■ Vitis と HBM(続)

複数 HBM リージョンにアクセスしてみる．bundle に気をつけて,
27 個の HBM リージョンにアクセス.
ちなみに, もう一組増やして 30 個利用しようとしたら P&R で配線できないというエラーがでた.

```
extern "C" {
    void vadd(int count,
              int* a_0, int* b_0, int* c_0,
              int* a_1, int* b_1, int* c_1,
              int* a_2, int* b_2, int* c_2,
              int* a_3, int* b_3, int* c_3,
              int* a_4, int* b_4, int* c_4,
              int* a_5, int* b_5, int* c_5,
              int* a_6, int* b_6, int* c_6,
              int* a_7, int* b_7, int* c_7,
              int* a_8, int* b_8, int* c_8);
}

void vadd(int count,
          int* a_0, int* b_0, int* c_0,
          int* a_1, int* b_1, int* c_1,
          int* a_2, int* b_2, int* c_2,
          int* a_3, int* b_3, int* c_3,
          int* a_4, int* b_4, int* c_4,
          int* a_5, int* b_5, int* c_5,
          int* a_6, int* b_6, int* c_6,
          int* a_7, int* b_7, int* c_7,
          int* a_8, int* b_8, int* c_8)
{
#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 m_axi     port=a_1 offset=slave bundle=gmem3
#pragma HLS INTERFACE s_axilite port=a_1 bundle=control
#pragma HLS INTERFACE m_axi     port=b_1 offset=slave bundle=gmem4
#pragma HLS INTERFACE s_axilite port=b_1 bundle=control
#pragma HLS INTERFACE m_axi     port=c_1 offset=slave bundle=gmem5
#pragma HLS INTERFACE s_axilite port=c_1 bundle=control
//
#pragma HLS INTERFACE m_axi     port=a_2 offset=slave bundle=gmem6
#pragma HLS INTERFACE s_axilite port=a_2 bundle=control
#pragma HLS INTERFACE m_axi     port=b_2 offset=slave bundle=gmem7
#pragma HLS INTERFACE s_axilite port=b_2 bundle=control
#pragma HLS INTERFACE m_axi     port=c_2 offset=slave bundle=gmem8
#pragma HLS INTERFACE s_axilite port=c_2 bundle=control
//
#pragma HLS INTERFACE m_axi     port=a_3 offset=slave bundle=gmem9
#pragma HLS INTERFACE s_axilite port=a_3 bundle=control
#pragma HLS INTERFACE m_axi     port=b_3 offset=slave bundle=gmem10
#pragma HLS INTERFACE s_axilite port=b_3 bundle=control
#pragma HLS INTERFACE m_axi     port=c_3 offset=slave bundle=gmem11
#pragma HLS INTERFACE s_axilite port=c_3 bundle=control
//
#pragma HLS INTERFACE m_axi     port=a_4 offset=slave bundle=gmem12
#pragma HLS INTERFACE s_axilite port=a_4 bundle=control
#pragma HLS INTERFACE m_axi     port=b_4 offset=slave bundle=gmem13
#pragma HLS INTERFACE s_axilite port=b_4 bundle=control
#pragma HLS INTERFACE m_axi     port=c_4 offset=slave bundle=gmem14
#pragma HLS INTERFACE s_axilite port=c_4 bundle=control
//
#pragma HLS INTERFACE m_axi     port=a_5 offset=slave bundle=gmem15
#pragma HLS INTERFACE s_axilite port=a_5 bundle=control
#pragma HLS INTERFACE m_axi     port=b_5 offset=slave bundle=gmem16
#pragma HLS INTERFACE s_axilite port=b_5 bundle=control
#pragma HLS INTERFACE m_axi     port=c_5 offset=slave bundle=gmem17
#pragma HLS INTERFACE s_axilite port=c_5 bundle=control
//
#pragma HLS INTERFACE m_axi     port=a_6 offset=slave bundle=gmem18
#pragma HLS INTERFACE s_axilite port=a_6 bundle=control
```

```
#pragma HLS INTERFACE m_axi     port=b_6 offset=slave bundle=gmem19
#pragma HLS INTERFACE s_axilite port=b_6 bundle=control
#pragma HLS INTERFACE m_axi     port=c_6 offset=slave bundle=gmem20
#pragma HLS INTERFACE s_axilite port=c_6 bundle=control
//
#pragma HLS INTERFACE m_axi     port=a_7 offset=slave bundle=gmem21
#pragma HLS INTERFACE s_axilite port=a_7 bundle=control
#pragma HLS INTERFACE m_axi     port=b_7 offset=slave bundle=gmem22
#pragma HLS INTERFACE s_axilite port=b_7 bundle=control
#pragma HLS INTERFACE m_axi     port=c_7 offset=slave bundle=gmem23
#pragma HLS INTERFACE s_axilite port=c_7 bundle=control
//
#pragma HLS INTERFACE m_axi     port=a_8 offset=slave bundle=gmem24
#pragma HLS INTERFACE s_axilite port=a_8 bundle=control
#pragma HLS INTERFACE m_axi     port=b_8 offset=slave bundle=gmem25
#pragma HLS INTERFACE s_axilite port=b_8 bundle=control
#pragma HLS INTERFACE m_axi     port=c_8 offset=slave bundle=gmem26
#pragma HLS INTERFACE s_axilite port=c_8 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];
        c_1[i] = a_1[i] + b_1[i];
        c_2[i] = a_2[i] + b_2[i];
        c_3[i] = a_3[i] + b_3[i];
        c_4[i] = a_4[i] + b_4[i];
        c_5[i] = a_5[i] + b_5[i];
        c_6[i] = a_6[i] + b_6[i];
        c_7[i] = a_7[i] + b_7[i];
        c_8[i] = a_8[i] + b_8[i];
    }
}
```

こんなコードを用意．design.cfg は,

```
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]
sp=vadd_1.a_1:HBM[3]
sp=vadd_1.b_1:HBM[4]
sp=vadd_1.c_1:HBM[5]
sp=vadd_1.a_2:HBM[6]
sp=vadd_1.b_2:HBM[7]
sp=vadd_1.c_2:HBM[8]
sp=vadd_1.a_3:HBM[9]
sp=vadd_1.b_3:HBM[10]
sp=vadd_1.c_3:HBM[11]
sp=vadd_1.a_4:HBM[12]
sp=vadd_1.b_4:HBM[13]
sp=vadd_1.c_4:HBM[14]
sp=vadd_1.a_5:HBM[15]
sp=vadd_1.b_5:HBM[16]
sp=vadd_1.c_5:HBM[17]
sp=vadd_1.a_6:HBM[18]
sp=vadd_1.b_6:HBM[19]
sp=vadd_1.c_6:HBM[20]
sp=vadd_1.a_7:HBM[21]
sp=vadd_1.b_7:HBM[22]
sp=vadd_1.c_7:HBM[23]
sp=vadd_1.a_8:HBM[24]
sp=vadd_1.b_8:HBM[25]
sp=vadd_1.c_8:HBM[26]
```

ビルドして実行.

という感じで HBM がたくさん並んだ.
リソース使用量は LUT と Register がそれぞれ 6,453 と 9,417．BRAM と DSP は使ってない.

実行時のメモリバンド幅をみてみると，それぞれ 1GBps を越える性能がでてる．よかった．

512bit 幅で読み書きするように修正．

```
    const int num = count / 16;


    ap_uint<512> tmp_a_0, tmp_b_0, tmp_c_0;
    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;
    }
```

バンド幅は 8GBps 〜 10.5GBpsっていう感じ
リソース使用量は，LUT とレジスタが，それぞれ，49,778 個と 111,190 個．
また BRAM を 207 個 (15.4% 相当) 利用している．

複数の HBM リージョンを使う．

C++ コードはかわらず，たとえば，

```
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];
    }
}
```

で，design.cfg で，

```
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[0:1]
sp=vadd_1.b_0:HBM[2:3]
sp=vadd_1.c_0:HBM[4:5]
```

とかする.

という感じに複数の HBM リージョンをぶらさげることができる.