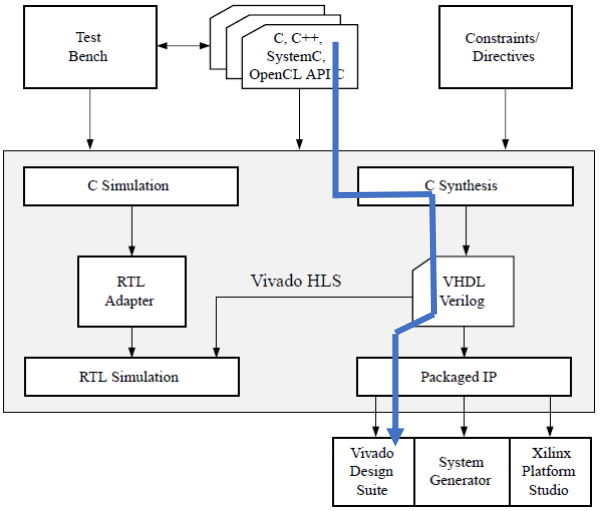
Deep Learning with FPGA
- YOLO V2 -

2020

Ando Ki, Ph.D.
adki@future-ds.com

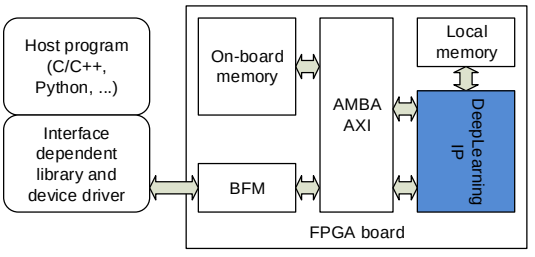
Contents

Xilinx HLS design flow

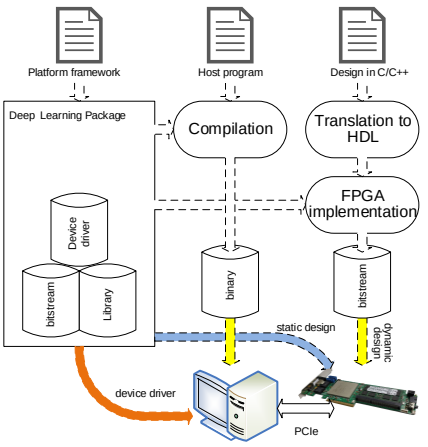


Straight forward solution

- Generic platform

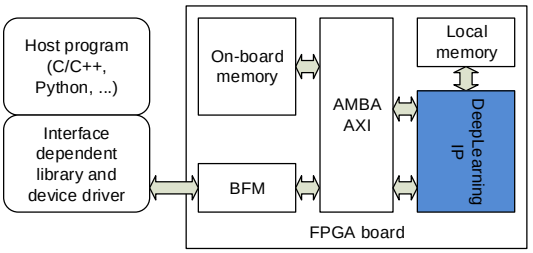


- Develop framework

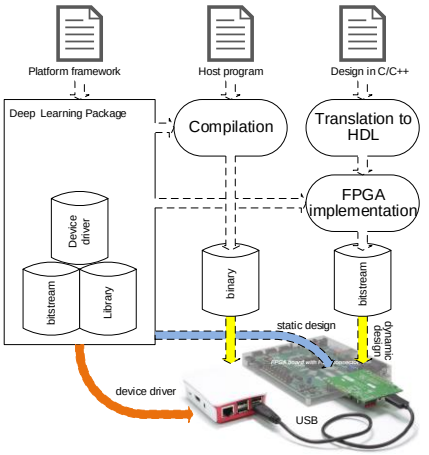


Straight forward solution

- Generic platform



- Develop framework



5

How to run in short (1/3)

- Prerequisites
  - Xilinx Vivado 2018.3
  - Xilinx Vivado HLS 2018.3
  - Xilinx SDK 2018.3
  - Xilinx PlatformUSB device driver installed (optional)
  - LibUSB-1.0.0 installed
  - Future Design Systems CON-FMC 2019.10
  - OpenCV 2

How to run in short (2/3)

$PROJECT/codes.fpga/Yolov2_fpag

- 1. HLS Synthesis
- - Go to 'hw/hls/tcl'
- - run 'make syn'

- 2. VIVADO Implementation using IP Integrator
- - Go to 'hw/impl/vivado.zed.confmc'
- - Invoke 'make'

- 3. SD card Image (Boot file) Generation
- - Go to 'hw/impl/vivado.zed.confmc/bootgen'
- - Invoke 'make'
- - Copy 'BOOT.bin' to the SD-Card and then turn on ZedBoard

- Use Vivado 2018.3.
- Do not forget to set Vivado environment

  $ source /opt/Xilinx/Vivado/2018.3/settings64.sh

- Do not forget to set Vivado-SDK environment

  $ source /opt/Xilinx/SDK/2018.3/settings64.sh

Use Vivado 2018.3

7

How to run in short (3/3)

- 4. Compile software
- - It requres OpenCV and CON-FMC
- - Go to 'sw.native/yolov2.confmc'
- - Invoke 'make clean; make'

- 5. Run software
- - Go to 'sw.native/yolov2.confmc'
- - Run 'make run' to deal with image file or 'make run.cam' to use Camera

- Do not forget to set CON-FMC environment

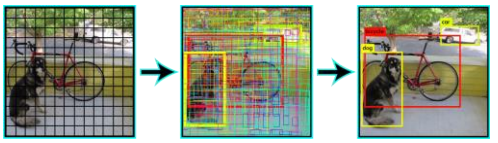
  $ source /opt/confmc/2019.10/settings.sh

- Do not forget to connect ZedBoard to the computer through USB
- Do not forget to turn on ZedBoard along with SD-Card containing proper BOOT.bin

8

Yolo V2

- YOLO (You Look Only Once, Yolo9000 – can detect over 9000 object categories by combining multiple data set and detection)



  - 1. Visit https://pjreddie.com/darknet/yolov2/
  - 2. Get Darknet (framework) and compile it
    - $ git clone https://github.com/pjreddie/darknet
    - $ cd darknet
    - $ make
  - 2. Get weights (194.49Mbyte)
    - $ mkdir weights; cd weights
    - $ wget https://pjreddie.com/media/files/yolov2.weights
  - 3. Run Yolo V2 using Darknet
    - $ ./darknet detect cfg/yolov2.cfg weights/yolov2.weights data/dog.jpg

9

Yolo V2

```
adki@AndoUbuntu: ~/work/projects/DeepLearningFpga/YoloV2/yolov2_org/darknet
[adki@AndoUbuntu] ./darknet detect cfg/yolov2.cfg weights/yolov2.weights data/dog.jpg
layer     filters    size              input                output
    0 conv     32  3 x 3 / 1   608 x 608 x   3   ->   608 x 608 x  32  0.639 BFLOPs
    1 max          2 x 2 / 2   608 x 608 x  32   ->   304 x 304 x  32
    2 conv     64  3 x 3 / 1   304 x 304 x  32   ->   304 x 304 x  64  3.407 BFLOPs
    3 max          2 x 2 / 2   304 x 304 x  64   ->   152 x 152 x  64
    4 conv    128  3 x 3 / 1   152 x 152 x  64   ->   152 x 152 x 128  3.407 BFLOPs
    5 conv     64  1 x 1 / 1   152 x 152 x 128   ->   152 x 152 x  64  0.379 BFLOPs
    6 conv    128  3 x 3 / 1   152 x 152 x  64   ->   152 x 152 x 128  3.407 BFLOPs
    7 max          2 x 2 / 2   152 x 152 x 128   ->    76 x  76 x 128
    8 conv    256  3 x 3 / 1    76 x  76 x 128   ->    76 x  76 x 256  3.407 BFLOPs
    9 conv    128  1 x 1 / 1    76 x  76 x 256   ->    76 x  76 x 128  0.379 BFLOPs
   10 conv    256  3 x 3 / 1    76 x  76 x 128   ->    76 x  76 x 256  3.407 BFLOPs
   11 max          2 x 2 / 2    76 x  76 x 256   ->    38 x  38 x 256
   12 conv    512  3 x 3 / 1    38 x  38 x 256   ->    38 x  38 x 512  3.407 BFLOPs
   13 conv    256  1 x 1 / 1    38 x  38 x 512   ->    38 x  38 x 256  0.379 BFLOPs
   14 conv    512  3 x 3 / 1    38 x  38 x 256   ->    38 x  38 x 512  3.407 BFLOPs
   15 conv    256  1 x 1 / 1    38 x  38 x 512   ->    38 x  38 x 256  0.379 BFLOPs
   16 conv    512  3 x 3 / 1    38 x  38 x 256   ->    38 x  38 x 512  3.407 BFLOPs
   17 max          2 x 2 / 2    38 x  38 x 512   ->    19 x  19 x 512
   18 conv   1024  3 x 3 / 1    19 x  19 x 512   ->    19 x  19 x1024  3.407 BFLOPs
   19 conv    512  1 x 1 / 1    19 x  19 x1024   ->    19 x  19 x 512  0.379 BFLOPs
   20 conv   1024  3 x 3 / 1    19 x  19 x 512   ->    19 x  19 x1024  3.407 BFLOPs
   21 conv    512  1 x 1 / 1    19 x  19 x1024   ->    19 x  19 x 512  0.379 BFLOPs
   22 conv   1024  3 x 3 / 1    19 x  19 x 512   ->    19 x  19 x1024  3.407 BFLOPs
   23 conv   1024  3 x 3 / 1    19 x  19 x1024   ->    19 x  19 x1024  6.814 BFLOPs
   24 conv   1024  3 x 3 / 1    19 x  19 x1024   ->    19 x  19 x1024  6.814 BFLOPs
   25 route  16
   26 conv     64  1 x 1 / 1    38 x  38 x 512   ->    38 x  38 x  64  0.095 BFLOPs
   27 reorg              / 2    38 x  38 x  64   ->    19 x  19 x 256
   28 route  27 24
   29 conv   1024  3 x 3 / 1    19 x  19 x1280   ->    19 x  19 x1024  8.517 BFLOPs
   30 conv    425  1 x 1 / 1    19 x  19 x1024   ->    19 x  19 x 425  0.314 BFLOPs
   31 detection
mask_scale: Using default '1.000000'
Loading weights from weights/yolov2.weights...Done!
data/dog.jpg: Predicted in 10.963610 seconds.
dog: 82%
truck: 64%
bicycle: 85%
[adki@AndoUbuntu]
```

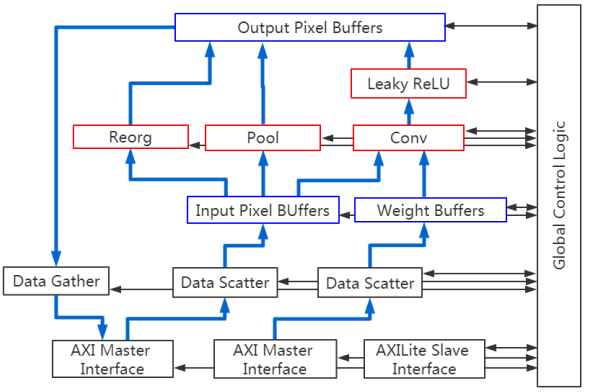
Darknet-19 classification network (not exactly)

Type	Filters	Size/Stride	Output
Convolutional	32	3×3	224×224
Maxpool		$2 \times 2/2$	112×112
Convolutional	64	3×3	112×112
Maxpool		$2 \times 2/2$	56×56
Convolutional	128	3×3	56×56
Convolutional	64	1×1	56×56
Convolutional	128	3×3	56×56
Maxpool		$2 \times 2/2$	28×28
Convolutional	256	3×3	28×28
Convolutional	128	1×1	28×28
Convolutional	256	3×3	28×28
Maxpool		$2 \times 2/2$	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Maxpool		$2 \times 2/2$	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	1000	1×1	7×7
Avgpool		Global	1000
Softmax			

10

YOLO V2 on FPGA

- Use Fixed-16 instead Float-3
- Overall architecture
  - AXI-Lite slave interface is responsible for reading and writing control, data and status register sets
  - AIX4 master interfaces: read input feature maps and write output feature maps
  - AIX4 master interfaces: read weights and write output weights
  - The Data Scatter module is designed to generate the corresponding write address and distribute the data read from the DRAM to the on-chip buffers.
  - The Data Gather module is designed to generate the DRAM write-back address and write the data in the output buffer back to the DRAM.



Prepare IP

- 1. HLS Synthesis
- - go to 'hw/hls/tcl'
- - run 'make'

$PROJECT/codes.fpga/Yolov2_fpag

add following in '.bashrc' file.

```
alias set_vivado='source /opt/XilinxWebpack/Vivado/2018.3/settings64.sh;₩
                  export XILINX_VIVADO_HLS=/opt/XilinxWebpack/Vivado/2018.3;₩
                  export XILINX_SDK=/opt/Xilinx/SDK/2018.3;₩
                  source ${XILINX_SDK}/settings64.sh'
```

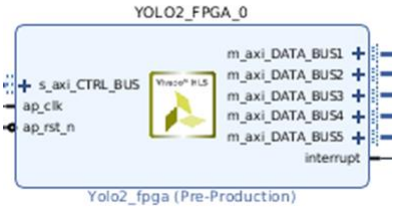
```
... ...
$ source /opt/XilinxWebpack/Vivado/2018.3/settings64.sh
$ export XILINX_VIVADO_HLS=/opt/XilinxWebpack/Vivado/2018.3
... ...
$ cd hw/hls/tcl
$ make
```

YOLO V2 core code

```
void YOLO2_FPGA(int *Input,int *Input1,int *Input2,int *Input3,int *Output,int *Output1,
                int *Weight,int *Beta,const int InFM_num,const int OutFM_num,
                const int Kernel_size,const int Kernel_stride,
                const int Input_w,const int Input_h,const int output_w,const int output_h,const int Padding,const bool IsNL,const bool IsBN,
                const int TM,const int TN,const int TR,const int TC,
                const int mLoops,const int nLoops,const int rLoops,const int cLoops,const int LayerType,
                const int InputQ,const int OutputQ,const int WeightQ,const int BetaQ,int trow_loops)
{

#pragma HLS INTERFACE m_axi depth=512 port=Input    offset=slave bundle=DATA_BUS1 num_read_outstanding=1 num_write_outstanding=1 max_read_burst_length=64 max_write_burst_length=64
#pragma HLS INTERFACE m_axi depth=512 port=Input1   offset=slave bundle=DATA_BUS2 num_read_outstanding=1 num_write_outstanding=1 max_read_burst_length=64 max_write_burst_length=64
#pragma HLS INTERFACE m_axi depth=512 port=Input2   offset=slave bundle=DATA_BUS3 num_read_outstanding=1 max_read_burst_length=64
#pragma HLS INTERFACE m_axi depth=512 port=Input3   offset=slave bundle=DATA_BUS4 num_read_outstanding=1 max_read_burst_length=64
#pragma HLS INTERFACE m_axi depth=512 port=Output   offset=slave bundle=DATA_BUS1 num_read_outstanding=1 num_write_outstanding=1 max_read_burst_length=64 max_write_burst_length=64
#pragma HLS INTERFACE m_axi depth=512 port=Output1  offset=slave bundle=DATA_BUS2 num_read_outstanding=1 num_write_outstanding=1 max_read_burst_length=64 max_write_burst_length=64
#pragma HLS INTERFACE m_axi depth=512 port=Weight   offset=slave bundle=DATA_BUS5 num_read_outstanding=1 max_read_burst_length=128
#pragma HLS INTERFACE m_axi depth=512 port=Beta     offset=slave bundle=DATA_BUS5 num_read_outstanding=1 max_read_burst_length=128

#pragma HLS INTERFACE s_axilite register port=return bundle=CTRL_BUS
#pragma HLS INTERFACE s_axilite register port=InFM_num bundle=CTRL_BUS
#pragma HLS INTERFACE s_axilite register port=OutFM_num bundle=CTRL_BUS
#pragma HLS INTERFACE s_axilite register port=Kernel_size bundle=CTRL_BUS
... ...
#pragma HLS INTERFACE s_axilite register port=Input bundle=CTRL_BUS
#pragma HLS INTERFACE s_axilite register port=Output bundle=CTRL_BUS
#pragma HLS INTERFACE s_axilite register port=Weight bundle=CTRL_BUS
#pragma HLS INTERFACE s_axilite register port=Beta bundle=CTRL_BUS
```

Bus groups

Control bus (slave)



13

Xilinx HLS script

- $ vivado_hls -f run_hls_syn.tcl

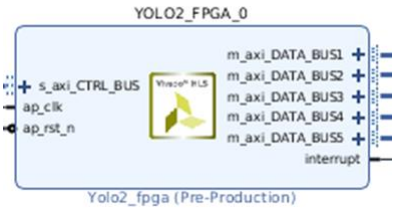
```
# Create a project
open_project proj_yolo
# Set the top-level function
set_top YOLO2_FPGA
# Add design files
add_files src/cnn.cpp
add_files src/cnn.h
# Add test bench & files
add_files -tb tb/main.cpp
add_files -tb tb/yolov2.h
add_files -tb tb/stb_image_write.h
add_files -tb tb/stb_image.h
add_files -tb tb/coco.names
add_files -tb tb/kite.jpg
...
```

```
# Create a solution
open_solution "solution1"
# Define technology and clock rate
set_part  {xc7z020clg484-1}
create_clock -period 6 -name default
csynth_design
export_design -format ip_catalog
tclapp::reset_tclstore
exit
```

14

Yolo V2 RTL

```
// ==============================================================
// RTL generated by Vivado(TM) HLS - High-Level Synthesis from C, C++ and OpenCL
// Version: 2019.2
// Copyright (C) 1986-2019 Xilinx, Inc. All Rights Reserved.
//
// ==============================================================

`timescale 1 ns / 1 ps

(* CORE_GENERATION_INFO="YOLO2_FPGA,hls_ip_2019_2,{HLS_INPUT_TYPE=cxx,HLS_INPUT_FLOAT=0,HLS_INPUT_FIXED=0,HLS_INPUT_PART=xc7z020-clg484-1,H ... ... *)

module YOLO2_FPGA (
        ap_clk,
        ap_rst_n,
        m_axi_DATA_BUS1_AWVALID,
        m_axi_DATA_BUS1_AWREADY,
        m_axi_DATA_BUS1_AWADDR,
...
        m_axi_DATA_BUS2_AWVALID,
        m_axi_DATA_BUS2_AWREADY,
        m_axi_DATA_BUS2_AWADDR,
...
        m_axi_DATA_BUS5_BID,
        m_axi_DATA_BUS5_BUSER,
        s_axi_CTRL_BUS_AWVALID,
        s_axi_CTRL_BUS_AWREADY,
        s_axi_CTRL_BUS_AWADDR,
```



HLS report

General Information

Date:	Tue Jun 9 17:04:27 2020
Version:	2018.3.1 (Build 2489210 on Tue Mar 26 04:40:43 MDT 2019)
Project:	proj_yolo
Solution:	solution1
Product family:	zynq
Target device:	xc7z020clg484-1

Performance Estimates

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	6.00	6.421	0.75

Latency (clock cycles)

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	3	-	-
Expression	-	0	0	889
FIFO	-	-	-	-
Instance	49	149	38933	54056
Memory	129	-	0	0
Multiplexer	-	-	-	4700
Register	-	-	2610	-
Total	178	152	41543	59645
Available	280	220	106400	53200
Utilization (%)	63	69	39	112

Detail

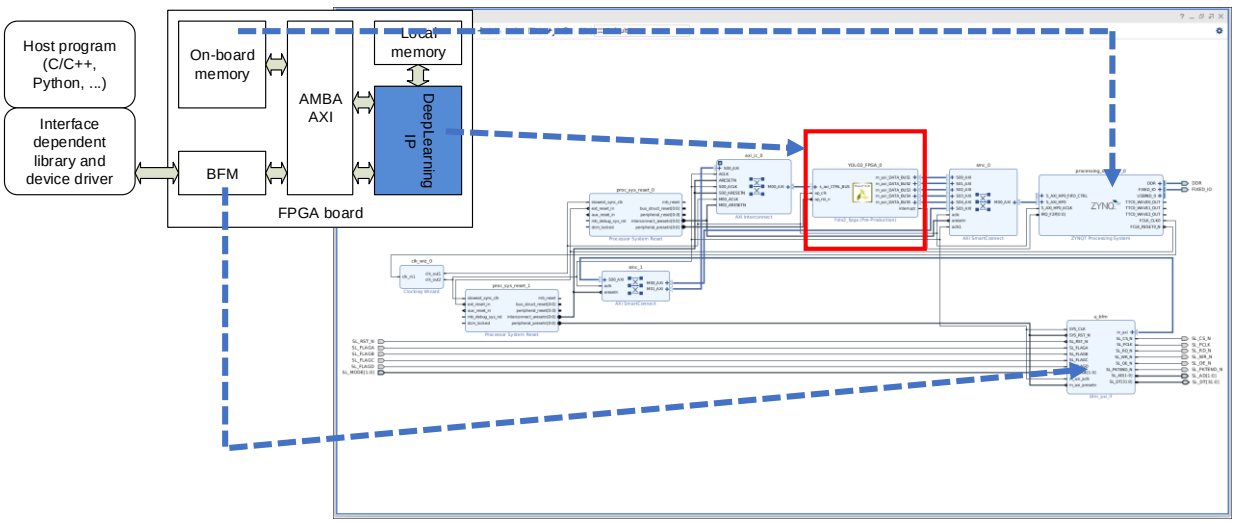
Prepare whole design

- 2. VIVADO IP Integrator
- - go to 'hw/impl/vivado.zed.confmc'
- - run 'make'

$PROJECT/codes.fpga/Yolov2_fpag

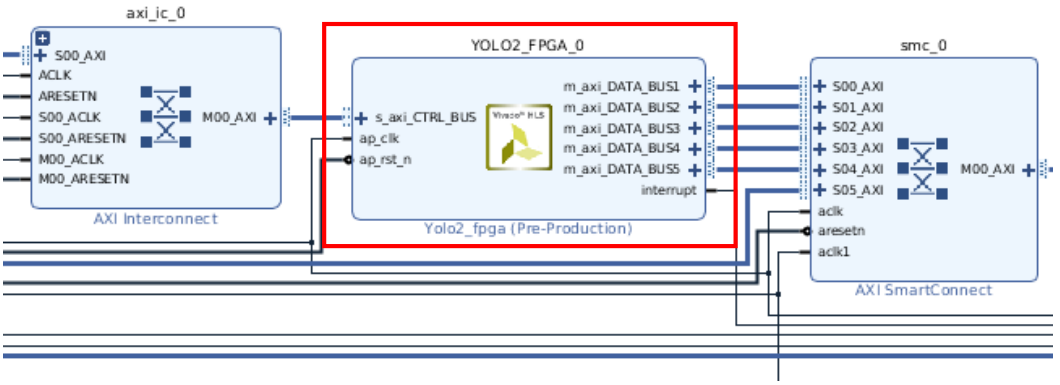
```
... ...
$ source /opt/XilinxWebpack/Vivado/2018.3/settings64.sh
... ...
$ cd hw/impl/vivado.zed.confmc
$ make
```


Xilinx Vivado IP integrator

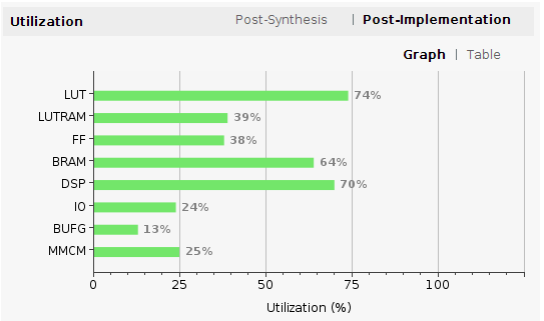


Yolo V2 IP

- Running at 200Mhz



Implementation report



Timing — Setup | Hold | Pulse Width

Worst Negative Slack (WNS):	-1.462 ns
Total Negative Slack (TNS):	-14445.755 ns
Number of Failing Endpoints:	37059
Total Number of Endpoints:	152946

Implemented Timing Report

Power — Summary | On-Chip

Total On-Chip Power:	**3.728 W**
Junction Temperature:	**68.0 °C**
Thermal Margin:	17.0 °C (1.4 W)
Effective θJA:	11.5 °C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Implemented Power Report

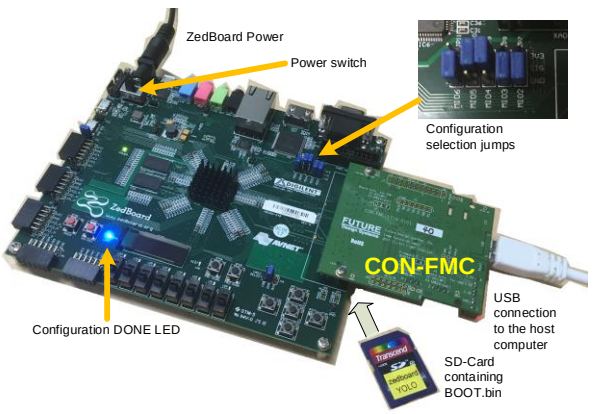
Prepare FPGA image

- 3. Boot file generation
- - go to 'hw/impl/vivado.zed.confmc/bootgen'
- - run 'make'
- - copy 'BOOT.bin' to the SD-CARD and then tun on ZedBoard

$PROJECT/codes.fpga/Yolov2_fpag

```
... ...
$ export XILINX_SDK=/opt/Xilinx/SDK/2018.3
$ source ${XILINX_SDK}/settings64.sh
... ...
$ cd hw/impl/vivado.zed.confmc/bootgen
$ make
```


HW setup



- 1. Turn off ZedBoard
- 2. Check configuration jumps
- 3. Insert SD-Card
- 4. Connect USB port
- 5. Turn on ZedBoard

You should see followings on you host computer.

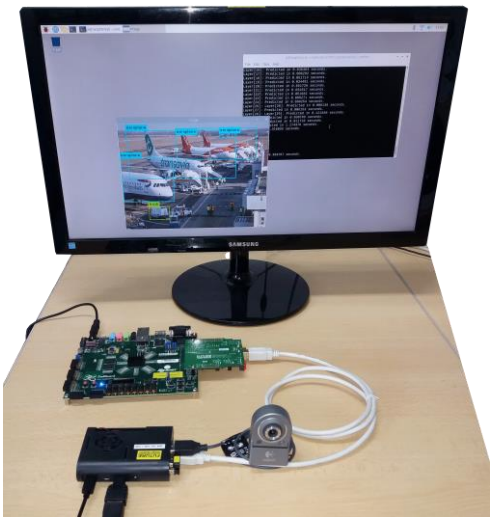
$ lsusb -d 04b4:
Bus 001 Device 017: ID ***04b4:00f3*** Cypress Semiconductor Corp.

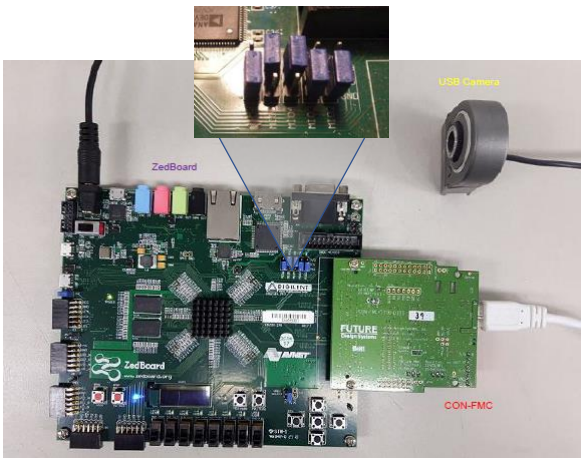
Running the host program through USB

- 4. CON-FMC software (OpenCV is required)
- - It requires OpenCV and CON-FMC
- - go to 'sw.native/yolov2.confmc' directory
- - run 'make clean; make'
- 5. Run
- - go to 'sw.native/yolov2.confmc' directory
- - run './yolov2 images/5.png'

```
... ...
$ souce /opt/confmc/2919.10/sttings.sh
... ...
$ cd sw.native/yolov2.confmc
$ make
$ ./yolov2
```


Running YOLO V2 HW IP along with CON-FMC





C-driven host program

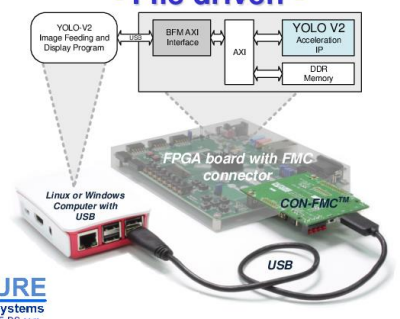
```
#include <opencv2/opencv.hpp>
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
#ifdef USE_CON_FMC
#include "conapi.h"
#include "trx_axi_api.h"
unsigned int card_id=0;
con_Handle_t handle=NULL;
#endif
#include "yolov2.h"
using namespace cv;

int main(int argc, char *argv[]) {

   // Open CON-FMC
   handle=conInit(card_id, CON_MODE_CMD,
            CONAPI_LOG_LEVEL_INFO);

    // Get class names
     //  Get label related things
     //  Write weights
     // Write biases
      while (1) {
          // read image
          image im = load_image_stb(input_imgfn, 3);
          // resize image
          image sized = letterbox_image(im, 416, 416);
          // get image data
          float *X = sized.data;
          // let HW IP go with image data in X
          yolov2_hls_ps(net, X, WEIGHT_BASE, BETA_BASE, MEM_BASE);
          // get bounding box
          detection *dets = get_network_boxes(...);
          // draw detection
      }
}
```

$PROJECT/codes.fpga/Yolov2_fpag/sw.native/yolov2.confmc/src/main.cpp

25

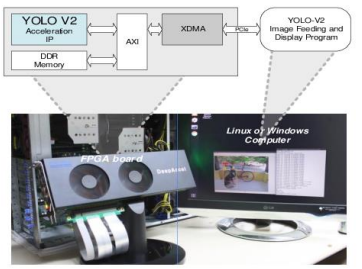
Example cases

Running YOLO V2 on FPGA
- File driven -



FUTURE Design Systems
www.FUTURE-DS.com

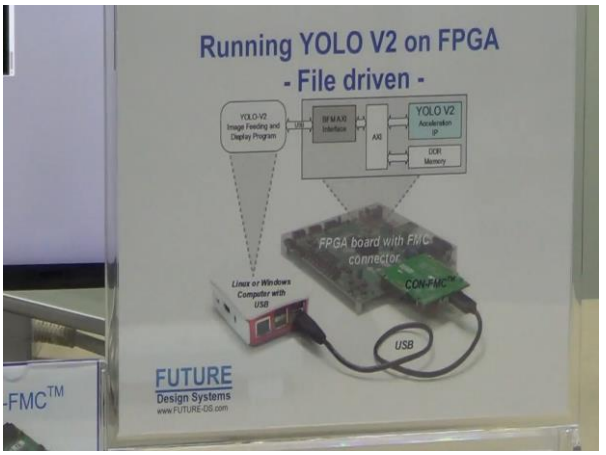
Running YOLO V2 on FPGA



FUTURE Design Systems
www.FUTURE-DS.com

26

Example cases



27

References

- A demo for accelerating YOLOv2 in xilinx's fpga pynq/zedboard, https://github.com/dhm2013724/yolov2_xilinx_fpga

28