Trigger state struct draft

Ilnur Gabdrakhmanov

JINR, VBLHEP

Dubna June 22, 2020

Main objectives:

- Save trigger logic in the compact and universal form
- Trigger states must be comparable to each other
- Flexible for extension
- Working for any logical expression

```
struct __attribute__ ((packed)) BmnTrigStruct
{
    bool BC1;
    bool BC2;
    bool BC3;
    bool VETO; // means not VETO
    UShort_t :8; // not used
    UShort_t ThrBD:5;
    UShort_t ThrSI:5;
    UShort_t :10; // not used
};
```

Beam Counters:

$$\begin{cases} \wedge BC2 & BC2 == true \\ indifferent & BC2 == false \end{cases}$$

BD and SI triggers:

$$\begin{cases} \wedge (BD >= 3) & ThrBD == 3 \\ indifferent & ThrBD == 0 \end{cases}$$

Example:

$$BC1 \wedge BC2 \wedge \overline{VETO} \wedge (BD >= 2) \wedge (SI >= 3)$$

```
BmnTrigStruct s;
s.BC1 = true;
s.BC2 = true;
s.VETO = true;
s.ThrBD = 2;
s.ThrSI = 3;
```

Bit scheme

BIT №	0	1	2	3	4..11	12..16	17..21	22..31
Name	BC1	BC2	BC3	VETO		BD	SI	
Length	1	1	1	1	8	5	5	10

- √ Volume = 32 bits
- √ Can be extended by additional 8 beam counters and 2 BD-like triggers
- × Does not support neither $\overline{BC}$ not || operation

Backup Slides

```
class BmnTrigState {
public:
    BmnTrigState();
    BmnTrigState(BmnTrigState &s);

    virtual ~BmnTrigState();

    Bool_t operator==(BmnTrigState &s) {
        Bool_t ret = kTRUE;
        ret = ret && (BC1 == s.BC1);
        ret = ret && (BC2 == s.BC2);
        ret = ret && (BC3 == s.BC3);
        ret = ret && (VETO == s.VETO);

        ret = ret && (ThrBD == s.ThrBD);
        ret = ret && (ThrSI == s.ThrSI);
        return ret;
    }

protected:
    // BC1:
    // +1 == " and BC1"
    //  0 == " indifferent to BC1"
    // -1 == " and not BC1"
    Int_t BC1;
    Int_t BC2;
    Int_t BC3;
    Int_t VETO;

    // ThrBD == v means "BD >= v"
    Int_t ThrBD;
    Int_t ThrSI;

};
```

Beam Counters:

$$\begin{cases} \wedge BC2 & BC2 == 1 \\ indifferent & BC2 == 0 \\ \wedge \overline{BC2} & BC2 == -1 \end{cases}$$

BD and SI triggers:

$$\begin{cases} \wedge (BD >= 3) & ThrBD == 3 \\ indifferent & ThrBD == 0 \end{cases}$$

Example:

$$BC1 \wedge BC2 \wedge \overline{VETO} \wedge (BD >= 2) \wedge (SI >= 3)$$

```
BmnTrigState s;
s.BC1 = 1;
s.BC2 = 1;
s.VETO = -1;
s.ThrBD = 2;
s.ThrSI = 3;
```

Main objectives:

- ≈ Save trigger logic in the compact and universal form
- √ Trigger states must be comparable to each other
- × Flexible for extension
- × Working for any logical expression

Better solution should be:

- > Parse trigger expression into logical tree
- > Create functions for simplification and comparisons of these trees