■ Vitis と HBM

HLS で HBM アクセスする場合のバンド幅とレイテンシについて実機確認とか文献調査とか.

Alveo U50 で測ってみた結果は次のような感じ.
特にケアしない (むしろ HBM をあえて分けて，足をひっぱってる) 場合に読み書き 30MBps/500 MBps で,
これをビット幅増やしたり burst_length 指定すると 490MBps/7050MBps に速度向上.
ビット幅そのままで bundle 指定して AXI を分割すると読み書き 1170GBps/1060GBps くらいで,
ビット幅を考慮して bundle もちゃんと指定すると 10GBps 程度出る.

(Alveo U280 で)uBench 使っても読み書き 10GBps 程度はでてることが確認できる.

すごく腑抜けたコード

Hello World 的なこんなコード.

```
miyo@dev-8800:~/vitis_au50/vadd_2$ cat vadd.cpp
extern "C" {
    void vadd(int count,
              int* a_0, int* b_0, int* c_0
              );
}

void vadd(int count,
          int* a_0, int* b_0, int* c_0
          )
{
#pragma HLS INTERFACE s_axilite port=count bundle=control
//
#pragma HLS INTERFACE m_axi     port=a_0 offset=slave
#pragma HLS INTERFACE s_axilite port=a_0 bundle=control
#pragma HLS INTERFACE m_axi     port=b_0 offset=slave
#pragma HLS INTERFACE s_axilite port=b_0 bundle=control
#pragma HLS INTERFACE m_axi     port=c_0 offset=slave
#pragma HLS INTERFACE s_axilite port=c_0 bundle=control
//
#pragma HLS INTERFACE s_axilite port=return bundle=control

    for(int i = 0; i < count; i++){
#pragma HLS PIPELINE
        c_0[i] = a_0[i] + b_0[i];
    }
}
```

HBM 分けてみたかったので，design.cfg はこんな感じ.

```
platform=xilinx_u50_gen3x16_xdma_201920_3
debug=1
profile_kernel=data:all:all:all

[connectivity]
nk=vadd:1:vadd_1
sp=vadd_1.a_0:HBM[0]
sp=vadd_1.b_0:HBM[1]
sp=vadd_1.c_0:HBM[2]
```

で，実行結果をプロファイルで見た.

腑抜けたコードだとここまで遅いのか ...

ちょっと気の利いたようなコードを書いてみる

```
#include <ap_int.h>

extern "C" {
    void vadd(int count,
              ap_uint<512>* a_0, ap_uint<512>* b_0, ap_uint<512>* c_0
              );
}

void vadd(int count,
          ap_uint<512>* a_0, ap_uint<512>* b_0, ap_uint<512>* c_0
          )
{
#pragma HLS INTERFACE s_axilite port=count bundle=control
//
#pragma HLS INTERFACE m_axi     port=a_0 offset=slave
#pragma HLS INTERFACE s_axilite port=a_0 bundle=control
#pragma HLS INTERFACE m_axi     port=b_0 offset=slave
#pragma HLS INTERFACE s_axilite port=b_0 bundle=control
#pragma HLS INTERFACE m_axi     port=c_0 offset=slave
#pragma HLS INTERFACE s_axilite port=c_0 bundle=control
//
#pragma HLS INTERFACE s_axilite port=return bundle=control

    ap_uint<512> tmp_a_0, tmp_b_0, tmp_c_0;
    int x[16];
    int y[16];
    int z[16];

    int num = count / 4; // 間違えた

#pragma HLS DATAFLOW
    for(int i = 0; i < num; i++){
#pragma HLS PIPELINE II=1
        tmp_a_0 = a_0[i];
        tmp_b_0 = b_0[i];
        for(int j = 0; j < 16; j++){
            tmp_c_0(j*32+31, j*32) = tmp_a_0.range(j*32+31,j*32) + tmp_b_0.range(j*32+31,j*32);
        }
        c_0[i] = tmp_c_0;
    }

}
```

オリジナルの vadd.c から，512bit 単位でメモリ読み書きするように変更.
Number of Transfers の値がおかしいのは，num の計算を間違ってるから.
それでもスループットは少しよくなった.

burst の指定をしてみた

```
#include <ap_int.h>

extern "C" {
    void vadd(int count,
              ap_uint<512>* a_0, ap_uint<512>* b_0, ap_uint<512>* c_0
              );
}

void vadd(int count,
          ap_uint<512>* a_0, ap_uint<512>* b_0, ap_uint<512>* c_0
          )
{
#pragma HLS INTERFACE s_axilite port=count bundle=control
//
#pragma HLS INTERFACE m_axi     port=a_0 offset=slave max_read_burst_length=16
#pragma HLS INTERFACE s_axilite port=a_0 bundle=control
#pragma HLS INTERFACE m_axi     port=b_0 offset=slave max_read_burst_length=16
#pragma HLS INTERFACE s_axilite port=b_0 bundle=control
#pragma HLS INTERFACE m_axi     port=c_0 offset=slave max_write_burst_length=16
#pragma HLS INTERFACE s_axilite port=c_0 bundle=control
//
```

```
#pragma HLS INTERFACE s_axilite port=return bundle=control

    ap_uint<512> tmp_a_0, tmp_b_0, tmp_c_0;

    int x[16];
    int y[16];
    int z[16];

    int num = count / (512/32);

    for(int i = 0; i < num; i++){
#pragma HLS unroll factor=16
#pragma HLS PIPELINE II=1
        tmp_a_0 = a_0[i];
        tmp_b_0 = b_0[i];
        for(int j = 0; j < 16; j++){
            tmp_c_0(j*32+31, j*32) = tmp_a_0.range(j*32+31,j*32) + tmp_b_0.range(j*32+31,j*32);
        }
        c_0[i] = tmp_c_0;
    }

}
```

max_{read,write}_burst_length を指定して，ついでにアンロールしてみた.
書く方は，だいぶ速くなった，かな．読む方は回数も多いしうまくいってない．残念

ポートを分ける

元のコードで同じ m_axi ポート使ってるのはダメ，という指摘を受けたので，bundle で分割.

```
extern "C" {
    void vadd(int count,
              int* a_0, int* b_0, int* c_0
              );
}

void vadd(int count,
          int* a_0, int* b_0, int* c_0
          )
{
#pragma HLS INTERFACE s_axilite port=count bundle=control
//
#pragma HLS INTERFACE m_axi     port=a_0 offset=slave bundle=gmem0
#pragma HLS INTERFACE s_axilite port=a_0 bundle=control
#pragma HLS INTERFACE m_axi     port=b_0 offset=slave bundle=gmem1
#pragma HLS INTERFACE s_axilite port=b_0 bundle=control
#pragma HLS INTERFACE m_axi     port=c_0 offset=slave bundle=gmem2
#pragma HLS INTERFACE s_axilite port=c_0 bundle=control
//
#pragma HLS INTERFACE s_axilite port=return bundle=control

    for(int i = 0; i < count; i++){
#pragma HLS PIPELINE
        c_0[i] = a_0[i] + b_0[i];
    }
}
```

速くなった !!

ビット幅と bundle の両方をケア

ビット幅を考慮しつつ (とりあえずの 512bit 幅), bundle でポート分けるのも忘れないバージョン

```
#include <ap_int.h>
```

```
extern "C" {
    void vadd(int count,
              ap_uint<512>* a_0, ap_uint<512>* b_0, ap_uint<512>* c_0
              );
}

void vadd(int count,
          ap_uint<512>* a_0, ap_uint<512>* b_0, ap_uint<512>* c_0
          )
{
#pragma HLS INTERFACE s_axilite port=count bundle=control
//
#pragma HLS INTERFACE m_axi     port=a_0 offset=slave bundle=gmem0
#pragma HLS INTERFACE s_axilite port=a_0 bundle=control
#pragma HLS INTERFACE m_axi     port=b_0 offset=slave bundle=gmem1
#pragma HLS INTERFACE s_axilite port=b_0 bundle=control
#pragma HLS INTERFACE m_axi     port=c_0 offset=slave bundle=gmem2
#pragma HLS INTERFACE s_axilite port=c_0 bundle=control
//
#pragma HLS INTERFACE s_axilite port=return bundle=control

    ap_uint<512> tmp_a_0, tmp_b_0, tmp_c_0;
    int x[16];
    int y[16];
    int z[16];

    int num = count / 16;

#pragma HLS DATAFLOW
    for(int i = 0; i < num; i++){
#pragma HLS PIPELINE II=1
        tmp_a_0 = a_0[i];
        tmp_b_0 = b_0[i];
        for(int j = 0; j < 16; j++){
            tmp_c_0(j*32+31, j*32) = tmp_a_0.range(j*32+31,j*32) + tmp_b_0.range(j*32+31,j*32);
        }
        c_0[i] = tmp_c_0;
    }
}
```

読み書きとも 10GBps 近くなった !!

ビット幅と bundle 両方をケア /burst_length も指定

```
#include <ap_int.h>

extern "C" {
    void vadd(int count,
              ap_uint<512>* a_0, ap_uint<512>* b_0, ap_uint<512>* c_0
              );
}

void vadd(int count,
          ap_uint<512>* a_0, ap_uint<512>* b_0, ap_uint<512>* c_0
          )
{
#pragma HLS INTERFACE s_axilite port=count bundle=control
//
#pragma HLS INTERFACE m_axi     port=a_0 offset=slave max_read_burst_length=16 bundle=gmem0
#pragma HLS INTERFACE s_axilite port=a_0 bundle=control
#pragma HLS INTERFACE m_axi     port=b_0 offset=slave max_read_burst_length=16 bundle=gmem1
#pragma HLS INTERFACE s_axilite port=b_0 bundle=control
#pragma HLS INTERFACE m_axi     port=c_0 offset=slave max_write_burst_length=16 bundle=gmem2
#pragma HLS INTERFACE s_axilite port=c_0 bundle=control
//
#pragma HLS INTERFACE s_axilite port=return bundle=control

    ap_uint<512> tmp_a_0, tmp_b_0, tmp_c_0;

    int x[16];
    int y[16];
    int z[16];

    int num = count / (512/32);

    for(int i = 0; i < num; i++){
```

```
    #pragma HLS unroll factor=16
    #pragma HLS PIPELINE II=1
            tmp_a_0 = a_0[i];
            tmp_b_0 = b_0[i];
            for(int j = 0; j < 16; j++){
                tmp_c_0(j*32+31, j*32) = tmp_a_0.range(j*32+31,j*32) + tmp_b_0.range(j*32+31,j*32);
            }
            c_0[i] = tmp_c_0;
        }

    }
```

このケースだとわざわざ指定するまでもなかった，みたい．

uBench

uBench の ubench で測定．
コード見るかぎり，読みっぱなし，書きっぱなしのチャンピオンデータ測定をしているように見える．
ちなみに，実行環境は Alveo U280.

```
~/uBench/ubench/off-chip_bandwidth/read/HBM/2ports_512bit$ make check TARGET=hw DEVICE=xilinx_u280
-es1_xdma_201910_1 HOST_ARCH=x86 SYSROOT=/
    g++       -I../../../../..//common/includes/xcl2       -pthread       -I/opt/xilinx/xrt/include
-I/tools/Xilinx/Vivado/2019.2 /include     -Wall     -O0     -g     -std=c++11     -fmessage-length=0
../../../../..//common/includes/xcl2 /xcl2 .cpp    src/host.cpp    src/krnl_config.h      -o    'ubench'
-L/opt/xilinx/xrt/lib -lOpenCL -lpthread  -lrt -lstdc++
./ubench ./build_dir.hw.xilinx_u280-es1_xdma_201910_1/ubench.xclbin
Found Platform
Platform Name: Xilinx
INFO: Reading ./build_dir.hw.xilinx_u280-es1_xdma_201910_1/ubench.xclbin
Loading: './build_dir.hw.xilinx_u280-es1_xdma_201910_1/ubench.xclbin'
Trying to program device[0]: xilinx_u280-es1_xdma_201910_1
Device[0]: program successful!
Creating a kernel [krnl_ubench:{krnl_ubench_1}] for CU(1)
Execution time = 0.00300834
Payload Size: 3.8147e-06MB - Bandwidth = 6.80775GB/s
Execution time = 0.00419801
Payload Size: 3.8147e-06MB - Bandwidth = 9.75701GB/s
Execution time = 0.00746985
Payload Size: 3.8147e-06MB - Bandwidth = 10.9668GB/s
Execution time = 0.014406
Payload Size: 3.8147e-06MB - Bandwidth = 11.3731GB/s
Execution time = 0.029443
Payload Size: 3.8147e-06MB - Bandwidth = 11.1293GB/s
Execution time = 0.0565577
Payload Size: 3.8147e-06MB - Bandwidth = 11.5875GB/s
Execution time = 0.11245
Payload Size: 3.8147e-06MB - Bandwidth = 11.656GB/s
Execution time = 0.224376
Payload Size: 3.8147e-06MB - Bandwidth = 11.6833GB/s
Execution time = 0.44852
Payload Size: 3.8147e-06MB - Bandwidth = 11.6893GB/s
Execution time = 0.896489
Payload Size: 3.8147e-06MB - Bandwidth = 11.6965GB/s
Execution time = 1.79279
Payload Size: 3.8147e-06MB - Bandwidth = 11.6977GB/s
perf_analyze profile -i profile_summary.csv -f html
ERROR: Could not open file profile_summary.csv
Makefile:130: recipe for target 'check' failed
make: *** [check] Error 5


~/uBench/ubench/off-chip_bandwidth/write/HBM/2ports_512bit$ make check TARGET=hw DEVICE=xilinx_u280
-es1_xdma_201910_1 HOST_ARCH=x86 SYSROOT=/
    g++       -I../../../../..//common/includes/xcl2       -pthread       -I/opt/xilinx/xrt/include
-I/tools/Xilinx/Vivado/2019.2 /include     -Wall     -O0     -g     -std=c++11     -fmessage-length=0
../../../../..//common/includes/xcl2 /xcl2 .cpp    src/host.cpp    src/krnl_config.h      -o    'ubench'
-L/opt/xilinx/xrt/lib -lOpenCL -lpthread  -lrt -lstdc++
./ubench ./build_dir.hw.xilinx_u280-es1_xdma_201910_1/ubench.xclbin
Found Platform
Platform Name: Xilinx
```

```
INFO: Reading ./build_dir.hw.xilinx_u280-es1_xdma_201910_1/ubench.xclbin
Loading: './build_dir.hw.xilinx_u280-es1_xdma_201910_1/ubench.xclbin'
Trying to program device[0]: xilinx_u280-es1_xdma_201910_1
Device[0]: program successful!
Creating a kernel [krnl_ubench:{krnl_ubench_1}] for CU(1)
Creating a kernel [krnl_ubench:{krnl_ubench_2}] for CU(2)
Execution time = 0.00226169
Payload Size: 7.62939e-06MB - Bandwidth = 9.05517GB/s
Execution time = 0.00379374
Payload Size: 7.62939e-06MB - Bandwidth = 10.7967GB/s
Execution time = 0.0079286
Payload Size: 7.62939e-06MB - Bandwidth = 10.3322GB/s
Execution time = 0.0145442
Payload Size: 7.62939e-06MB - Bandwidth = 11.2649GB/s
Execution time = 0.02898
Payload Size: 7.62939e-06MB - Bandwidth = 11.3071GB/s
Execution time = 0.058234
Payload Size: 7.62939e-06MB - Bandwidth = 11.2539GB/s
Execution time = 0.116943
Payload Size: 7.62939e-06MB - Bandwidth = 11.2082GB/s
Execution time = 0.230614
Payload Size: 7.62939e-06MB - Bandwidth = 11.3672GB/s
Execution time = 0.463979
Payload Size: 7.62939e-06MB - Bandwidth = 11.2998GB/s
Execution time = 0.92165
Payload Size: 7.62939e-06MB - Bandwidth = 11.3772GB/s
Execution time = 1.84582
Payload Size: 7.62939e-06MB - Bandwidth = 11.3616GB/s
perf_analyze profile -i profile_summary.csv -f html
ERROR: Could not open file profile_summary.csv
Makefile:130: recipe for target 'check' failed
make: *** [check] Error 5


~/uBench/ubench/off-chip_latency/HBM/32bit_per_access$ make check TARGET=hw DEVICE=xilinx_u280-es1
_xdma_201910_1 HOST_ARCH=x86 SYSROOT=/
    g++        -I../../../..//common/includes/xcl2        -pthread        -I/opt/xilinx/xrt/include
-I/tools/Xilinx/Vivado/2019.2 /include     -Wall     -O0     -g     -std=c++11     -fmessage-length=0
../../../..//common/includes/xcl2 /xcl2 .cpp    src/host.cpp    src/krnl_config.h       -o    'ubench'
-L/opt/xilinx/xrt/lib -lOpenCL -lpthread  -lrt -lstdc++
./ubench ./build_dir.hw.xilinx_u280-es1_xdma_201910_1/ubench.xclbin
Found Platform
Platform Name: Xilinx
INFO: Reading ./build_dir.hw.xilinx_u280-es1_xdma_201910_1/ubench.xclbin
Loading: './build_dir.hw.xilinx_u280-es1_xdma_201910_1/ubench.xclbin'
Trying to program device[0]: xilinx_u280-es1_xdma_201910_1
Device[0]: program successful!
Creating a kernel [krnl_ubench:{krnl_ubench_1}] for CU(1)
Execution time = 0.00582883
Payload Size: 6.10352e-05MB - Latency = 0.109799GB/s
Execution time = 0.0113186
Payload Size: 0.00012207MB - Latency = 0.113088GB/s
Execution time = 0.0226948
Payload Size: 0.000244141MB - Latency = 0.112801GB/s
Execution time = 0.0449585
Payload Size: 0.000488281MB - Latency = 0.113883GB/s
Execution time = 0.0901793
Payload Size: 0.000976562MB - Latency = 0.113552GB/s
Execution time = 0.183912
Payload Size: 0.00195312MB - Latency = 0.111357GB/s
Execution time = 0.367893
Payload Size: 0.00390625MB - Latency = 0.111337GB/s
Execution time = 0.736642
Payload Size: 0.0078125MB - Latency = 0.111207GB/s
Execution time = 1.50362
Payload Size: 0.015625MB - Latency = 0.108963GB/s
Execution time = 3.21045
Payload Size: 0.03125MB - Latency = 0.102067GB/s
Execution time = 6.54912
Payload Size: 0.0625MB - Latency = 0.100068GB/s
Execution time = 13.2363
Payload Size: 0.125MB - Latency = 0.0990244GB/s
Execution time = 26.6355
Payload Size: 0.25MB - Latency = 0.0984188GB/s
Execution time = 53.4147
Payload Size: 0.5MB - Latency = 0.0981542GB/s
Execution time = 106.987
Payload Size: 1MB - Latency = 0.09801GB/s
perf_analyze profile -i profile_summary.csv -f html
ERROR: Could not open file profile_summary.csv
Makefile:130: recipe for target 'check' failed
```

```
make: *** [check] Error 5
```