現状報告

2012/8/10

京都ATLAS meeting

田代 拓也

testbench

- VME 読出を使って、system完成
- 仕様

  - PPGでBW,SWのpatternを作成
    → 任意のpatternを入力可能

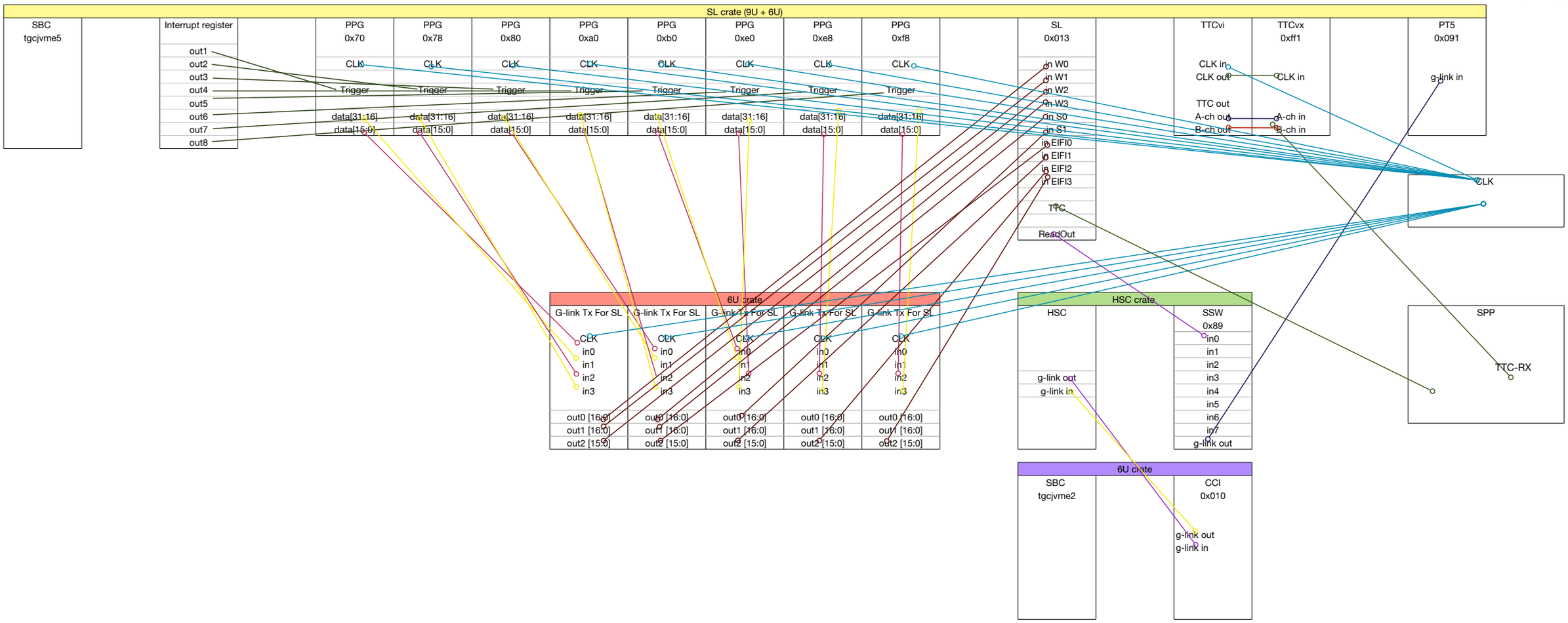
  - PSBの出力(160 bit)をSSW,PT5で読出
    → 入力との比較により、動作検証

KEK と同じ仕様

- PT5からのS-link読出に関しては、cableの取り替えを行った → cableの問題でないことだけ確認

testbench 配線

ver.1 2012/08/02



現状ではHSC crateのみPSB testbenchと共存中

Online software

設定file(新registerの定義,初期値の代入)準備

bash

```
// Delay
static const unsigned int NUM_BW_ENDCAP_INPUT = NUM_LINK_ENDCAP/2;
static const unsigned int NUM_BW_FORWARD_INPUT = NUM_LINK_FORWARD/2;
static const unsigned int NUM_EIFI_INPUT = NUM_LINK_EIFI;
static const TgcVmeApiTm::Register DELAY_BW_0;
static const TgcVmeApiTm::Register DELAY_BW_1;
static const TgcVmeApiTm::Register DELAY_BW_2;
static const TgcVmeApiTm::Register DELAY_BW_3;
static const TgcVmeApiTm::Register DELAY_BW_4;
static const TgcVmeApiTm::Register DELAY_BW_5;
static const TgcVmeApiTm::Register DELAY_BW[NUM_BW_ENDCAP_INPUT];
static const TgcVmeApiTm::Register DELAY_EIFI_0;
static const TgcVmeApiTm::Register DELAY_EIFI_1;
static const TgcVmeApiTm::Register DELAY_EIFI_2;
static const TgcVmeApiTm::Register DELAY_EIFI_3;
static const TgcVmeApiTm::Register DELAY_EIFI[NUM_EIFI_INPUT];
static const TgcVmeApiTm::Register DELAY_BCR;
static const TgcVmeApiTm::Register DELAY_NIM;

// Mask
static const TgcVmeApiTm::Register MASK_SSC;
static const TgcVmeApiTm::Register MASK_HL;
static const TgcVmeApiTm::Register MASK_NIM;
static const TgcVmeApiTm::Register MASK_EIFI;
static const TgcVmeApiTm::Register MASK_WIRE;
static const TgcVmeApiTm::Register MASK_STRIP;

// Prescale
static const unsigned int NUM_SSC_FORWARD = 8;
static const unsigned int NUM_SSC_ENDCAP = 19;
static const unsigned int NUM_SSC = NUM_SSC_ENDCAP;
static const TgcVmeApiTm::Register PRESCALE_NIM;
static const TgcVmeApiTm::Register PRESCALE_SSC[NUM_SSC];

// Misc
static const TgcVmeApiTm::Register INHIBIT_TRIGGER_SETTING;
static const TgcVmeApiTm::Register PASS_THROUGH;
static const TgcVmeApiTm::Register ALLOW_CONSECUTIVE_TRIGGER;
// phase0
static const TgcVmeApiTm::Register INNER_COINCIDENCE_FLAG;
static const TgcVmeApiTm::Register INNER_PT_FLAG;
static const TgcVmeApiTm::Register INNER_ROI_FLAG_0;
static const TgcVmeApiTm::Register INNER_ROI_FLAG_1;
static const TgcVmeApiTm::Register INNER_ROI_FLAG_2;
static const TgcVmeApiTm::Register INNER_ROI_FLAG_3;
static const TgcVmeApiTm::Register INNER_ROI_FLAG_4;
};

endif // TgcSectorLogicApiTm_hpp
* eof */
```

新register定義

gcSectorLogicApiTm_tashiro.hpp 322,5 Bot

NewRegisters

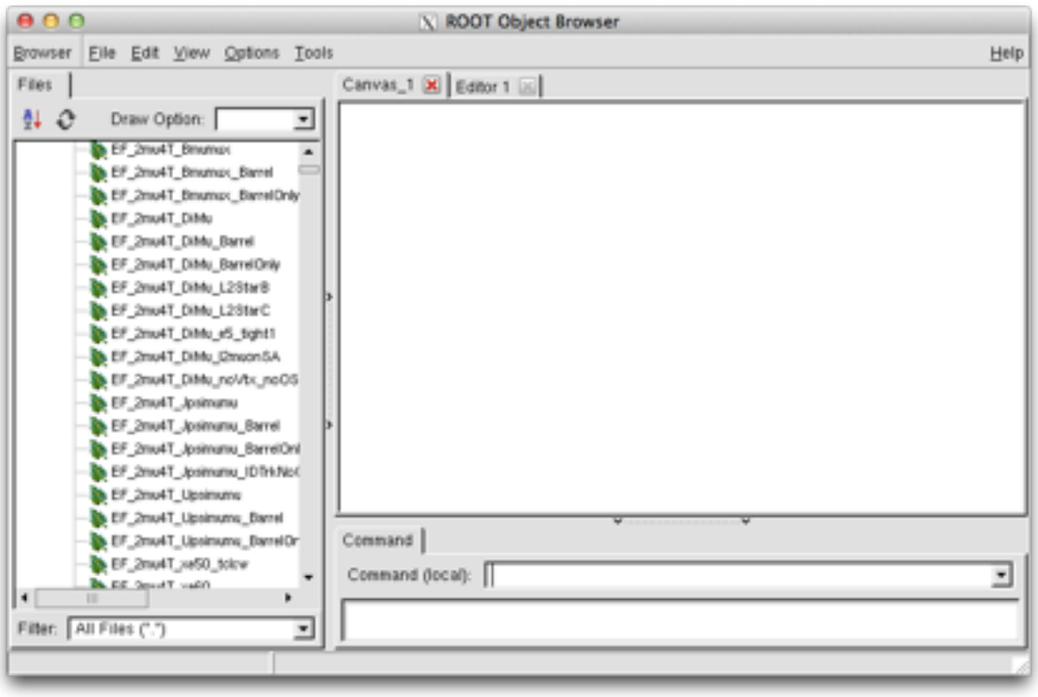
```
  {0xd4, "PRESCALE_SSC_12", 0xffffffff, 0x0, 32},
  {0xd8, "PRESCALE_SSC_13", 0xffffffff, 0x0, 32},
  {0xdc, "PRESCALE_SSC_14", 0xffffffff, 0x0, 32},
  {0xe0, "PRESCALE_SSC_15", 0xffffffff, 0x0, 32},
  {0xe4, "PRESCALE_SSC_16", 0xffffffff, 0x0, 32},
  {0xe8, "PRESCALE_SSC_17", 0xffffffff, 0x0, 32},
  {0xec, "PRESCALE_SSC_18", 0xffffffff, 0x0, 32},
};



//
// misc
//
const TgcVmeApiTm::Register
TgcSectorLogicApiTm::INHIBIT_TRIGGER_SETTING = {0x148, "INHIBIT_TRIGGER_SETTING", 0xffffffff, 10000,
32};

const TgcVmeApiTm::Register
TgcSectorLogicApiTm::PASS_THROUGH = {0x44, "PASS_THROUGH", 0x1, 0x0, 1};

const TgcVmeApiTm::Register
TgcSectorLogicApiTm::ALLOW_CONSECUTIVE_TRIGGER = {0x194, "ALLOW_CONSECUTIVE_TRIGGER", 0x3, 0x3, 2};

//
// phase0
//
const TgcVmeApiTm::Register
TgcSectorLogicApiTm::INNER_COINCIDENCE_FLAG = {0x198, "INNER_COINCIDENCE_FLAG", 0x1, 0x0, 1};

const TgcVmeApiTm::Register
TgcSectorLogicApiTm::INNER_PT_FLAG = {0x19c, "INNER_PT_FLAG", 0xff, 0xff, 8};

const TgcVmeApiTm::Register
TgcSectorLogicApiTm::INNER_ROI_FLAG_0 = {0x1a0, "INNER_ROI_FLAG_0", 0xffffffff, 0xffffffff, 32};

const TgcVmeApiTm::Register
TgcSectorLogicApiTm::INNER_ROI_FLAG_1 = {0x1a4, "INNER_ROI_FLAG_1", 0xffffffff, 0xffffffff, 32};

const TgcVmeApiTm::Register
TgcSectorLogicApiTm::INNER_ROI_FLAG_2 = {0x1a8, "INNER_ROI_FLAG_2", 0xffffffff, 0xffffffff, 32};

const TgcVmeApiTm::Register
TgcSectorLogicApiTm::INNER_ROI_FLAG_3 = {0x1ac, "INNER_ROI_FLAG_3", 0xffffffff, 0xffffffff, 32};

const TgcVmeApiTm::Register
TgcSectorLogicApiTm::INNER_ROI_FLAG_4 = {0x1b0, "INNER_ROI_FLAG_4", 0xffffffff, 0xffffffff, 32};
// ------------------------------------------------------------------
// Constructors and Destructor
// ------------------------------------------------------------------
TgcSectorLogicApiTm::TgcSectorLogicApiTm(const TgcVmeApiTm* sbc)
```

新register定義

TgcSectorLogicApiTm_tashiro.cc 174,4 21

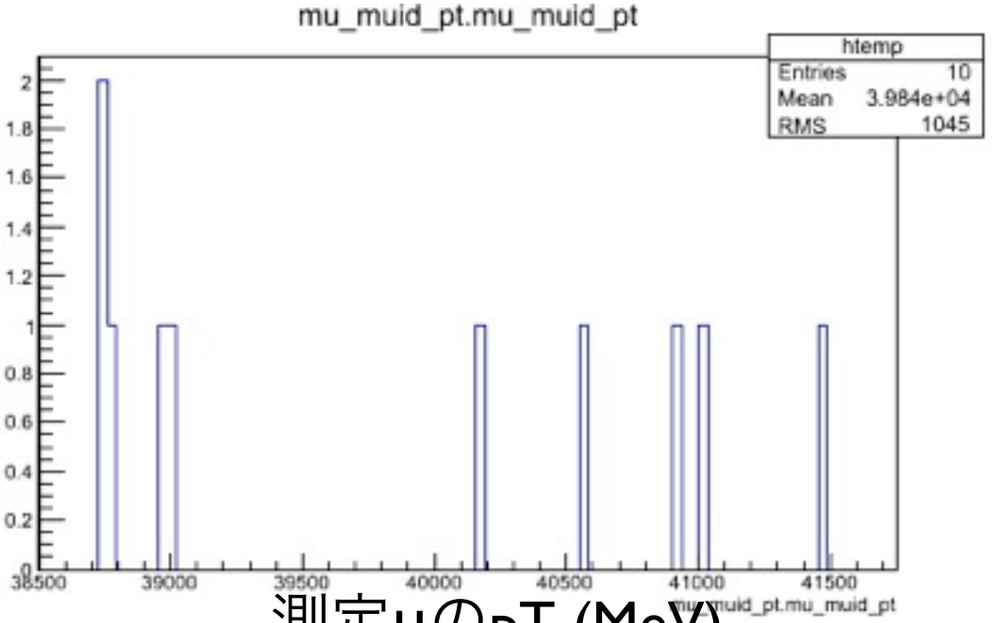
trigT1TGC

- 基本的な使い方習得
  - muon generation
  → hit simulation
  → digitization
  → reconstruction
  →DPD

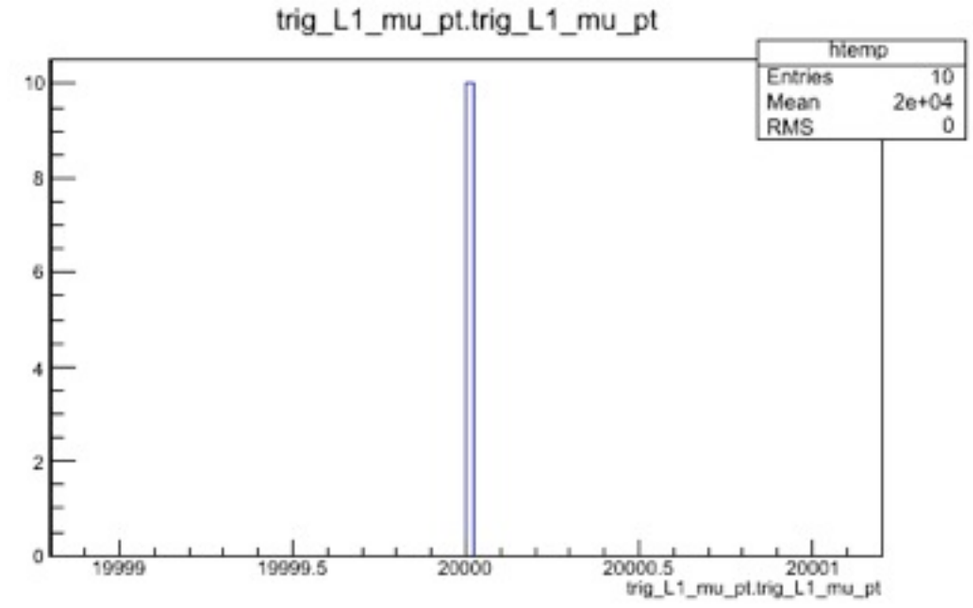


作ったDPD

例 : 40 GeV μ 10個



測定μのpT (MeV)



μのpT trigger (MeV)