Deep Learning with FPGA

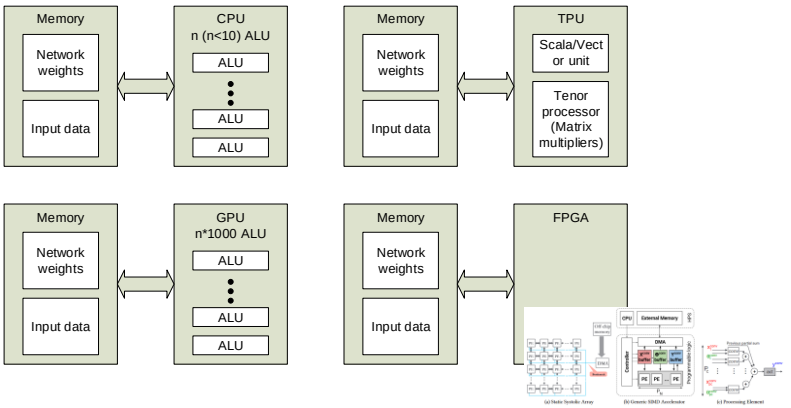
2020

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

HW accelerators

- HW accelerators
- One approach: OpenCL
- OpenCL platform & memory model
- OpenCL programming and design flow
- OpenCL C language restrictions for kernel
- Matrix multiplication OpenCL kernel
- OpenCL and FPGA
- What HLS does
- OpenCL port to Xilinx FPGA: SDAccel
- Example case
- Kernel expression
- Future Design Systems' solution

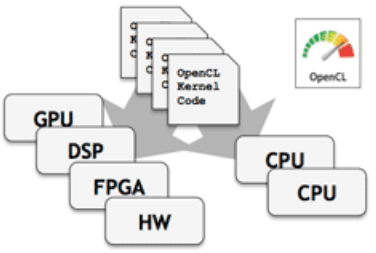
HW accelerators



3

One approach: OpenCL

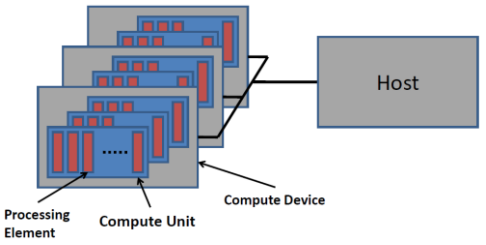
- Open Computing Language
  - Multiple CPU machines with multiple co-processors, all from different vendors, can work together.
- One code can be executed on CPUs, GPUs, DSPs, FPGA and hardware
  - Dynamically interrogate system load and balance work across available processors
- OpenCL = Two APIs and Kernel language
  - C Platform Layer API to query, select and initialize compute devices
  - C Runtime API to build and execute kernels across multiple devices



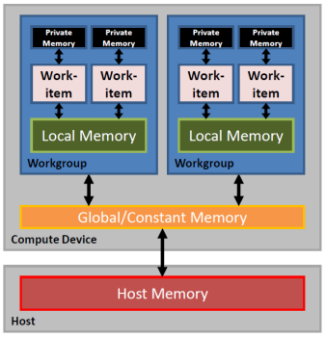
4

OpenCL platform & memory model

- Platform model
  - one host + one or more compute devices (CD)
  - each compute device is composed of one or more compute units (CU)
  - each compute unit is further divided into one or more processing units (PU)

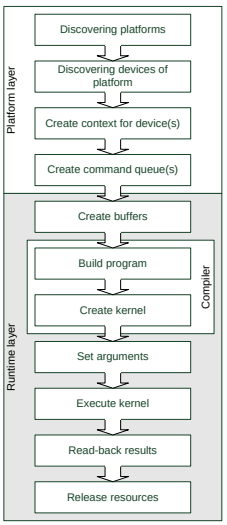


- Memory model
  - Host memory
  - Global memory
  - Constant memory
  - Local memory
  - Private memory



5

OpenCL programming and design flow

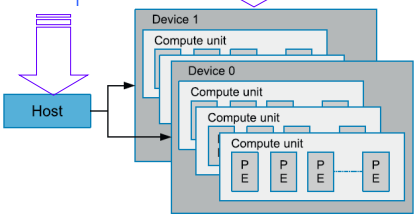


Host program

```
#include <CL/cl.h>
int main(){
  clGetPlatformIDs(…);
  clGetDeviceIdIDs(…);
  clCreateContext(…):
  clCreateCommandQueue(…);
  clCreateProgramWithSource(…);
  clBuildProgram(…);
  clCreateKernel(…);
  clCreateBuffer(…);
  clEnqueueWriteBuffer(…);
  clSetKernelArg(…);
  clEnqueueNDRangeKernel(…);
  clEnqueueReadBuffer(…);
  clRelease(…);
}
```

Kernel

```
__kernel
void vecadd(__global int *A,
            __global int *B,
            __global int *C)
{
   int idx = get_global_id(0);
   C[idx] = A[idx] + B[idx];
}
```



6

OpenCL C language restrictions for kernel

- Key restrictions
  - No function pointer
  - No bit-fields
  - No variable length arrays
  - No recursion
  - No standard headers
- Data types

Scalar Type	Vector Type (n = 2, 4, 8, 16)	API Type for host app
char, uchar	charn, ucharn	cl_char<n>, cl_uchar<n>
short, ushort	shortn, ushortn	cl_short<n>, cl_ushort<n>
int, uint	intn, uintn	cl_int<n>, cl_uint<n>
long, ulong	longn, ulongn	cl_long<n>, cl_ulong<n>
float	floatn	cl_float<n>

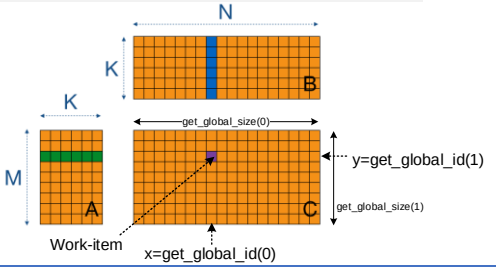
- Address space qualifiers
  - __global
    - memory objects allocated in global memory pool
  - __local
    - fast local memory pool
    - sharing between work-items
  - __constant
    - read-only allocation in global memory pool
  - __private
    - accessible by work-item
    - kernel arguments are private

7

Matrix multiplication OpenCL kernel

```
__kernel
void multiply ( __global float *outputC
              , __global float *inputA
              , __global float *inputB
              ,        int  widthA // K
              ,        int  widthB)  // N
{
   int x = get_global_id(0);
   int y = get_global_id(1);
   float result = 0.0;
   for (int i=0; i<widthA; i++) {
      result += inputA[y*widthA+i]*inputB[i*widthB+x];
   }
   outputC[y*widthB+x] = result;
}
```

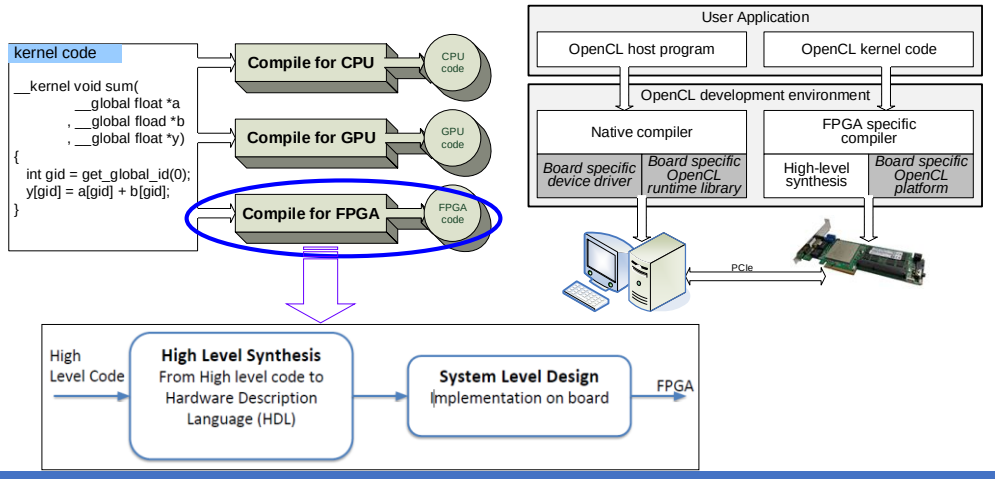
built in functions of kernels regarding work-item
- get_work_dim(): number of dimensions in use
- get_global_id(dim): unque index of a work-item
- get_global_size(dim): number of global work-items
- get_local_id(dim): unique index of the work-item within the work-group
- get_local_size(dim): number of work-items within the work-group
- get_group_id(dim): index of the work-group
- get_num_groups(dim): number of work-group
- Note that the size of work-groups or work-items cannot vary during a kernel call



8

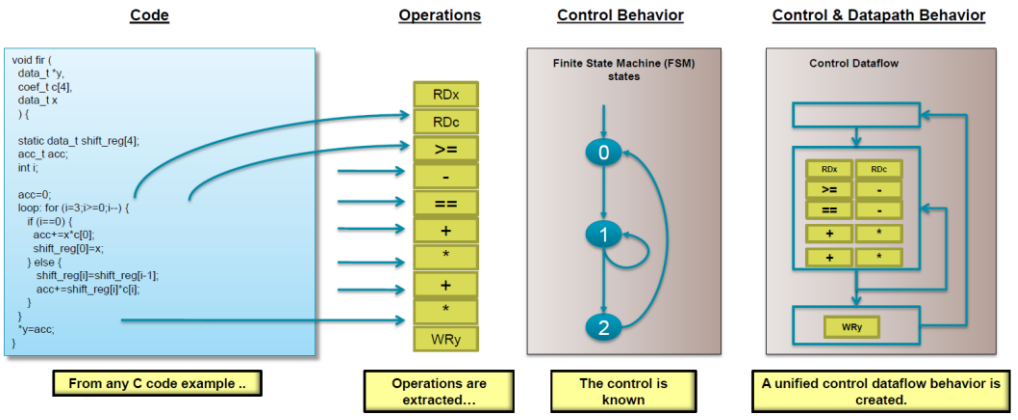
OpenCL and FPGA

- This requires C/C++ to RTL translation to get FPGA specific implementation.

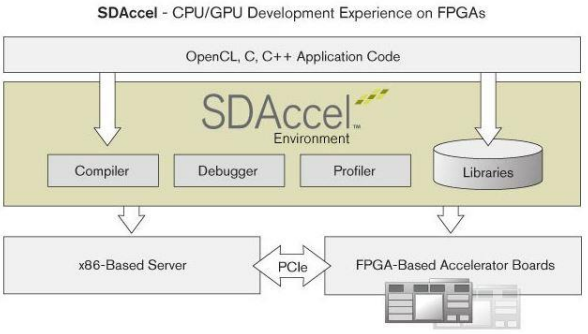


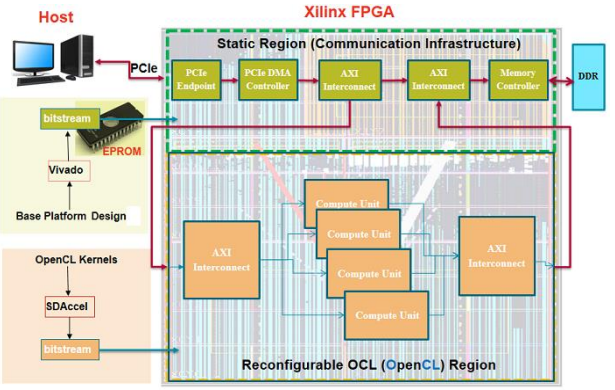
What HLS does

- HLS: High Level Synthesis
  - Operator extraction, control & datapath synthesis



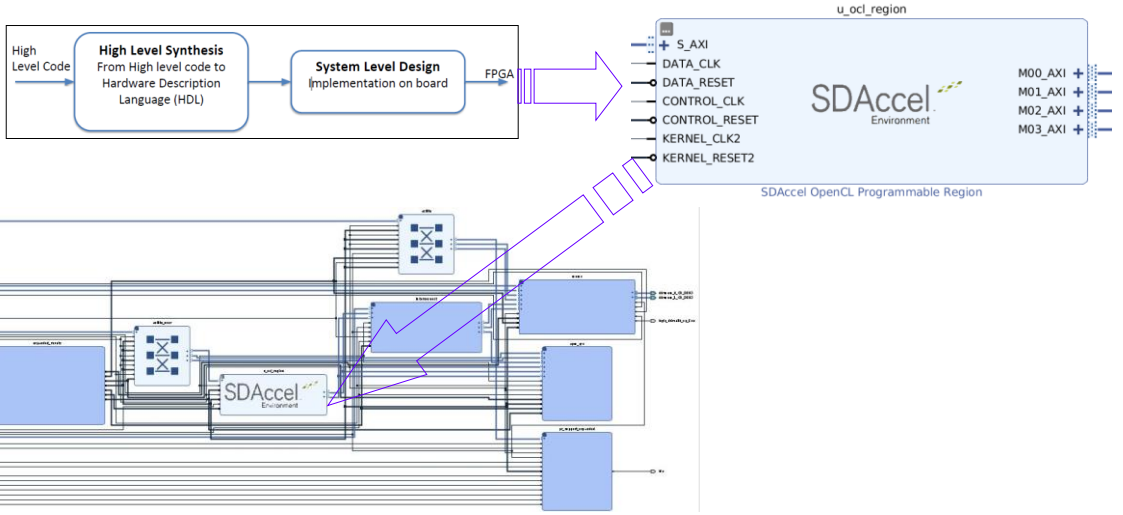
OpenCL port to Xilinx FPGA: SDAccel





11

Example case

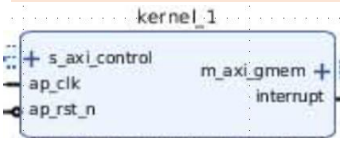


12

Kernel expression

- SDAccel kernel can be written in
  - OpenCL C
  - C/C++
  - RTL (Verilog or VHDL).
- OpenCL C kernel
  - specify work group size for better implementation
  - must be called from the host as an NDRange kernel

SDAccel OpenCL kernel module




```
__kernel
__attribute__((reqd_work_group_size(10,1,1)))
void vector_add ( __global const float *in_a
                , __global const float *in_b
                , __global float *out)
{
    size_t idx = get_global_id(0);
    out[idx] = in_a[idx] + in_b[idx];
}
```

13

Kernel expression

- C/C++ kernel
  - use standard AXI master and AXI Lite interface as for Vivdo HLS
  - include the kernel code within an extern “C” block
  - must be called from the host as a simple task
  - a function with a void return value
  - Global variable is not supported

SDAccel OpenCL kernel module


```
extern "C" {
   void vector_add(float *in_a, float *in_b, float *out)   {
     #pragma HLS INTERFACE m_axi depth=10 port=in_a bundle=gemm0
    #pragma HLS INTERFACE m_axi depth=10 port=in_b bundle=gemm0
    #pragma HLS INTERFACE m_axi depth=10 port=out bundle=gemm0

     #pragma HLS INTERFACE s_axilite register port=in_a bundle=control
    #pragma HLS INTERFACE s_axilite register port=in_b bundle=control
    #pragma HLS INTERFACE s_axilite register port=out bundle=control
     #pragma HLS INTERFACE s_axilite register port=return bundle=control

      for (int i=0; i<100; i++) {
            #pragma HLS_PIPELINE
            out[i] = in_a[i] + in_b[i];
       }
   }
}
```

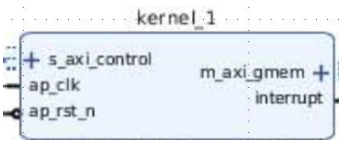
14

Kernel expression

- RTL kernel

```
module krnl_vadd_rtl #(
  parameter integer  C_S_AXI_CONTROL_DATA_WIDTH = 32,
  parameter integer  C_S_AXI_CONTROL_ADDR_WIDTH = 6,
  parameter integer  C_M_AXI_GMEM_ID_WIDTH = 1,
  parameter integer  C_M_AXI_GMEM_ADDR_WIDTH = 64,
  parameter integer  C_M_AXI_GMEM_DATA_WIDTH = 32)
(   // System signals
  input  wire  ap_clk,
  input  wire  ap_rst_n,
  // AXI4 master interface
  output wire      m_axi_gmem_AWVALID,
  ....
  // AXI4-Lite slave interface
  input  wire      s_axi_control_AWVALID,
);
...
endmodule
```

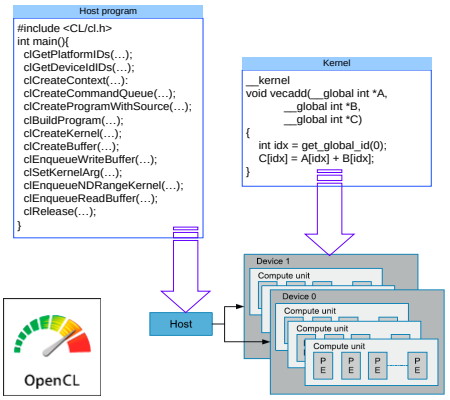
Why 'in_a', 'in_b' and 'out' are not shown?

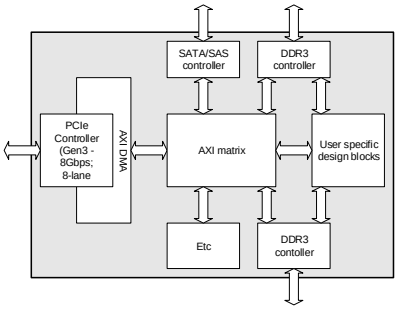


15

Future Design Systems' solution (1/2)

Heterogeneous computing, Reconfigurable computing
Big-data, machine-learning, deep-learing
OpenCL™ (Open Computing Language)





16

Future Design Systems' solution (2/2)

- PCB : 22 layer
  - Power (8) & GND (5), signal (9)
- Power:
  - 0.85V(VCCINT), 0.9V, 1.2V, 1.8V, 2.5V, 3.3V
  - 150W / FPGA
  - 300W / board (75W/PCIe+75W/6PIN+150W/8PIN)
- FPGA
  - Two Xilinx Virtex UltraScale+ VU9P
- Memory
  - DDR4: MT40A256M16 (256M x 16bit = 4Gbit/chip)
  - 512MByte / chip
  - 4 chips / channel → 2GByte / channel
  - 3 channels / FPGA → 6GByte / FPGA
  - 2 FPGA / board → 12GByte / board
- 5x performance & 16x performance/watt compare to GPU-TitanX
  - 3K CUDA cores @ 1GHz
  - 6 Teraflops FP32

17