教育・研究報告

PYNQ-Z1ボードを用いた深層学習 BNN-PYNQ の Microsoft Windows 10用開発環境の構築

総合情報学部教授　桑　門　秀　典

1．はじめに

ニューラル・ネットワークを用いた画像認識は、その高い認識率から実用性が高い技術として注目されている。従来、ニューラル・ネットワークによる画像認識には、浮動小数点による実数演算を多用していたが、1 ビットまたは 2 ビットに量子化した値を用いた binarized neural network（BNN）でも高い認識率が実現できることが示された[1)]。BNN の処理は、二値演算が主となるため、プログラマブルロジックなどのハードウェア実装に向いている。プログラマブルロジック（FPGA）とマイクロプロセッサを組み合わせたオンチップシステム上に BNN を実装することができれば、小型化、低消費電力、リアルタイム性が求められる IoT デバイスで、ニューラル・ネットワークによる画像認識を実現することができる。

PYNQ は、Xilinx 社のオンチップシステム Zynq を使用した組み込みシステム開発用オープンソースプロジェクトである[2)]。PYNQ の特徴は、ブラウザ上で動作する対話型実行環境 Jupyter Notebook からプログラミング言語 Python を用いて、プログラマブルロジックとマイクロプロセッサを制御できることである。PYNQ は、Zynq を搭載した 4 種類のボードを公式にサポートしている。これらのボードのプログラマブルロジック部分の開発には、Xilinx 社 Vivado Design Suite を用いる。Vivado Design Suite には、Vivado HL Design Edition, Vivado HL System Edition, Vivado HL WebPACK の 3 種類があり、無償の Vivado HL WebPACK でこれらのボード上の開発が可能である。Vivado HL WebPACK には、プログラミング言語 C/C++ のソースコードから Verilog などのハードウェア記述言語に高位合成を行う Vivado HLS とハードウェア記述言語から論理合成・配置配線を行う Vivado の二つのツールが含まれている[3)]。

Xilinx 社は、PYNQ で動作する BNN である BNN-PYNQ を GitHub で公開している[4)]。BNN-PYNQ は、Python パッケージ管理コマンド pip で GitHub からインストールでき、機械学習のベンチマークに使われる CIFAR-10[5)]や MNIST[6)]を用いたデモも含まれている。公式サイトによれば、BNN-PYNQ は Linux 上の Vivado 2018.2でテストされているので、BNN-PYNQ の開発を行うためには、Linux の PC を用意するか、または Microsoft Windows 上に仮想 Linux マシンを用意する必要がある。

しかし、Vivado HL WebPACK はCPU やメモリ等の計算機資源を多く必要とするソフトウェアなので、仮想マシン上で Vivado HL WebPACK を実行すると、計算機資源が不足し、開発効率の低下や操作性の悪化の可能性がある。授業・実習などで PYNQ を利用する場合、Microsoft Windows の PC のシェアが大きいこと、仮想マシンを作成する時間を省くことの理由から、Microsoft Windows 上で BNN-PYNQ の開発ができた方が良いと考え、本稿では BNN-PYNQ の Microsoft Windows 10への移植を行う。

2．準備

本稿では、PYNQ がサポートしている Digilent 社の PYNQ-Z1 ボードを用いた[7]。PYNQ の公式サイトの PYNQ-Z1 Setup Guide にしたがって、イメージファイル PYNQ-Z1 v2.5 PYNQ Image を書き込んだ microSD カードから PYNQ-Z1 を起動した。PYNQ-Z1 への給電は、microUSB Type-B ケーブルで可能だが、安定した給電のために、出力12V 2 A のスイッチング AC アダプタを使用した。PYNQ-Z1 をイーサネットケーブルで LAN に接続し、PYNQ-Z1 上で起動している Jupyter Notebook に Microsoft Windows 10上の Web ブラウザでアクセスできることを確認した。Python パッケージ管理コマンド pip を用いて、BNN-PYNQ を公式サイト[4]から PYNQ-Z1 の microSD カードにインストールした後、PYNQ-Z1 上で動作している Jupyter Notebook に Web ブラウザでアクセスし、付属のデモプログラムの動作を確認した。その後、プログラマブルロジックのプログラミング情報（bitstream）の生成に必要なソースコードを公式サイト[4]から、バージョン管理コマンド git[8]を用いて、PYNQ-Z1 の microSD カードにインストールした。このソースコードは、Linux 上の Vivado 2018.2 でテストされている（Linux ディストリビューションは明記されていない）。

BNN-PYNQ の実体は、プログラミング言語 C++ のヘッダファイルのみで実装されている深層学習フレームワーク tiny-cnn[9]を修正した xilinx-tiny-cnn である。xilinx-tiny-cnn をコンパイルするために、他のパッケージは不要である。xilinx-tiny-cnn の C++ のソースコードを Vivado HLS で Verilog などのハードウェア記述言語に高位合成した後、Vivado で論理合成・配置配線を行い、プログラマブルロジック（FPGA）の回路情報である bitstream を生成する。BNN-PYNQ には、高位合成と論理合成・配置配線を行う Bash スクリプト make-hw.sh 及び make-hw.sh から起動された Vivado HLS が実行する Tcl スクリプト hls-syn.tcl と Vivado が実行する Tcl スクリプトが用意されている。したがって、Linux 上の Vivado HL WebPACK 2018.2を用いる場合、make-hw.sh を実行するだけで、bitstream を生成できる。

3．BNN-PYNQ の bitstream の生成に必要な修正

本稿では、Ubuntu 16.04, Ubuntu 18.04, Microsoft Windows 10 上の Vivado HL

WebPACK 2018.2, Vivado HL WebPACK 2019.2を用いる。Ubuntu 16.04上の Vivado HL WebPACK 2018.2を用いる場合、修正は不要だが、それ以外の場合には、本節で述べる修正が必要である。本節で述べる修正は、高位合成に関係する修正であり、論理合成・配置配線には直接は関係しない。

3.1 bitstream 生成用スクリプト make-hw.sh と hls-syn.tcl の修正

Ubuntu 16.04, Ubuntu 18.04の場合、Bash スクリプト make-hw.sh を実行する Bash のシェル環境があるが、Microsoft Windows 10の場合、Bash スクリプトを実行するシェル環境が必要である。文献[10]では、Bash スクリプト make-hw.sh を PowerShell スクリプトに移植する検討をしている。本稿では、Microsoft Windows 10用の分散型バージョン管理システム Git for Windows に付属するシェル環境 Git Bash[8]を用いることにした。Git Bash には、make-hw.sh 内で使われている UNIX コマンドが用意されているので、make-hw.sh の修正箇所が少なくてすむからである。具体的には、Git Bash で make-hw.sh が実行できるように、make-hw.sh を以下のように修正する。

⑴ Vivado HL WebPACK のコマンド vivado_hls.bat, vivado.bat へのパスを設定する。Git Bash の which コマンドでは、これらバッチファイルへのパスが取得できないためである。

⑵ vivado_hls.bat の引数に含まれるパスの区切り文字には、Windows 形式の￥の代わりに／を用いる。これは、Tcl スクリプト hls-syn.tcl では、パス表記に／を用いるためである。

⑶ make-hw.sh で使われる mkdir コマンド、cut コマンドは Git Bash のコマンドを用いるようにし、Vivado HL WebPACK 2018.2の SDK に含まれるこれらのコマンドを使わないようにパスを設定する。make-hw.sh では、SDK に含まれるこれらのコマンドがサポートしていないオプションを使用するためである。

⑷ Vivado HL WebPACK 2018.2では、C/C++ コンパイラとして clang と gcc/g++ をサポートしているが、Vivado HL WebPACK 2019.2では、gcc/g++ のみをサポートしている。

hls-syn.tcl では、C/C++ コンパイラとして clang を明示的に指定しているので、Vivado HL WebPACK 2019.2を使う場合は、その部分を削除する。なお、C/C++ コンパイラは Vivado HL WebPACK に付属するので、別途インストールする必要はない。

3.2 Microsoft Windows 用 Vivado HL WebPACK の修正

Microsoft Windows 10用の Vivado HL WebPACK の場合、2018.2, 2019.2ともに下記の修正が必要である。この修正は、文献[11]と文献[12]でも指摘されている。

win64/tools/clang/include/c++/4.5.2/exception_ptr.h: 132行目
（修正前） const type_info*
（修正後） const class type_info*
win64/tools/clang/include/c++/4.5.2/nested_exception.h: 110行目
（修正前） __throw_with_nested（_Ex&&, const nested_exception*＝0）
（修正後） __throw_with_nested（_Ex&&, const nested_exception*）
win64/tools/clang/include/c++/4.5.2/nested_exception.h: 122行目
（修正前） __throw_with_nested（_Ex&&__ex, const nested_exception*＝0）
（修正後） __throw_with_nested（_Ex&&__ex, const nested_exception*）

3.3 深層学習フレームワーク xilinx-tiny-cnn の修正

Microsoft Windows 10用の Vivado HL WebPACK の場合、2018.2, 2019.2ともに下記の修正が必要である。

BNN-PYNQ/bnn/src/xilinx-tiny-cnn/tiny_cnn/util/aligned_allocator.h: 131行目
（修正前）::free（ptr);
（修正後）_mm_free（ptr);
BNN-PYNQ/bnn/src/library/finn-hlslib/bnn-library.h: 51行目
（追　記）#include<cstdlib>

4．生成した回路情報 bitstream の検証

BNN-PYNQ では、5 種類のネットワーク（cnvW1A1, cnvW1A2, cnvW2A2, lfcW1A1, lfcW1A2）の bitstream を生成する。Vivado HL WebPACK のバージョン、OS の種類、高位合成時の周期に応じた 6 種類の開発環境で上述の 5 種類のネットワークの bitstream を生成した。

4.1 周期の制約について

bitstream を生成するスクリプト make-hw.sh では、高位合成時の周期（clock）を 5 ns、クロック周期のばらつきの度合いを表す Uncertainty を12.5%に設定している。したがって、5 ns にした場合には、高位合成時の周期制約は実質4.575ns になる。しかし、表 1 に示すように、高位合成時の見積もりでは、Vivado HL WebPACK や OS によらず、この制約を満たすことができなかった。高位合成時の周期を10ns、Uncertainty を12.5%にした場合、高位合成時の周期の制約は実質8.75ns になるが、この制約も満たせない場合があった。

表 1　高位合成終了時の見積もり周期

Vivado HL WebPACK	高位合成時 Clock [ns]	OS	cnvW1A1 [ns]	cnvW1A2 [ns]	cnvW2A2 [ns]	lfcW1A1 [ns]	lfcW1A2 [ns]
2018.2	5	Ubuntu 16.04	9.883	9.883	8.611	12.514	12.514
		Microsoft Windows 10	9.883	9.883	8.611	12.514	12.514
	10	Ubuntu 16.04	9.883	9.883	8.750	12.514	12.514
		Microsoft Windows 10	9.883	9.883	8.750	12.514	12.514
2019.2	5	Ubuntu 18.04	8.101	8.101	8.205	8.611	8.247
		Microsoft Windows 10	8.101	8.101	8.205	8.611	8.247
	10	Ubuntu 18.04	8.750	8.750	8.750	8.750	8.750
		Microsoft Windows 10	8.750	8.750	8.750	8.750	8.750

一方、論理合成・配置配線時の周期は、Vivado が読み込む Tcl スクリプト make-vivado-proj.tcl によれば、10ns（周波数100MHz）である。Vivado が出力した論理合成・配置配線後の最大遅延解析におけるタイミングパスの最悪スラック（Worst Negative Slack: WNS）を表 2 に示す。表 2 において、プラスの値は制約の10ns に対して、それだけの余裕時間があることを意味する。高位合成時に周期制約を満たすことができなくても、論理合成・配置配線の最適化の結果、10ns の制約を満たすことができる場合がある。WNS がマイナスの値は 10ns よりその絶対値の時間だけ超えていることを意味する。マイナスの値の場合、配置配線ができても、その回路は安定して動作しない可能性がある。

表 2　論理合成・配置配線後の最悪スラック

Vivado HL WebPACK	高位合成時 Clock [ns]	OS	cnvW1A1 [ns]	cnvW1A2 [ns]	cnvW2A2 [ns]	lfcW1A1 [ns]	lfcW1A2 [ns]
2018.2	5	Ubuntu 16.04	0.164	−0.607	0.028	0.265	0.273
		Microsoft Windows 10	0.164	−0.607	0.028	0.265	0.273
	10	Ubuntu 16.04	0.113	0.185	0.190	0.247	0.126
		Microsoft Windows 10	0.113	配置配線不可	0.190	0.247	0.025
2019.2	5	Ubuntu 18.04	0.078	配置配線不可	0.114	0.241	−0.418
		Microsoft Windows 10	0.078	配置配線不可	0.114	0.241	−0.418
	10	Ubuntu 18.04	0.114	0.159	0.104	0.260	0.013
		Microsoft Windows 10	0.116	0.159	0.104	0.132	0.013

4.2　ブロック使用率

高位合成時、論理合成・配置配線時のブロック使用量を表 3 から表 7 に示す。これらの表において、BRAM は RAM の、DSP は DSP の、FF は flip flop の、LUT は look up table の見積もり量または使用量を表し、括弧内は PYNQ-Z1 が内蔵する各ブロックの量である。また、これらの表において、「論理合成」の列は論理合成・配置配線後の使用量である。全体的に、高位合成時の FF と LUT の見積もりブロック数が、PYNQ-Z1 が内蔵するブロック数を超えているが、論理合成・配置配線時には、内蔵するブロック数以内になっている場合が多い。次に、ネットワークごとに述べる。

表3　ネットワーク cnvW1A1 のブロック使用状況

Vivado HL WebPACK	高位合成時 Clock [ns]	OS	BRAM（280）		DSP（220）		FF（106400）		LUT（53200）	
			高位合成	論理合成	高位合成	論理合成	高位合成	論理合成	高位合成	論理合成
2018.2	5	Ubuntu 16.04	257	96	48	24	182042	41312	121964	26080
		Microsoft Windows 10	257	96	48	24	182042	41312	121964	26080
	10	Ubuntu 16.04	257	96	36	0	32578	34168	118371	24806
		Microsoft Windows 10	257	96	36	0	32578	34168	118371	24806
2019.2	5	Ubuntu 18.04	257	96	24	24	181188	42053	129622	29635
		Microsoft Windows 10	257	96	24	24	181188	42053	129622	29635
	10	Ubuntu 18.04	257	96	24	0	35594	37714	120648	25367
		Microsoft Windows 10	257	96	24	0	35594	37716	120648	25411

表4　ネットワーク cnvW1A2 のブロック使用状況

Vivado HL WebPACK	高位合成時 Clock [ns]	OS	BRAM（280）		DSP（220）		FF（106400）		LUT（53200）	
			高位合成	論理合成	高位合成	論理合成	高位合成	論理合成	高位合成	論理合成
2018.2	5	Ubuntu 16.04	284	83	52	26	226561	56276	157286	40328
		Microsoft Windows 10	284	83	52	26	226561	56276	157286	40328
	10	Ubuntu 16.04	284	83	39	0	49792	47946	140457	38753
		Microsoft Windows 10	284	配置配線不可	39	配置配線不可	49792	配置配線不可	140457	配置配線不可
2019.2	5	Ubuntu 18.04	284	配置配線不可	26	配置配線不可	202908	配置配線不可	152857	配置配線不可
		Microsoft Windows 10	284	配置配線不可	26	配置配線不可	202908	配置配線不可	152857	配置配線不可
	10	Ubuntu 18.04	284	83	26	0	57858	53105	144546	37474
		Microsoft Windows 10	284	83	26	0	57858	53105	144546	37474

表5　ネットワーク cnvW2A2 のブロック使用状況

VivadoHL WebPACK	高位合成時 Clock [ns]	OS	BRAM（280）		DSP（220）		FF（106400）		LUT（53200）	
			高位合成	論理合成	高位合成	論理合成	高位合成	論理合成	高位合成	論理合成
2018.2	5	Ubuntu 16.04	362	4	64	32	195376	51388	110421	37320
		Microsoft Windows 10	362	4	64	32	195376	51388	110421	37320
	10	Ubuntu 16.04	362	4	48	0	43304	44964	105280	37090
		Microsoft Windows 10	362	4	48	0	43304	44964	105280	37090
2019.2	5	Ubuntu 18.04	362	4	32	32	191292	51321	117694	40022
		Microsoft Windows 10	362	4	32	32	191292	51321	117694	40022
	10	Ubuntu 18.04	362	4	32	0	42549	44952	104339	34679
		Microsoft Windows 10	362	4	32	0	42549	44952	104339	34679

表 6　ネットワーク lfcW1A1 のブロック使用状況

Vivado HL WebPACK	高位合成時 Clock [ns]	OS	BRAM（280）		DSP（220）		FF（106400）		LUT（53200）	
			高位合成	論理合成	高位合成	論理合成	高位合成	論理合成	高位合成	論理合成
2018.2	5	Ubuntu 16.04	220	208	8	4	55336	33470	137522	28360
		Microsoft Windows 10	220	208	8	4	55336	33470	137522	28360
	10	Ubuntu 16.04	220	208	6	0	24373	23831	127091	25253
		Microsoft Windows 10	220	208	6	0	24373	23831	127091	25253
2019.2	5	Ubuntu 18.04	220	208	4	4	41034	30674	133941	25025
		Microsoft Windows 10	220	208	4	4	41034	30674	133941	25025
	10	Ubuntu 18.04	220	208	2	0	28565	28295	129718	24621
		Microsoft Windows 10	220	208	2	0	28565	28295	129718	24597

表 7　ネットワーク lfcW1A2 のブロック使用状況

Vivado HL WebPACK	高位合成時 Clock [ns]	OS	BRAM（280）		DSP（220）		FF（106400）		LUT（53200）	
			高位合成	論理合成	高位合成	論理合成	高位合成	論理合成	高位合成	論理合成
2018.2	5	Ubuntu 16.04	236	208	8	4	108910	45338	214732	48981
		Microsoft Windows 10	236	208	8	4	108910	45338	214732	48981
	10	Ubuntu 16.04	236	208	6	0	42197	31544	190733	42597
		Microsoft Windows 10	236	208	6	0	42197	31537	190733	42220
2019.2	5	Ubuntu 18.04	236	208	4	4	71999	44801	201828	47916
		Microsoft Windows 10	236	208	4	4	71999	44801	201828	47916
	10	Ubuntu 18.04	236	208	2	0	59230	41057	200053	42333
		Microsoft Windows 10	236	208	2	0	59230	41057	200053	42333

⑴　cnvW1A1（表 3）：3 章で述べた修正で、bitstream が生成できる。Vivado HL WebPACK のバージョンが同じであれば、OS が違っても、生成される bitstream が使用するブロック数はあまり変わらない。

⑵　cnvW1A2（表 4）：Ubuntu 16.04と Microsoft Windows 10上の Vivado HL WebPACK 2018.2では、bitstream は生成できるが、論理合成・配置配線時に周期制約を0.313ns 違反するエラーが発生した。BNN-PYNQ の公式サイトにこのエラーに関する言及は見当たらなかった。Vivado HL WebPACK 2019.2では、論理合成・配置配線時に配置配線ができない旨のエラーが発生し、bitstream が生成できなかった。そこで、高位合成時の周期を10ns に設定して、高位合成を行った後、論理合成・配置配線を行った結果、bitstream を生成できた。

⑶　cnvW2A2（表 5）：3 章で述べた修正で、bitstream が生成できる。Vivado HL WebPACK のバージョンが同じであれば、OS が違っても、生成される bitstream が使用するブロック数は変わらない。

⑷　lfcW1A1（表 6）：3 章で述べた修正で、bitstream が生成できる。Vivado HL WebPACK 2018.2の場合、高位合成時の見積もり周期が10ns を超えているが、論理合成・配置配線時

に10ns 以内になった。Vivado HL WebPACK のバージョンが同じであれば、OS が違っても、生成される bitstream が使用するブロック数はあまり変わらない。

⑸ lfcW1A2（表7）：Ubuntu 18.04上の Vivado HL WebPACK 2019.2の場合、高位合成時の周期を 5 ns にすると、論理合成・配置配線時に配置エラーが発生し、bitstream を生成できなかった。Microsoft Windows 10の場合は、bitstream を生成できるが、論理合成・配置配線時に周期10ns を0.154ns 超過している。Vivado HL WebPACK 2018.2の場合、高位合成時の周期が10ns を超えているが、論理合成・配置配線時に10ns 以内になった。Vivado HL WebPACK のバージョンが同じであれば、OS が違っても、生成される bitstream が使用するブロック数は変わらない。

4.3 高位合成時と論理合成・配置配線時の使用メモリ量

高位合成時と論理合成・配置配線時の Vivado HL WebPACK の最大使用メモリ量を表 8 に示す。この表において、「論理合成」の列は、論理合成または配置配線の使用メモリの大きい方の値を記している。配置配線ができなかった場合は、Vivado が停止するまでの最大使用メモリを示している。

表 8　高位合成と論理合成・配置配線の最大メモリ使用量［GB］

Vivado HL WebPACK	高位合成時 Clock [ns]	OS	cnvW1A1		cnvW1A2		cnvW2A2		lfcW1A1		lfcW1A2	
			高位合成	論理合成	高位合成	論理合成	高位合成	論理合成	高位合成	論理合成	高位合成	論理合成
2018.2	5	Ubuntu 16.04	1.60	3.41	1.71	3.79	1.48	3.76	1.99	3.31	1.99	3.76
		Microsoft Windows 10	1.24	2.75	1.33	3.32	1.21	3.22	1.31	2.65	1.34	3.23
	10	Ubuntu 16.04	1.58	3.34	1.69	3.64	1.46	3.70	1.96	3.23	1.96	3.56
		Microsoft Windows 10	1.23	2.66	1.32	1.87	1.20	3.14	1.31	2.54	1.33	2.97
2019.2	5	Ubuntu 18.04	1.66	3.67	1.76	3.82	1.52	3.84	2.19	3.57	2.21	3.87
		Microsoft Windows 10	1.30	2.99	1.38	3.07	1.27	3.28	1.41	2.85	1.45	3.39
	10	Ubuntu 18.04	1.65	3.63	1.75	3.83	1.50	3.77	2.20	3.56	2.20	3.83
		Microsoft Windows 10	1.30	2.92	1.38	3.23	1.26	3.17	1.43	2.85	1.45	3.25

表 8 から、Microsoft Windows 10上に仮想Linuxマシンを作成し、Vivado HL WebPACK を使う場合、Vivado HL WebPACK 用に 4 GB 程度のメモリを確保する必要があることがわかる。表 8 では、Microsoft Windows 10上で実行した方が Ubuntu 上で実行するよりも、ネットワークによらず、使用メモリ量が少ない。しかし、Ubuntu 16.04と Ubuntu 18.04の場合、Vivado のデフォルトのスレッド数は 8，Microsoft Windows 10の場合、デフォルトのスレッド数は 2 である。スレッドあたりのメモリ量で評価すれば、Micorsoft Windows 10の方が使用メモリ量が少ないとは言い切れない。

4.4　生成した bitstream を用いた識別テスト

生成した bitstream を用いて識別テストを行った。cnvW1A1, cnvW1A2, cnvW2A2 については、CIFAR-10のテスト画像10000枚に対して、識別テストを行い、公式サイトで配布されている bitstream による結果と一致することを確認した。また、lfcW1A1, lfcW1A2については、MNIST の 手書き文字画像10000枚に対して、識別テストを行い、公式サイトで配布されている bitstream による結果と一致することを確認した。

これらの識別テストを行った時の画像 1 枚当たりの識別に要した平均時間を表 9 に示す。bitstream の回路の動作周波数はすべて100MHz である。bitstream は生成する環境に依存するが（表 2 -表 8 ）、識別に要する時間は環境にはほとんど依存しない。Vivado HL WebPACK 2018.2を用いて高位合成時に 5 ns の周期で cnvW1A2 の bitstream を生成すると、論理合成・配置配線時に周期制約を満たせなかったが（表 2 ）、この識別テストでは正常に動作した。Vivado HL WebPACK 2019.2を用いて高位合成時に周期 5 ns で設計した lfcW1A2 も同様である。

表 9　識別テストにおける画像 1 枚あたりの平均所要時間

Vivado HL WebPACK	高位合成時 Clock [ns]	OS	cnvW1A1 [us]	cnvW1A2 [us]	cnvW2A2 [us]	lfcW1A1 [us]	lfcW1A2 [us]
			CIFAR-10	CIFAR-10	CIFAR-10	MNIST	MNIST
公式配布			1583.28	1629.01	4866.36	8.41	8.41
2018.2	5	Ubuntu 16.04	1583.20	1628.54	4866.80	8.41	8.41
		Microsoft Windows 10	1583.31	1628.98	4866.68	8.41	8.41
	10	Ubuntu 16.04	1583.00	1628.03	4866.18	8.41	8.41
		Microsoft Windows 10	1582.90	配置配線不可	4866.00	8.41	8.41
2019.2	5	Ubuntu 18.04	1583.26	配置配線不可	4866.39	8.41	8.41
		Microsoft Windows 10	1582.88	配置配線不可	4866.39	8.41	8.41
	10	Ubuntu 18.04	1583.13	1627.98	4866.23	8.41	8.41
		Microsoft Windows 10	1583.00	1628.27	4865.94	8.41	8.41

5. まとめ

本稿では、BNN-PYNQ を Microsoft Windows 10上に移植し、その動作確認を行った。Microsoft Windows 10上で生成される bitstream と Ubuntu 上で生成される bitstream は、必ずしも同一ではないが、BNN-PYNQ の 5 種類のネットワークでは、同等の性能が実現できることが確認できた。高位合成時の設計周期を10ns にすれば、Microsoft Windows 10上に Linux の仮想マシンを用意することなく、Microsoft Windows 10上の Vivado HL WebPACK 2019.2で BNN-PYNQ の開発ができるようになった。

今回使用した PYNQ-Z1 ボードより内蔵ブロック数が多い開発ボードを用いれば、PYNQ-Z1 で配置配線ができなかった問題が解決できる可能性がある。内蔵ブロック数が多い開発ボードでの動作確認が今後の課題である。

参考文献

1） M. Courbariaux and Y. Bengio, “BinaryNet: Training Deep Neural Networks with Weights and Activations Constrained to +1 or −1,” CoRR, abs/1602.02830,2016.

2） PYNQ-Python productivity for Zynq, http://www.pynq.io/（2020年 6 月14日閲覧）.

3） Xilinx, Vivado Design Suite 評価版および WebPACK、https://japan.xilinx.com/products/design-tools/vivado/vivado-webpack.html#webpack（2020年 6 月14日閲覧）.

4） BNN-PYNQ, https://github.com/Xilinx/BNN-PYNQ（2020年 6 月14日閲覧）.

5） CIFAR-10 and CIFAR-100 datasets, https://www.cs.toronto.edu/~kriz/cifar.html（2020年 6 月14日閲覧）.

6） Y. LeCun, C. Cortes, and C. J. C. Burges, THE MNIST DATABASE of handwritten digits, http://yann.lecun.com/exdb/mnist/（2020年 6 月14日閲覧）.

7） Digilent Inc., PYNQ-Z1, https://reference.digilentinc.com/reference/programmable-logic/pynq-z1/start（2020年 6 月14日閲覧）.

8） Git for Windows, https://gitforwindows.org/（2020年 6 月14日閲覧）.

9） tiny-dnn/tiny-dnn: header only, dependency-free deep learning framework in C++14, https://github.com/tiny-dnn/tiny-dnn（2020年 6 月14日閲覧）.

10） 竹中瑞樹、BNN-PYNQ パッケージにおけるハードウェア設計再構築の Windows 上での実行可能性調査、2019年度関西大学総合情報学部卒業論文、2020年 3 月.

11） 鈴木量三朗、FPGA 人工知能のポテンシャルを探る、Interface 2018年 8 月号、CQ 出版、pp.161-165、2018年.

12） 新千葉ガーベージ・コレクション、http://ryos36.hatenablog.com/entry/2017/09/09/230941（2020年 6 月14日閲覧）.