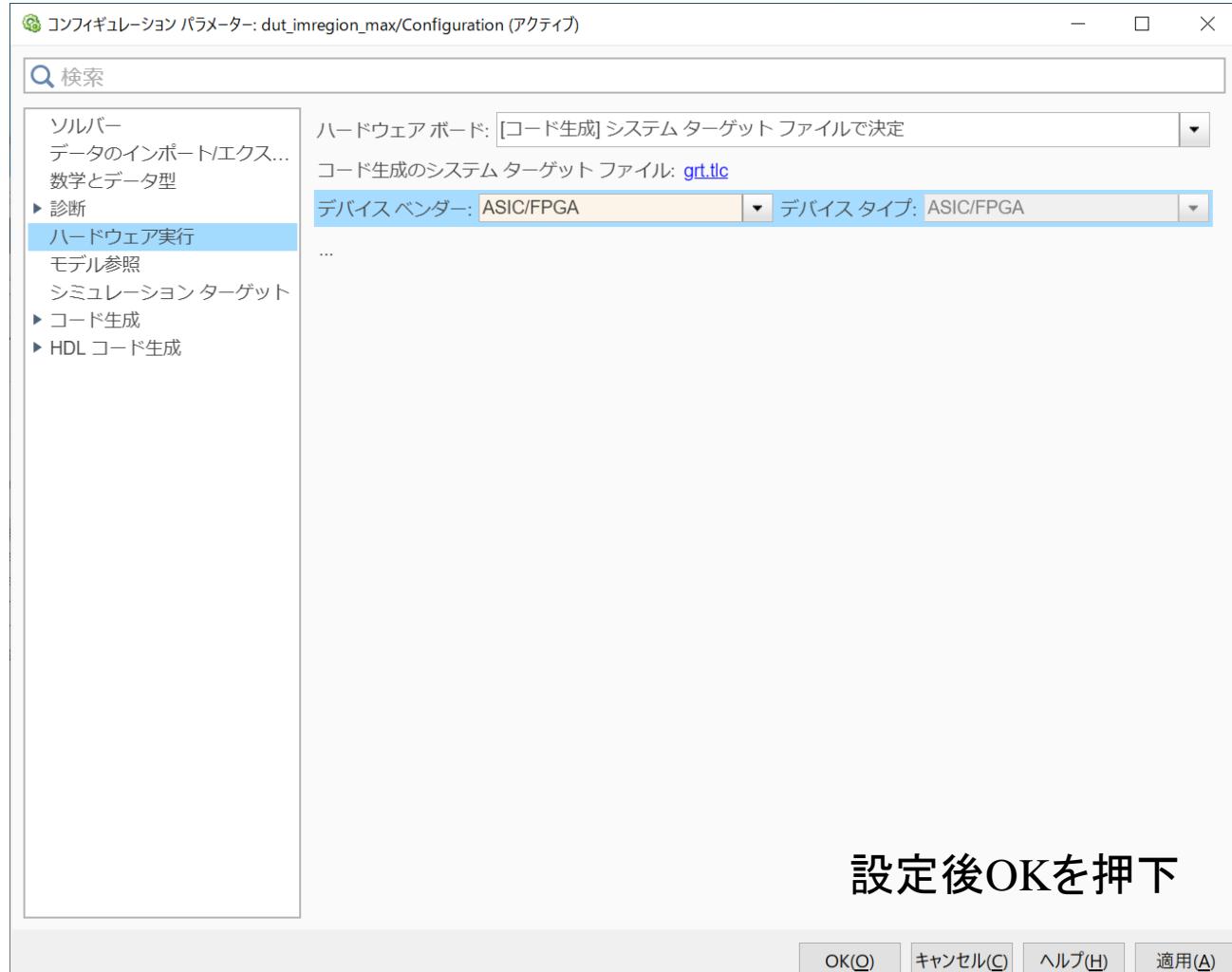


Conditional input branch execution

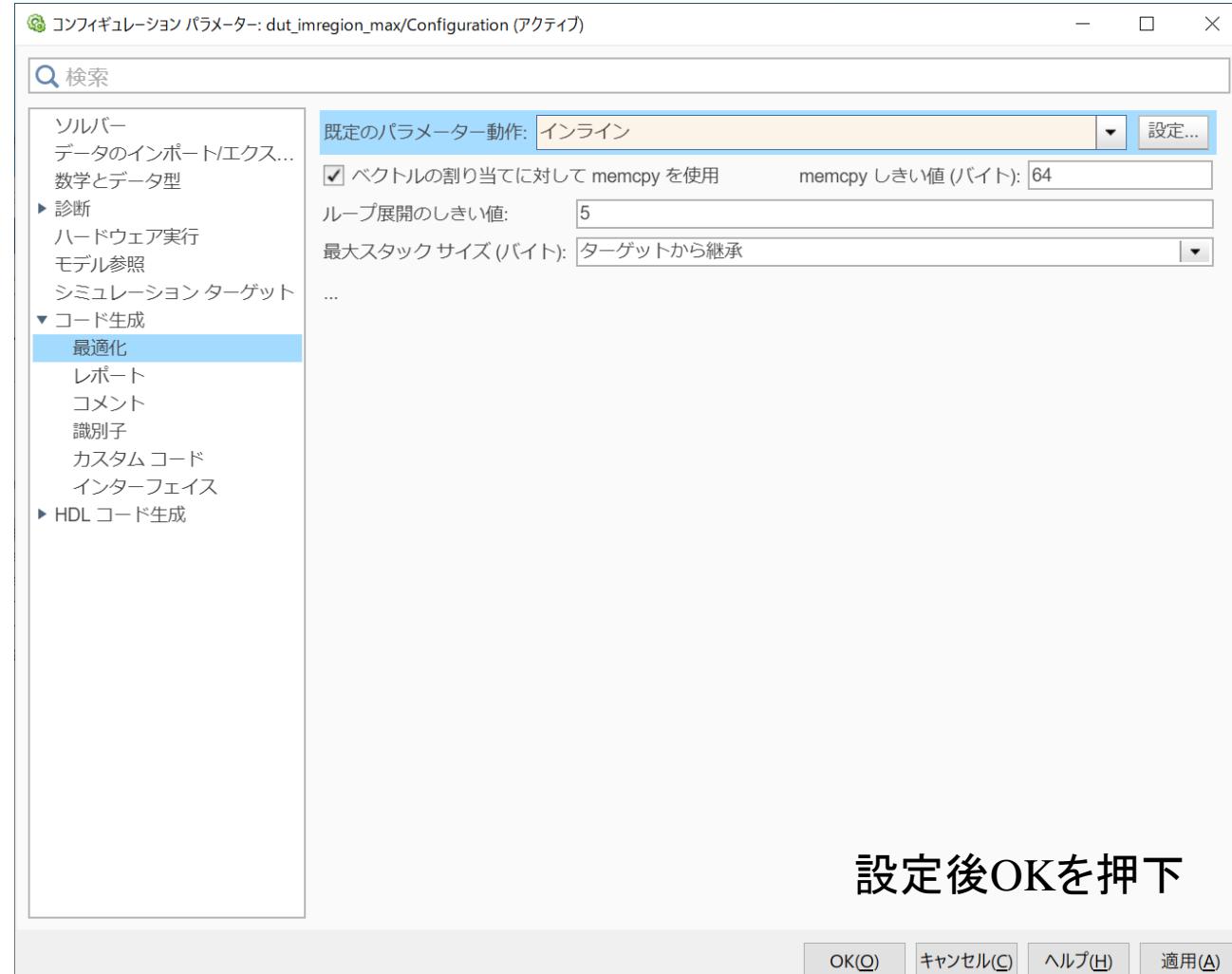




Device type

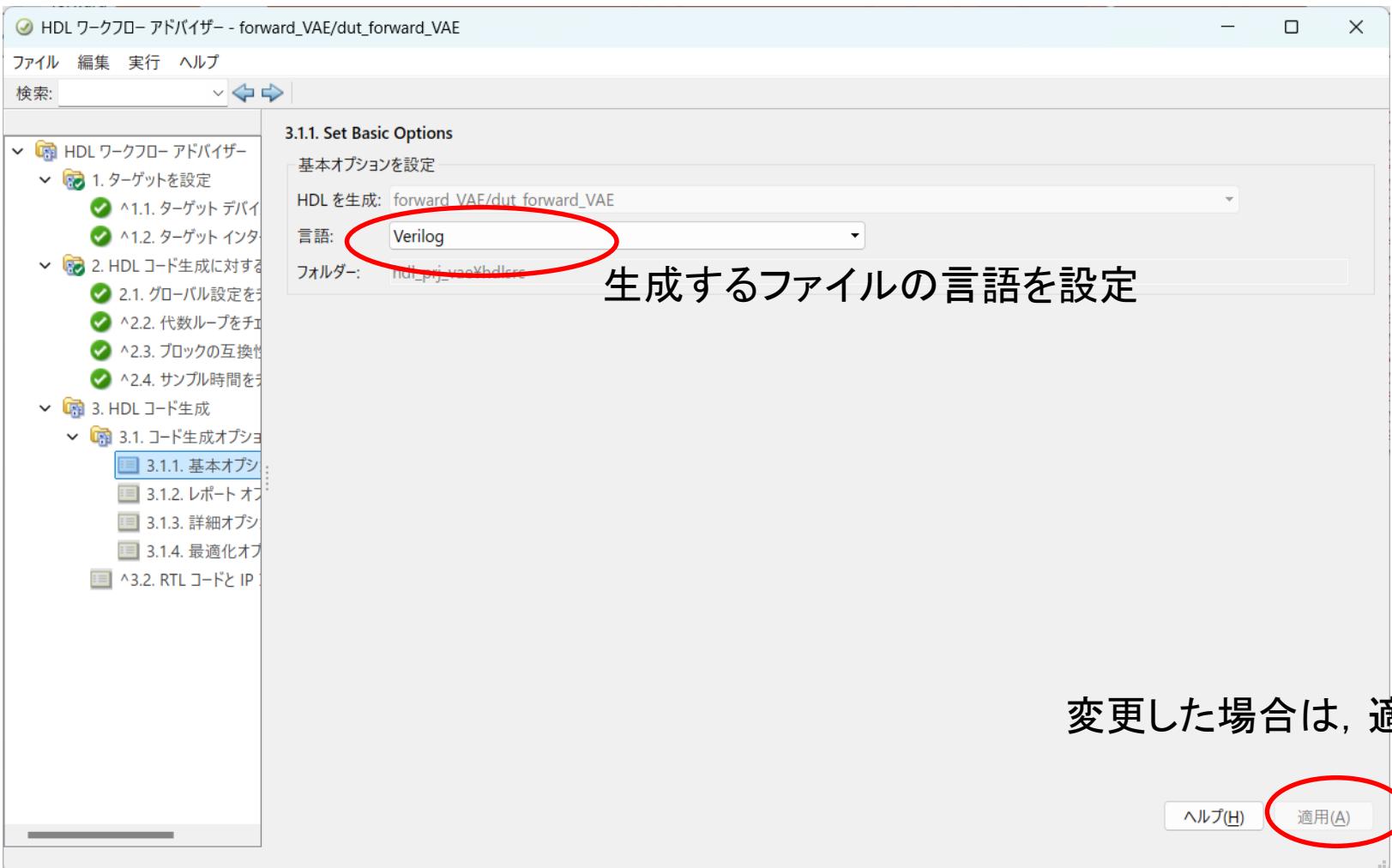


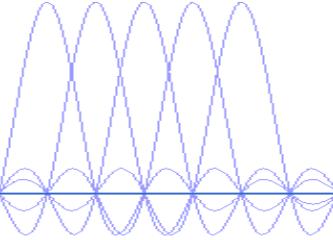
Default parameter behavior



HDL ワークフローアドバイザー

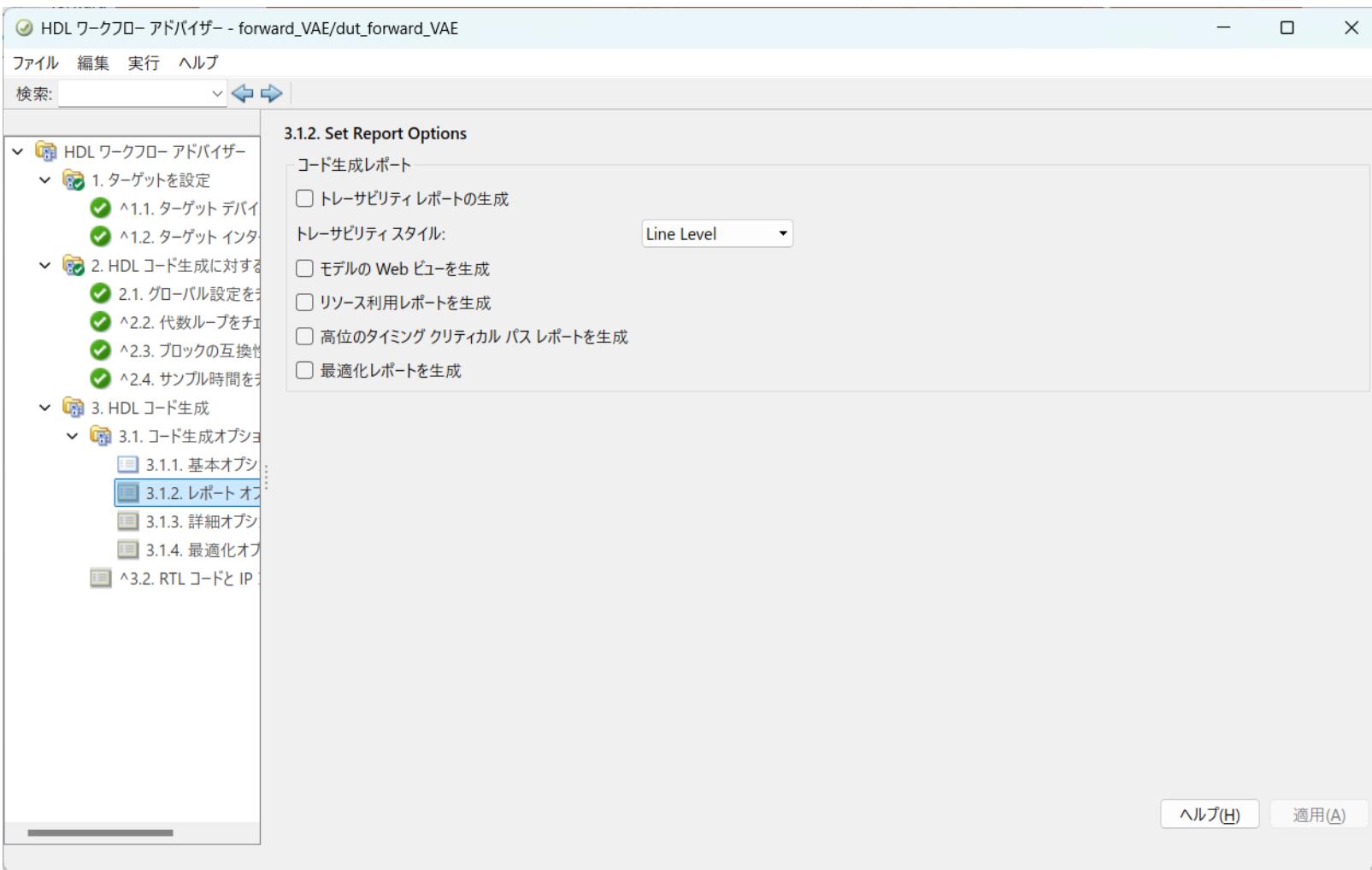
HDLコード生成に対する設定

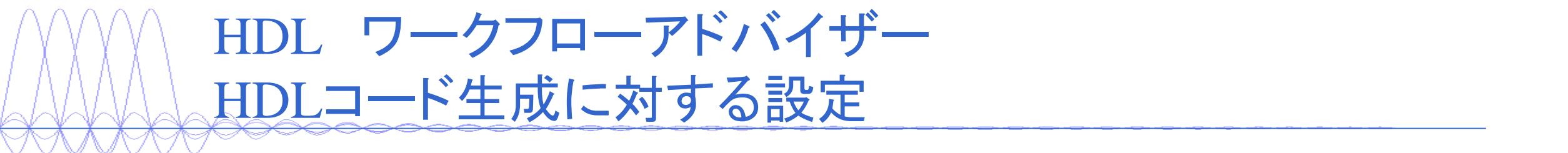




HDL ワークフローアドバイザー

HDLコード生成に対する設定

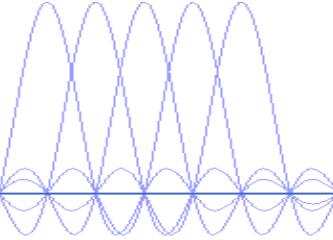




HDL ワークフローアドバイザー

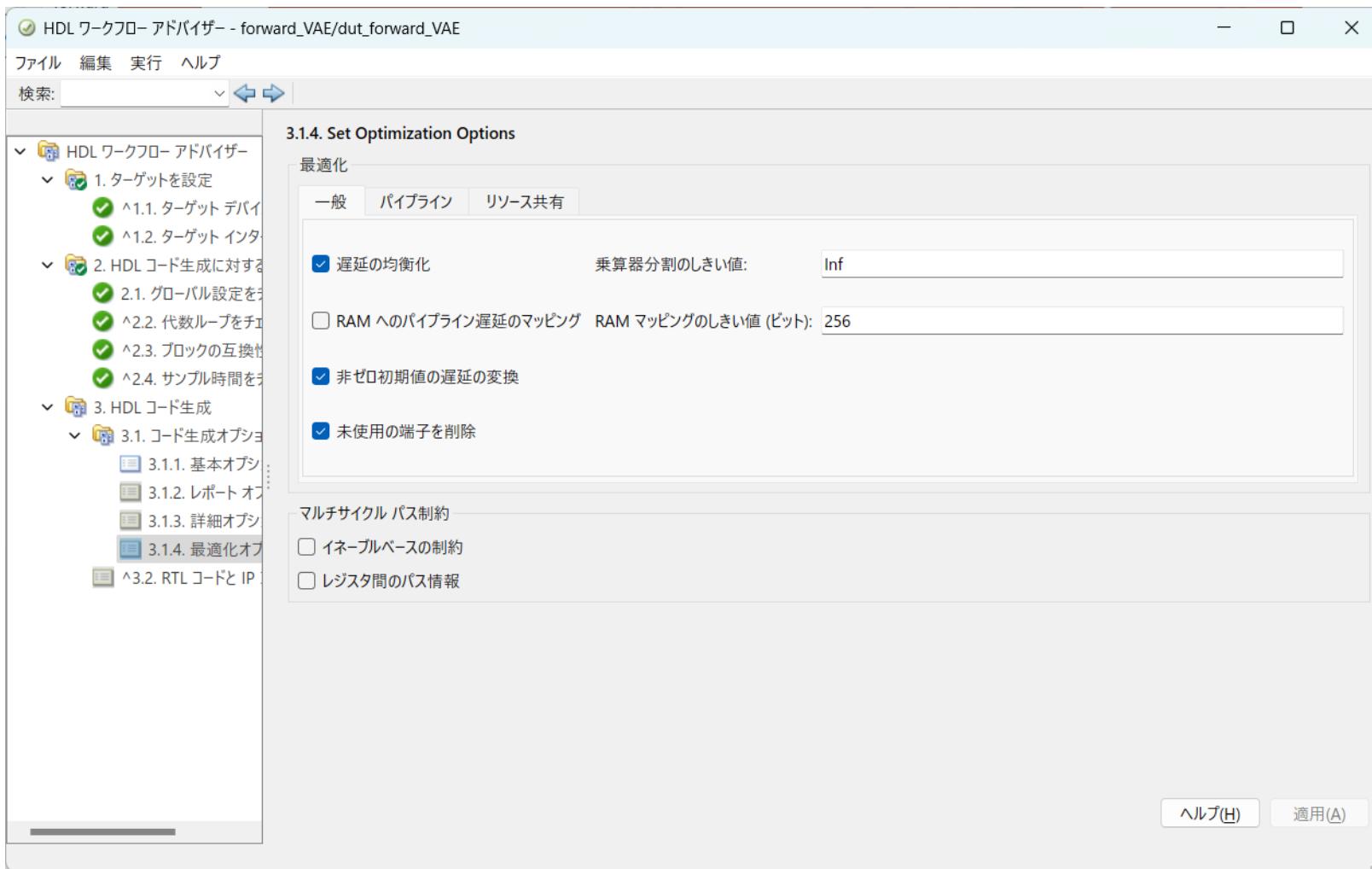
HDLコード生成に対する設定

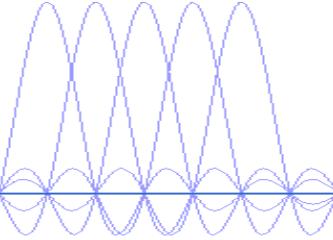




HDL ワークフローアドバイザー

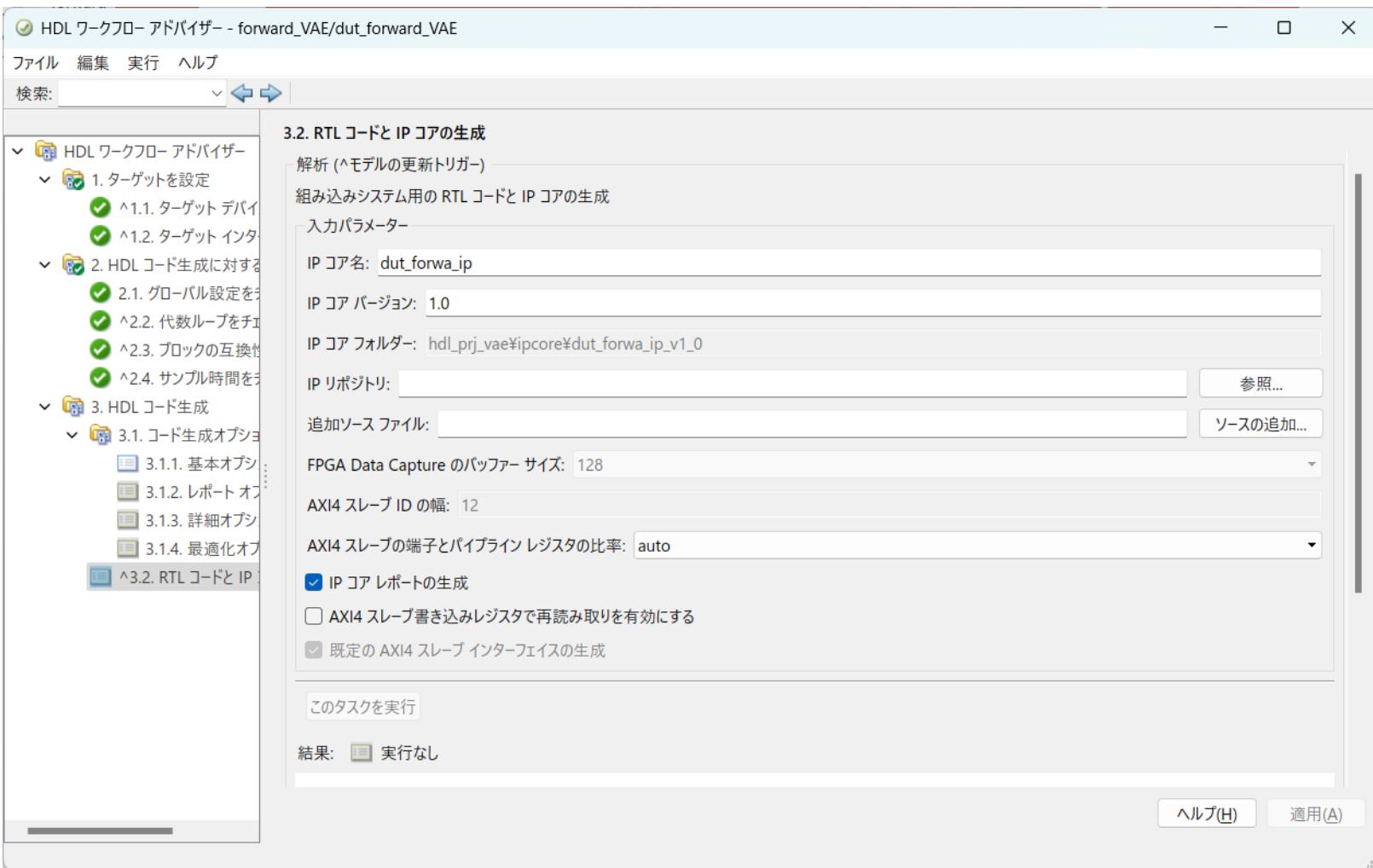
HDLコード生成に対する設定





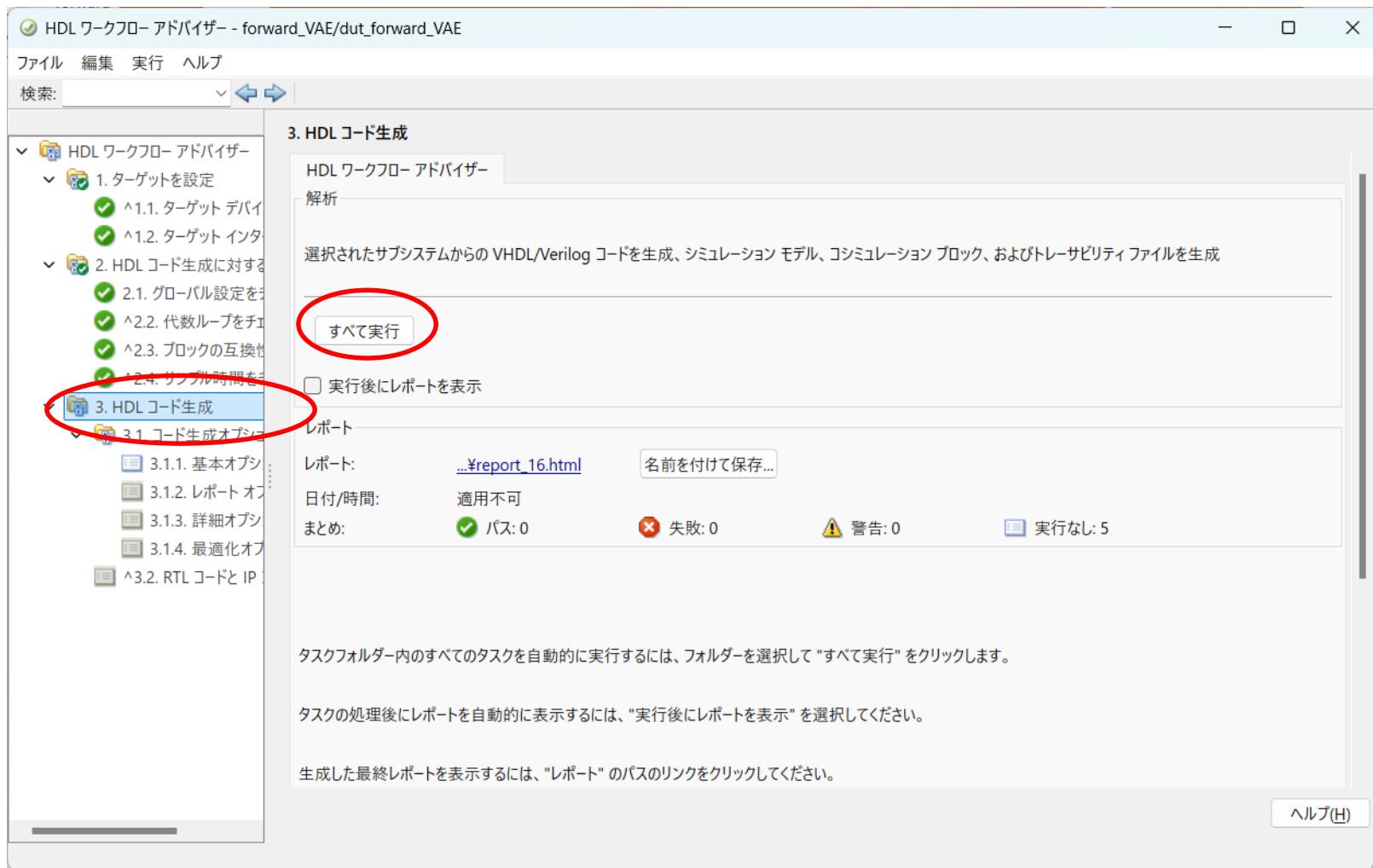
HDL ワークフローアドバイザー

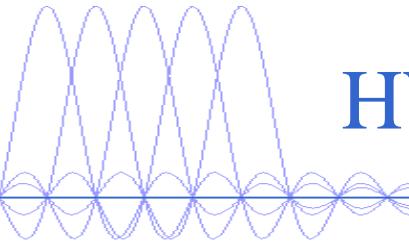
HDLコード生成に対する設定



HDL ワークフローアドバイザー

HDLコード生成に対する設定

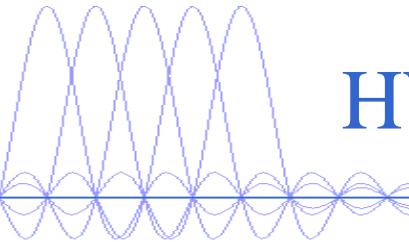




HW Summary



HDLコードの生成が終了すると、概要が表示される



HW Summary

コード生成レポート

検索: 大文字小文字を区別

Contents

- Summary
- Clock Summary
- コードインターフェイスポート
- IP コアの生成レポート

既定以外のブロックプロパティ

ブロック パラメーター: dut_forward_VAE (現在のアーキテクチャ Module)

ProcessorFPGASynchronization	Free running
------------------------------	--------------

ブロック パラメーター: w2_11 (現在のアーキテクチャ default)

IOInterface	AXI4-Lite
IOInterfaceMapping	x"100"

ブロック パラメーター: w2_12 (現在のアーキテクチャ default)

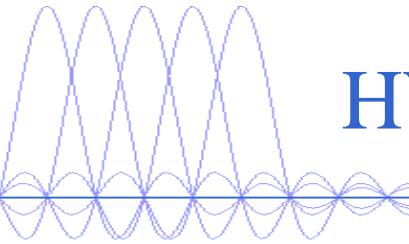
IOInterface	AXI4-Lite
IOInterfaceMapping	x"104"

ブロック パラメーター: w2_13 (現在のアーキテクチャ default)

IOInterface	AXI4-Lite
IOInterfaceMapping	x"108"

ブロック パラメーター: w2_14 (現在のアーキテクチャ default)

OK(○) ヘルプ(H)



HW Summary

コード生成レポート

検索: 大文字小文字を区別

Contents

- Summary
- Clock Summary
- コードインターフェイスレポート
- IP コアの生成レポート

Referenced Models

IOInterfaceMapping

プロックパラメーター: a3_6 (現在のアーキテクチャ default)

IOInterface	AXI4-Lite
IOInterfaceMapping	x"228"

プロックパラメーター: a3_7 (現在のアーキテクチャ default)

IOInterface	AXI4-Lite
IOInterfaceMapping	x"22C"

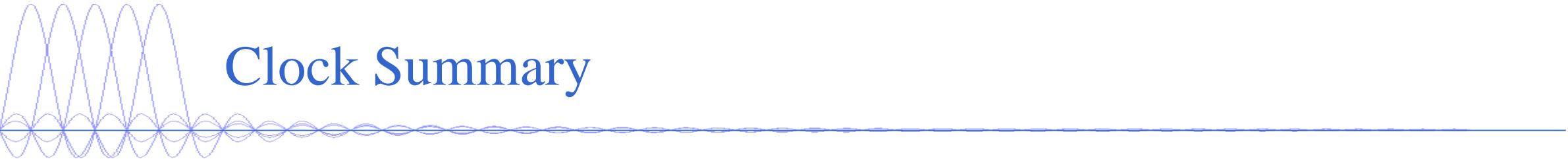
プロックパラメーター: a3_8 (現在のアーキテクチャ default)

IOInterface	AXI4-Lite
IOInterfaceMapping	x"230"

プロックパラメーター: a3_9 (現在のアーキテクチャ default)

IOInterface	AXI4-Lite
IOInterfaceMapping	x"234"

OK(○) ヘルプ(H)



Clock Summary

コード生成レポート

検索: 大文字小文字を区別

Contents

- Summary
- Clock Summary**
- コードインターフェイスレポート
- IP コアの生成レポート

Referenced Models

Clock Report for forward_VAE

Rate Information

Model Base Rate	1
DUT Base Rate	1

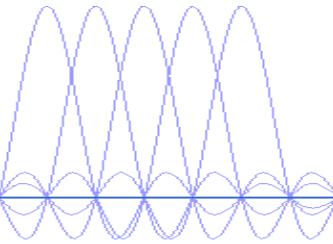
Clock Enable Table

Clock Enable Name	Sample Time
ce_out	1

Output Signal Table

Output Signal	Clock Enable Name	Sample Time	Latency
z2_1	ce_out	1	3
z2_2	ce_out	1	3
a2_1	ce_out	1	3
a2_2	ce_out	1	3
z3_1	ce_out	1	3
z3_2	ce_out	1	3
z3_3	ce_out	1	3
z3_4	ce_out	1	3
z3_5	ce_out	1	3

OK(O) ヘルプ(H)



インターフェイスレポート

コード生成レポート

検索: 大文字小文字を区別

Contents

- Summary
- Clock Summary
- コードインターフェイスレポート
- IP コアの生成レポート

Referenced Models

forward_VAE のコードインターフェイス レポート

入力端子

ポート名	データ型	ビット
clk	boolean	1
reset	boolean	1
clk_enable	boolean	1
w2_11	sfix24_En16	24
w2_12	sfix24_En16	24
w2_13	sfix24_En16	24
w2_14	sfix24_En16	24
w2_15	sfix24_En16	24
w2_16	sfix24_En16	24
w2_17	sfix24_En16	24
w2_18	sfix24_En16	24
w2_19	sfix24_En16	24
w2_21	sfix24_En16	24
w2_22	sfix24_En16	24
w2_23	sfix24_En16	24
w2_24	sfix24_En16	24
w2_25	sfix24_En16	24
w2_26	sfix24_En16	24
w2_27	sfix24_En16	24
w2_28	sfix24_En16	24
...		

OK(O) ヘルプ(H)

コード生成レポート

検索: 大文字小文字を区別

Contents

Summary

Clock Summary

コードインターフェイスレポート

IP コアの生成レポート

Referenced Models

IP Core Generation Report for forward_VAE

Summary

IP core name	dut_forwa_ip
IP core version	1.0
IP core folder	hdl_prj_vae¥ipcore¥dut_forwa_ip_v1_0
IP core zip file name	dut_forwa_ip_v1_0.zip
Target platform	Generic Xilinx Platform
Target tool	Xilinx Vivado
Target language	Verilog
Model	forward_VAE
Model version	1.1
HDL Coder version	3.17
IP core generated on	15-Dec-2024 13:06:05
IP core generated for	dut_forward_VAE

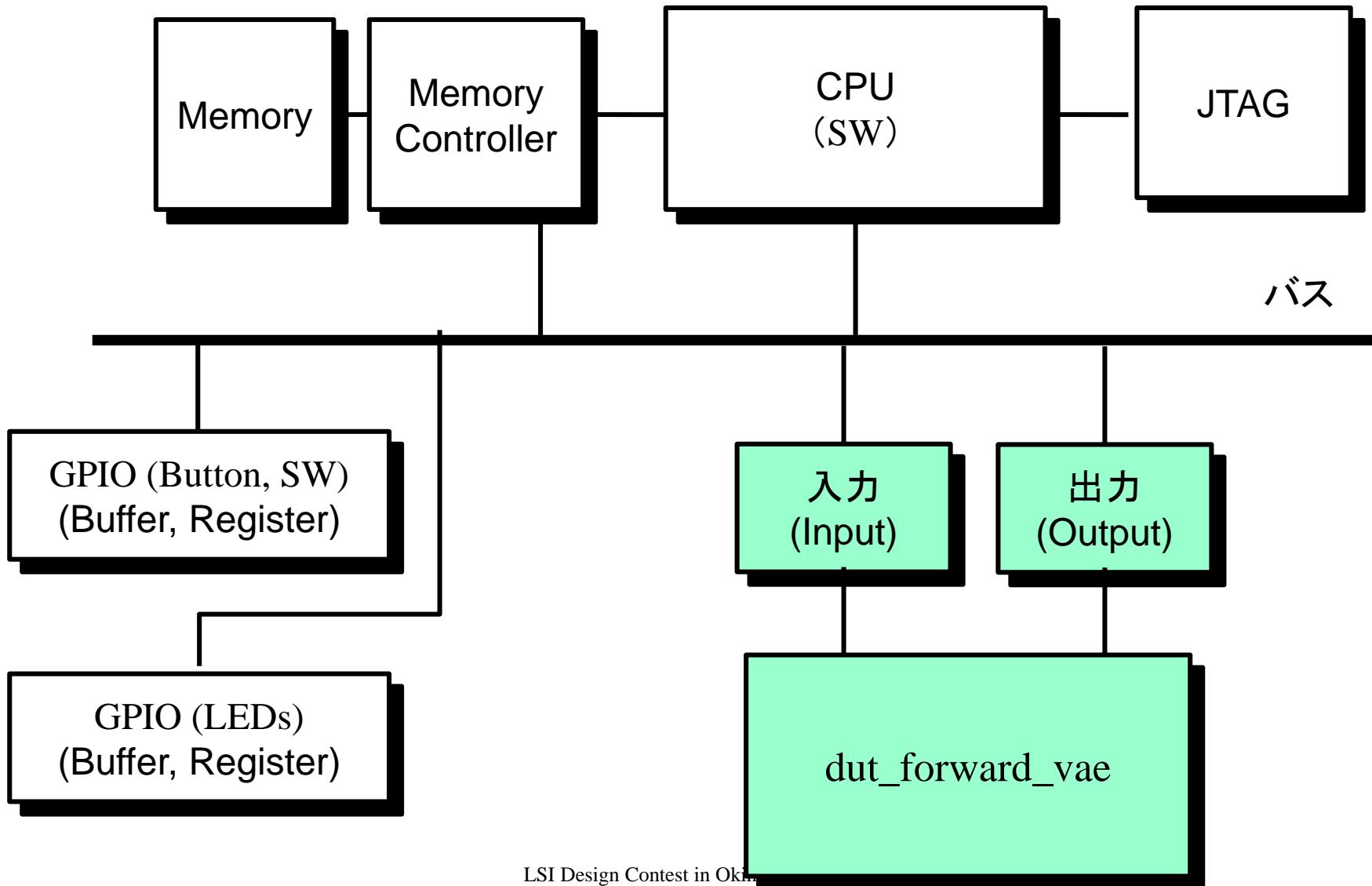
Target Interface Configuration

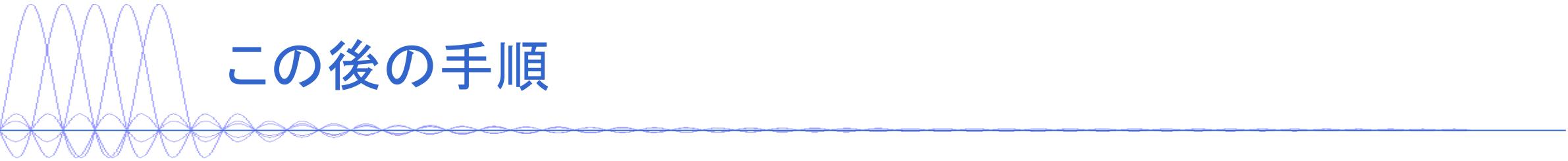
You chose the following target interface configuration for [forward_VAE](#) :

Processor/FPGA synchronization mode: **Free running**

OK(OK) ヘルプ(H)

dut_forward_vae回路の実装とFPGA上での検証

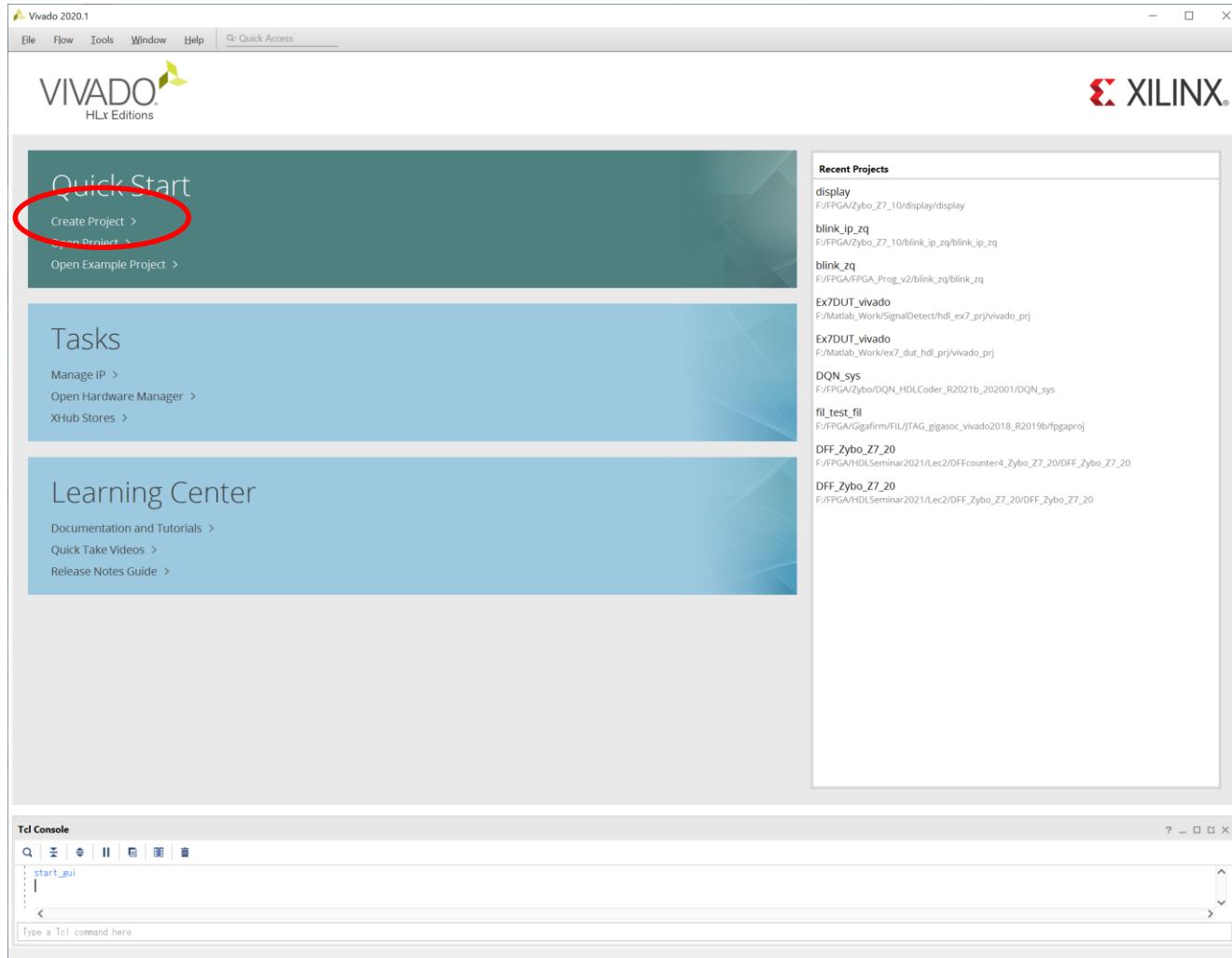


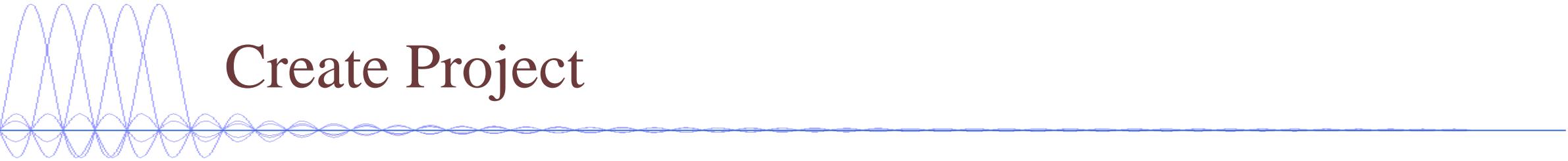


この後の手順

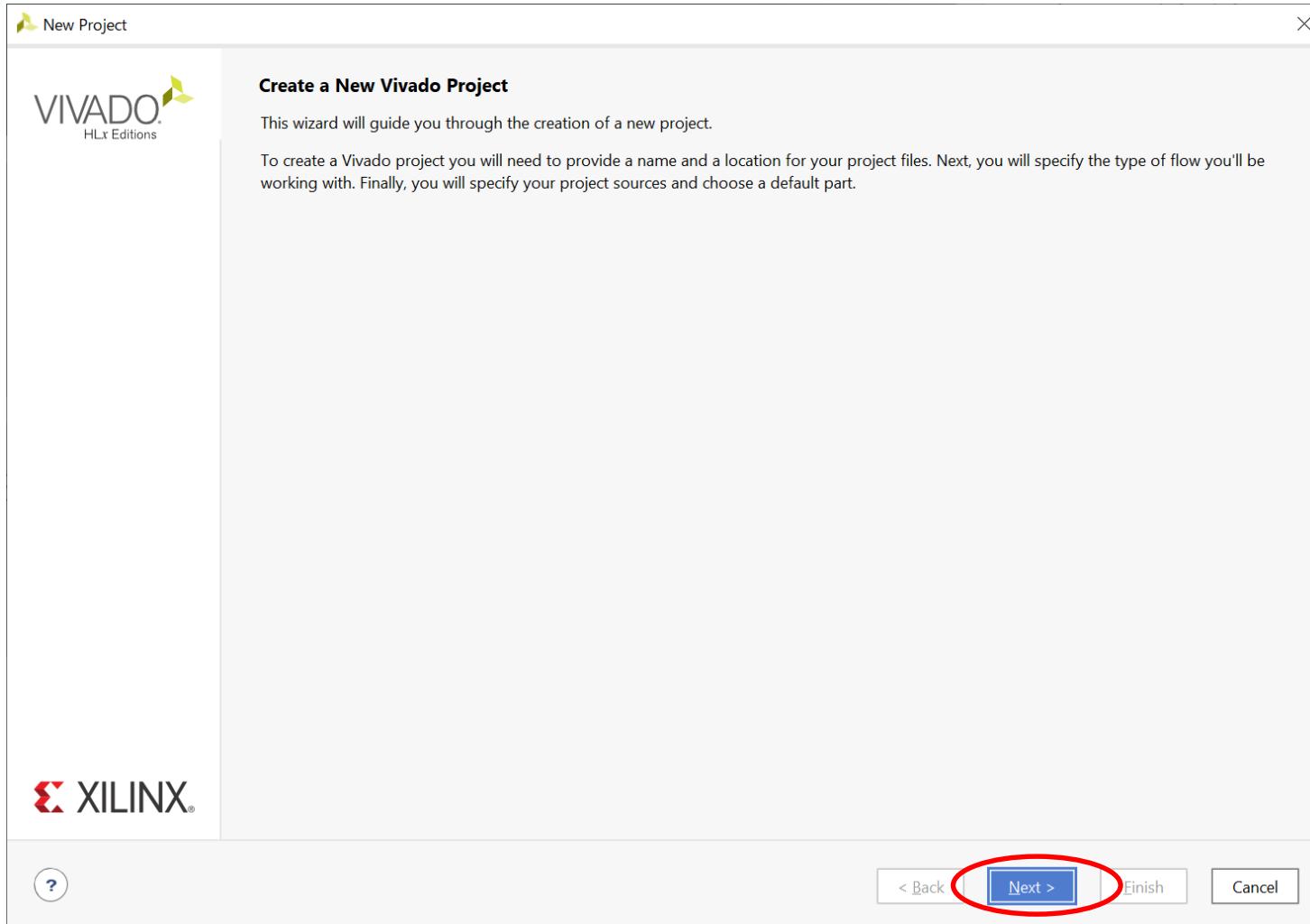
- HDLCoderによる、HW作成が終了
- CPU System(common_sys)と同じシステムを作成
- IPとしてHWが生成されているため、CPU System(common_sys)にVAEのIPを結合
 - IPを読み込み
 - IPを結合
 - wrapperを生成
 - 合成
 - ソフトウェア作成

Create Project

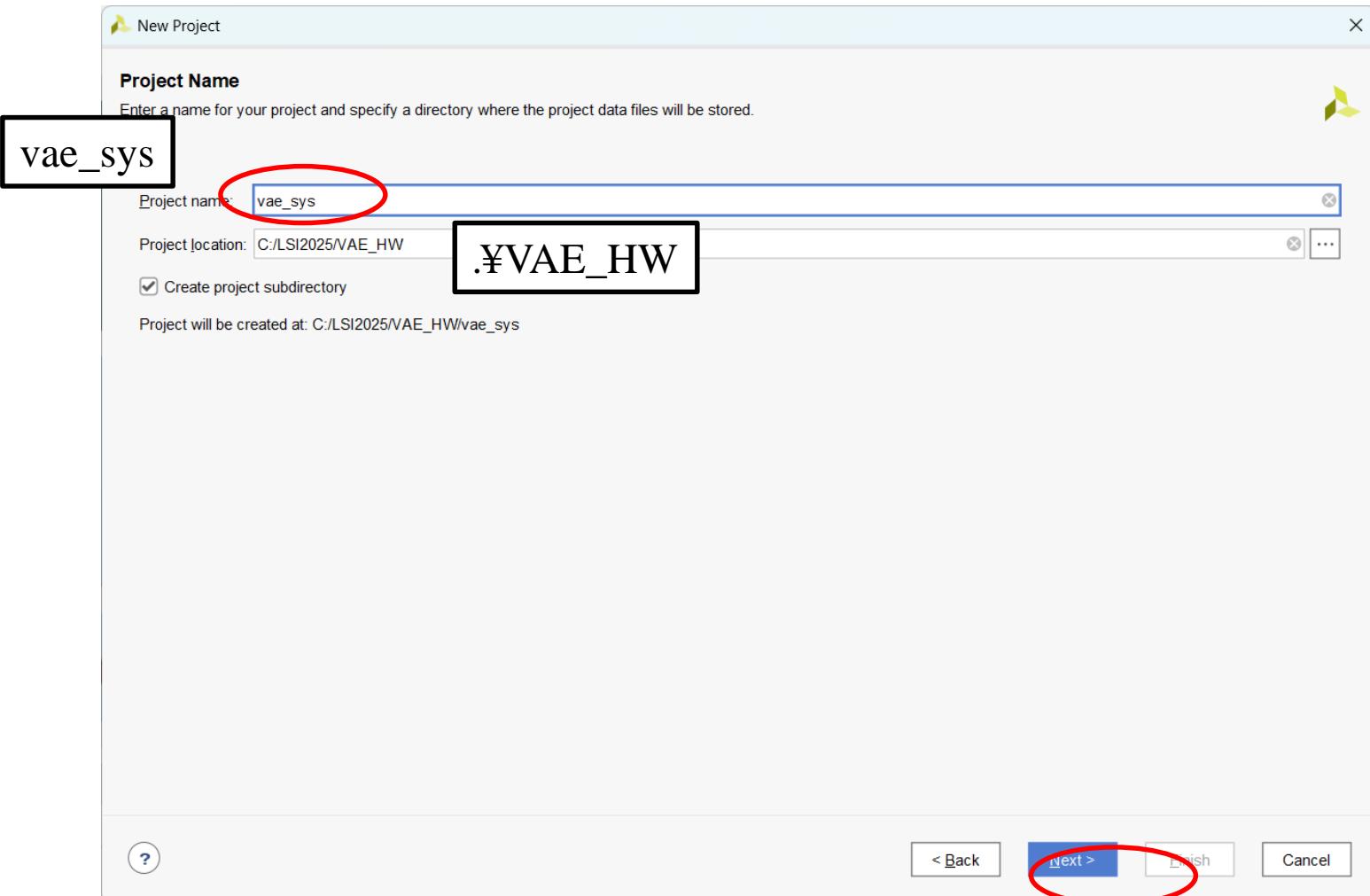




Create Project



Create Project



Create Project

New Project

Project Type
Specify the type of project to create.

RTL Project
You will be able to add sources, create block designs in IP Integrator, generate IP, run RTL analysis, synthesis, implementation, design planning and analysis.
 Do not specify sources at this time

Post-synthesis Project
You will be able to add sources, view device resources, run design analysis, planning and implementation.
 Do not specify sources at this time

I/O Planning Project
Do not specify design sources. You will be able to view part/package resources.

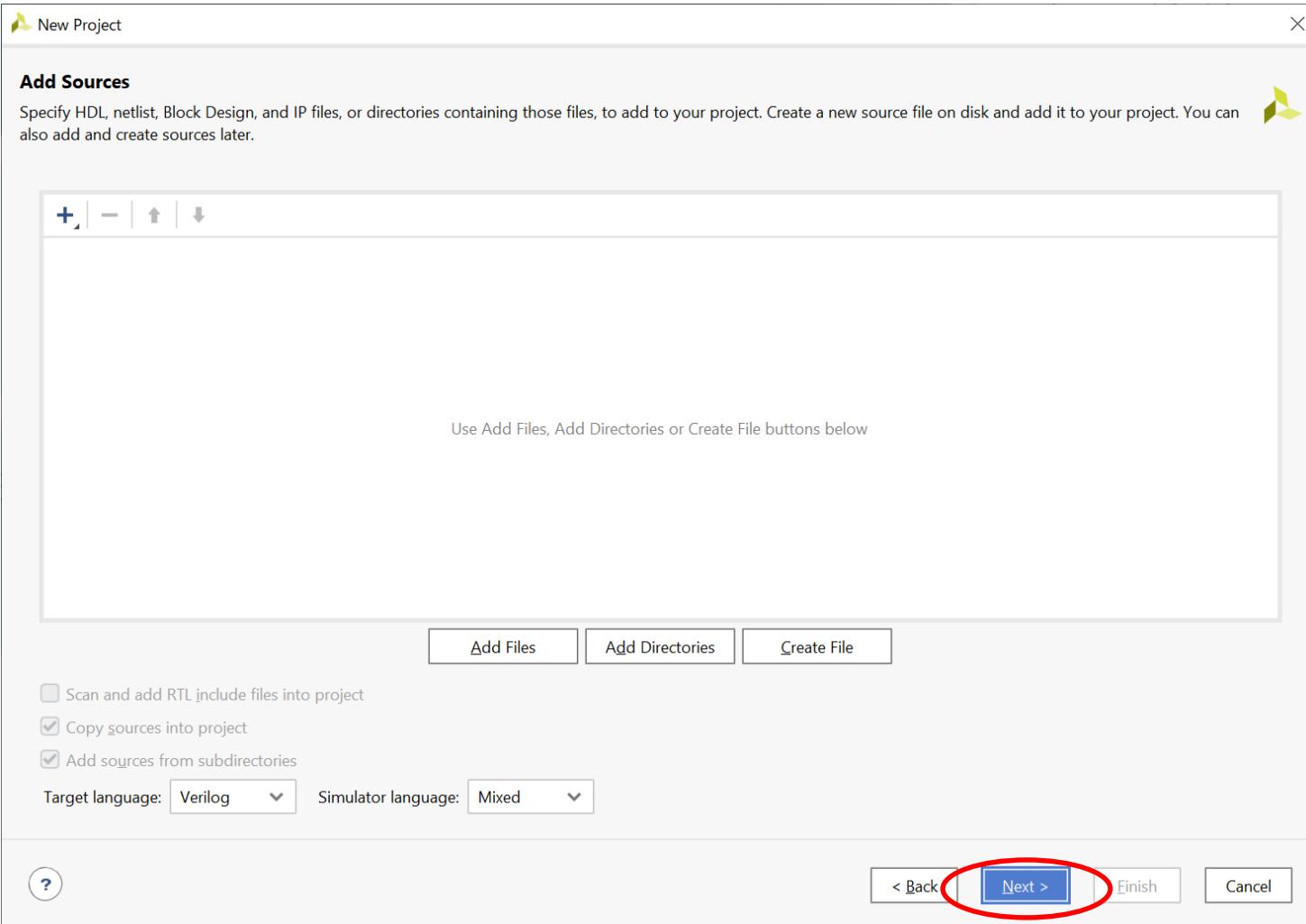
Imported Project
Create a Vivado project from a Synplify, XST or ISE Project File.

Example Project
Create a new Vivado project from a predefined template.

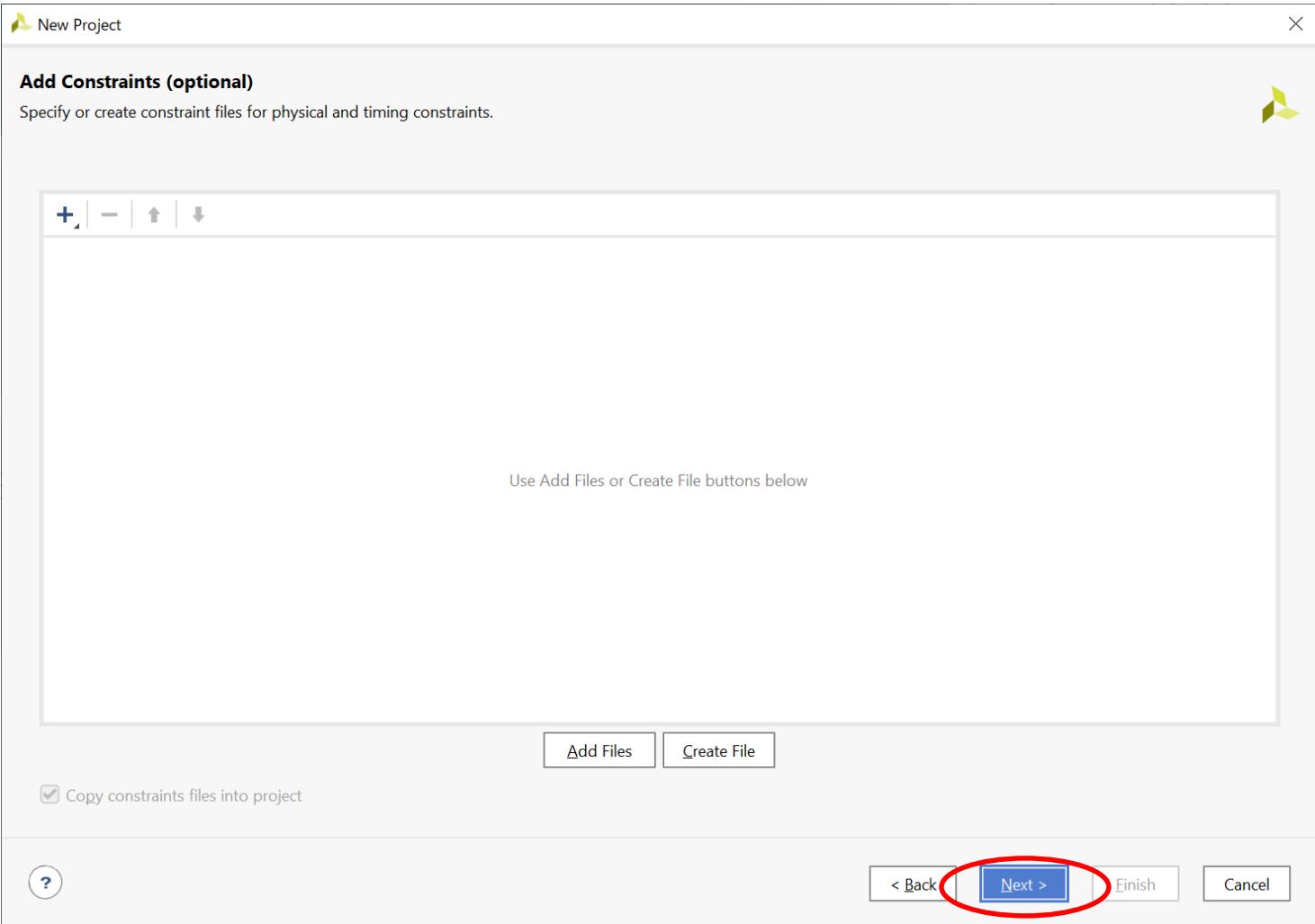
[?](#)

< Back Next > Finish Cancel

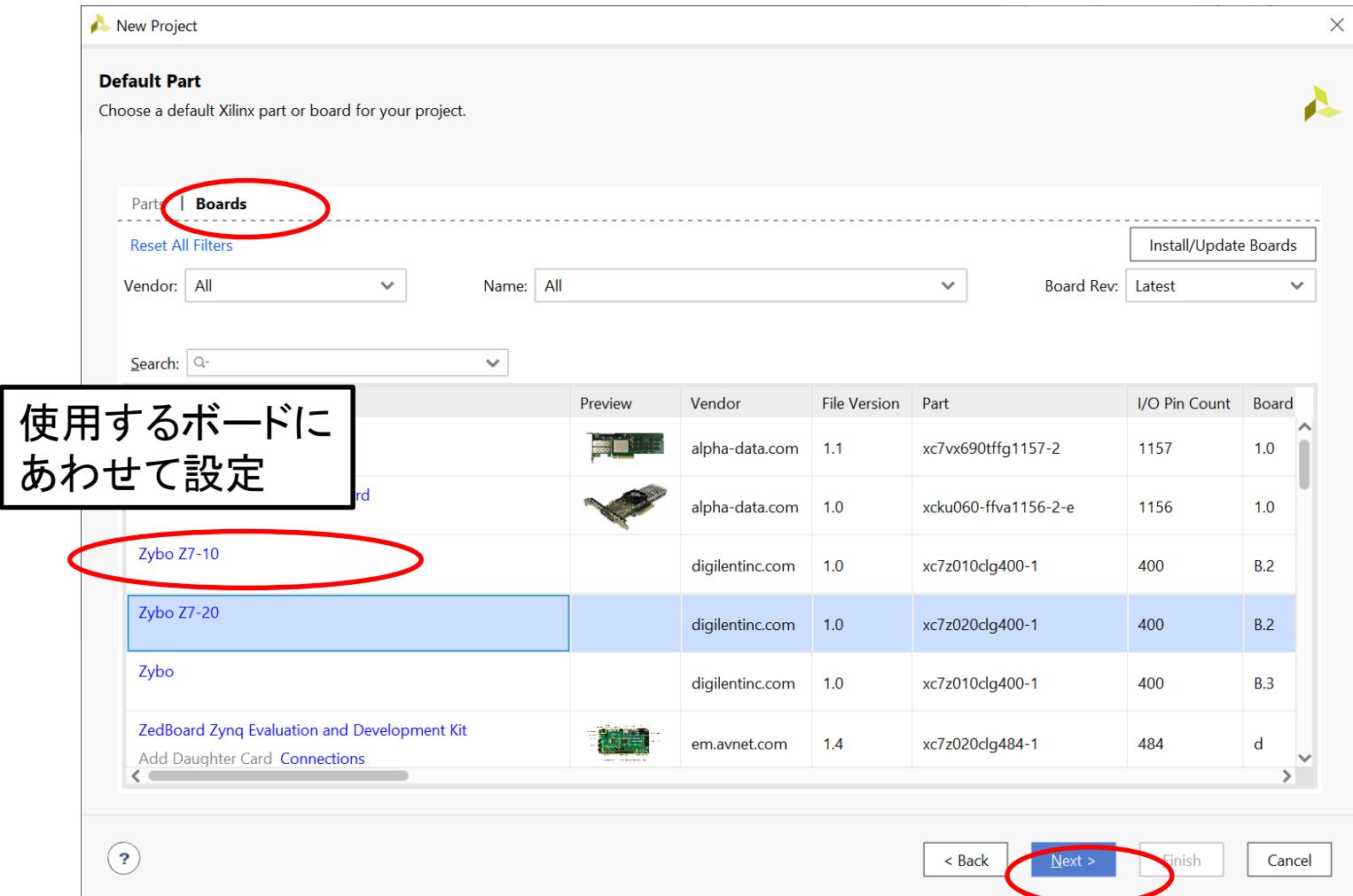
Create Project

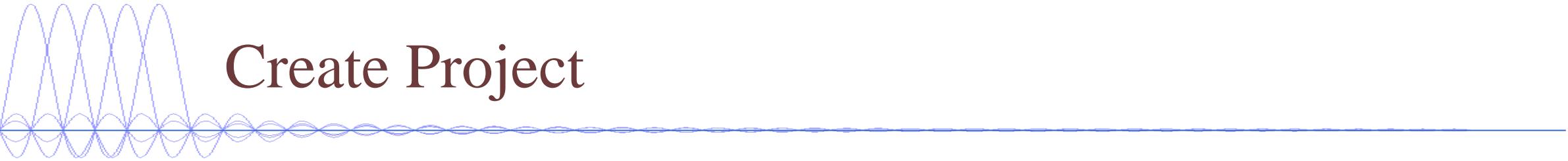


Create Project

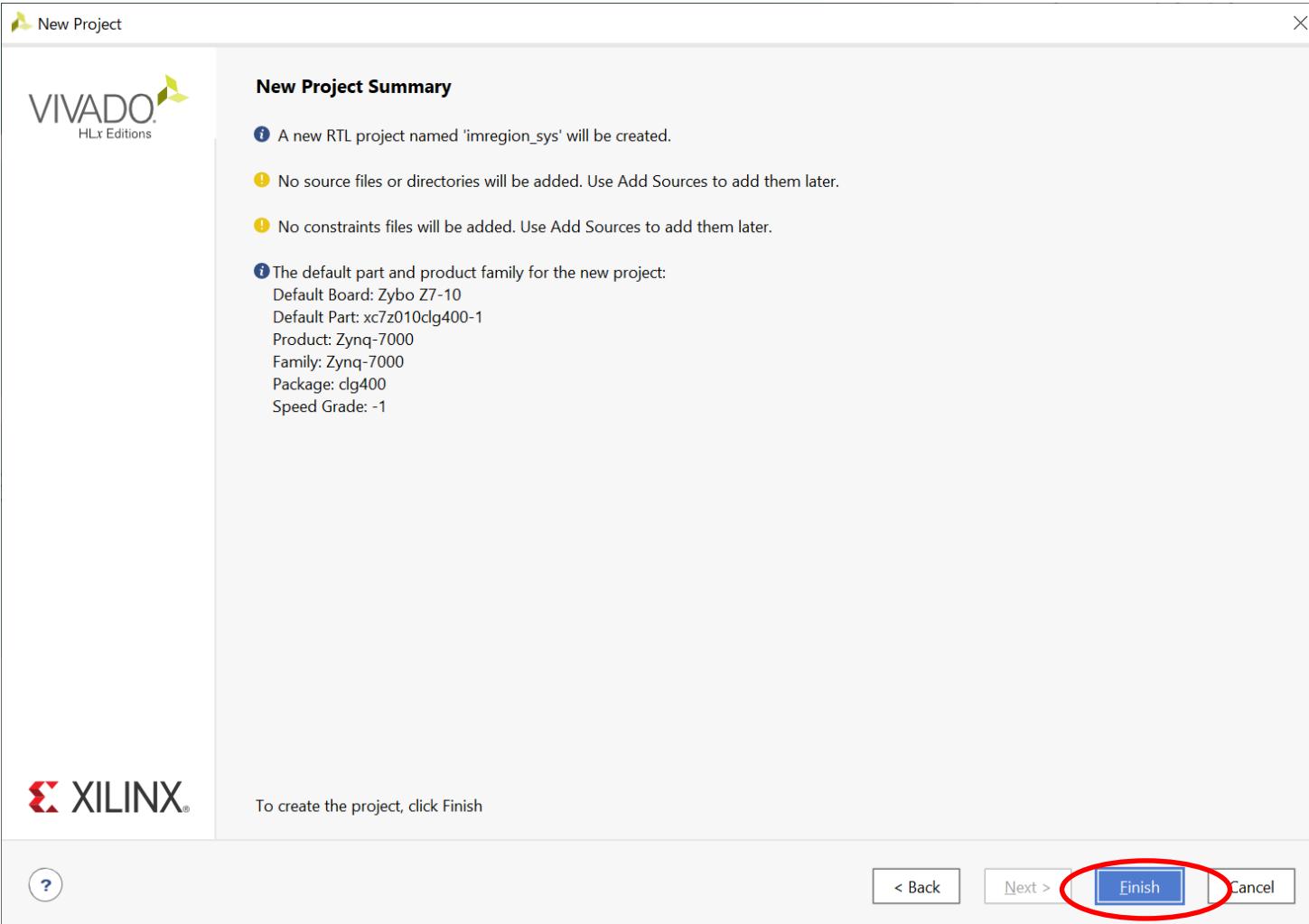


Create Project

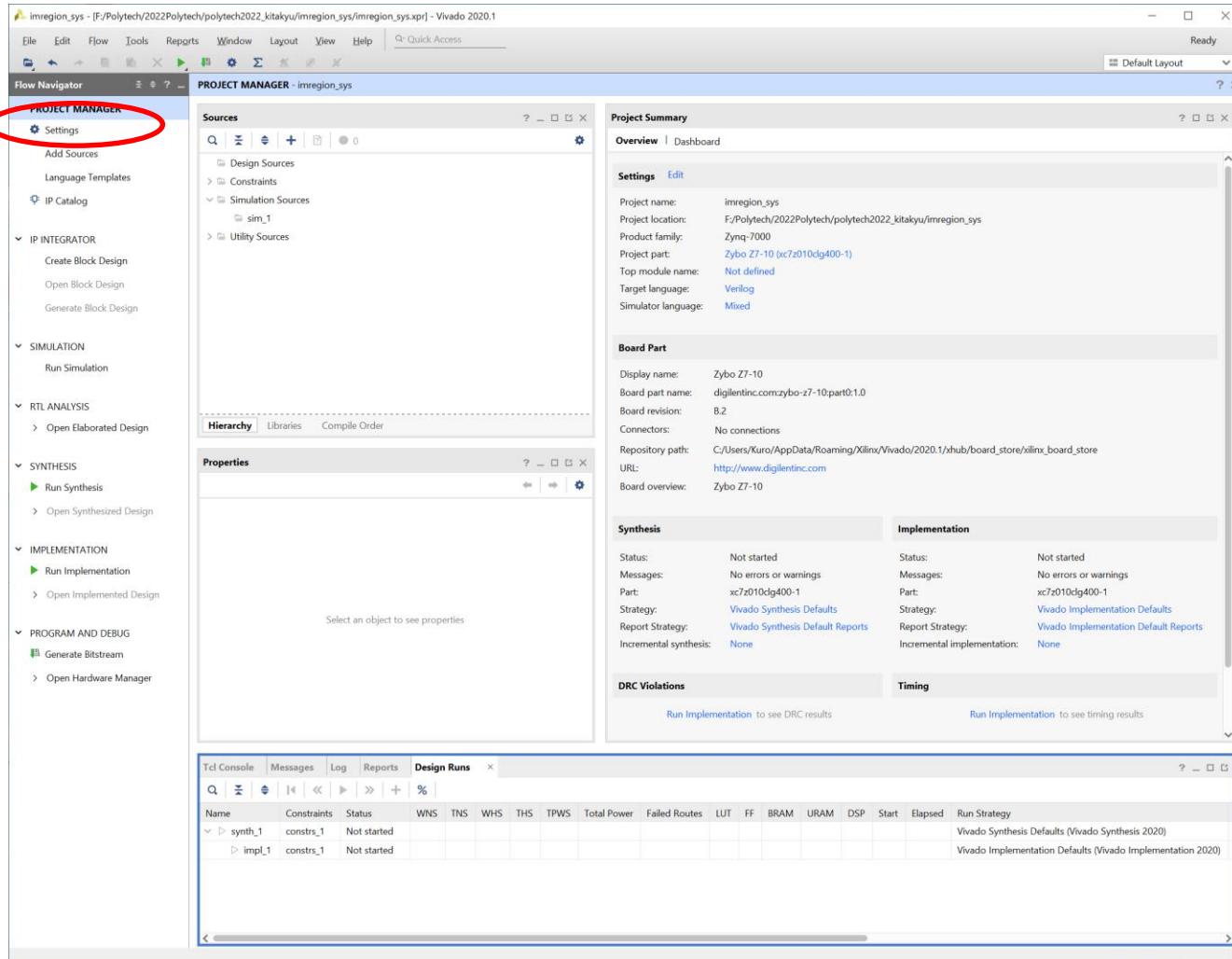




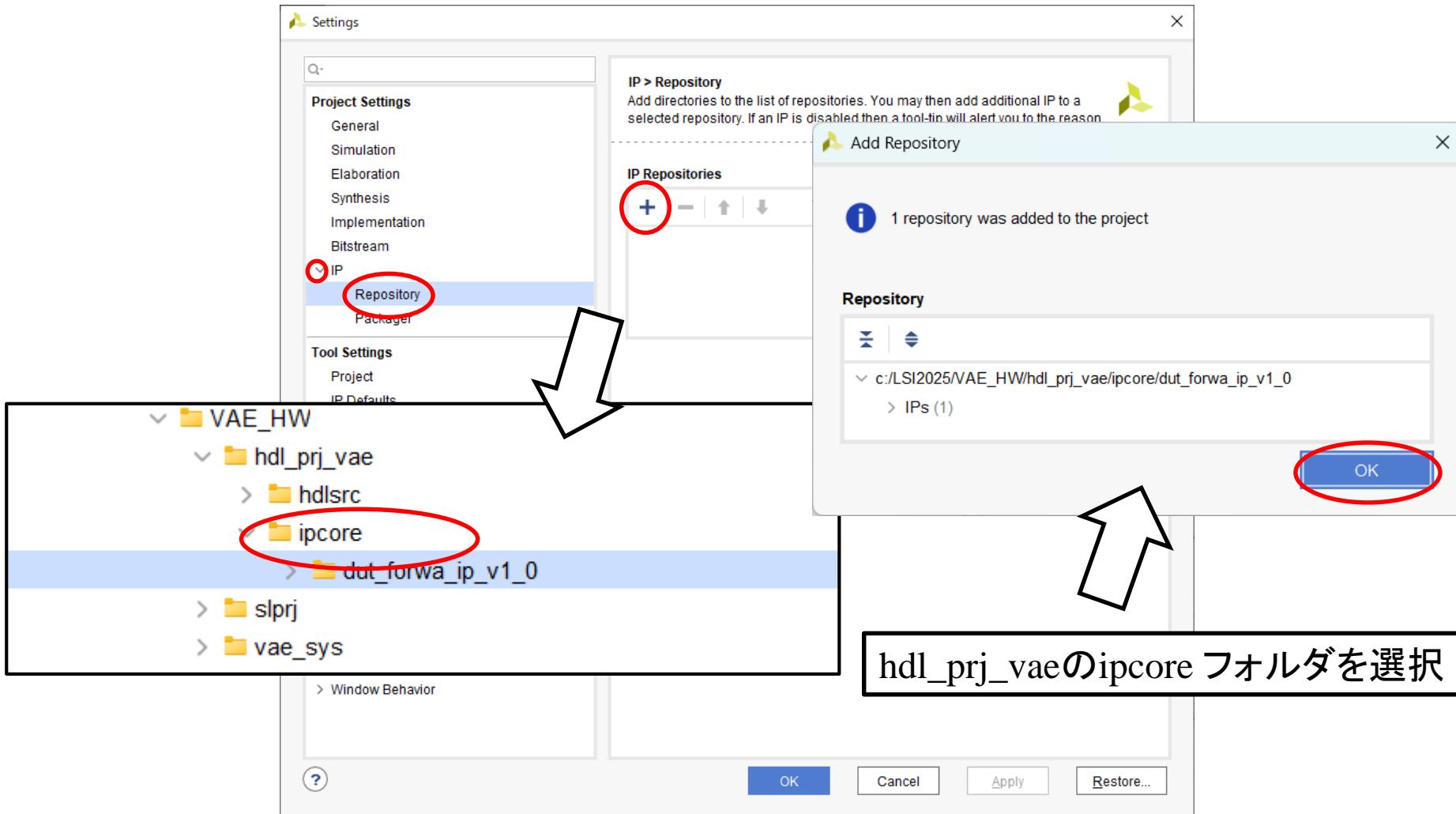
Create Project



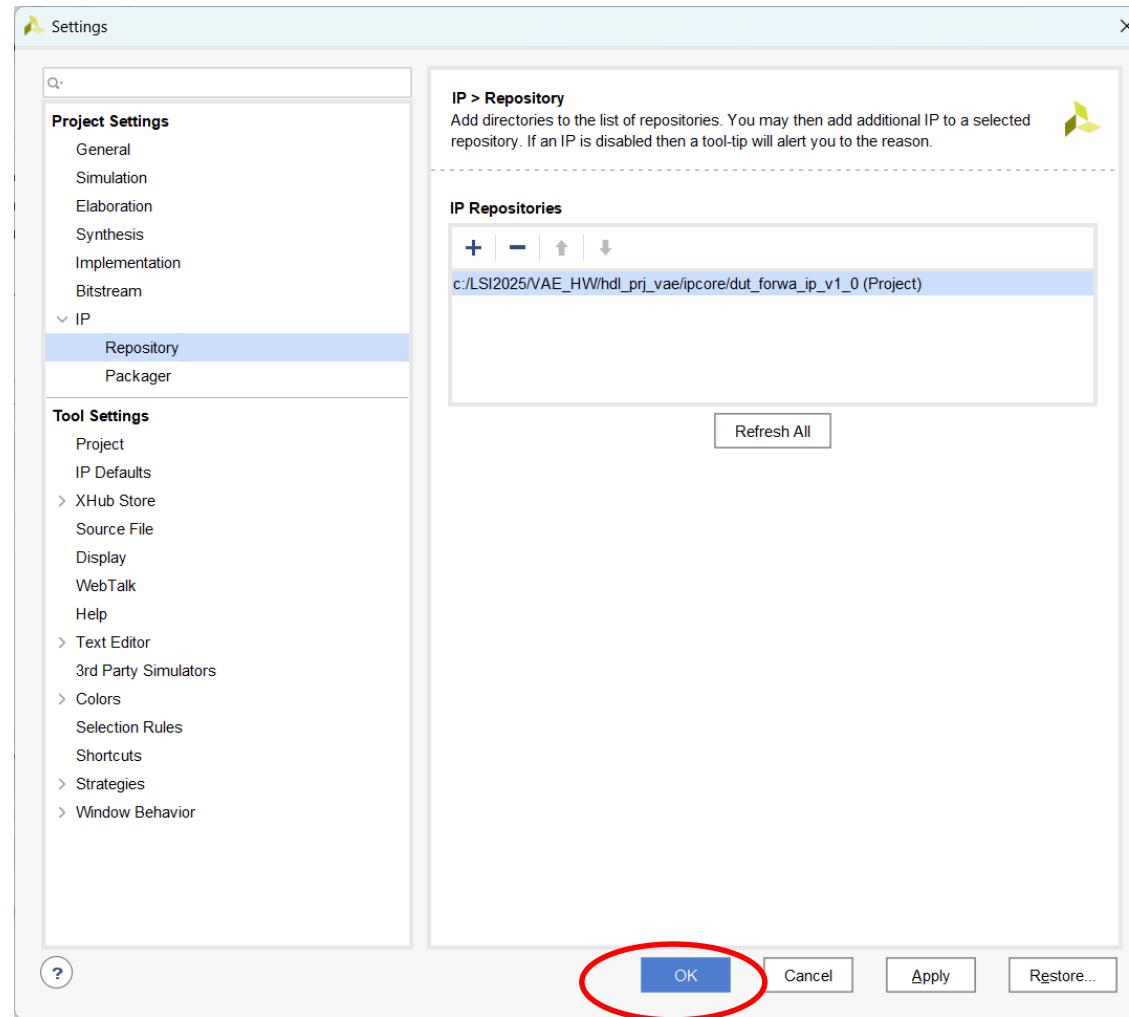
Add IP



Add IP

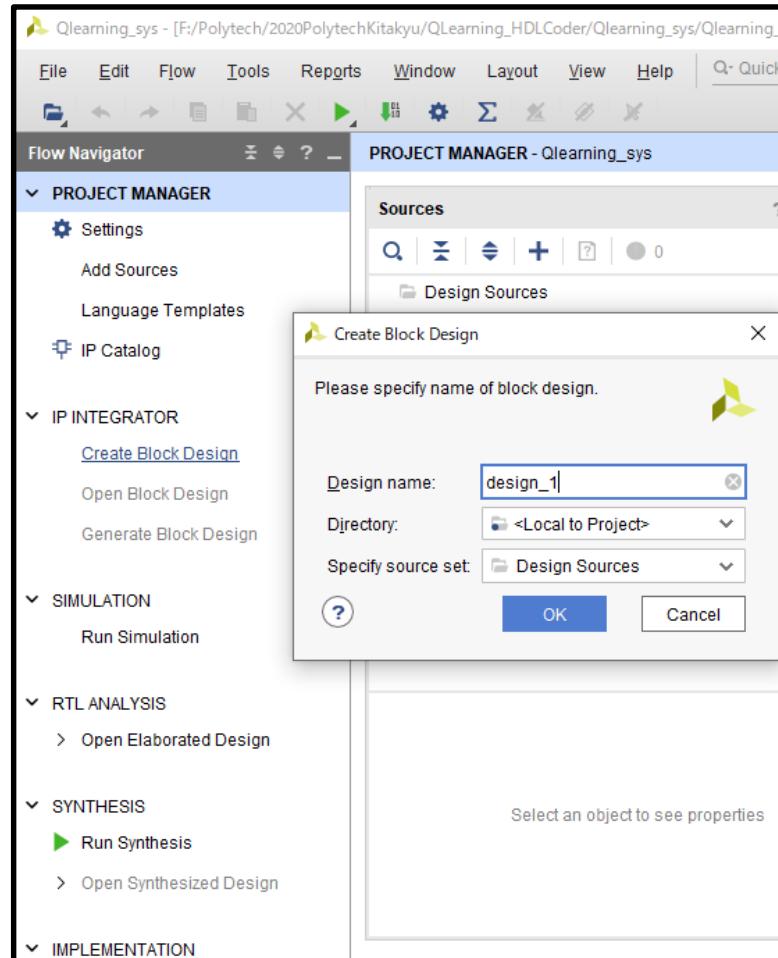


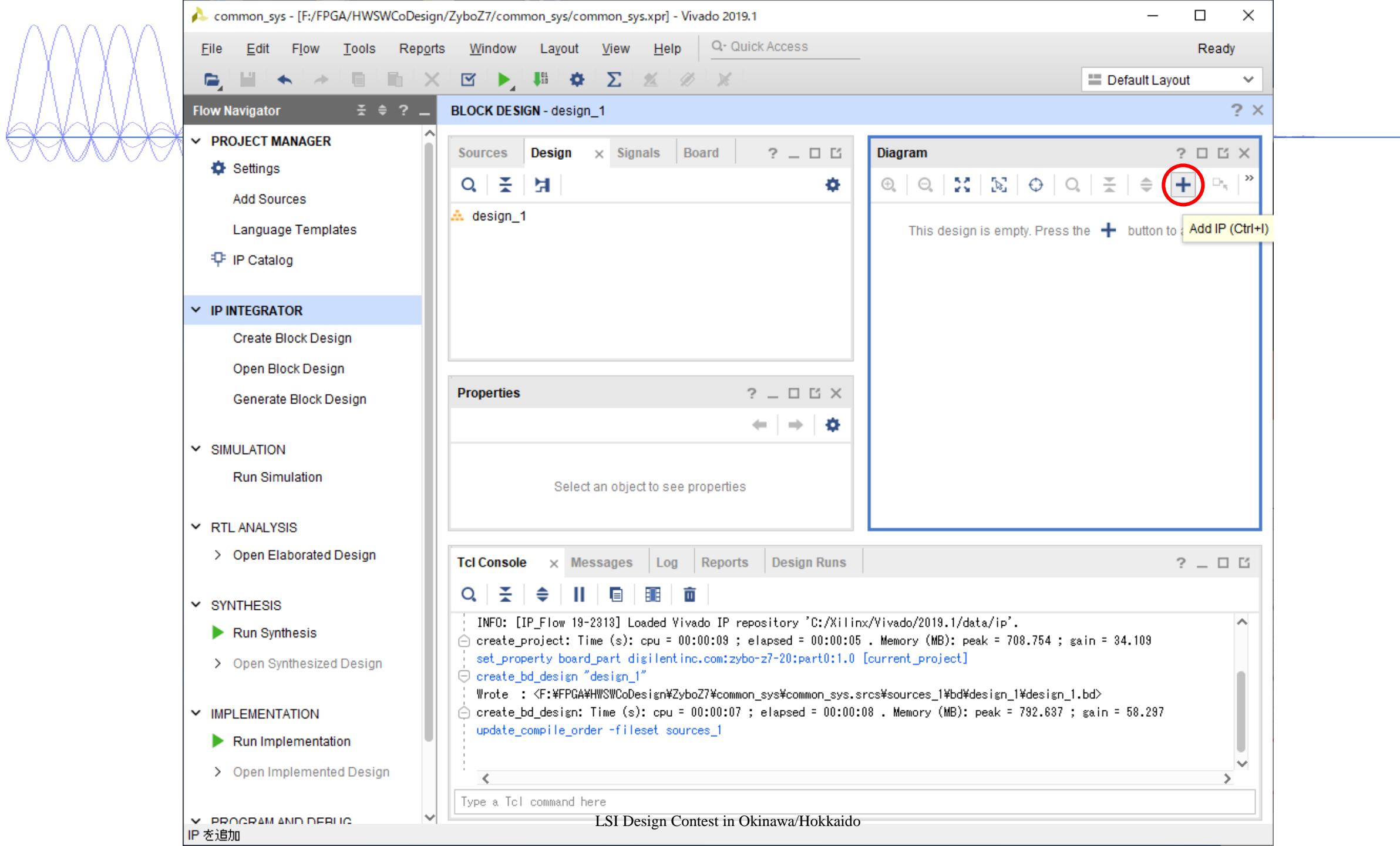
Add IP



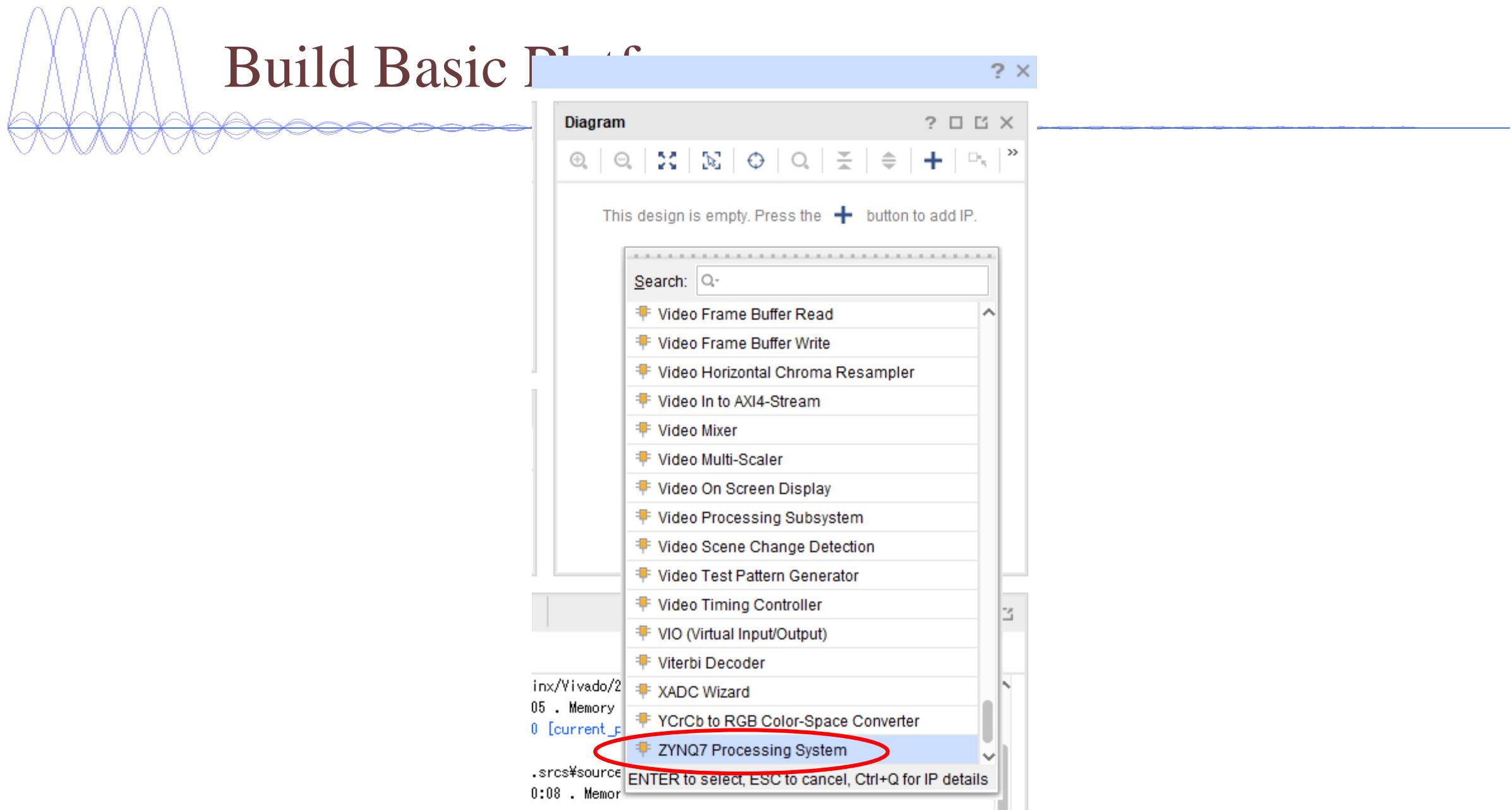


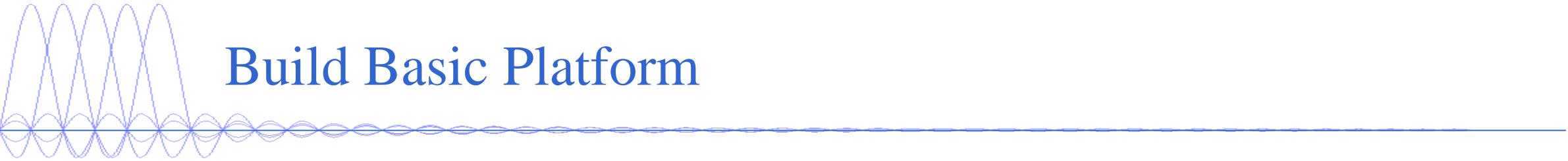
Build Basic Platform



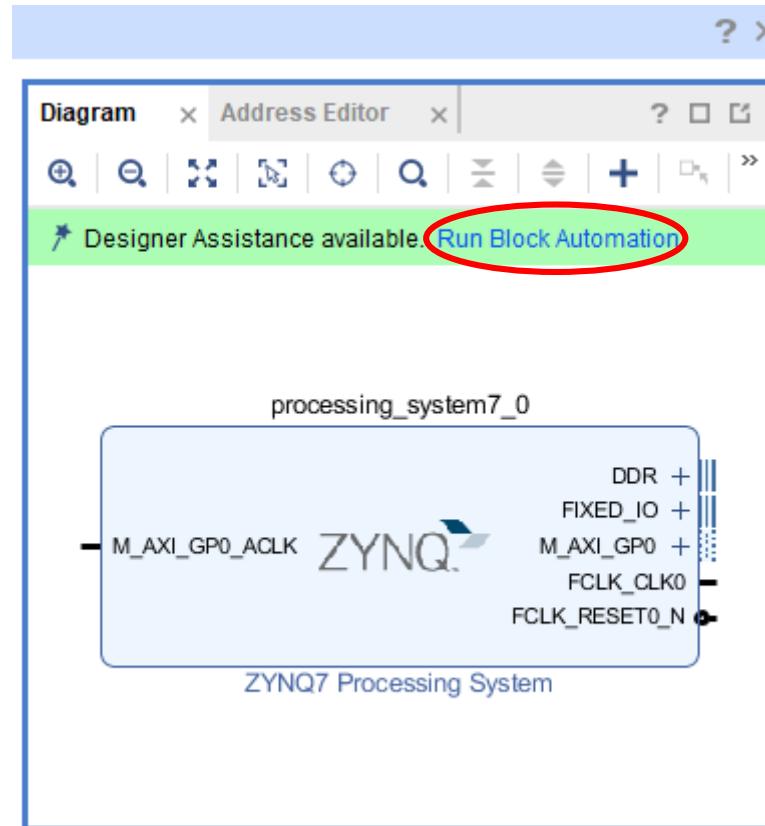


Build Basic Project

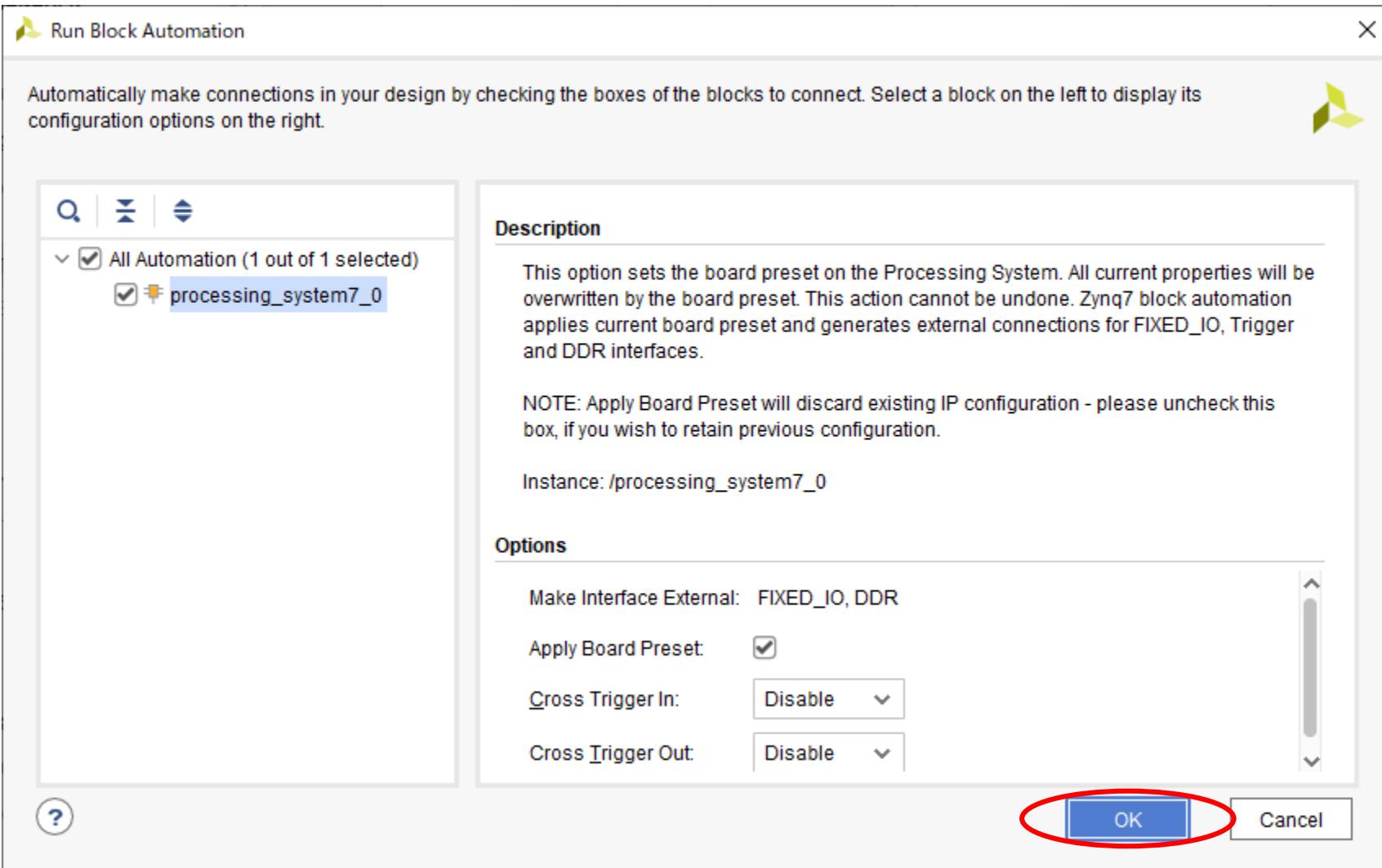


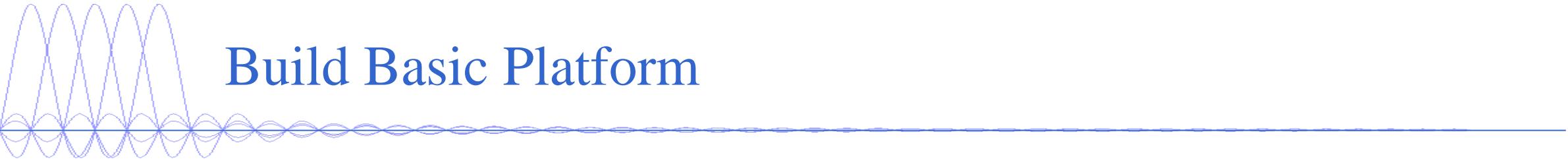


Build Basic Platform

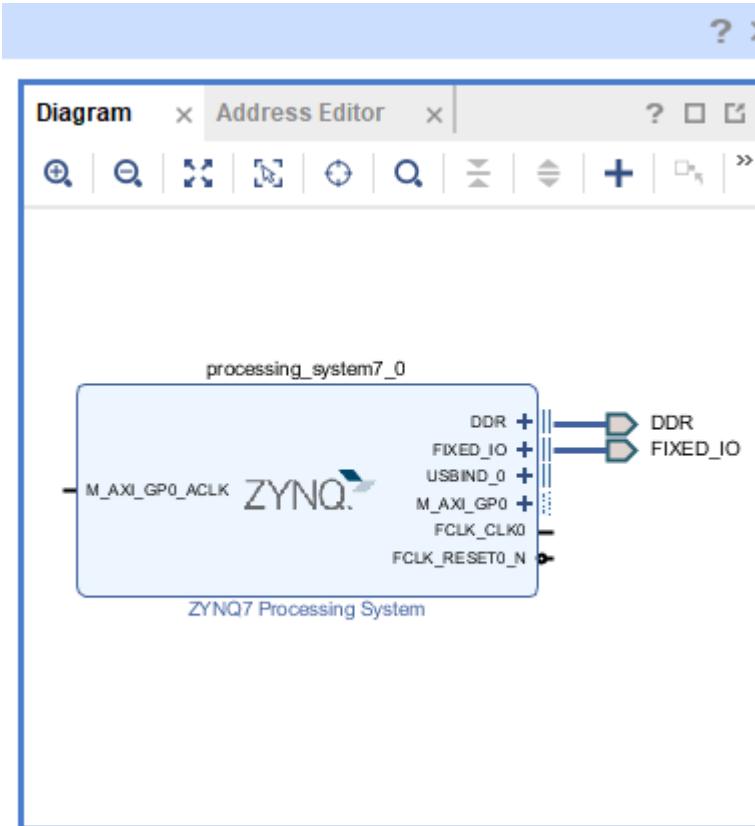


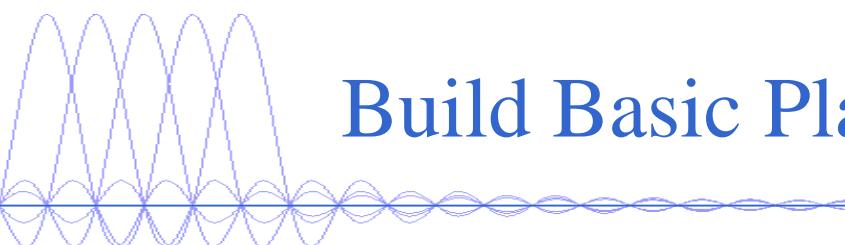
Build Basic Platform



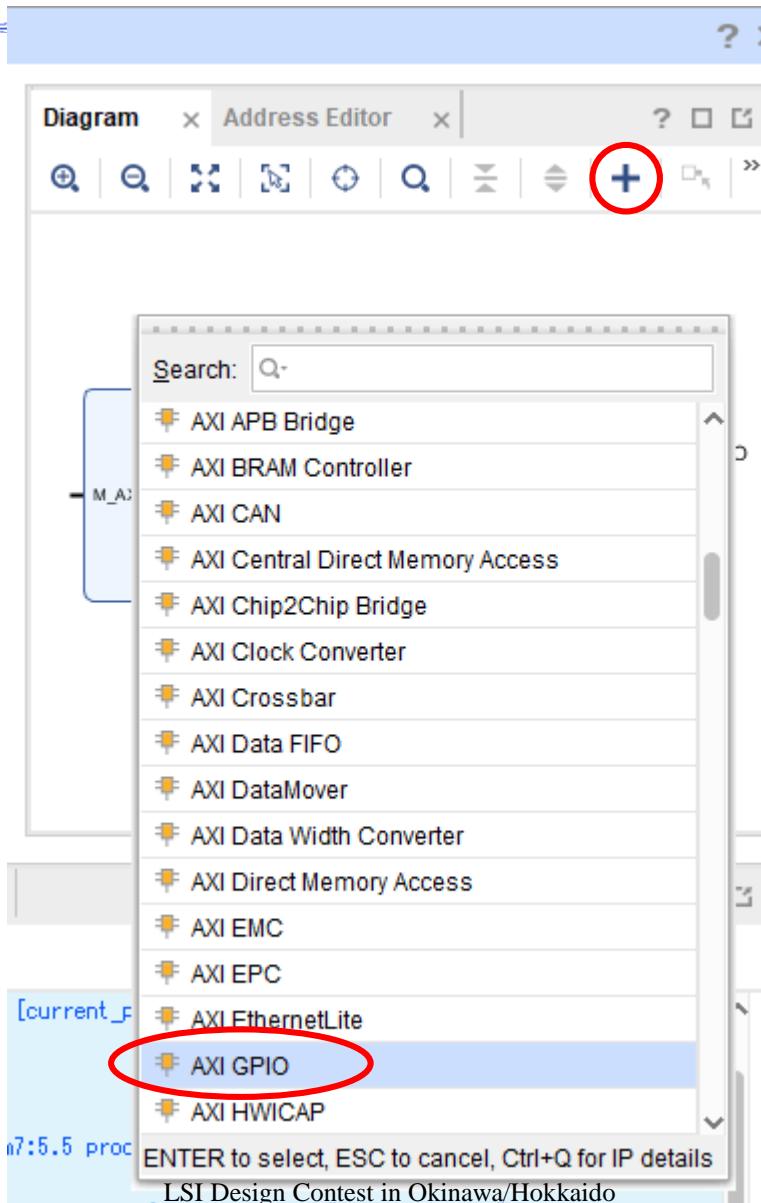


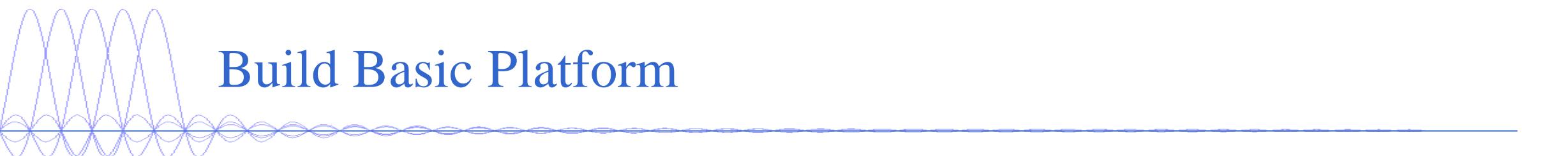
Build Basic Platform



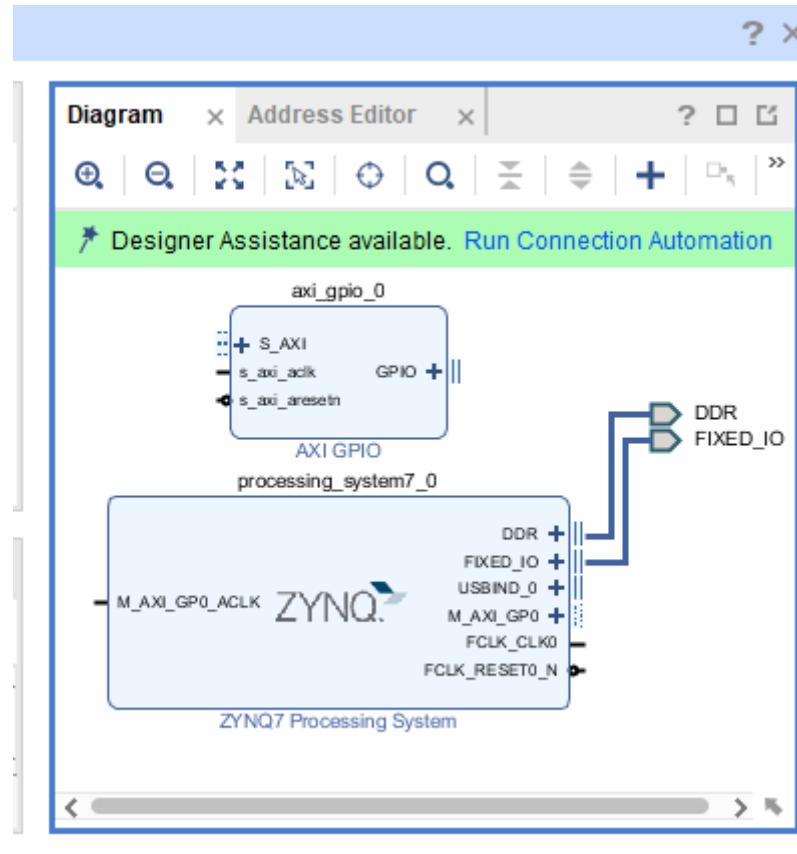


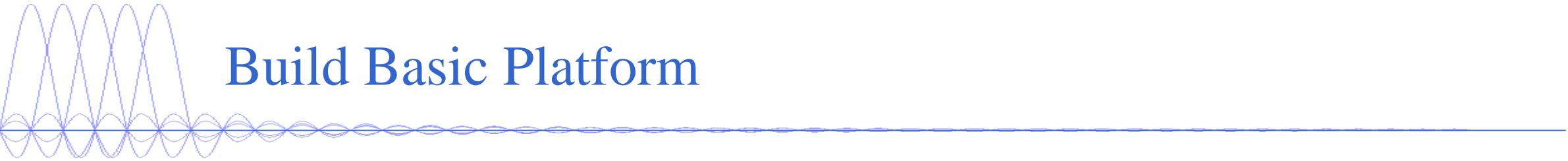
Build Basic Platform



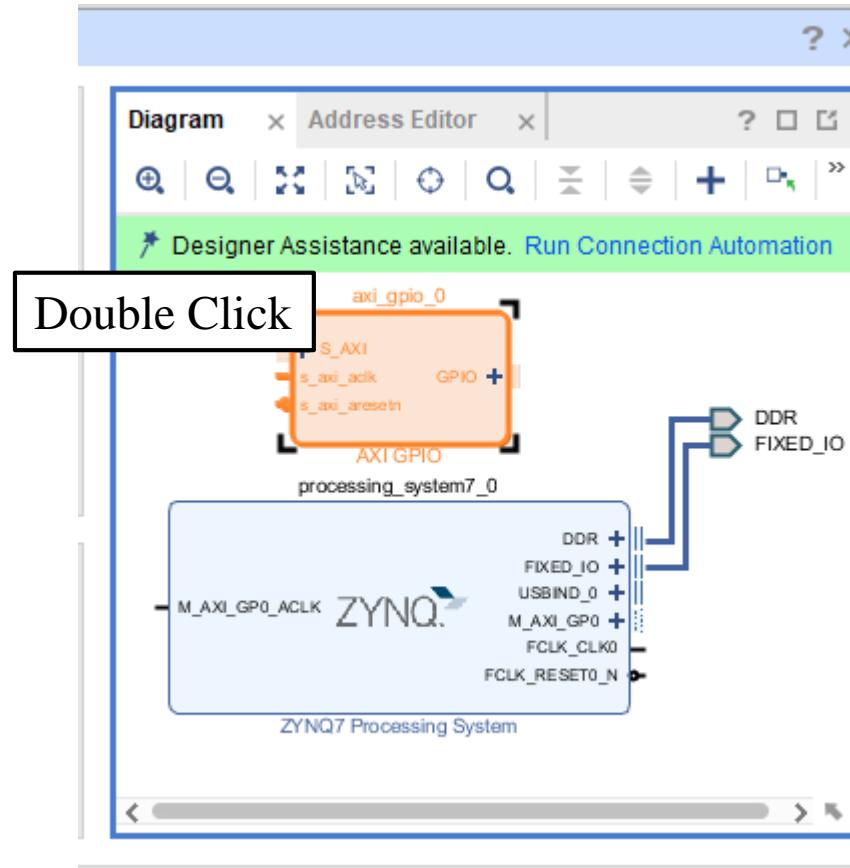


Build Basic Platform





Build Basic Platform



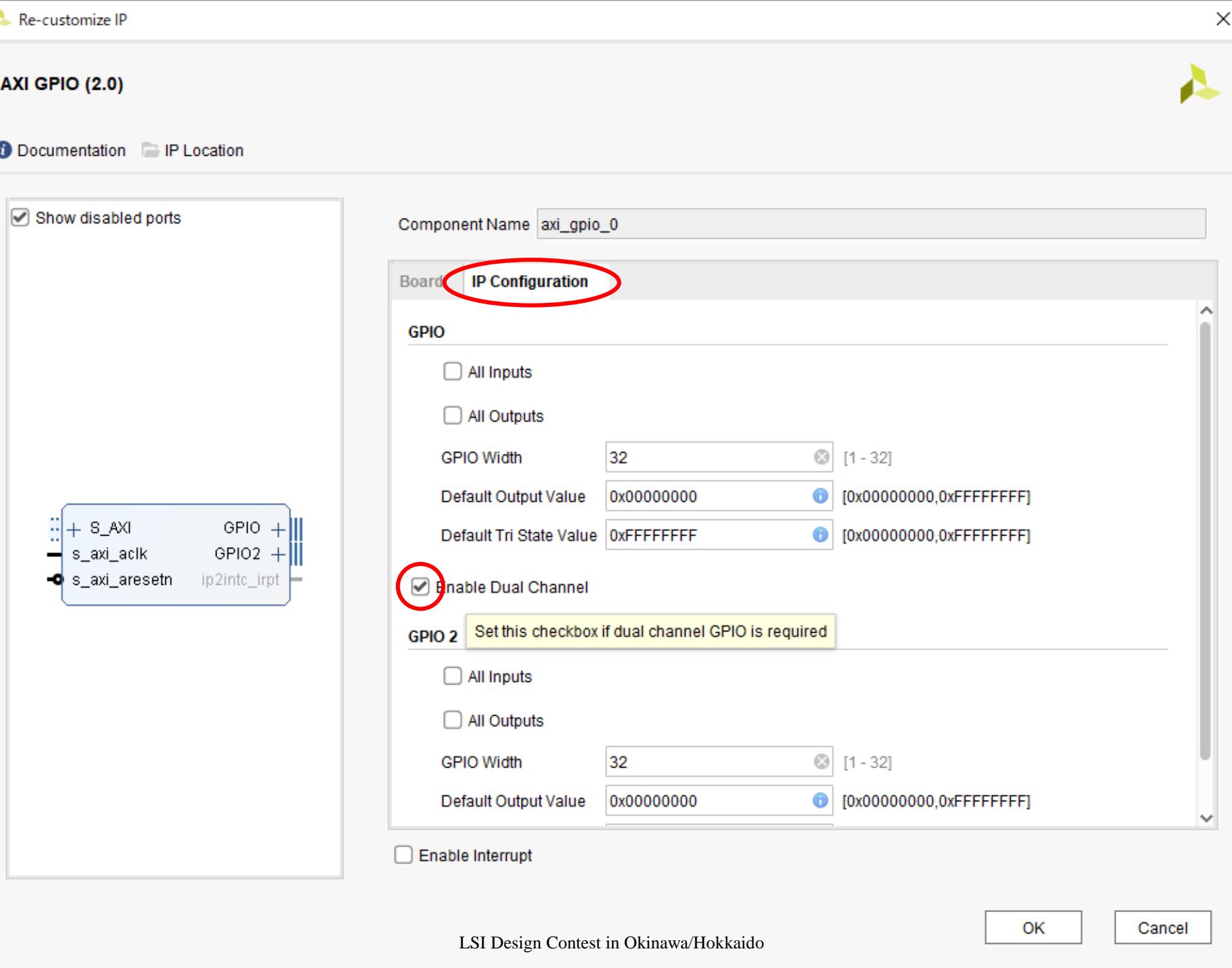
AXI GPIO (2.0)

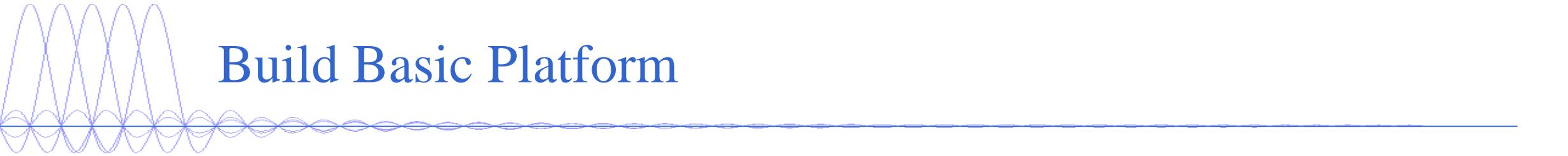
[Documentation](#) [IP Location](#) Show disabled portsComponent Name [Board](#)[IP Configuration](#)

Associate IP interface with board interface

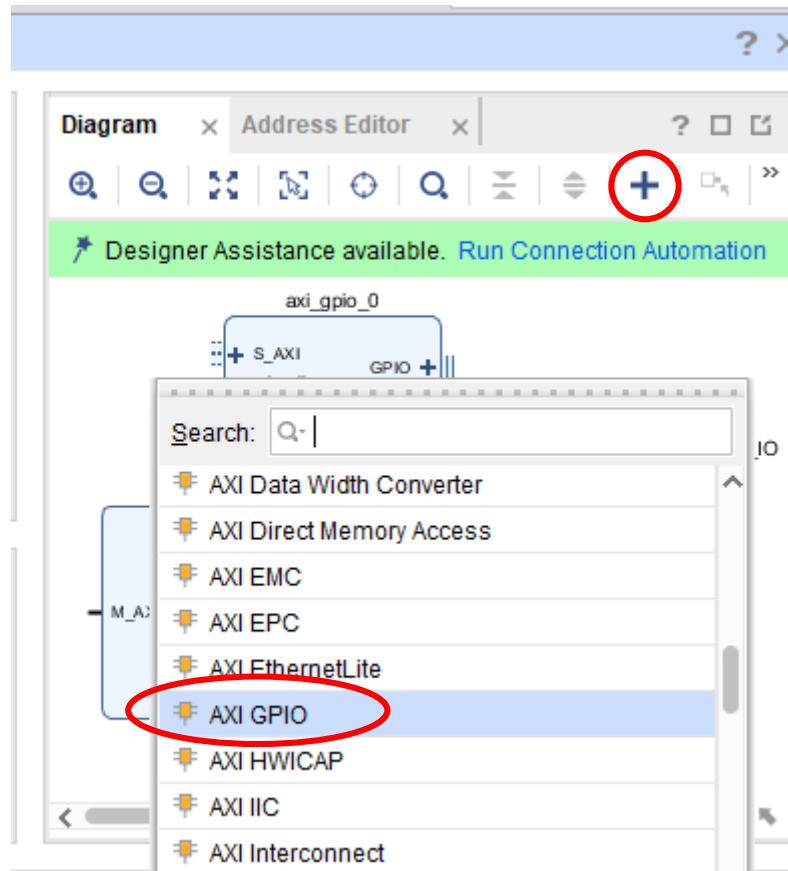
IP Interface	Board Interface
GPIO	Custom
GPIO2	Custom

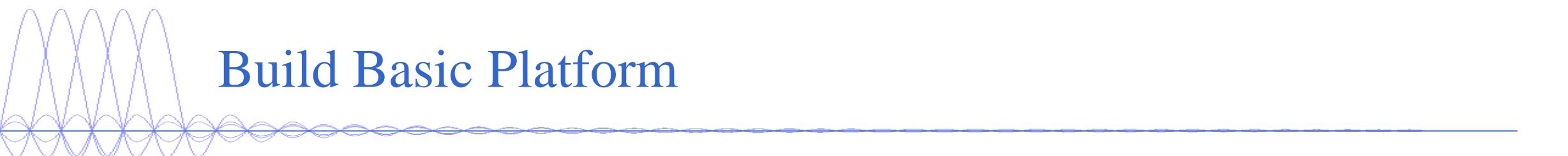
[Clear Board Parameters](#) Enable Interrupt[OK](#)[Cancel](#)



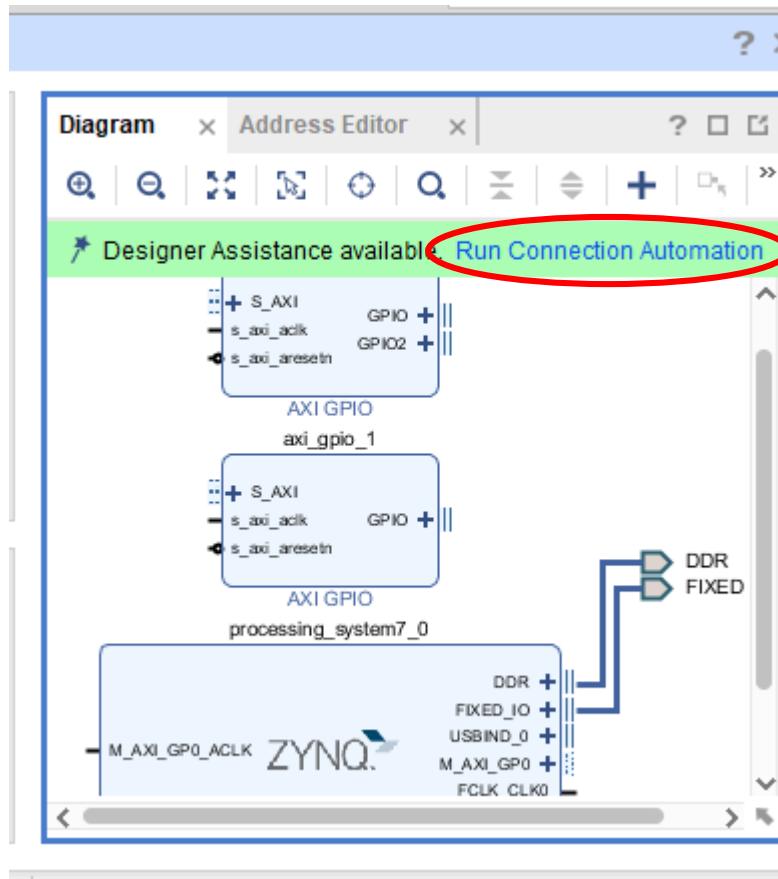


Build Basic Platform

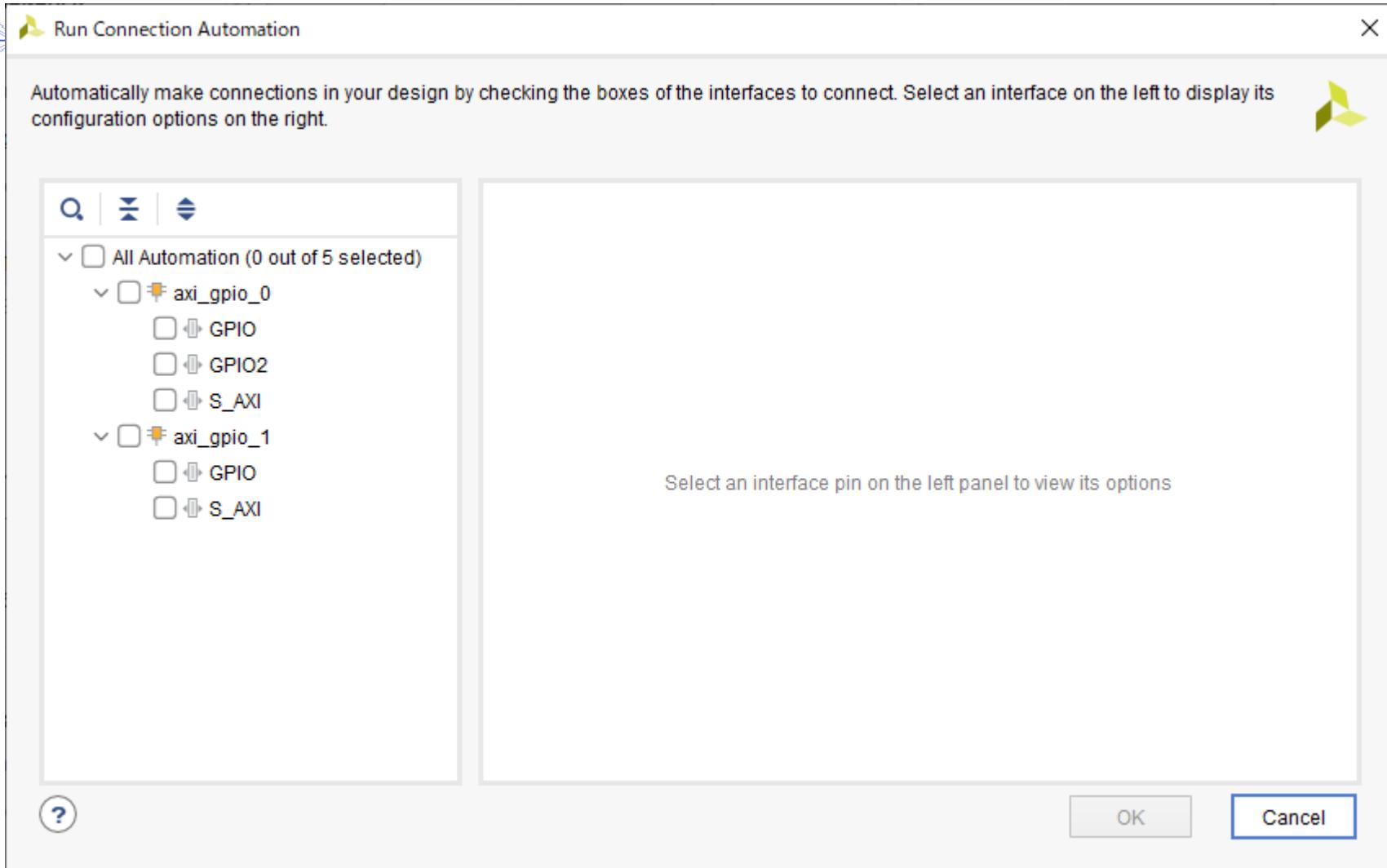




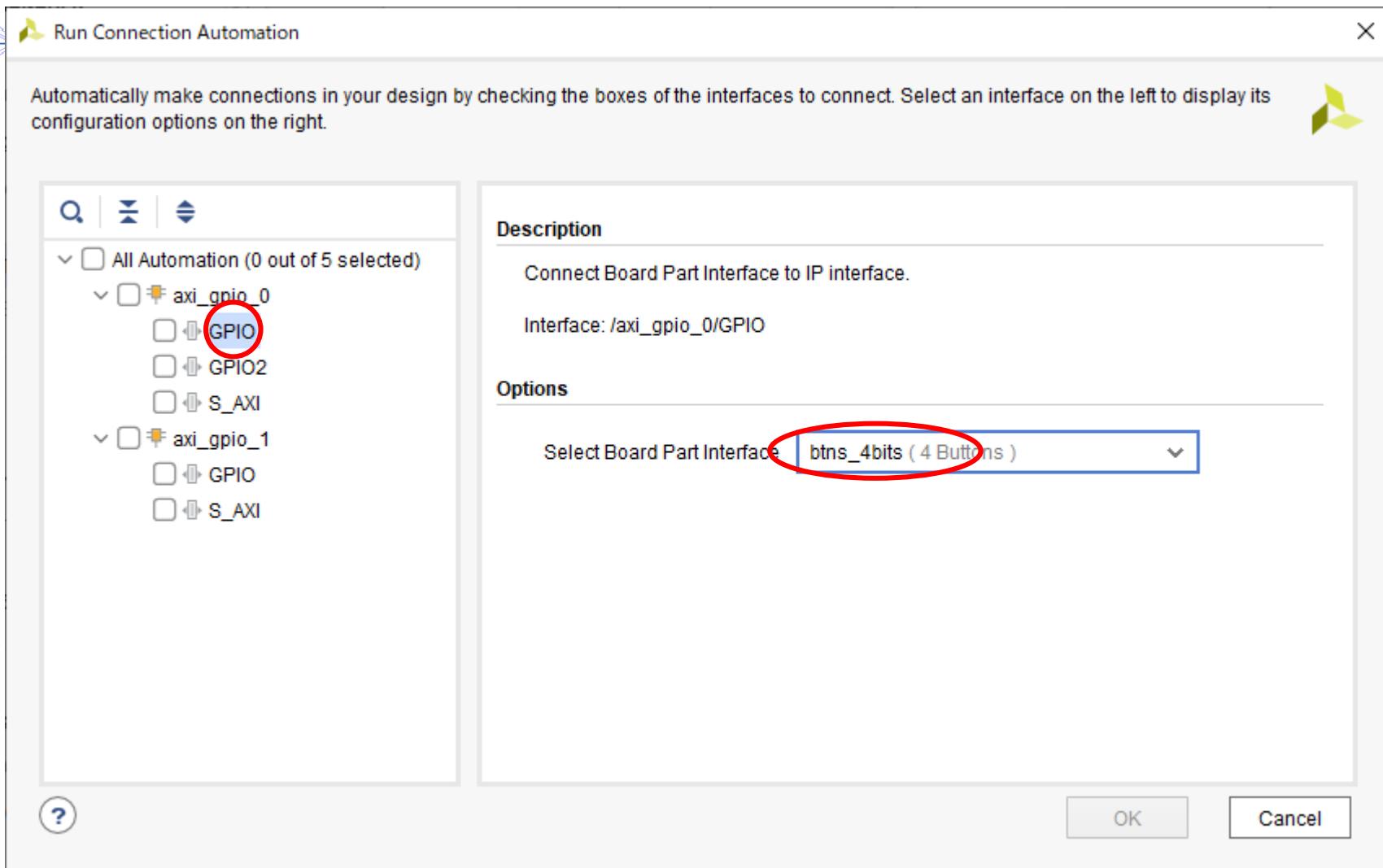
Build Basic Platform



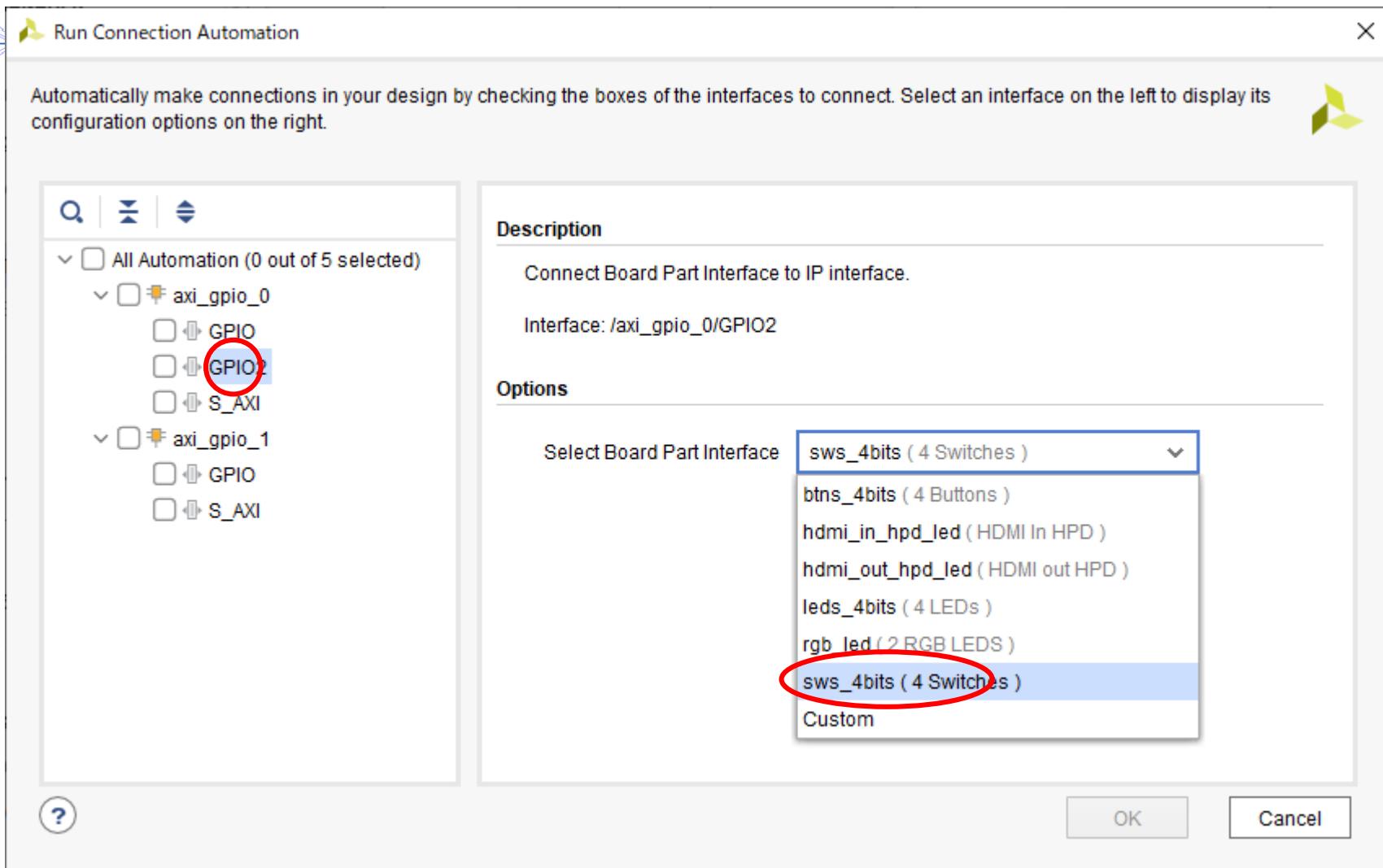
Build Basic Platform



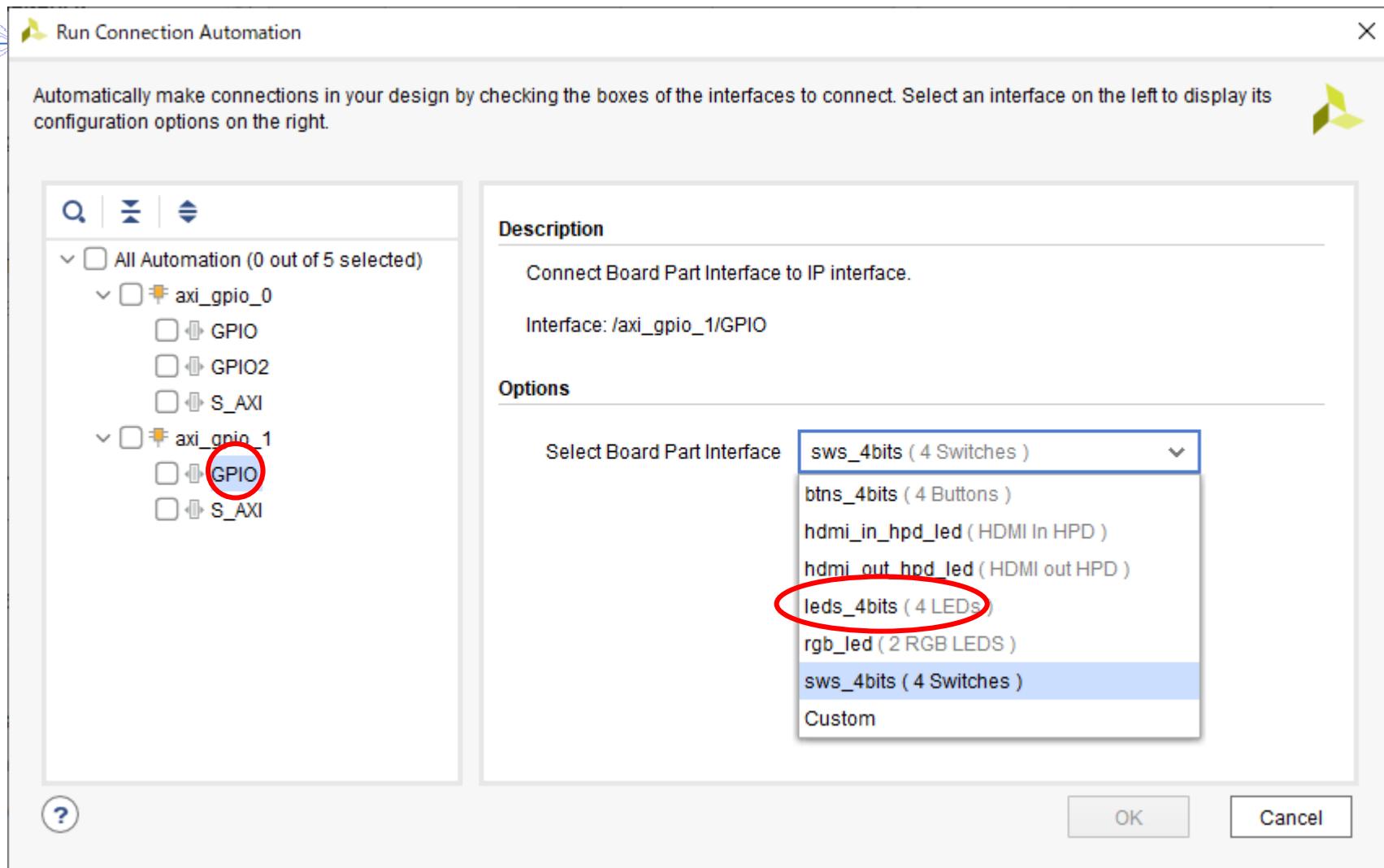
Build Basic Platform



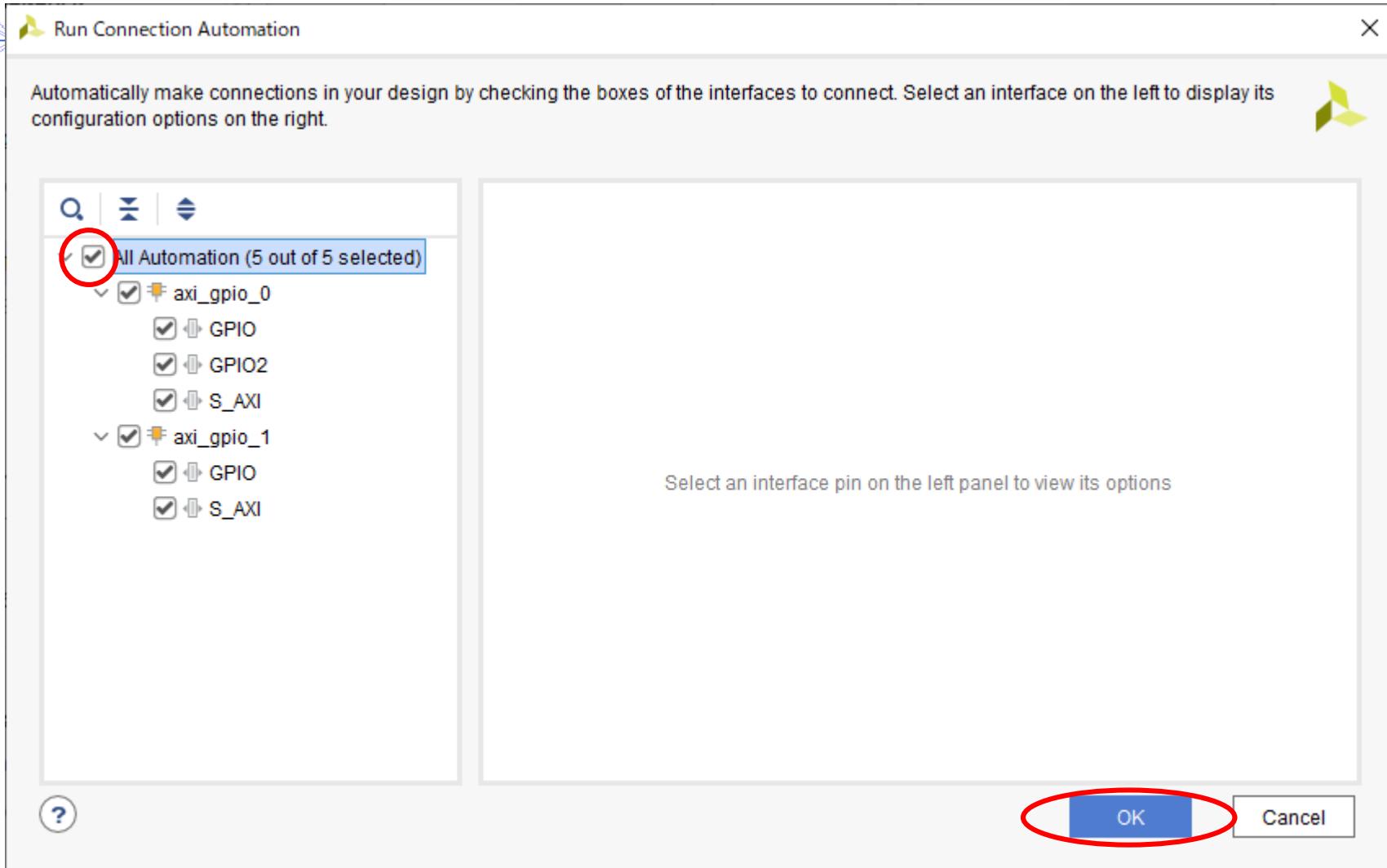
Build Basic Platform

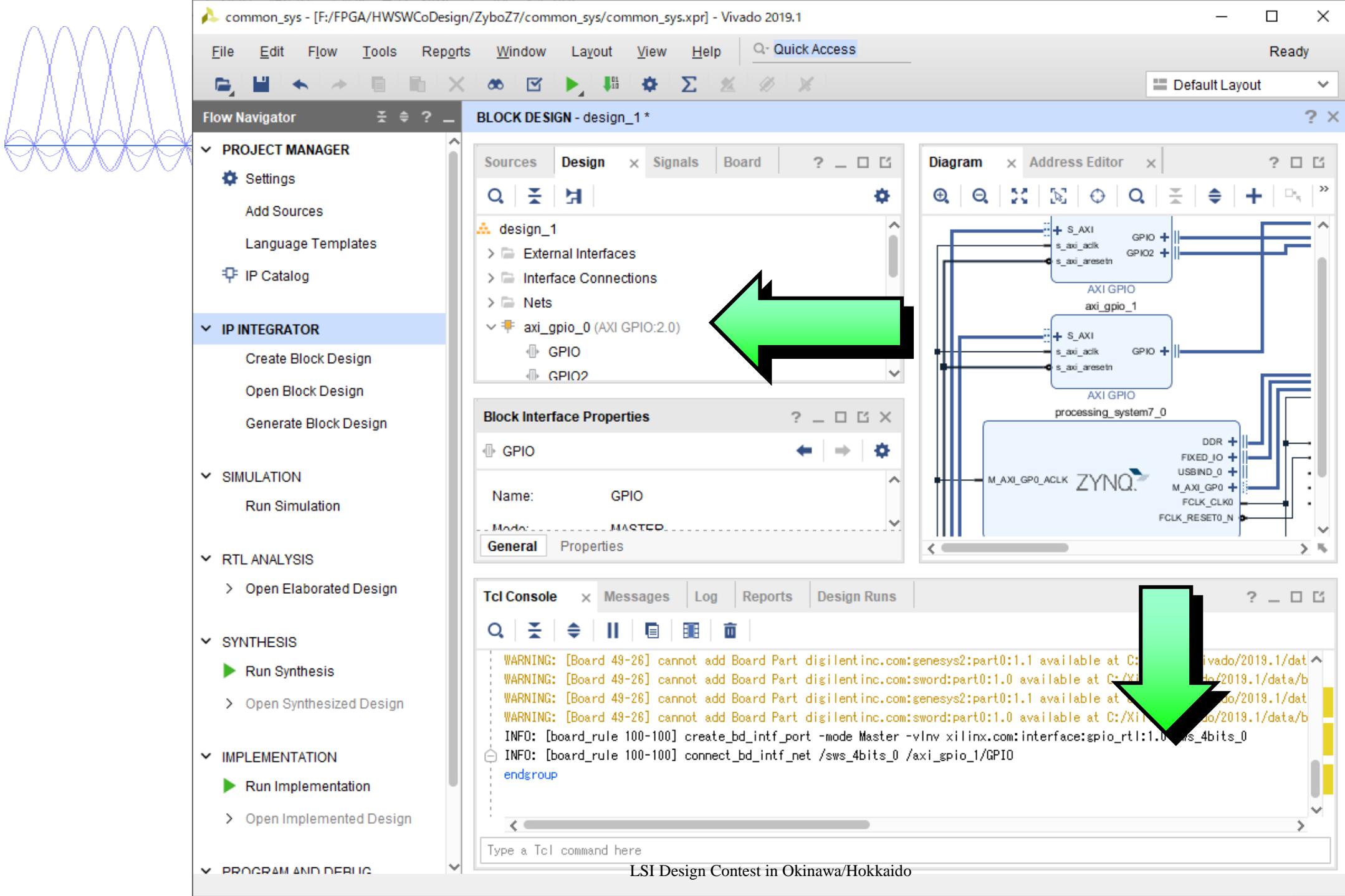


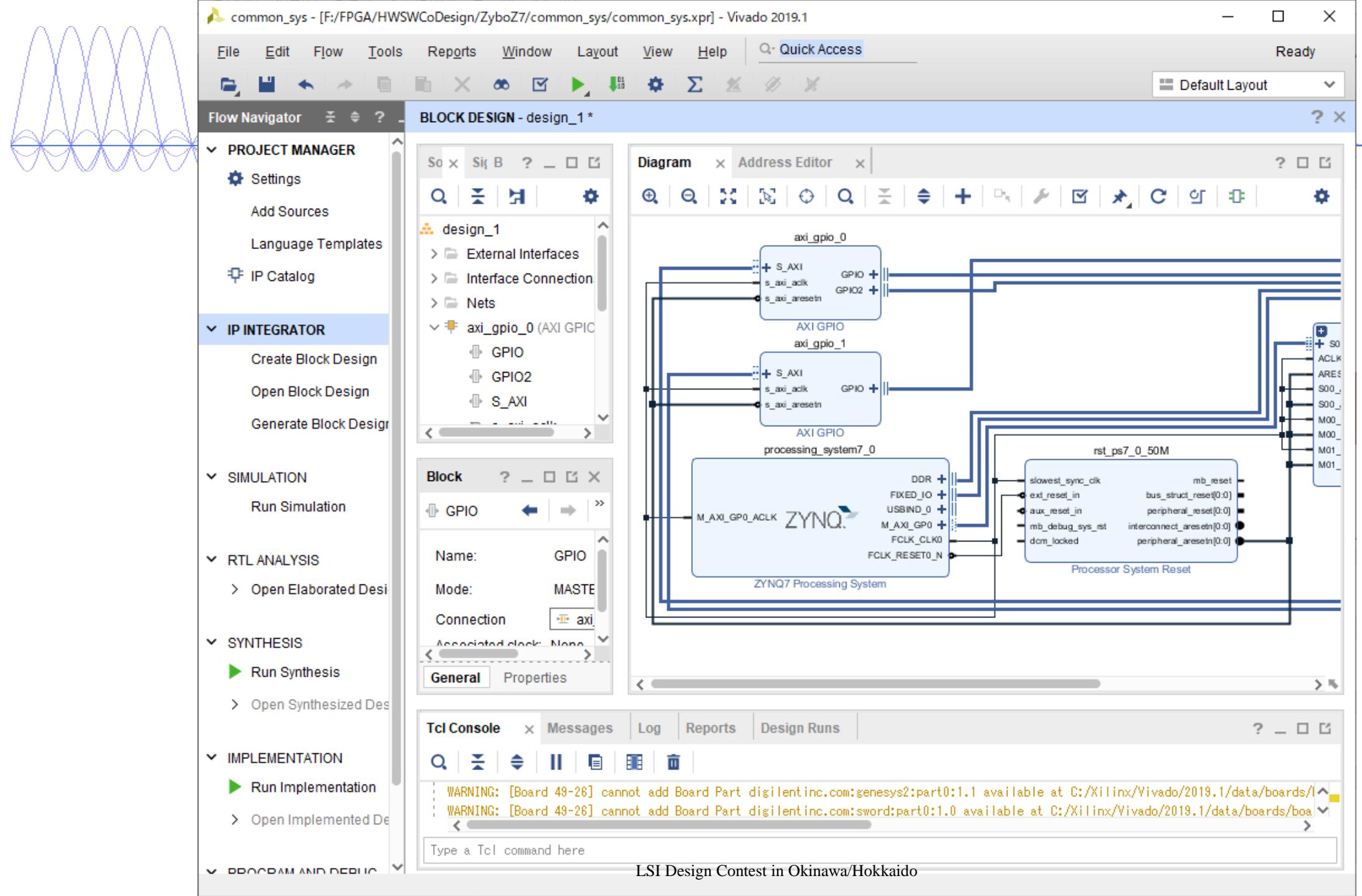
Build Basic Platform

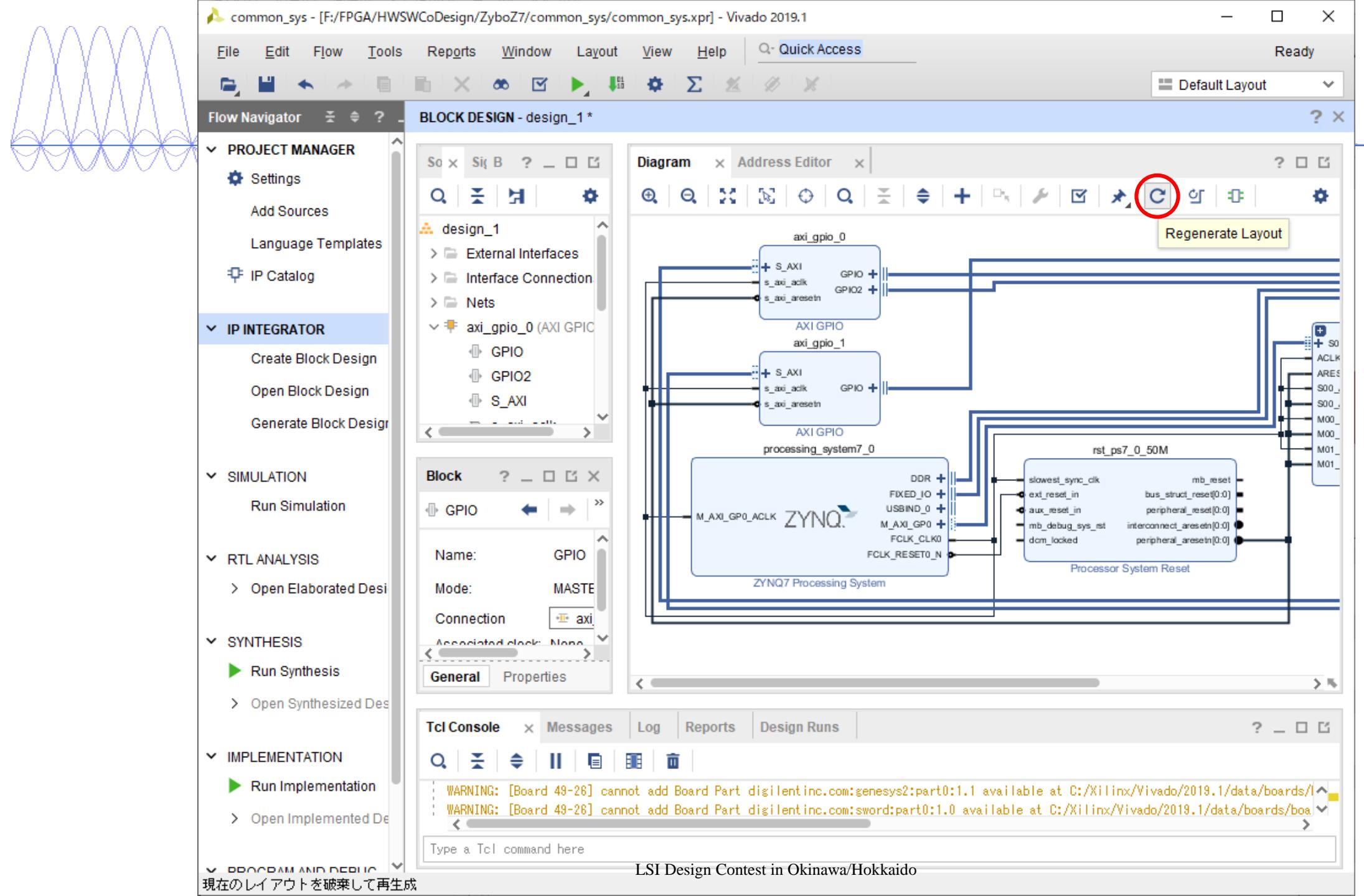


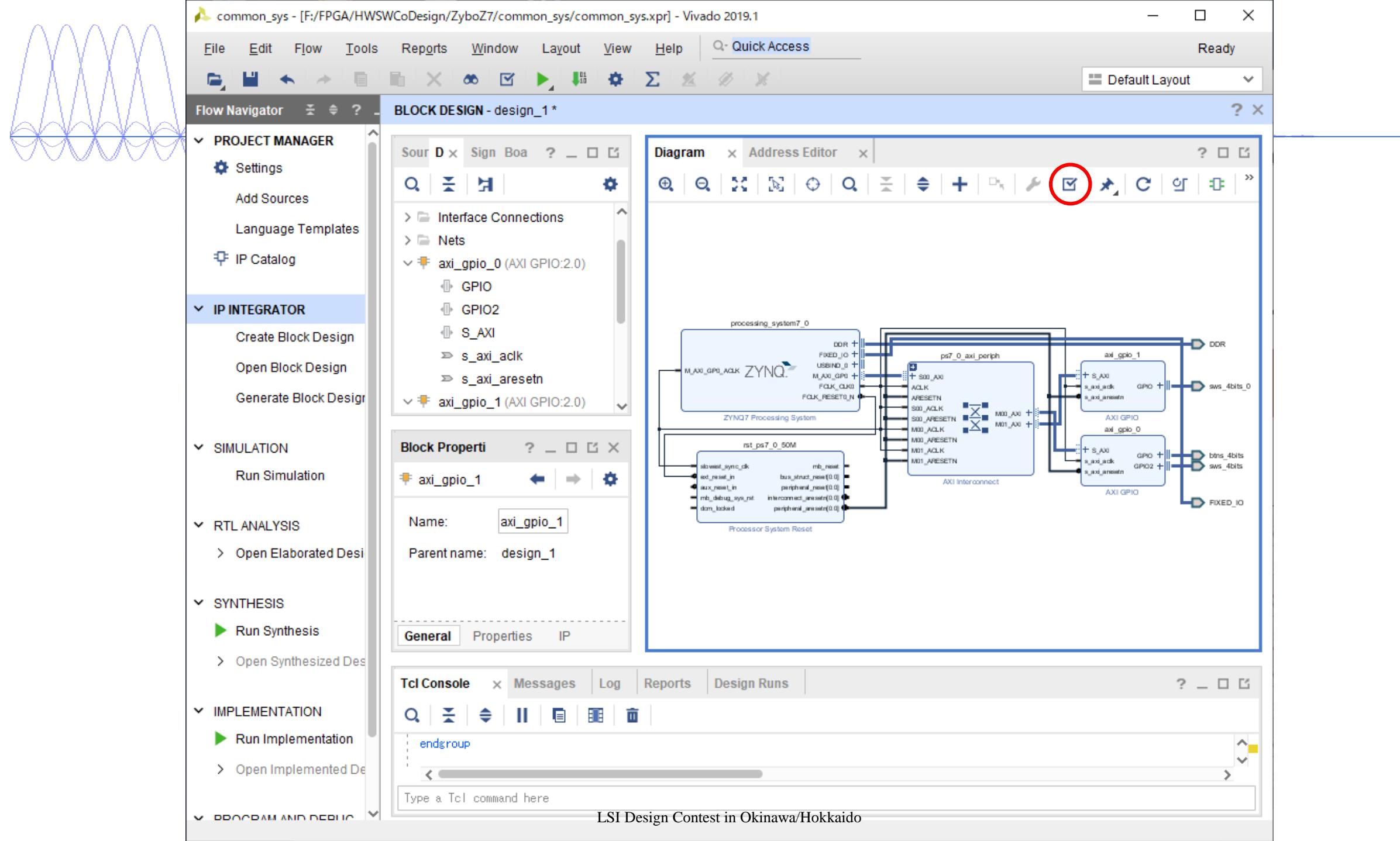
Build Basic Platform

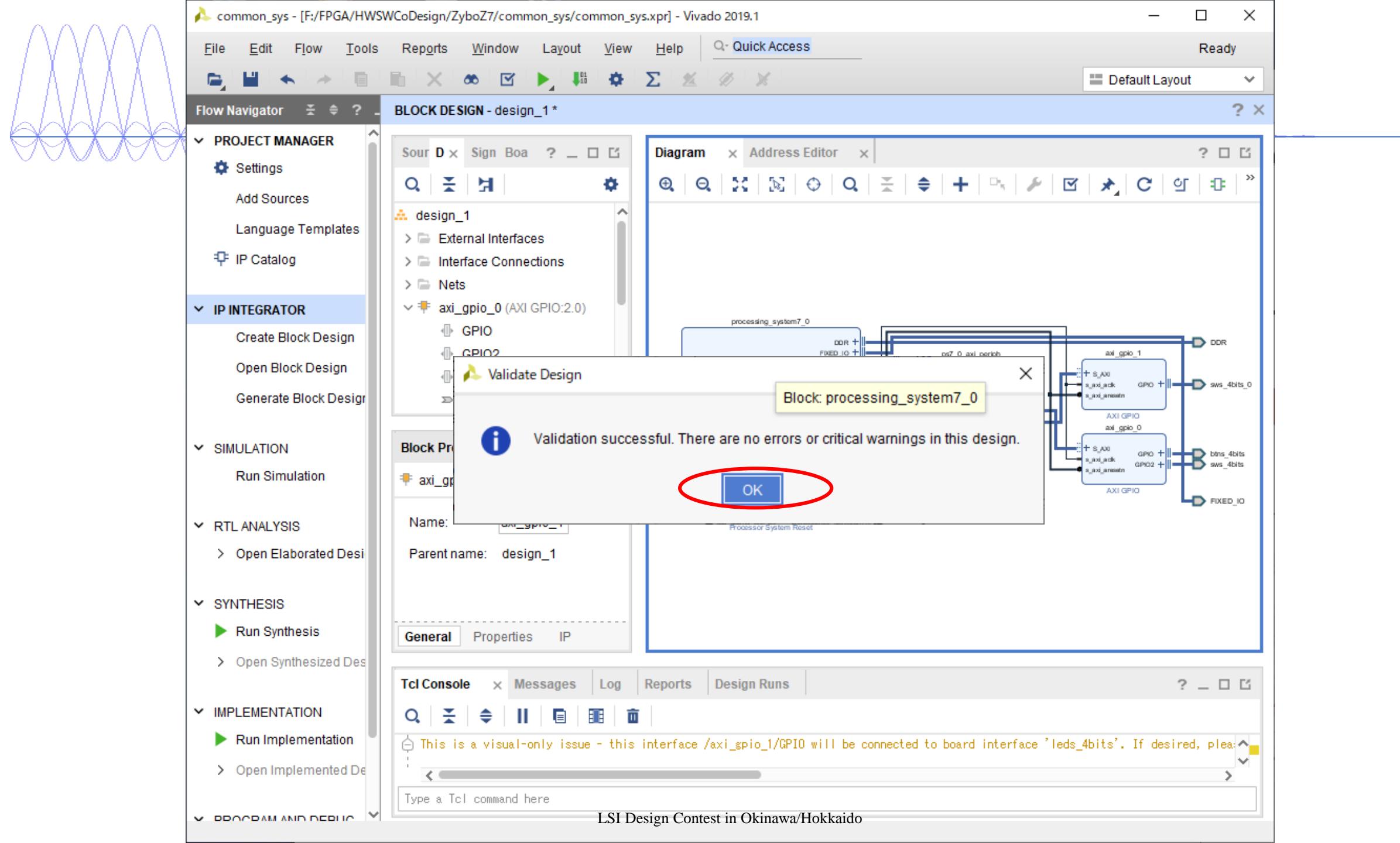


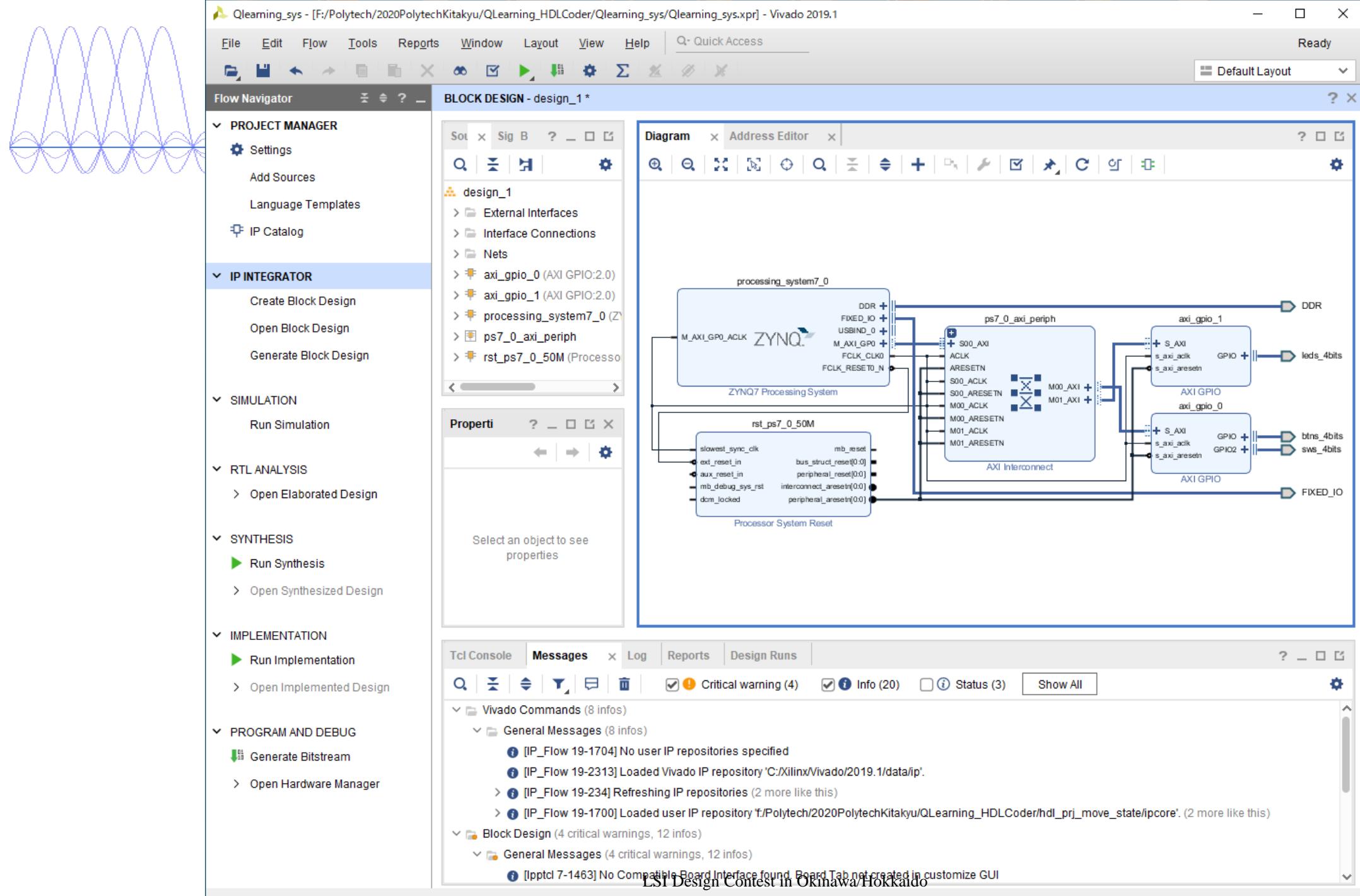


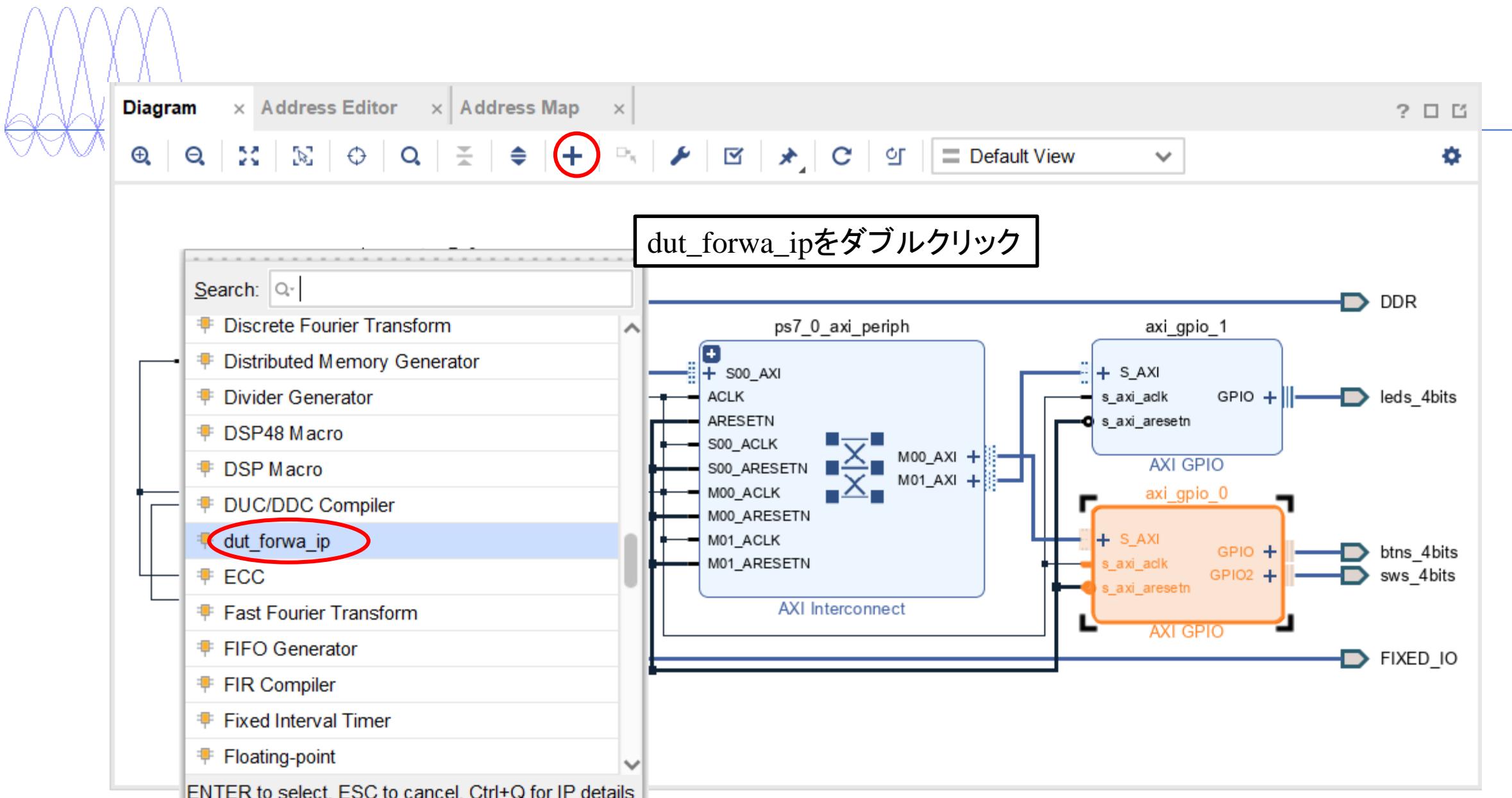


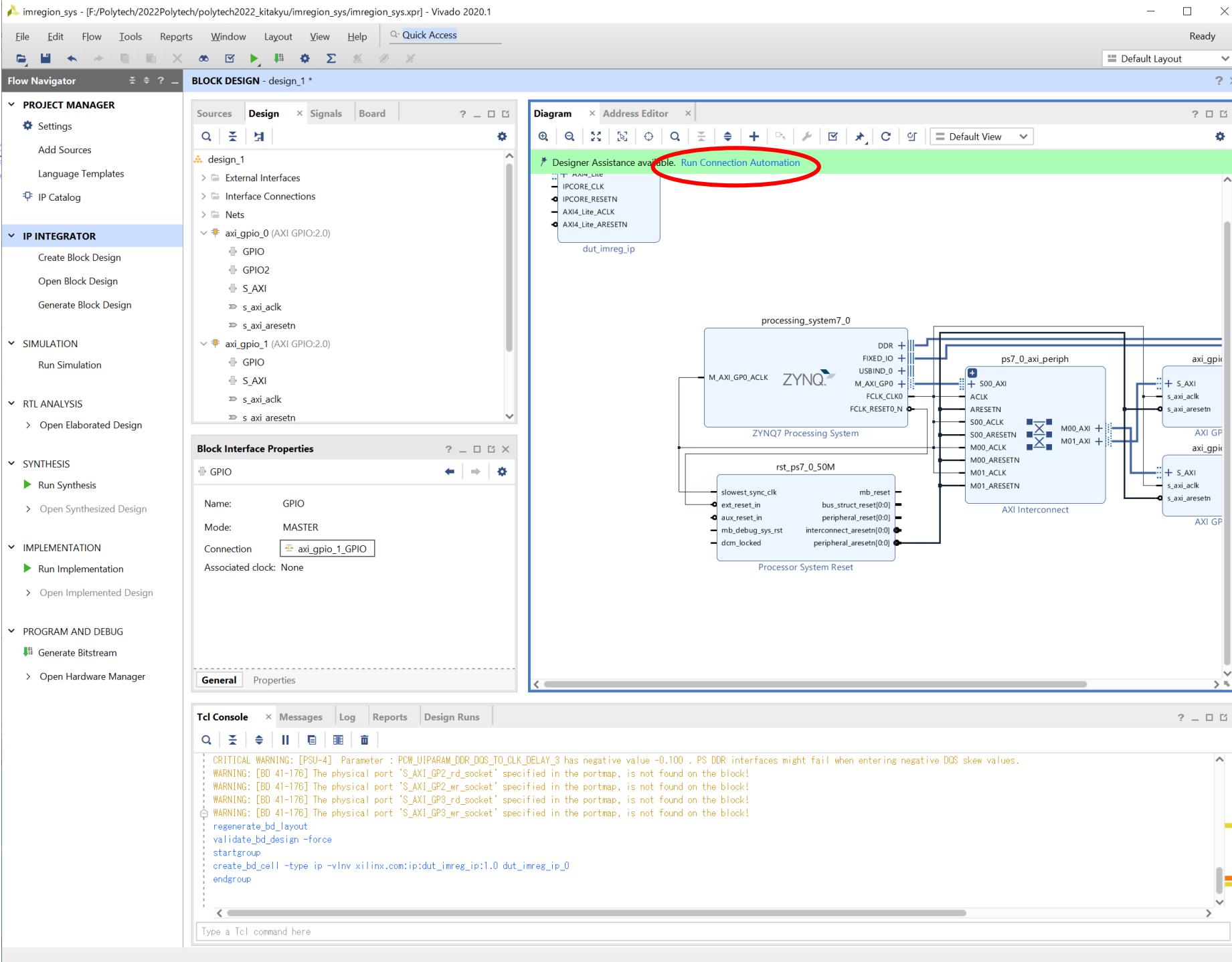


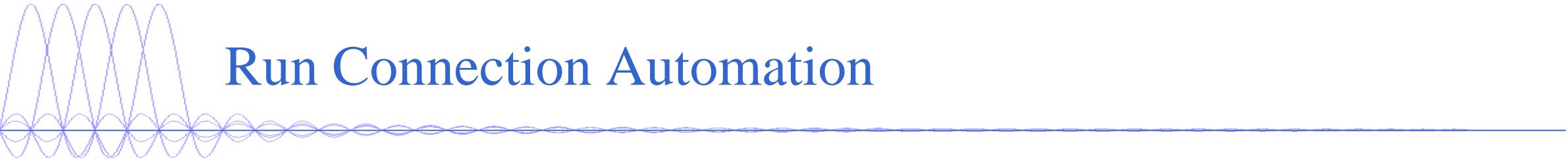




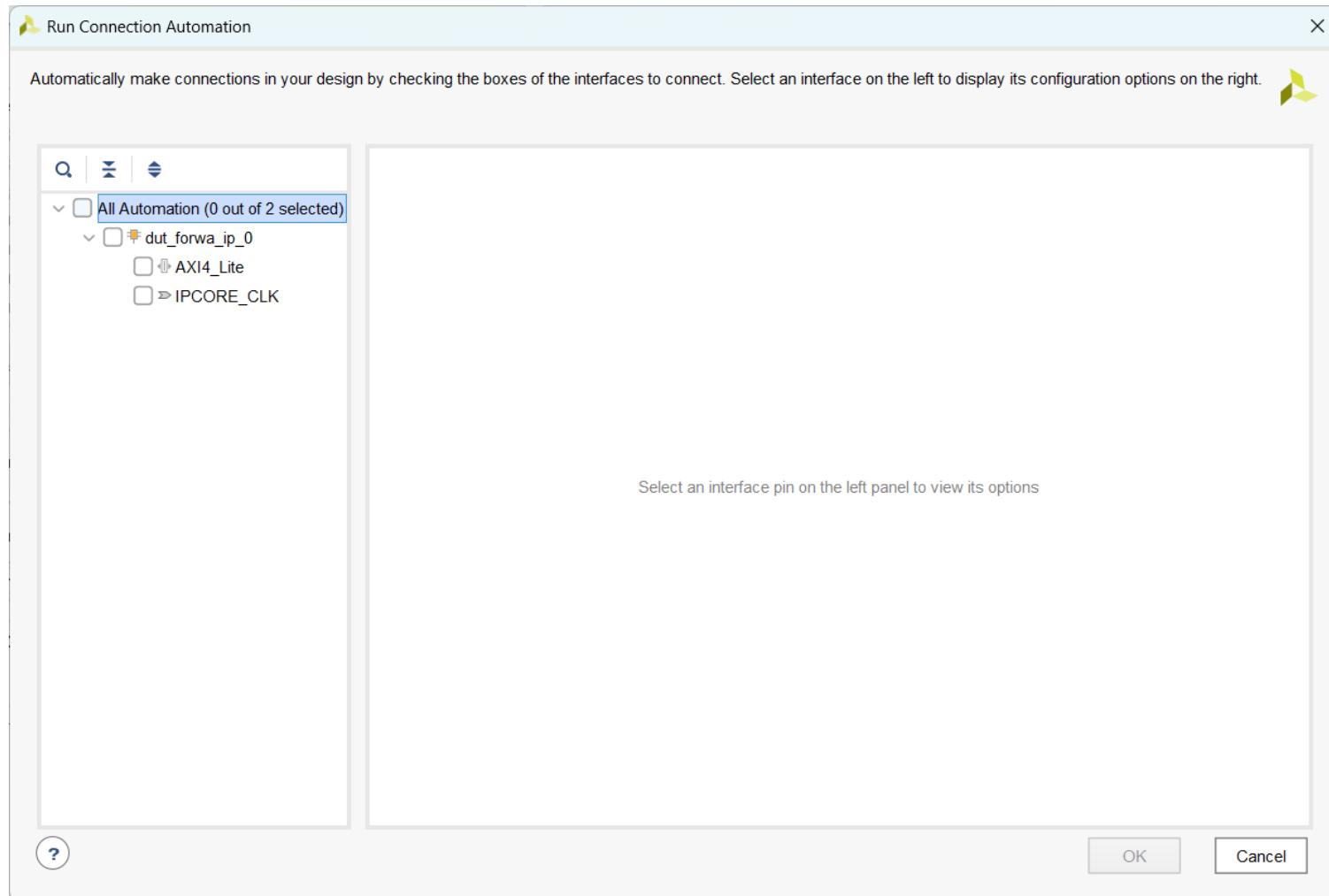


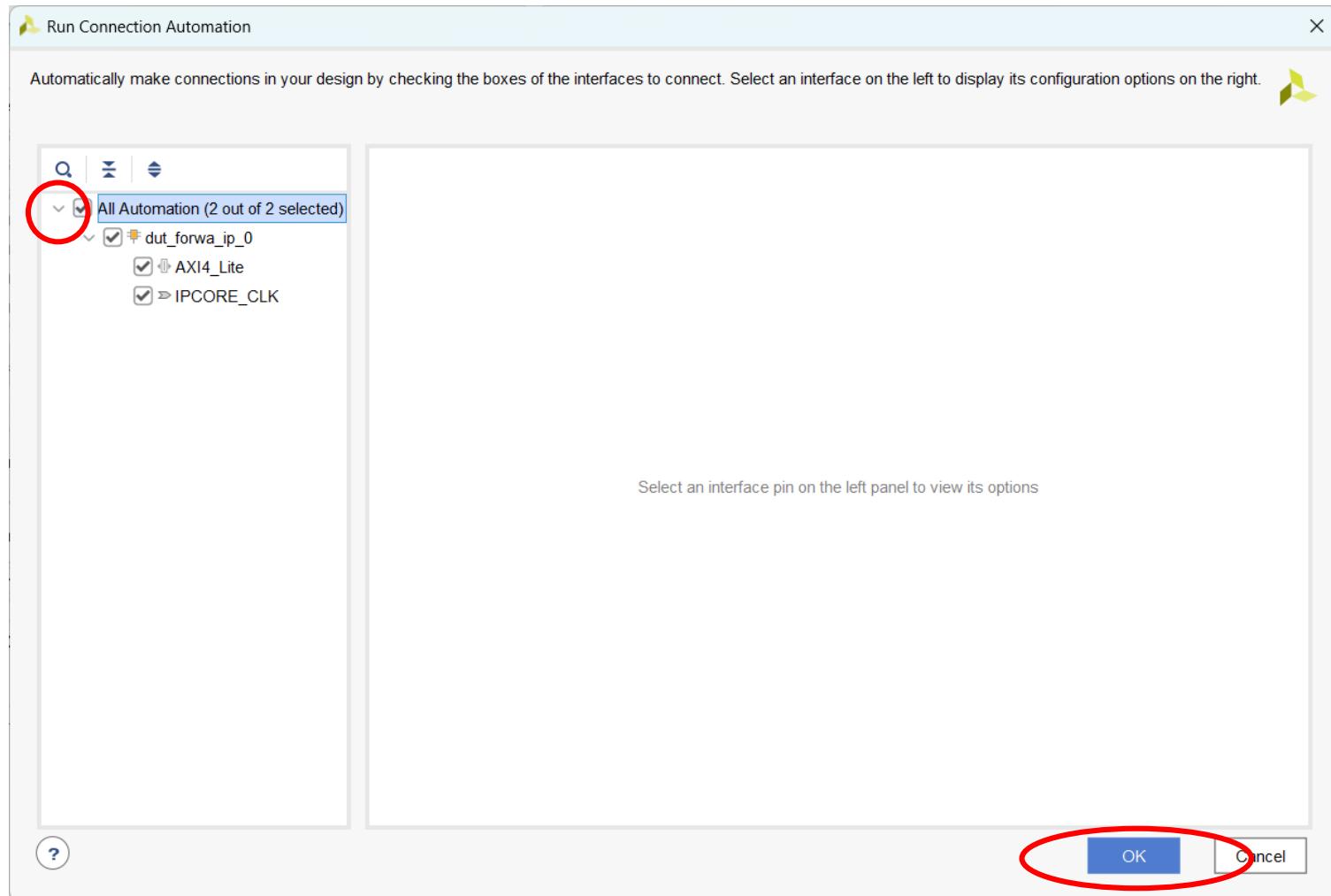
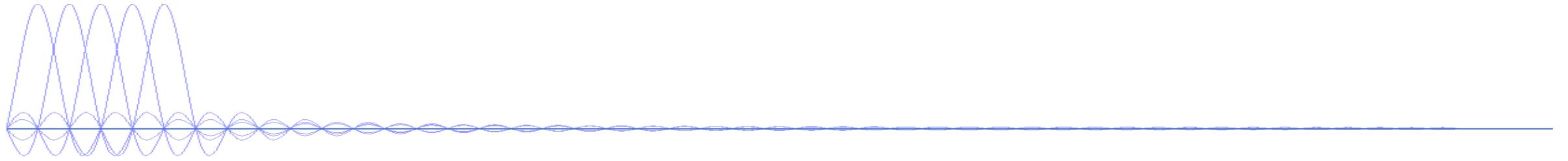


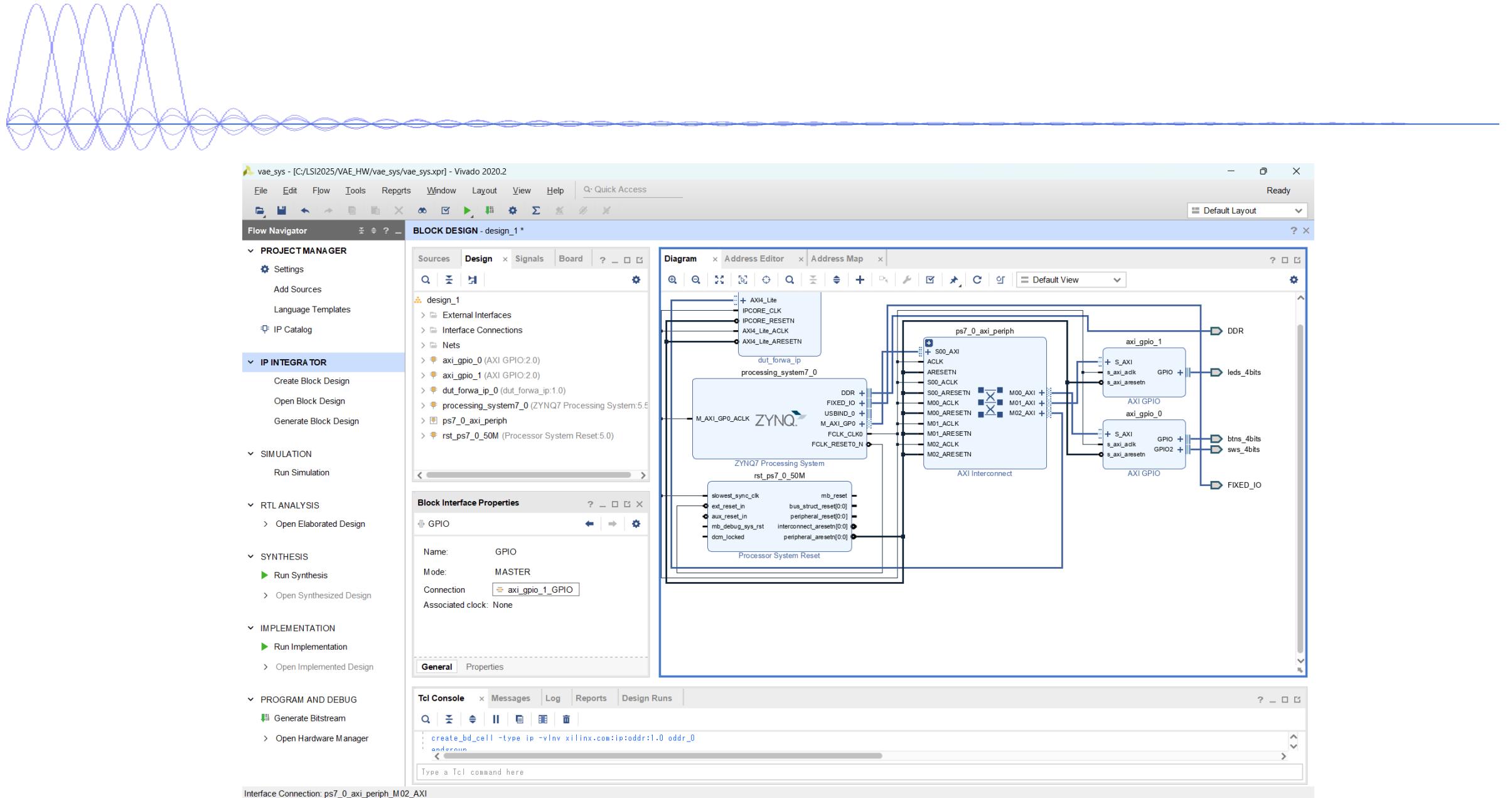


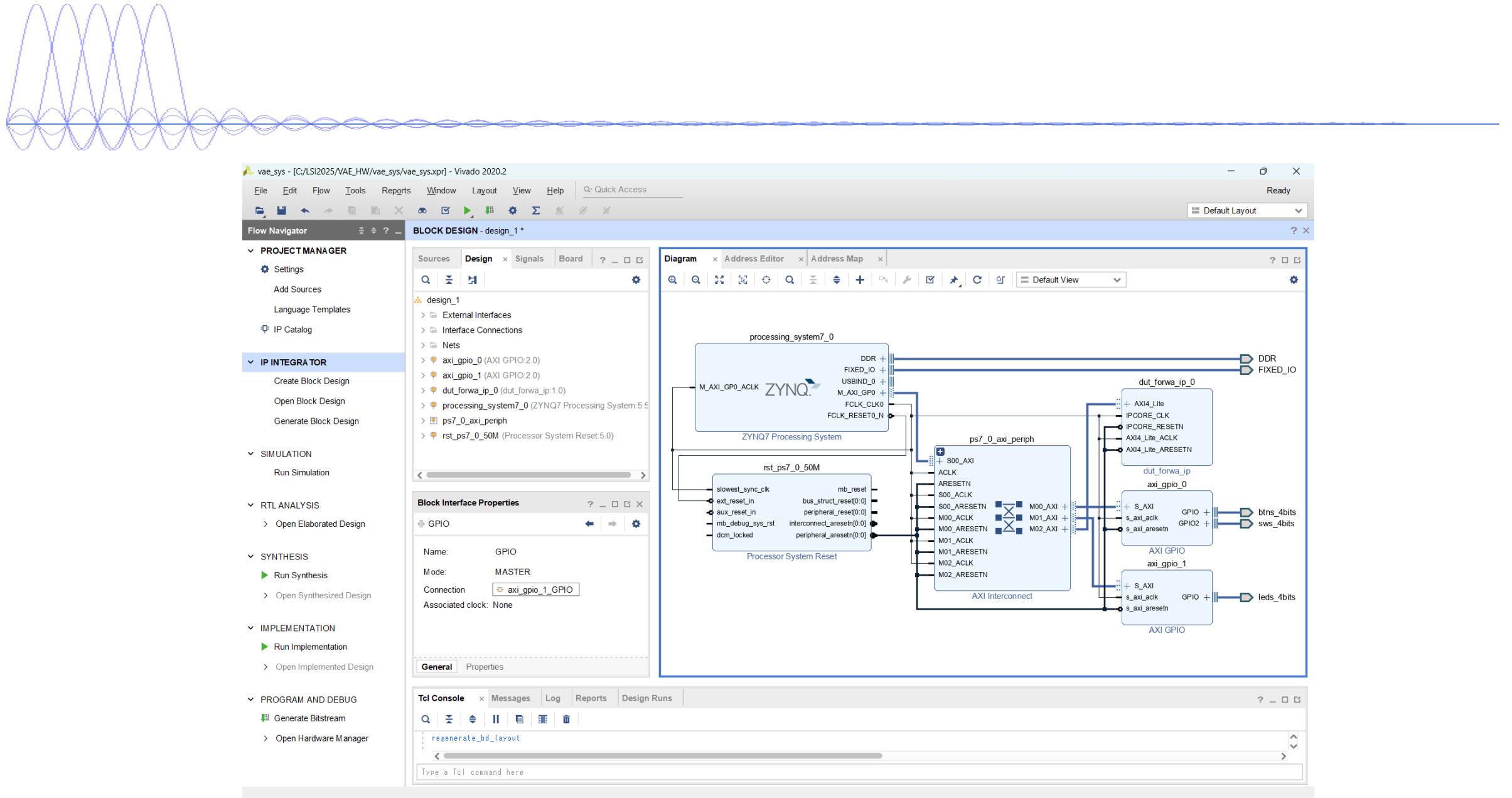


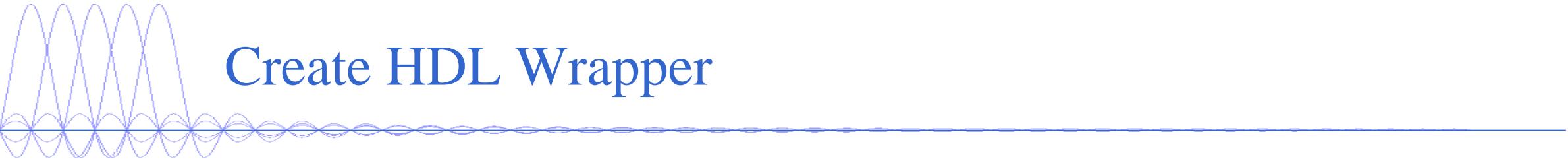
Run Connection Automation



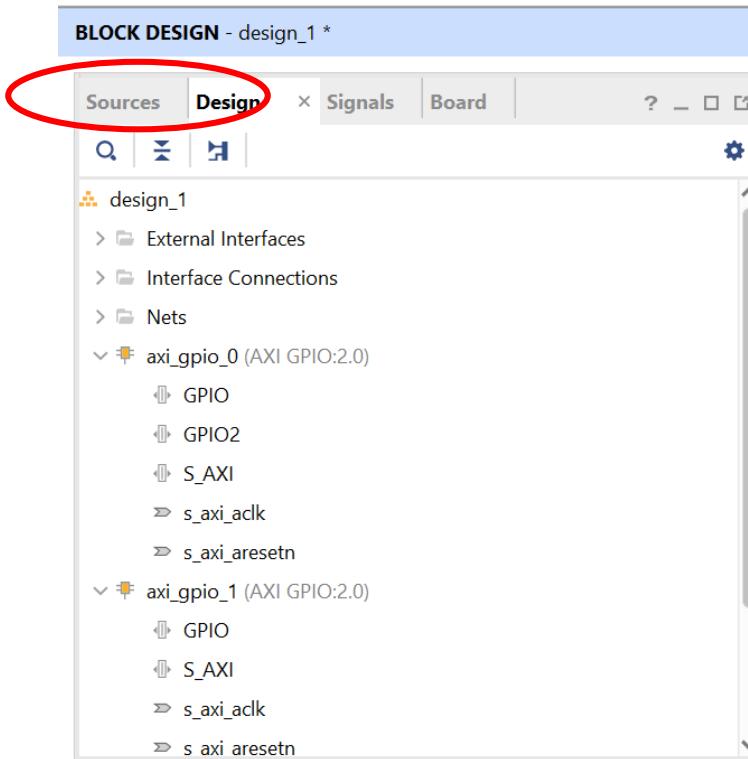


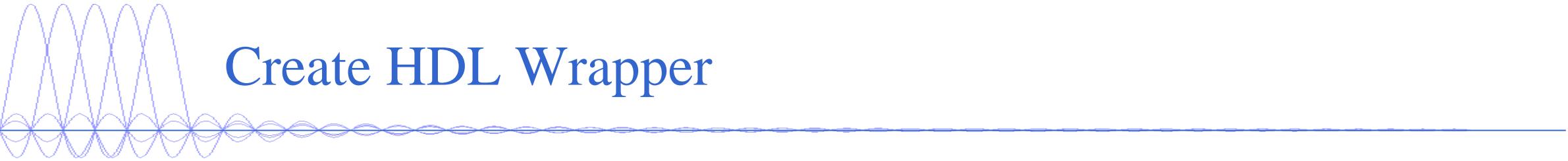




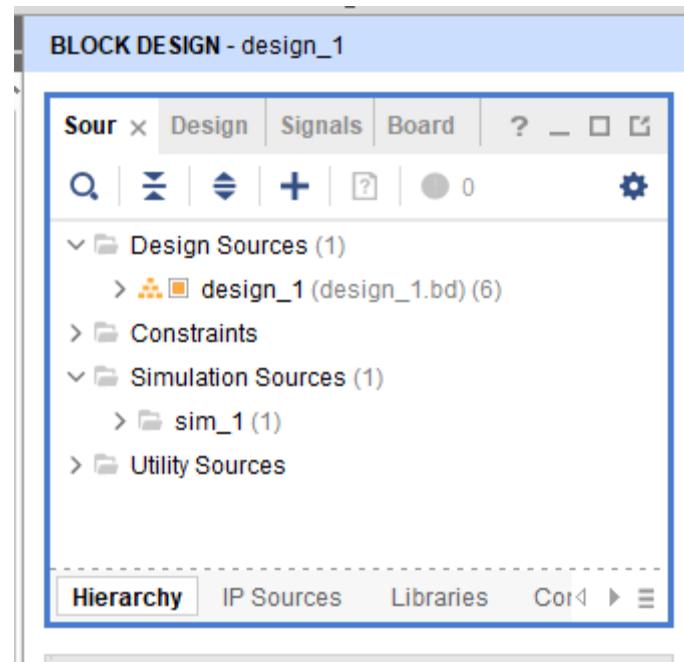


Create HDL Wrapper



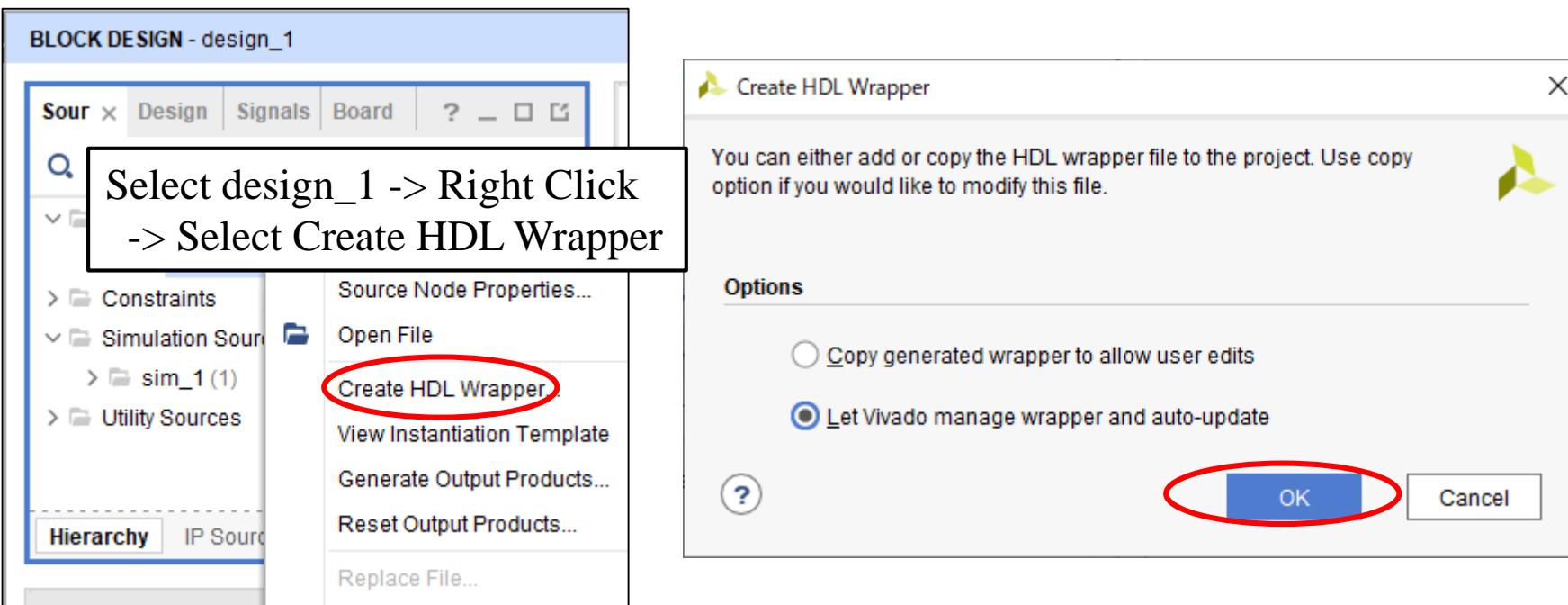


Create HDL Wrapper

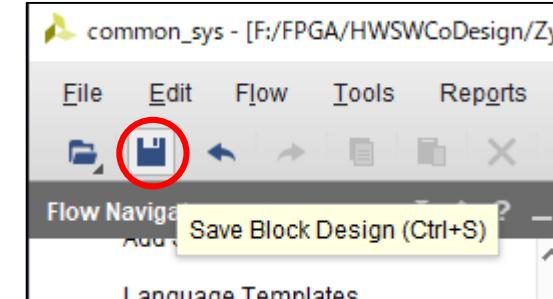
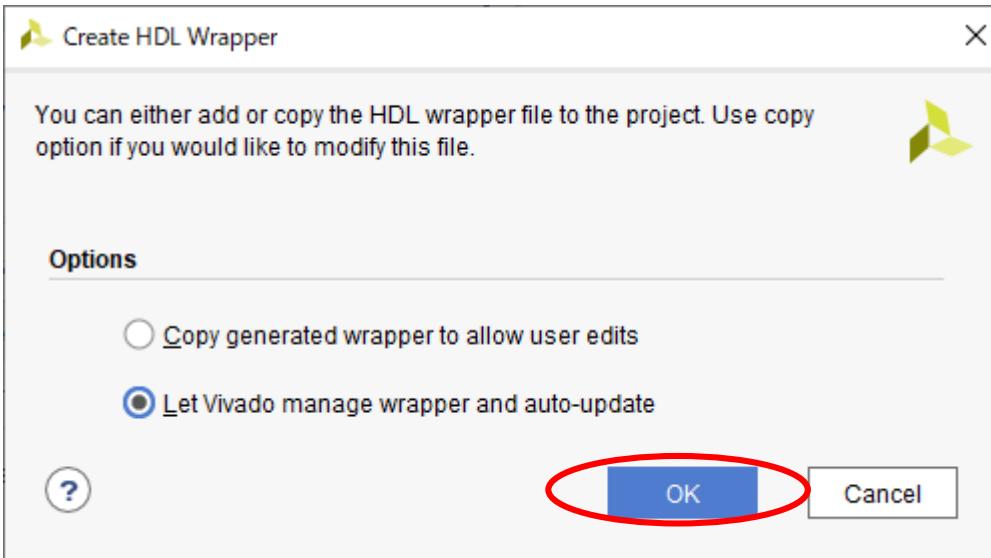


Create HDL Wrapper

design_1を一度左でクリックして、
その後右クリック

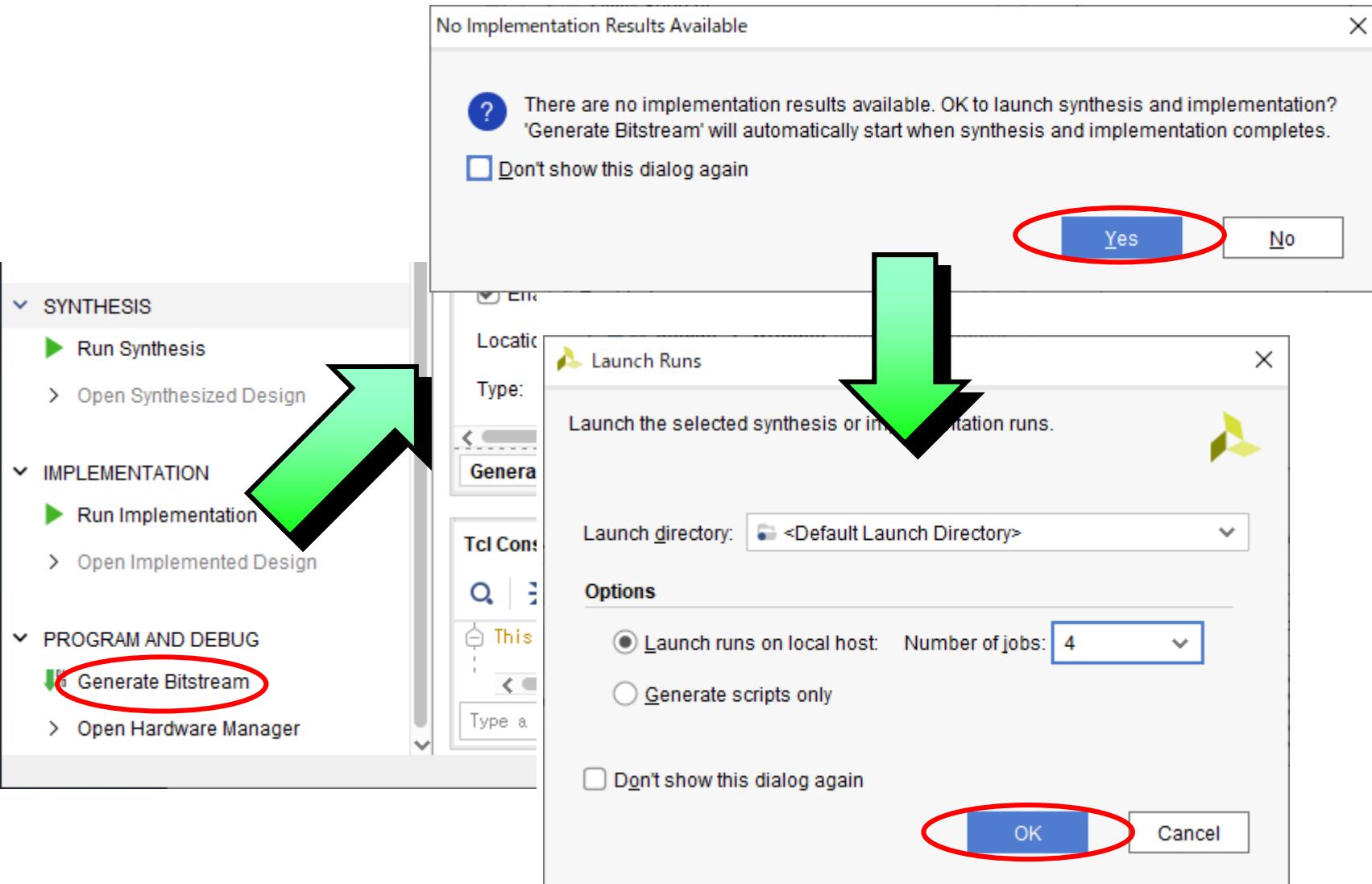


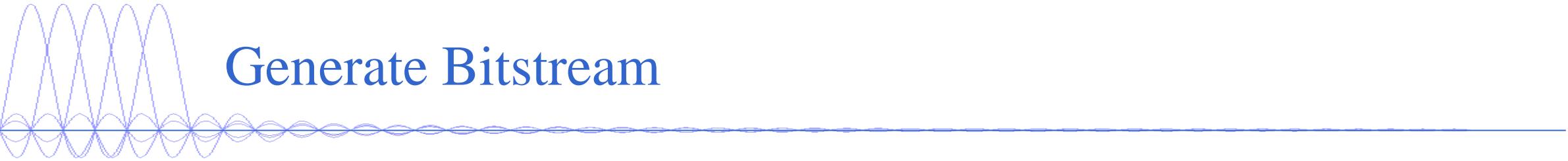
Create HDL Wrapper



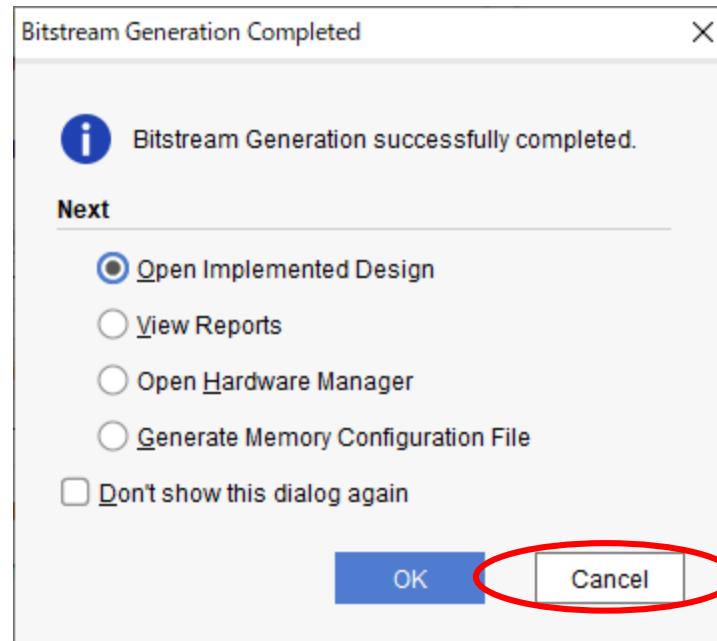
再びWrapperを作るとこのような
ダイアログが表示される

Generate Bitstream

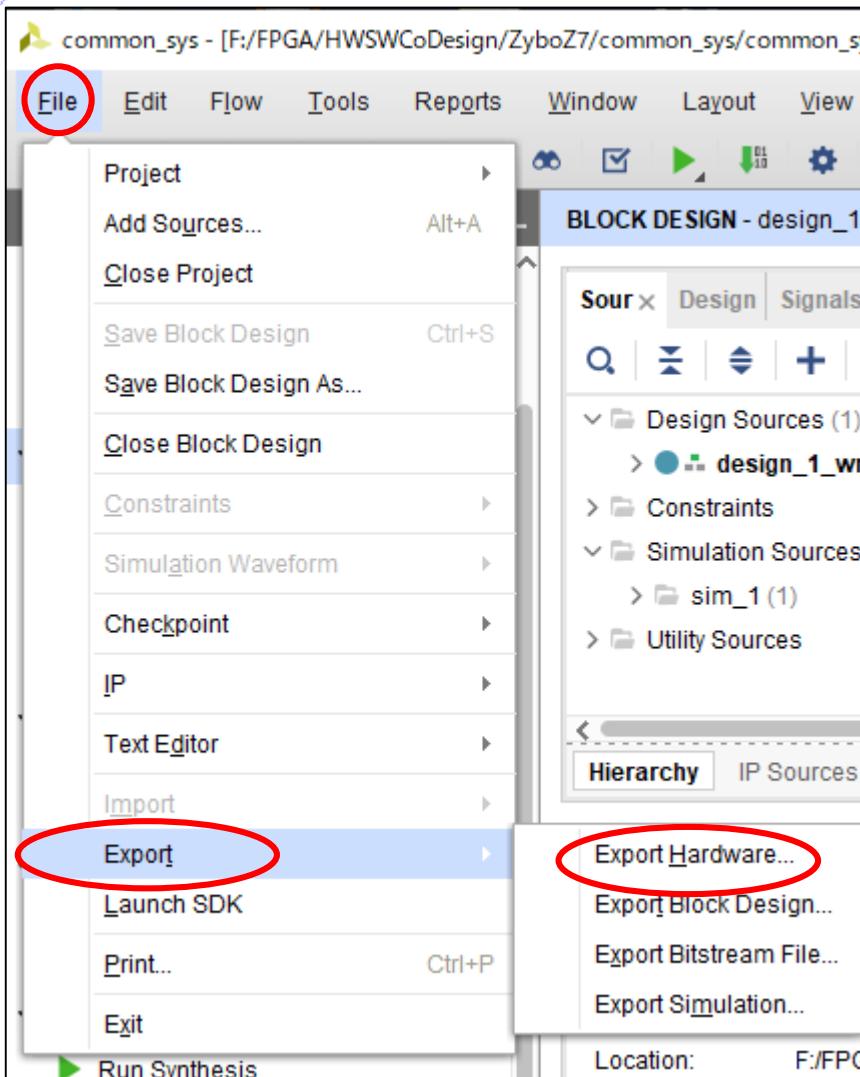


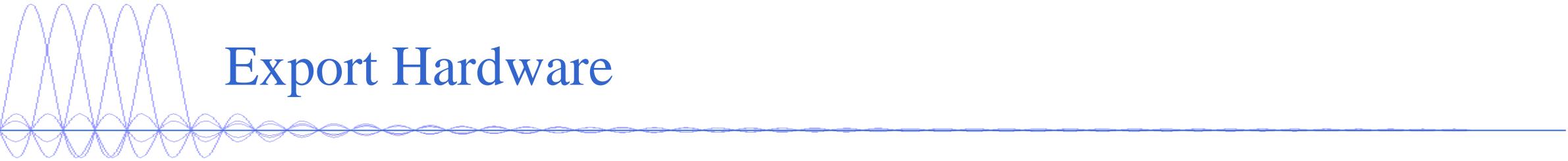


Generate Bitstream

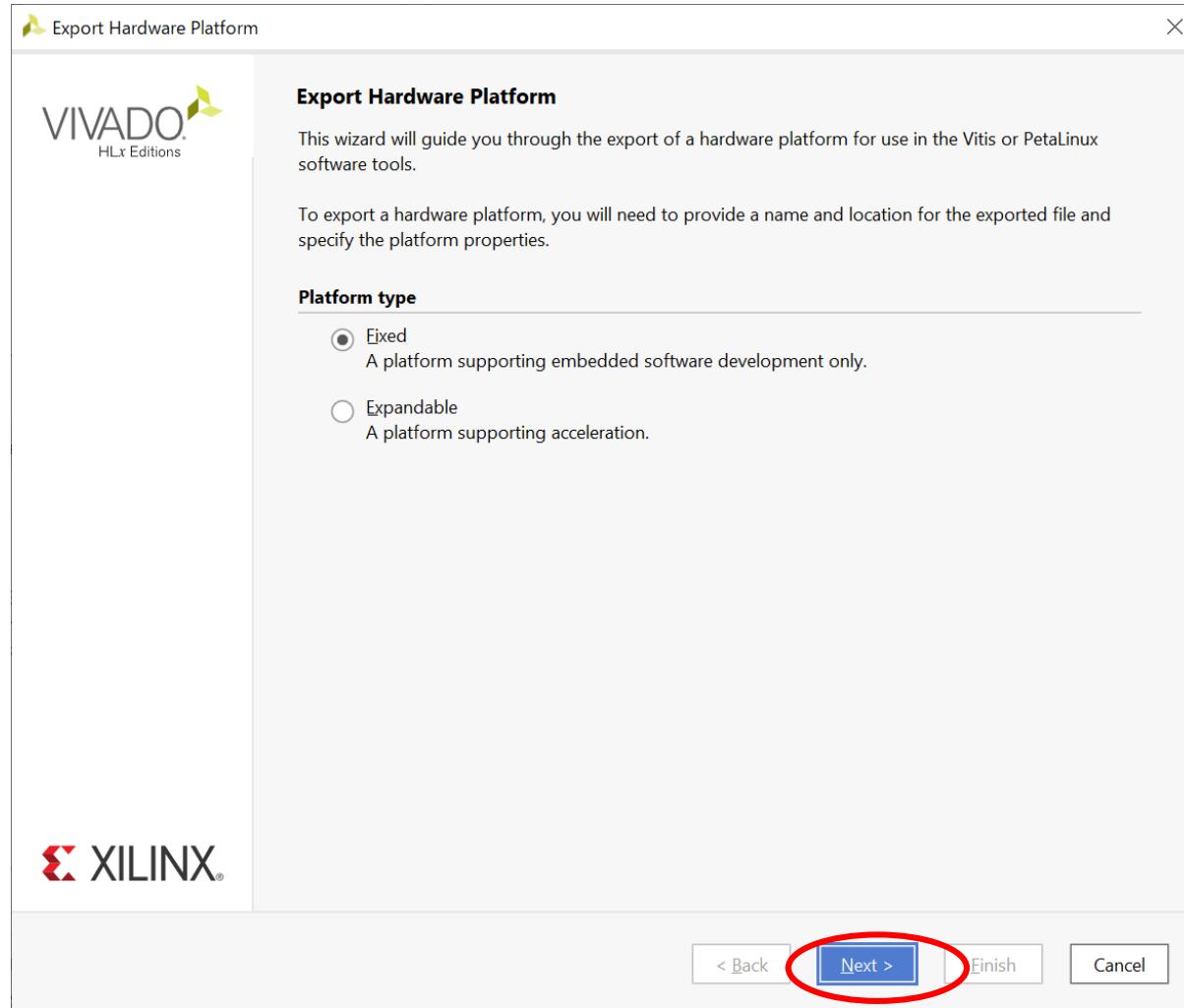


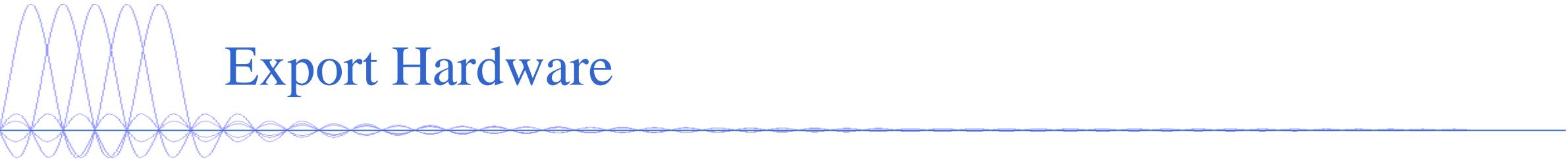
Export Hardware



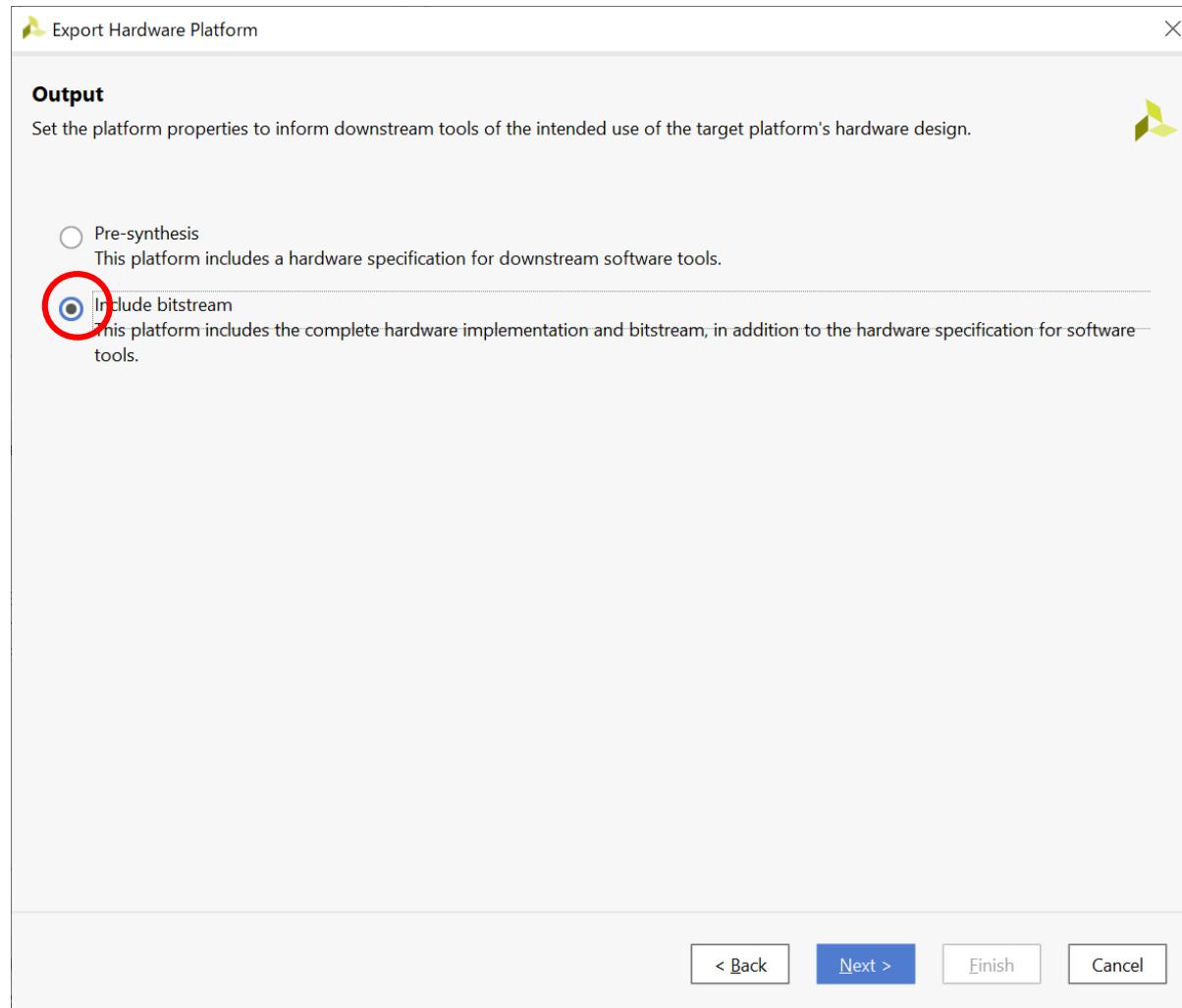


Export Hardware



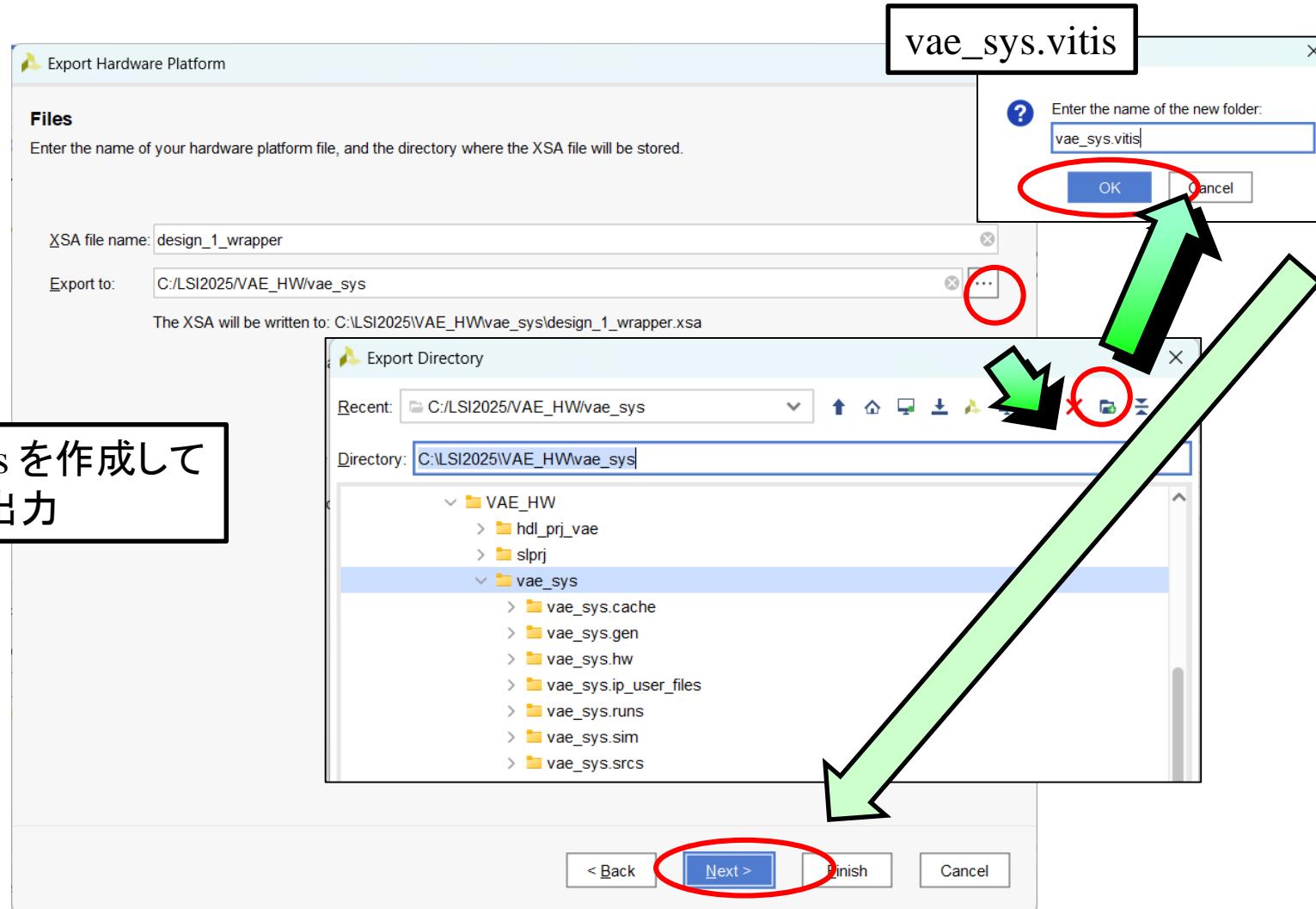


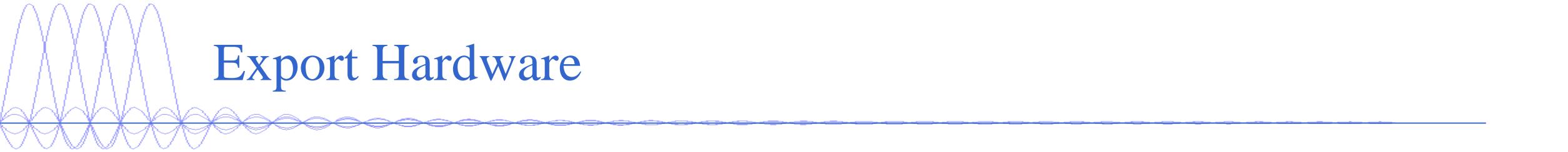
Export Hardware



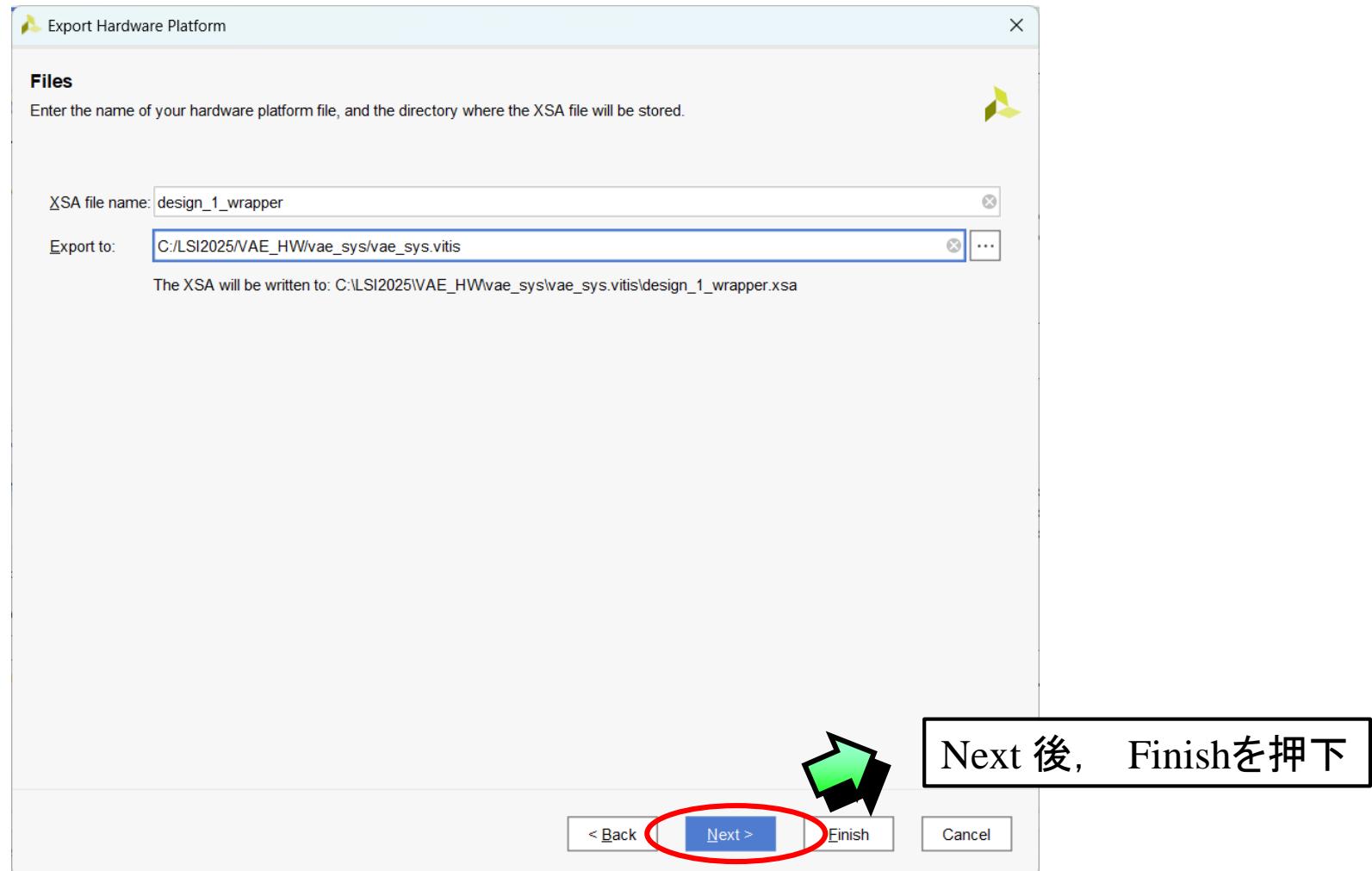
Export Hardware

vae_sys.vitis を作成して
XSA fileを出力

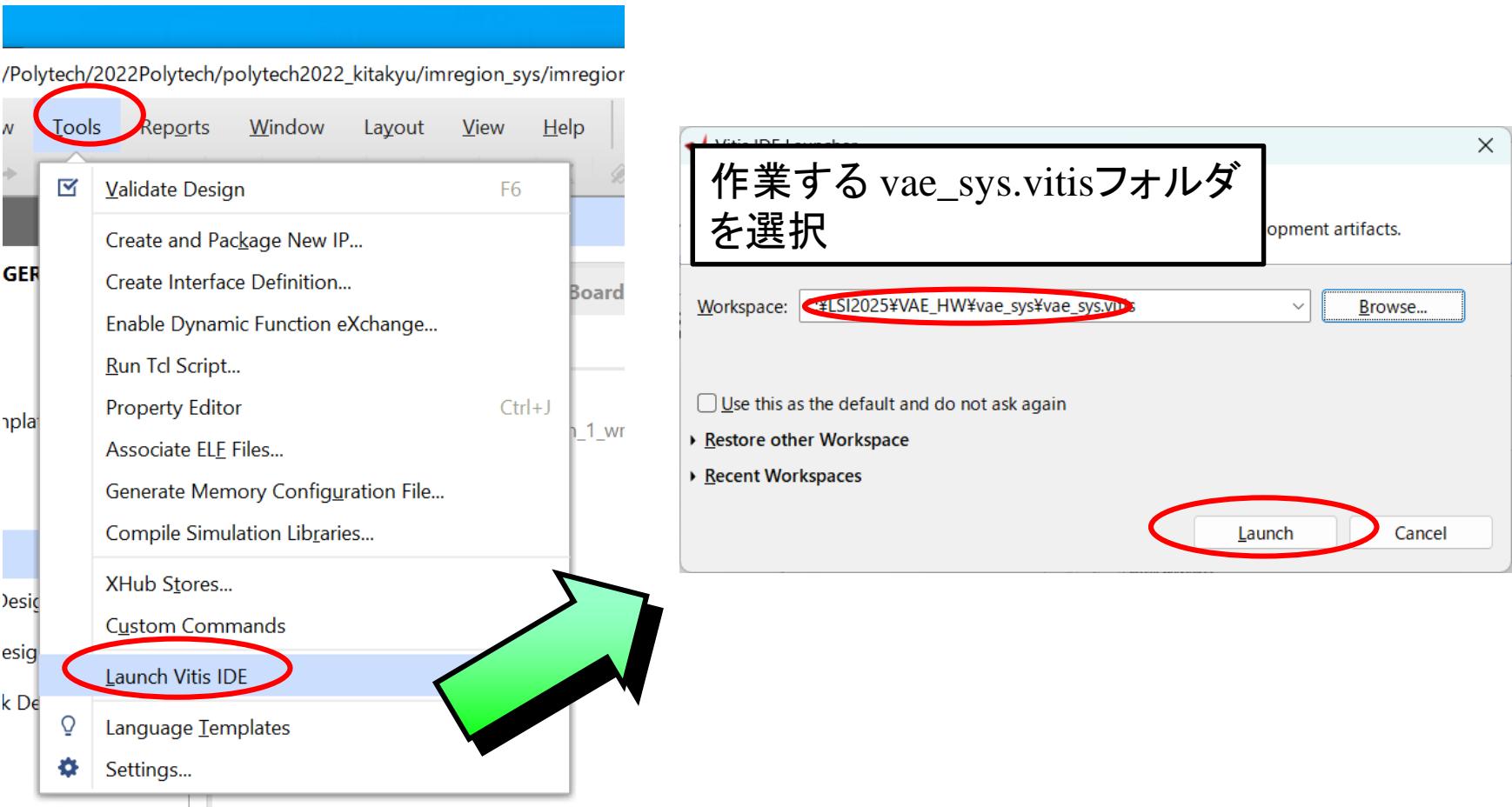


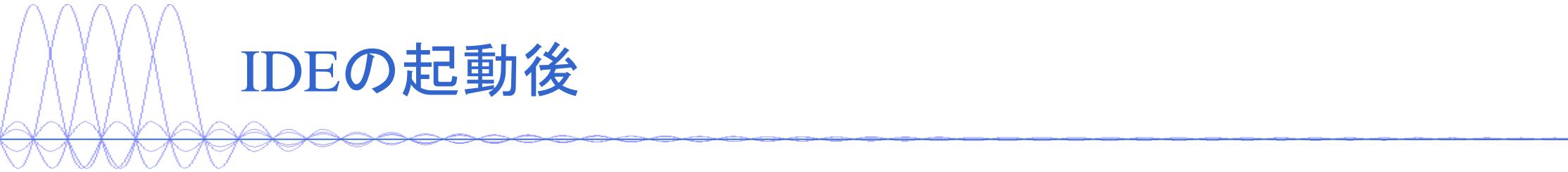


Export Hardware

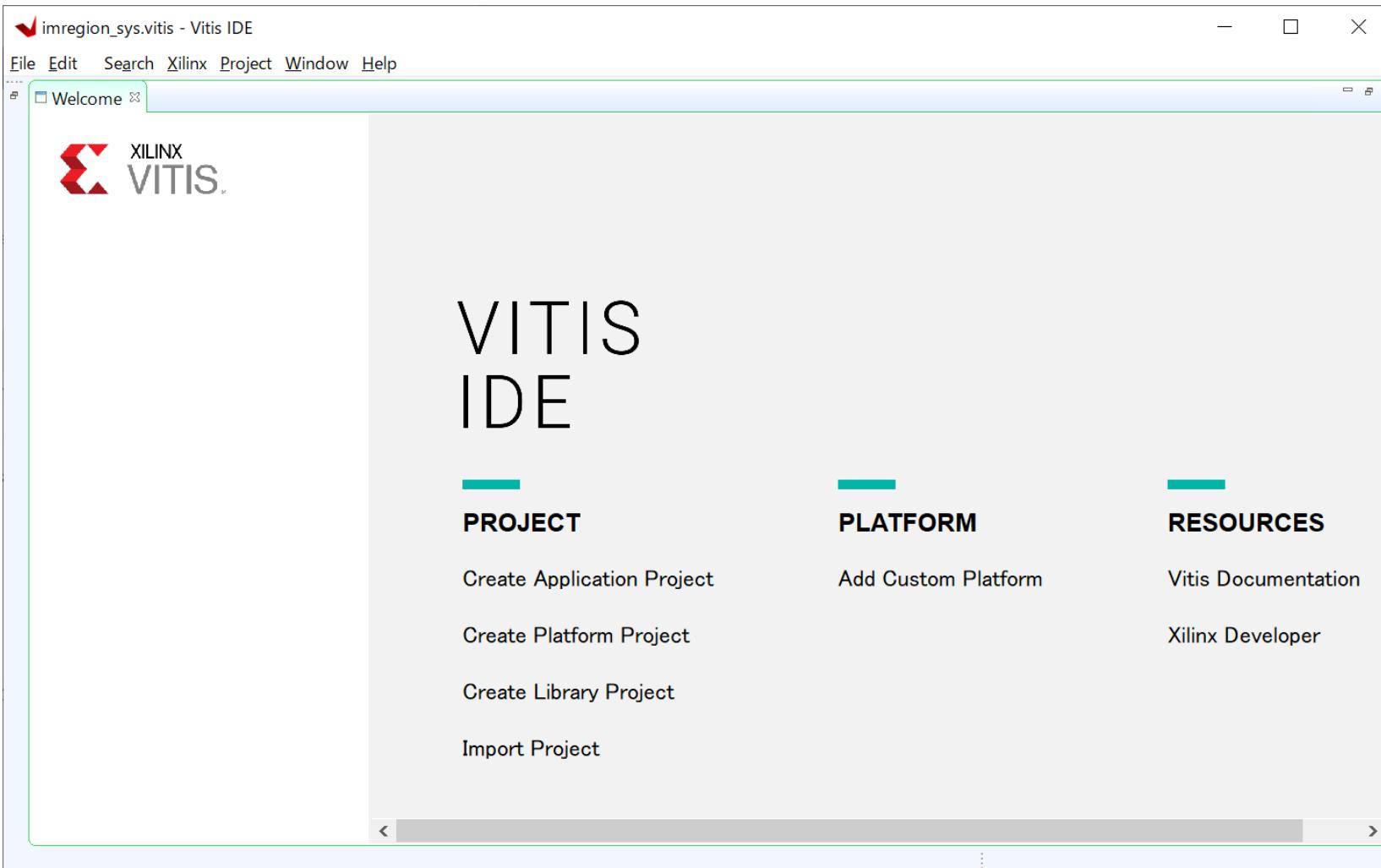


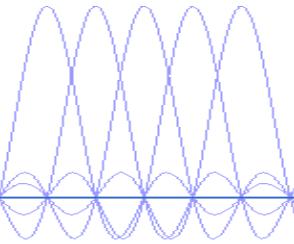
Launch Vitis IDK





IDEの起動後



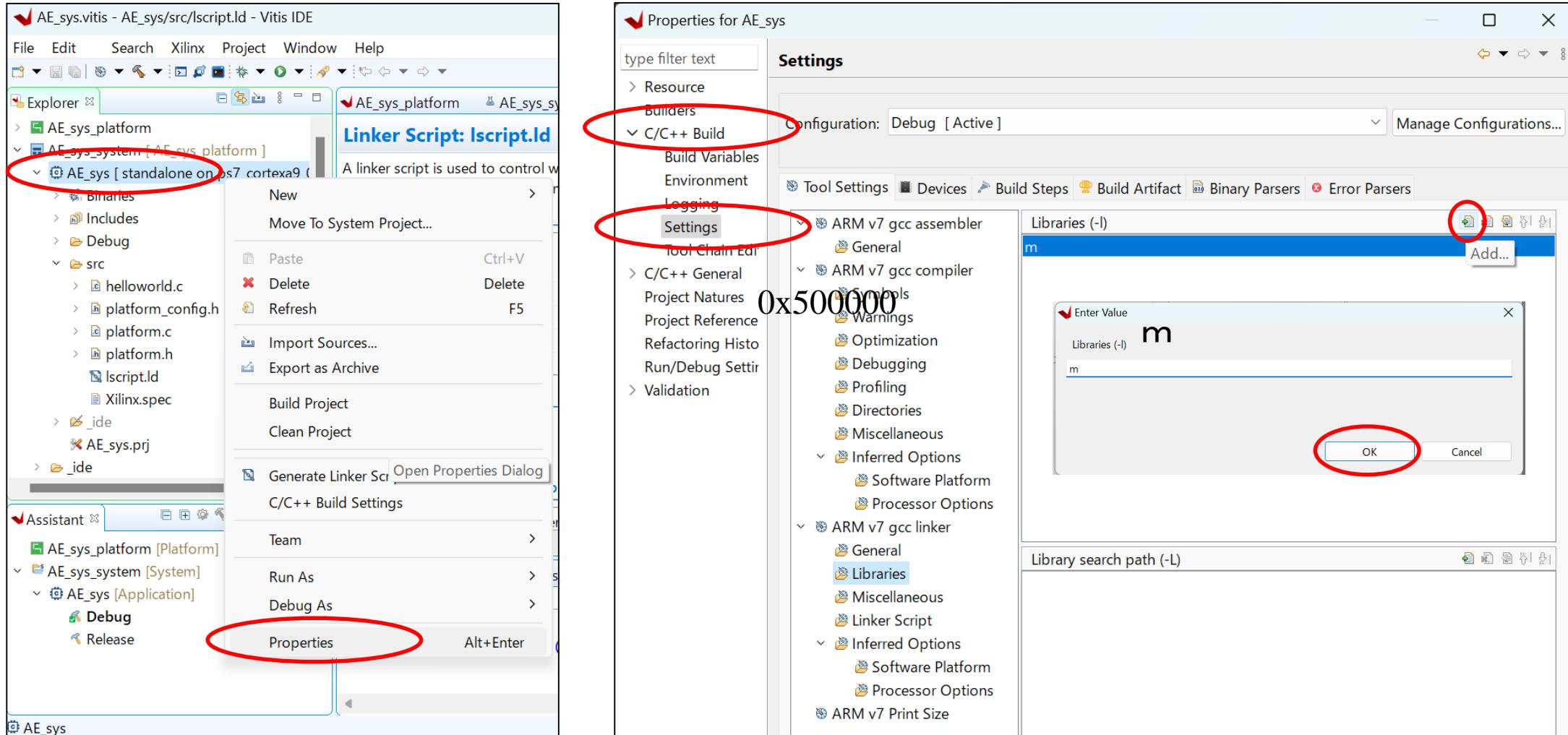


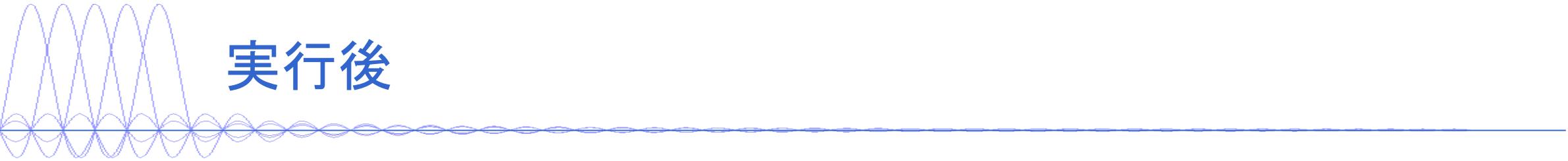
CPU Systemの作成

■ ソフトウェアの設定と実行

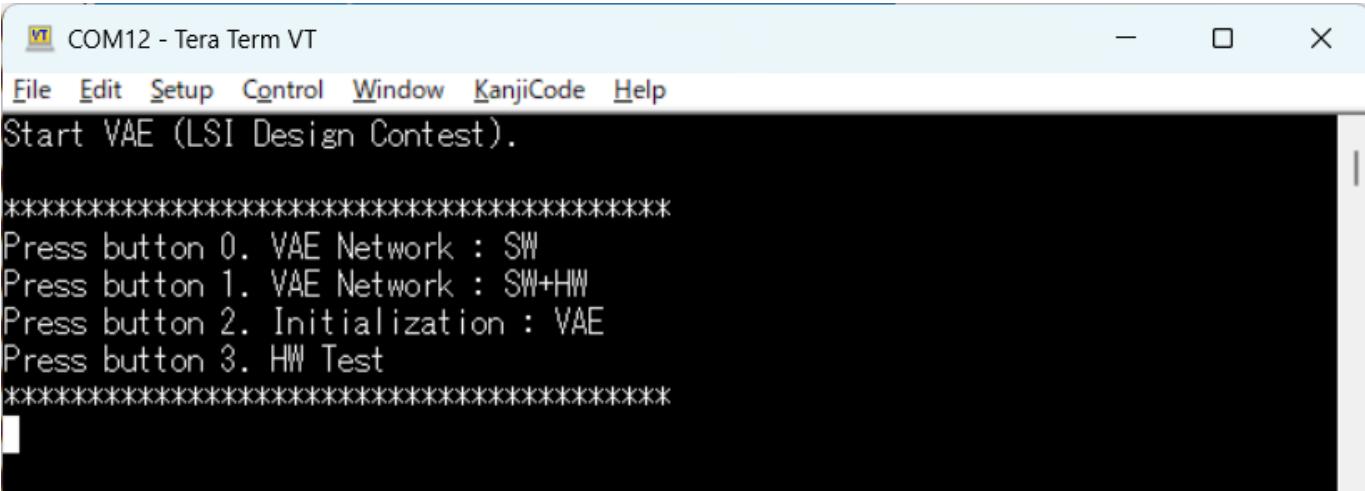
- Create Platform Project
 - Platform project name : vae_sys_platform
- SDカード利用できるために、Board Support Package を設定
 - Modify BSP Settings : xiffs
- File → New → Application Project
 - Application Project Name : vae_sys
 - SW development templates : Hello World → Finish
- 利用可能なメモリを拡大
 - src → lscript.ld の Stack Size : 0x500000, Heap Size : 0x500000 → セーブ
- 数学ライブラリの結合
 - 次ページを参照
- vae_sys.c の中身をコピー
- 実行

数学ライブラリの追加について これでmath.hが利用できる





実行後



```
COM12 - Tera Term VT
File Edit Setup Control Window KanjiCode Help
Start VAE (LSI Design Contest).

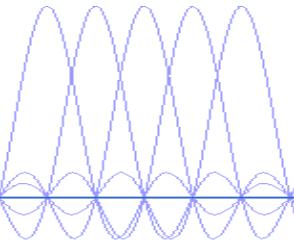
*****
Press button 0. VAE Network : SW
Press button 1. VAE Network : SW+HW
Press button 2. Initialization : VAE
Press button 3. HW Test
*****
```

使い方：

ボタン2を押下 → ボタン0を押下 (SWでの演算が可能)

ボタン2を押下 → ボタン1を押下 (SW+HWでの演算が可能)

必ずボタン2を押して一度初期化を行う



SWでの演算結果

```
tart VAE (LSI Design Contest).
```

```
*****
```

```
Press button 0. VAE Network : SW
```

```
Press button 1. VAE Network : SW+HW
```

```
Press button 2. Initialization : VAE
```

```
Press button 3. HW Test
```

```
*****
```

```
button 2 pressed
```

```
Initialization
```

```
*****
```

```
Press button 0. VAE Network : SW
```

```
Press button 1. VAE Network : SW+HW
```

```
Press button 2. Initialization : VAE
```

```
Press button 3. HW Test
```

```
*****
```

```
button 0 pressed
```

```
AE Network (SW).
```

```
Initial Weight
```

```
W2_mean
```

```
0.588014 0.188152 0.205019 0.727240 0.473846 0.019107 0.602449 0.664369 0.449151
```

```
0.699109 0.043809 0.106063 0.679400 0.448296 0.752598 0.961778 0.606630 0.225364
```

```
b2_mean
```

```
-0.500000 -0.500000
```

```
W2_var
```

```
0.670174 0.257996 0.960910 0.282165 0.797923 0.382708 0.285827 0.238987 0.883539
```

```
0.735767 0.095542 0.251767 0.768254 0.544037 0.381651 0.740268 0.437722 0.289281
```

```
b2_var
```

```
-0.500000 -0.500000
```

```
W3
```

```
0.784507 0.200527
```

```
0.758954 0.502395
```

```
0.417785 0.895382
```

```
0.225769 0.255921
```

```
0.420098 0.867232
```

```
0.064364 0.016488
```

```
0.596433 0.552497
```

```
0.837324 0.527905
```

```
0.892486 0.923350
```

```
b3
```

```
-0.500000 -0.500000 -0.500000 -0.500000 -0.500000 -0.500000 -0.500000 -0.500000 -0.500000
```

```
Epoch = 0
```

```
Epoch = 1000
```

```
Epoch = 2000
```

```
Epoch = 3000
```

```
Epoch = 4000
```

```
Epoch = 5000
```

```
Epoch = 6000
```

```
Epoch = 7000
```

```
Epoch = 8000
```

```
Epoch = 9000
```

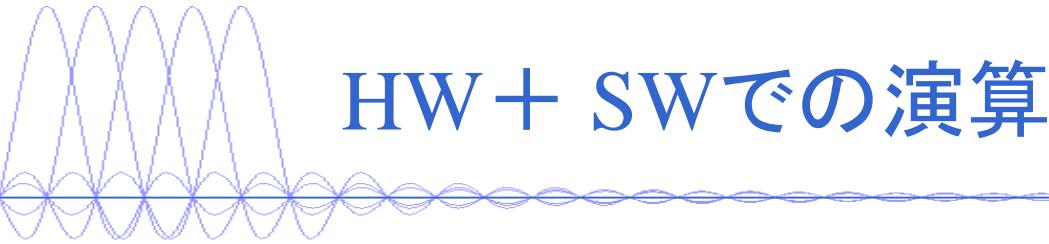
```
Final Weight
```



SWでの演算結果

```
W2_mean  
0.179194 0.234335 -0.203802 0.773423 0.018842 0.065289 0.193628 0.710551 0.040330  
0.416022 -0.480468 -0.177024 0.155124 0.689486 0.228321 0.678690 0.082353 -0.057723  
b2_mean  
-0.908821 -0.783087  
W2_var  
-0.190991 -0.006546 0.099745 0.017623 0.201300 0.118166 -0.575338 -0.025555 0.022374  
0.055935 -0.299210 -0.428065 0.373502 0.258957 -0.013101 0.060436 0.042970 -0.390551  
b2_var  
-1.361165 -1.179832  
W3  
1.258139 2.040584  
2.848507 -0.443623  
1.161464 2.359458  
2.765394 -0.547727  
-2.377854 1.513441  
2.731892 -0.637901  
1.204009 2.195866  
2.861325 -0.431063  
1.308027 2.396284  
b3  
1.464553 -0.889741 1.235160 -0.769971 -0.102778 -0.691040 1.343821 -0.906065 1.177172  
  
a2_mean : 1.084127, 0.062207,  
a2_var : 0.139167, 0.156140,  
z : 1.270903, 0.450115,  
a3 : 0.981694, 0.926270, 0.977540, 0.924004, 0.079909, 0.923706, 0.979415, 0.926653, 0.980510,  
a2_mean : -0.680629, 0.766363,  
a2_var : 0.152401, 0.180054,  
z : -0.540721, 0.977158,  
a3 : 0.941488, 0.053989, 0.948470, 0.057298, 0.934737, 0.057782, 0.944721, 0.053431, 0.943291,
```

```
Generate Image  
z = -0.500000 1.000000  
a3 =  
0.996510 0.003608 0.997340  
0.004067 0.994839 0.004125  
0.996909 0.003532 0.996766  
z = 0.000000 0.500000  
a3 =  
0.999708 0.001190 0.999762  
0.001436 0.997310 0.001507  
0.999730 0.001153 0.999698  
z = 1.000000 0.200000  
a3 =  
0.999987 0.007671 0.999986  
0.009388 0.976739 0.010119  
0.999986 0.007430 0.999984  
z = 1.200000 0.200000  
a3 =  
1.000000 0.081446 0.999999  
0.097979 0.747235 0.106832  
0.999999 0.079184 0.999999  
z = 1.400000 0.200000  
a3 =  
1.000000 0.642580 1.000000  
0.683985 0.114542 0.707343  
1.000000 0.636475 1.000000  
z = 1.500000 0.200000  
a3 =  
1.000000 0.979784 1.000000  
0.982719 0.004443 0.984658  
1.000000 0.979363 1.000000  
  
*****  
Press button 0. VAE Network : SW  
Press button 1. VAE Network : SW+HW  
Press button 2. Initialization : VAE  
Press button 3. HW Test  
*****
```

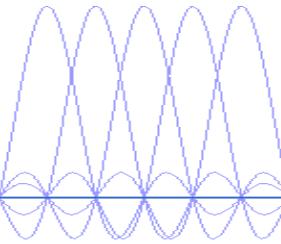


HW + SWでの演算結果

```
*****
Press button 0. VAE Network : SW
Press button 1. VAE Network : SW+HW
Press button 2. Initialization : VAE
Press button 3. HW Test
*****
button 2 pressed
Initialization

*****
Press button 0. VAE Network : SW
Press button 1. VAE Network : SW+HW
Press button 2. Initialization : VAE
Press button 3. HW Test
*****
button 1 pressed
Variational Auto Encoder (SW+HW).
Initial Weight
W2_mean
0.588014 0.188152 0.205019 0.727240 0.473846 0.019107 0.602449 0.664369 0.449151
0.699109 0.043809 0.106063 0.679400 0.448296 0.752598 0.961778 0.606630 0.225364
b2_mean
-0.500000 -0.500000
W2_var
0.670174 0.257996 0.960910 0.282165 0.797923 0.382708 0.285827 0.238987 0.883539
0.735767 0.095542 0.251767 0.768254 0.544037 0.381651 0.740268 0.437722 0.289281
b2_var
-0.500000 -0.500000
```

W3
0.784507 0.200527
0.758954 0.502395
0.417785 0.895382
0.225769 0.255921
0.420098 0.867232
0.064364 0.016488
0.596433 0.552497
0.837324 0.527905
0.892486 0.923350
b3
-0.500000 -0.500000 -0.500000 -0.500000 -0.500000 -0.500000 -0.500000 -0.500000
Epoch = 0
Epoch = 1000
Epoch = 2000
Epoch = 3000
Epoch = 4000
Epoch = 5000
Epoch = 6000
Epoch = 7000
Epoch = 8000
Epoch = 9000
Final Weight



HW + SWでの演算結果

Final Weight

W2_mean

0.177518 0.236045 -0.205478 0.775133 0.015456 0.067000 0.191952 0.712262 0.038655
0.413173 -0.478237 -0.179873 0.157355 0.684405 0.230553 0.675842 0.084584 -0.060572

b2_mean

-0.910497 -0.785936

W2_var

-0.191342 -0.007761 0.099394 0.016409 0.202164 0.116951 -0.575689 -0.026770 0.022023
0.055336 -0.297332 -0.428664 0.375380 0.256480 -0.011223 0.059837 0.044848 -0.391150

b2_var

-1.361516 -1.180431

W3

1.242990 2.041184

2.850877 -0.401960

1.140052 2.363663

2.769402 -0.502146

-2.394659 1.467228

2.737215 -0.592074

1.185725 2.198252

2.863028 -0.389662

1.288780 2.398916

b3

1.472936 -0.902572 1.239915 -0.785921 -0.081036 -0.707869 1.350480 -0.918344 1.185057

Final Weight

W2_mean

0.177518 0.236045 -0.205478 0.775133 0.015456 0.067000 0.191952 0.712262 0.038655
0.413173 -0.478237 -0.179873 0.157355 0.684405 0.230553 0.675842 0.084584 -0.060572

b2_mean

-0.910497 -0.785936

W2_var

-0.191342 -0.007761 0.099394 0.016409 0.202164 0.116951 -0.575689 -0.026770 0.022023
0.055336 -0.297332 -0.428664 0.375380 0.256480 -0.011223 0.059837 0.044848 -0.391150

b2_var

-1.361516 -1.180431

W3

1.242990 2.041184

2.850877 -0.401960

1.140052 2.363663

2.769402 -0.502146

-2.394659 1.467228

2.737215 -0.592074

1.185725 2.198252

2.863028 -0.389662

1.288780 2.398916

b3

1.472936 -0.902572 1.239915 -0.785921 -0.081036 -0.707869 1.350480 -0.918344 1.185057

a2_mean : 1.082535, 0.056870,

a2_var : 0.138311, 0.156798,

z : 1.261207, 0.414187,

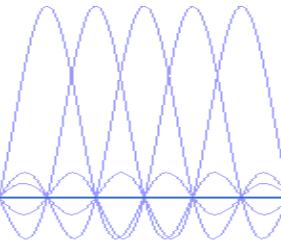
a3 : 0.979660, 0.924149, 0.975571, 0.924149, 0.075851, 0.924149, 0.977020, 0.924149, 0.978378,

a2_mean : -0.692429, 0.747025,

a2_var : 0.152275, 0.179156,

z : -0.486223, 1.131432,

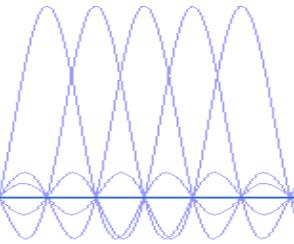
a3 : 0.960358, 0.060089, 0.966919, 0.063721, 0.939911, 0.063721, 0.962677, 0.060089, 0.962677,



HW + SWでの演算結果

```
Generate Image
z = -0.500000 1.000000
a3 =
0.997717 0.004152 0.998355
0.004678 0.995196 0.004697
0.997987 0.004113 0.997954
z = 0.000000 0.500000
a3 =
0.999811 0.001381 0.999854
0.001663 0.997493 0.001726
0.999826 0.001355 0.999811
z = 1.000000 0.200000
a3 =
0.999992 0.008875 0.999992
0.010834 0.978201 0.011555
0.999991 0.008698 0.999990
z = 1.200000 0.200000
a3 =
1.000000 0.093002 1.000000
0.111325 0.758170 0.120188
1.000000 0.091406 1.000000
z = 1.400000 0.200000
a3 =
1.000000 0.674968 1.000000
0.713723 0.119472 0.734030
1.000000 0.671588 1.000000
z = 1.500000 0.200000
a3 =
1.000000 0.982434 1.000000
0.984951 0.004600 0.986543
1.000000 0.982253 1.000000
```

```
*****
Press button 0. VAE Network : SW
Press button 1. VAE Network : SW+HW
Press button 2. Initialization : VAE
Press button 3. HW Test
*****
```



応用実習3: HW/SW協調実装

- ○や×以外の信号を入れたシステムを構築
 - △や1など
 - 学習はmファイルで行い、その係数のみをC言語に入れる

