SWEST21 セッションs5b

組込みAI技術の最前線

～AISingのEdge向けAIアルゴリズムの紹介と、
FPGAによるAI実装 LUT-Networkの開発記

名古屋大学

山本 椋太

アジェンダ

- 自己紹介
- FPGA × 深層学習
- LUT-Network とは
- 実際にやってみた
- 色々な比較

自己紹介

- 名前: 山本 椋太 (やまもと りょうた)

- 所属: 名古屋大学大学院
  情報学研究科
  博士後期課程 3年

- 研究内容:
  - 高位合成による機械学習フレームワークの開発
  - C言語から状態遷移表をリバースで抽出する
    ツールの開発
    - たまに，展示会などで講演させていただいています.
  - 要求仕様書から曖昧さを抽出する研究

深層学習

- 深層学習は様々な分野から注目を集めている.
  - もちろん，組込みでも！

- 組込みで動かそうと思うと，大変…
  - メモリが足りない！
    - MB・GBオーダーの訓練済みデータが必要になることもある
  - 電力を抑えたい！
    - (最近は省電力GPUデバイスもあるが) GPUは電力が大きい.
  - 速度がほしい！
    - リアルタイム性が要求されるシステムもある.

組込み × 深層学習

色々なデバイスで，DNNを動かす取り組みが盛んである.

- Jetson (GPU, Nvidia社)
  - 最近では，IoT向けに
    Jetson nano が登場した.

- PYNQ (SoC, Xilinx社)
  - BNN-PYNQ (FINN) とともに,
    Pythonを用いて
    簡単にDNNができる.
  - Weight や Activation を
    量子化 (1bit / 2bit)

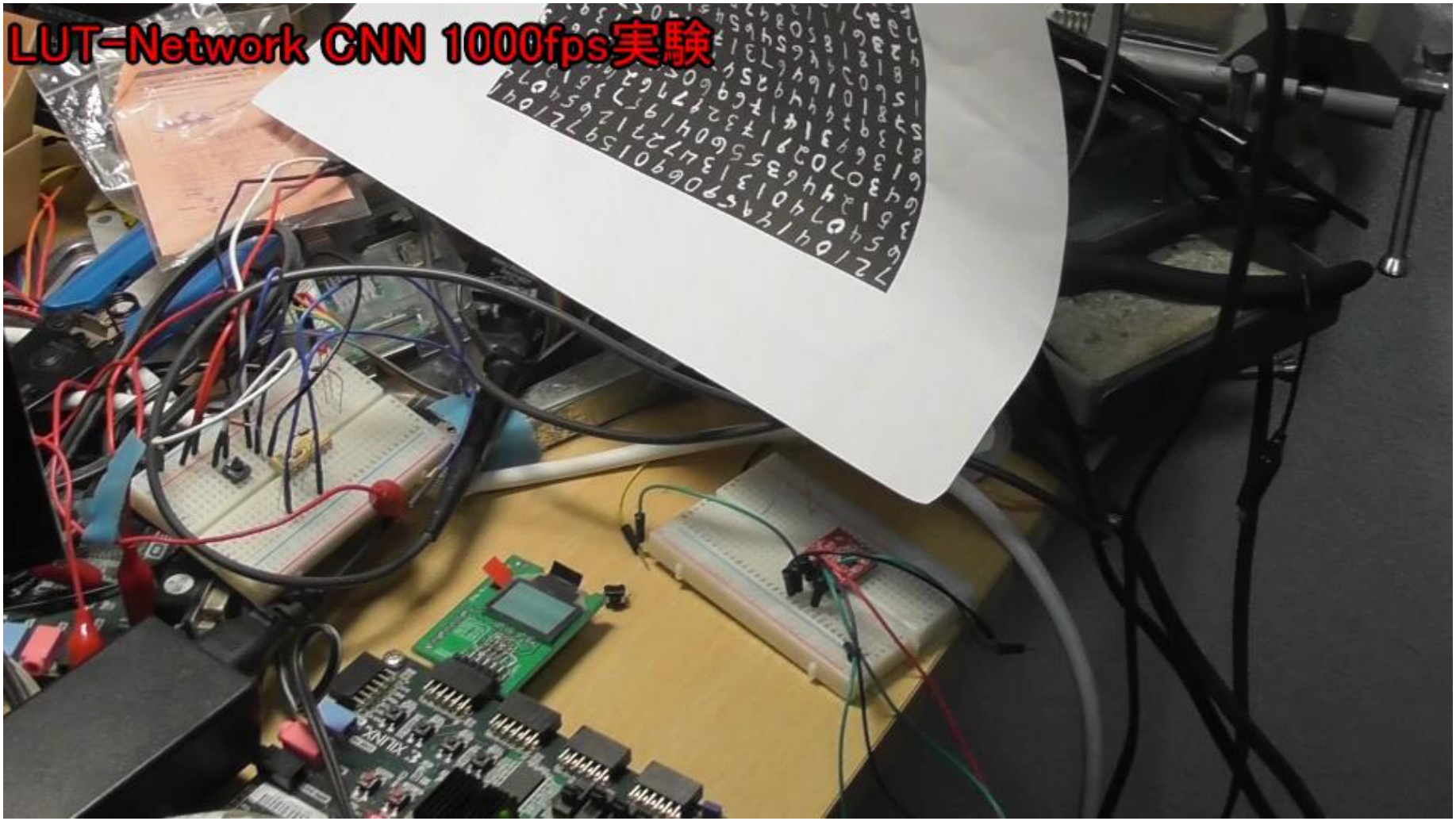
FPGA × 深層学習！

- FPGA向けHDLの設計は専門性が高いため，
  様々なフレームワークが研究・開発されている．
  - GUINNESS (GUI based Nerural Network)
    - 東工大 中原先生らによって開発されている．
    - Weight / Activation は 1bit に量子化される．
    - https://github.com/HirokiNakahara/GUINNESS
  - BNN-PYNQ
    - Xilinx社によって開発されている．
    - Weight / Activation は 1bit または 2bit に量子化される．
    - https://github.com/Xilinx/BNN-PYNQ
- 今回は，LUT-Networkを使ってみよう！

LUT-Network 用の環境 **BinaryBrain**

- BinaryBrain とは
  - 渕上氏 (@Ryuz88) による LUT(Look-up Table)-Network用の学習・推論を行う環境
  - (2値化なので，精度の悪さは認識して始めた)
- 特徴
  - バイナリ入力・多値出力
  - ニューラルネットワークのFPGA化
  - バイナリネットだが変調技術により回帰分析が可能
  - 独自の確率的LUTのモデルにより，高速に学習
  - C++で記述
  - GPU(CUDA)に対応
  - 高速かつ微小リソースなFPGAアクセラレータを生成
    - MNISTコア単体

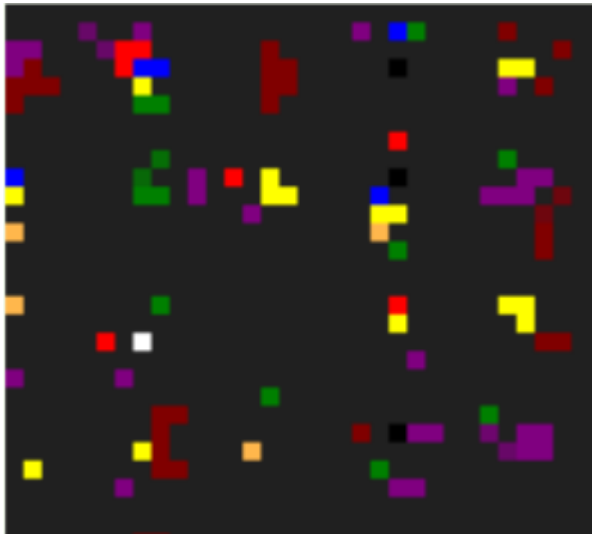
動いている様子: 動画



動いている様子: 出力例

- MNIST (手書きの数字 0から9)
- 正しい位置で画像を読み込んだ場合のみ正しい結果を得られている
- 出力結果の色は抵抗のカラーコードと対応している

入力画像



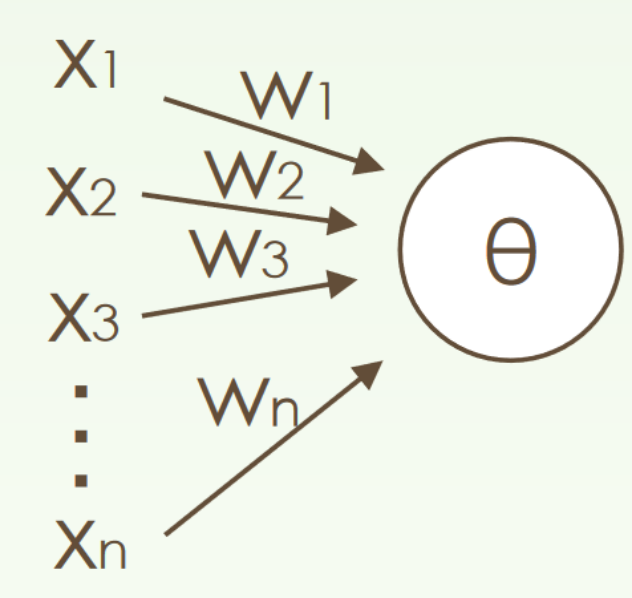
出力画像

通常のDNN と LUT-Network の違い (1/2)

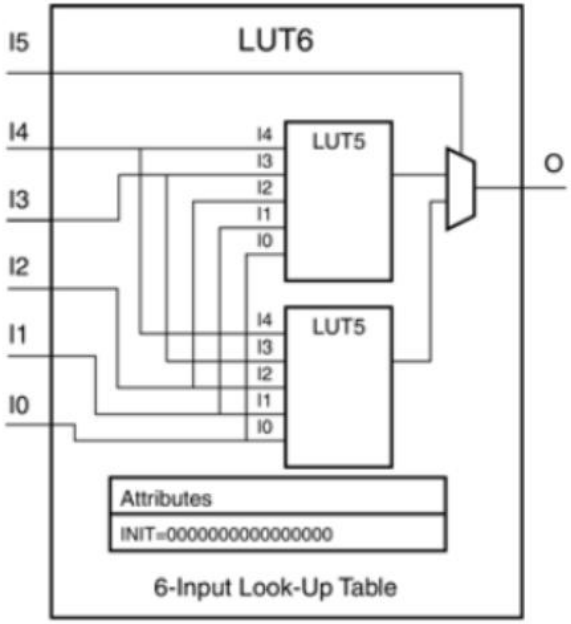
ニューロン を使うか LUT を使うか

- ニューロンでは，$\theta = w_1x_1 + w_2x_2 + \cdots + w_nx_n$

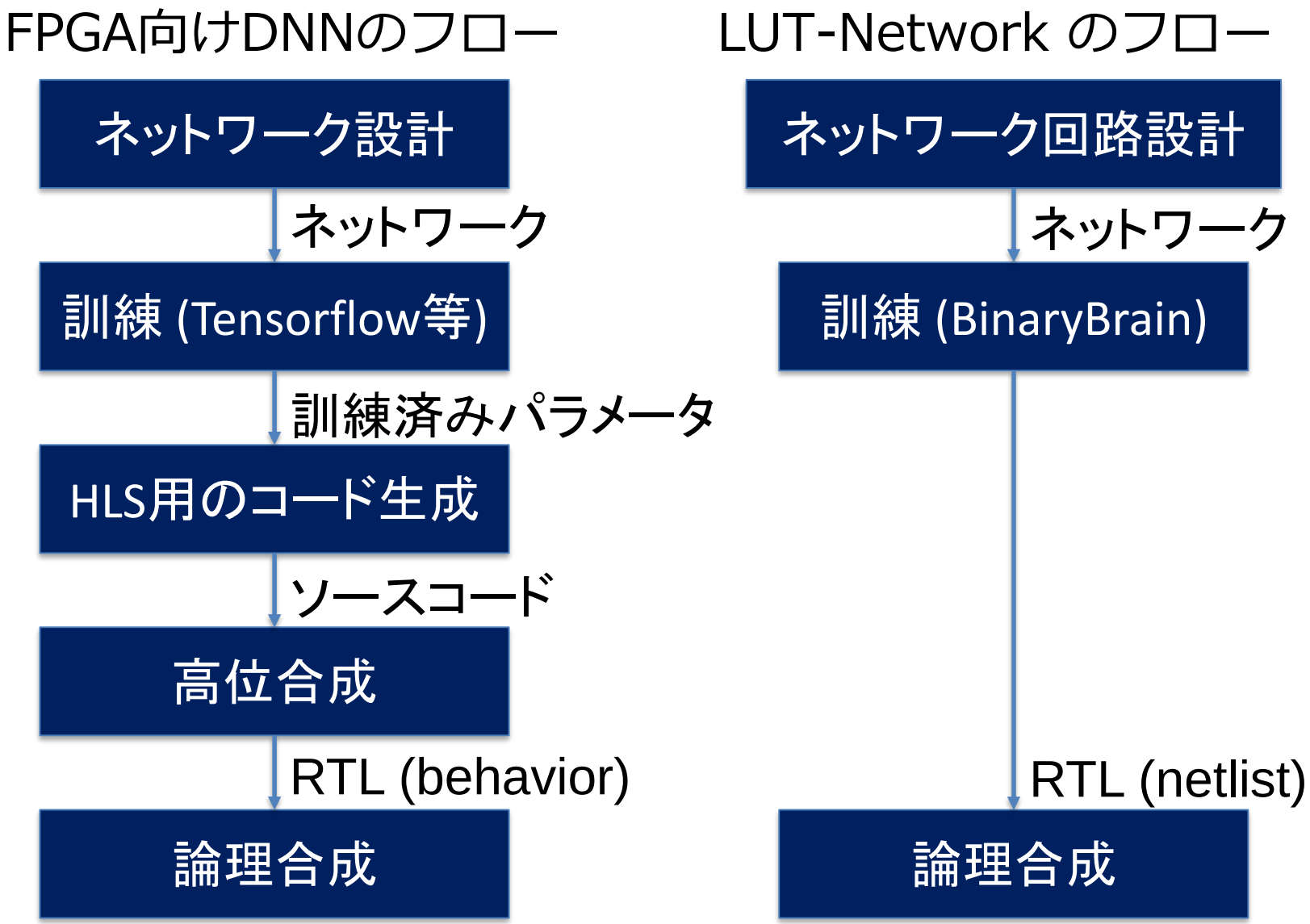
- LUTは，特定の入力パターンに対して割り当てられた出力結果を渡すだけ



Primitive: 6-Input Lookup Table with General Output



通常のDNN と LUT-Network の違い (2/2)



テーブル化

- 入力の組み合わせ全てに対して、計算結果を表にする
- テーブル化の例

  例) バイナリネットワークは各レイヤの入出力が２値
  - 32bit入力のパターン数は，4Gである.
  - 出力も32bitだとすれば，4G × 32 bit = 128 Gbitのテーブルがあれば良い.

$f(x) = x^2$ (ただし，xは0から3の整数)
→ x に値を代入して計算せず，
対応表から答えを引く.

入力	出力
0	0
1	1
2	4
3	9

バイナリ変調

- DNN部分は入力から出力まですべてバイナリ

- DNNの入力前にオーバーサンプリングでバイナリ変調して，２値化した値を入力
  - 浮動小数点の入力をオーバーサンプリングしつつPWM変調など施したバイナリに変換するクラス
  - BinaryToRealクラスというオーバーサンプリングされたバイナリを数えて浮動小数点に戻すクラス
- 回帰問題を解けるようになる！
  https://www.slideshare.net/HirokiNakahara1/fpgax2019/16

- 出口でまた積算して、多値に戻す
  - 一般の1bit ADCなどでも行われている方式

確率的LUTモデル

Stochastic 演算による乗算

- 例えばビット長 10 のビット列A が以下を考える.

  A = 00 0110 0111

- 1の出現回数は5回であるので，0.5を表す.
- ビット列B を考える.

  B = 00 0101 1100

- 1の出現回数は2回であるので，0.4を表す.
- ここで，A×Bは，A と B の論理積から求められる.
  - A and B = 00 0100 0100
  - このとき1の出現確率は 0.2 であり，乗算できている.
  - 誤差が出るため，高精度が要求される場合には注意

実際にやってみた

- Github から clone する
  - https://github.com/ryuz/BinaryBrain

- 基本は書いてある手順通りにすすめる.

- ただし，今回試した環境が RTX2080Tiで，
  CUDAが想定されているバージョンと異なるため，
  - g++ のバージョン問題
  - CUDAのバージョン問題
- に対応する必要があった.

- ツールインストールは割愛

学習の様子 (1/2)

```
    ---------------------------------------------------------------
    [Sequential]
      ---------------------------------------------------------------
      [StochasticLut6]
       input  shape : {3, 3, 64}       output shape : {256}
      ---------------------------------------------------------------
      [StochasticLut6]
       input  shape : {256}       output shape : {64}
    ---------------------------------------------------------------
    [ConvolutionCol2Im]
     input  shape : {64}     output shape : {8, 8, 64}
  ---------------------------------------------------------------
  [StochasticMaxPooling2x2]
   filter size : (2, 2)
   input  shape : {8, 8, 64}   output shape : {4, 4, 64}
  ---------------------------------------------------------------
  [StochasticLut6]
   input  shape : {4, 4, 64}   output shape : {1024}
  ---------------------------------------------------------------
  [StochasticLut6]
   input  shape : {1024}   output shape : {360}
  ---------------------------------------------------------------
  [StochasticLut6]
   input  shape : {360}   output shape : {60}
  ---------------------------------------------------------------
  [StochasticLut6]
   input  shape : {60}   output shape : {10}
---------------------------------------------------------------
----------------------------------
epoch_size            : 8
mini_batch_size       : 32
lut_binarize          : 1
file_read             : 0
----------------------------------
fitting start : MnistStochasticLutCnn
[4% (2880/60000)]  loss : 2.177  accuracy : 0.269792
```

学習の様子 (2/2)

```
fitting start : MnistStochasticLutCnn
    122.93s epoch[  1] test accuracy : 0.7233 train accuracy : 0.7190
    359.22s epoch[  2] test accuracy : 0.8250 train accuracy : 0.8192
    596.50s epoch[  3] test accuracy : 0.7637 train accuracy : 0.7568
    833.67s epoch[  4] test accuracy : 0.8130 train accuracy : 0.8123
   1070.93s epoch[  5] test accuracy : 0.8254 train accuracy : 0.8182
   1308.10s epoch[  6] test accuracy : 0.8688 train accuracy : 0.8637
   1545.35s epoch[  7] test accuracy : 0.8098 train accuracy : 0.8045
   1782.51s epoch[  8] test accuracy : 0.8212 train accuracy : 0.8158
fitting end
```

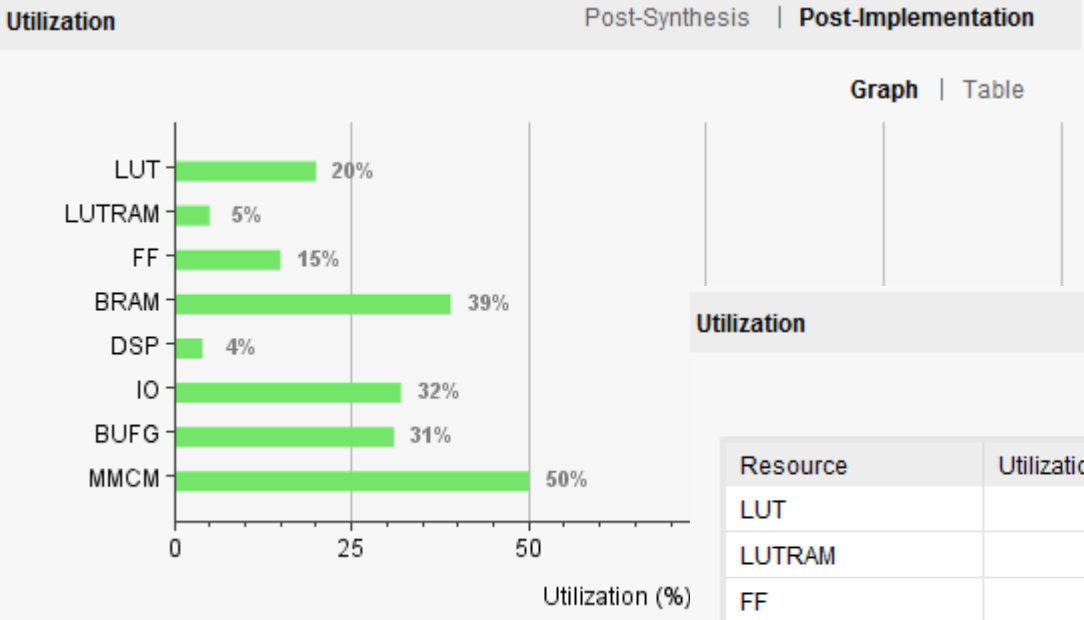
実行

- シミュレーションはできており，実機もカメラを購入すれば，動作させられる.
  - シミュレーションはVivado シミュレータを使用した
  - シミュレーションしてくれるスクリプトも用意されている
  - vsdファイルが生成されているため，波形ビューワをインストールして確認できる

- ここまでのフローは，非常に簡単である.

性能

- ネットワーク構成は7段
- 遅延も7サイクルのみで，LUT使用数は 1182個



Utilization　Post-Synthesis | **Post-Implementation**

Graph | **Table**

Resource	Utilization	Available	Utilization %
LUT	10858	53200	20.41
LUTRAM	833	17400	4.79
FF	16236	106400	15.26
BRAM	54	140	38.57
DSP	9	220	4.09
IO	40	125	32.00
BUFG	10	32	31.25
MMCM	2	4	50.00

今後

- Intel (旧Altera) 対応
- シミュレーションはできており，実機もカメラを購入すれば，動作させられる.
- ここまでのフローは，非常に簡単である.

今後の発表予定

- 2019 IEEE 9th International Conference on Consumer Electronics (ICCE-Berlin)
  - Sep. 8-11

- Fast and Light-weight Binarized Neural Network Implemented in an FPGA using LUT-based Signal Processing and its Time-domain Extension for Multi-bit Processing

- Ryuji Fuchikami ((Personal), Japan); Fumio Issiki (Finekit, Japan)

- https://edas.info/p25749#S1569571697