■ AFU/PAC 開発メモ その 1

まずは Hello World に毛をはやす.
サンプルの hw/samples/hello_afu/build_synth は，PAC 上にレジスタを作って
SW から読み書きする，というもの.
レジスタを一つ追加してみる.

```
    logic [63:0] scratch_reg;
    logic [63:0] scratch_reg2; // 追加
```

を用意して，レジスタ番号は 32bit 単位っぽいので，64bit 幅の場合，2 飛びで指定.

```
    always_ff @(posedge clk)
    begin
        if (reset)
        begin
            scratch_reg <= '0;
            scratch_reg2 <= '0;
        end
        else
        begin
            // set the registers on MMIO write request
            // these are user-defined AFU registers at offset 0x40.
            if (cp2af_sRxPort.c0.mmioWrValid == 1)
            begin
                case (mmioHdr.address)
                    16'h0020: scratch_reg <= cp2af_sRxPort.c0.data[63:0];
                    16'h0022: scratch_reg2 <= cp2af_sRxPort.c0.data[63:0] * 2; // 追加
                endcase
            end
        end
    end
```

SW から書いた値を 2 倍して保存してみることに.
読む方にも対応したアドレスへのアクセスを追加

```
    //
    // Handle MMIO reads.
    //
    always_ff @(posedge clk)
    begin
        if (reset)
        begin
            af2cp_sTxPort.c1.hdr <= '0;
            af2cp_sTxPort.c1.valid <= '0;
            af2cp_sTxPort.c0.hdr <= '0;
            af2cp_sTxPort.c0.valid <= '0;
            af2cp_sTxPort.c2.hdr <= '0;
            af2cp_sTxPort.c2.mmioRdValid <= '0;
        end
        else
        begin
... 略 ...
                    // Scratch Register.  Return the last value written
                    // to this MMIO address.
                    16'h0020: af2cp_sTxPort.c2.data <= scratch_reg;
                    16'h0022: af2cp_sTxPort.c2.data <= scratch_reg2; // 追加
... 略 ...
        end
    end
```

Quartus 上で合成だけ通しておきたければ, Quartus プロジェクトの build_synth/build/dcp.qpf を開いて,
afu_synth をベースに afu_dev を作って Analysis&Synthesis だけ実行しておくといい.
gbs 作るときには，特にファイルを追加したりしているわけではないので，build_synth の下で,

```
% ${OPAE_PLATFORM_ROOT}/bin/run.sh
```

とすれば，それでいい．
ソフトウェアからのアクセスではバイト単位でアドレス指定する必要があって，

```
/* test */
res = fpgaWriteMMIO64(afc_handle, 0, 0x88, 0xa5a5a5a5a5a5a5a5);
res = fpgaReadMMIO64(afc_handle, 0, 0x88, &data);
printf("[test] Reading Scratch Register (Byte Offset=%08x) = %08lx\n", 0x88, data);
res = fpgaReadMMIO64(afc_handle, 0, 0x80, &data);
printf("[test] Reading Scratch Register (Byte Offset=%08x) = %08lx\n", 0x80, data);
```

こんな感じに．

- 0x88(PAC 上では 0x22) に 0xa5a5a5a5a5a5a5a5 を書いて読むと，2 倍された 0x14b4b4b4b4b4b4b4 が読める
- 0x88 にデータを書いても，0x80 には影響ない

ことが確認できて OK．