TCU Upgrade Strawman

Jeff Landgraf (jml@bnl.gov)
6/7/06

Design Criteria

The criteria I propose for the new TCU is:

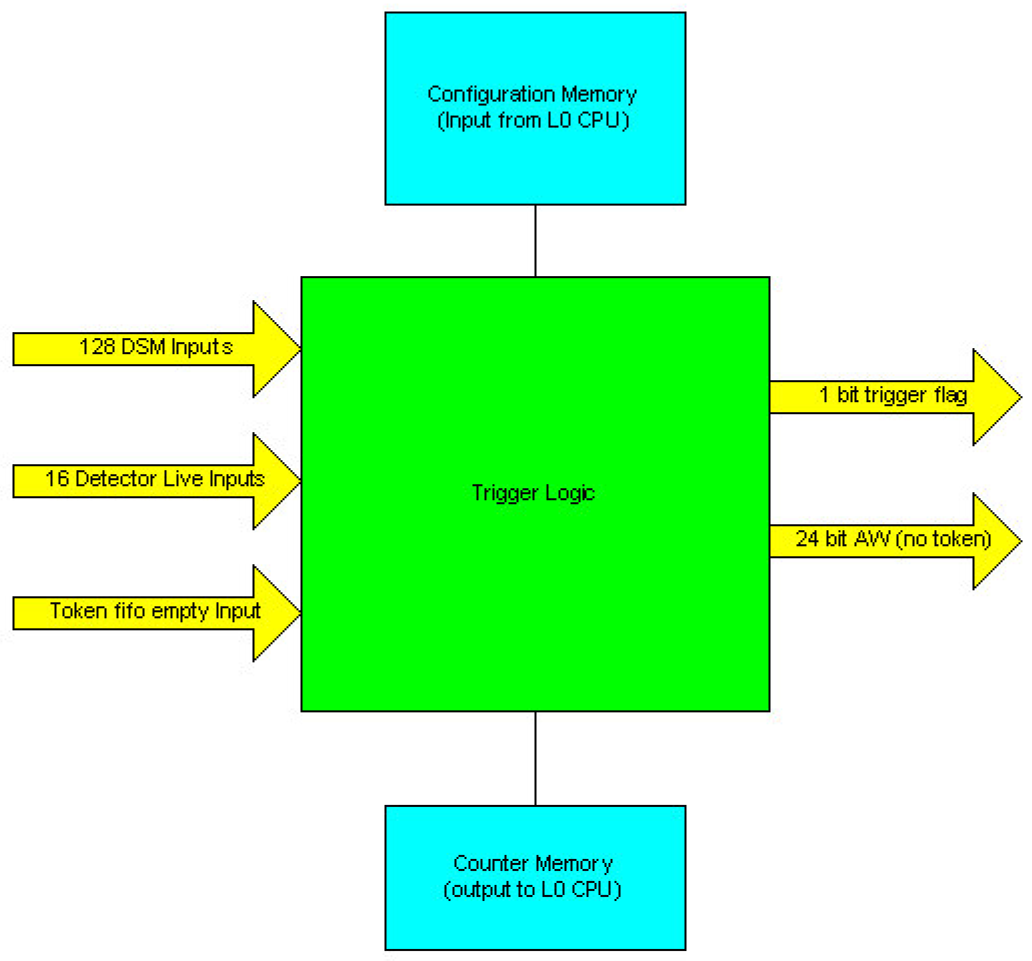
1. The interfaces have minimal changes:
    a. TCU interface stays identical
    b. DSM interface changes slightly in that more "lastDSM" bits are supported. At least 32 bits should be supported. If at all possible, I would support a larger number up to 128.
    c. I expect the interface to the L0 cpu would remain through VME, although there would be some shifting around as regards the width of fifo's and the memory maps for the counters.
    d. The run control interface would remain the same for the shift. The configuration code in L1 would be tremendously simplified.
    e. Need support more triggers. At least 64 are neccessary & 128 preferable.
2. Remove the PW and TW lookup tables.
3. Prescale counters need to be at least 32 bits
4. The counters in the TCU should be capable of obtaining the full accounting of the trigger deadtimes & luminosities. Effectively, they can and should duplicate formatted output of the "Tonko Scaler board."

The second criteria is the key. Obviously, the lookup table mechanism does not scale. Already, the current TCU had to break the TW LUT into the PW & TW tables to fit into real memory. At 32 input bits, we would need 4Gig, just for the PW LUT. At 128 bits we would need 3.4 x 10^23 Petabytes, which is significantly more than the amount of memory existing on earth.

The removal of the lookup tables will also make configuration much easier, reduce the number of different tier1 files we need to use and eliminate numerous complex work-a-rounds and esoteric limitations in the current trigger.

Overview

I'm going to propose logic to provide the trigger decision only and action word only and various counters. I assume that the token handling and distribution of signals remains the same I am currently assuming that all of this logic will live on a single FPGA. However, I don't make any assumptions about what else might live on this FPGA. In particular the token handling logic could also be there.



I will detail the configuration parameters later, but the approximate size of the configuration memory (assuming 128 DSM input bits & 128 triggers) is 20k.

I'm not certain what the appropriate counter memory size will be. It depends to some extent on how powerful the device is. For the minimum mix we would need ~3k (assuming 128 DSM bits & 128 triggers). For a slightly more complete mix we would need up to 2MB of counter memory.

The algorithms are primarily implemented in passive logic. The number of logic gates should be ~650,000 on the assumption of 1 gate per Boolean operation.

Additionally, there are a few pieces of programmed logic. This would be 128 instances of the pre-scale logic for each trigger, which requires minimal counter memory ~< 1k. Also, the logic to implement the counters.

My impression is that these requirements easily fit on a large, modern FPGA, but I don't have any experience in this area.

Design

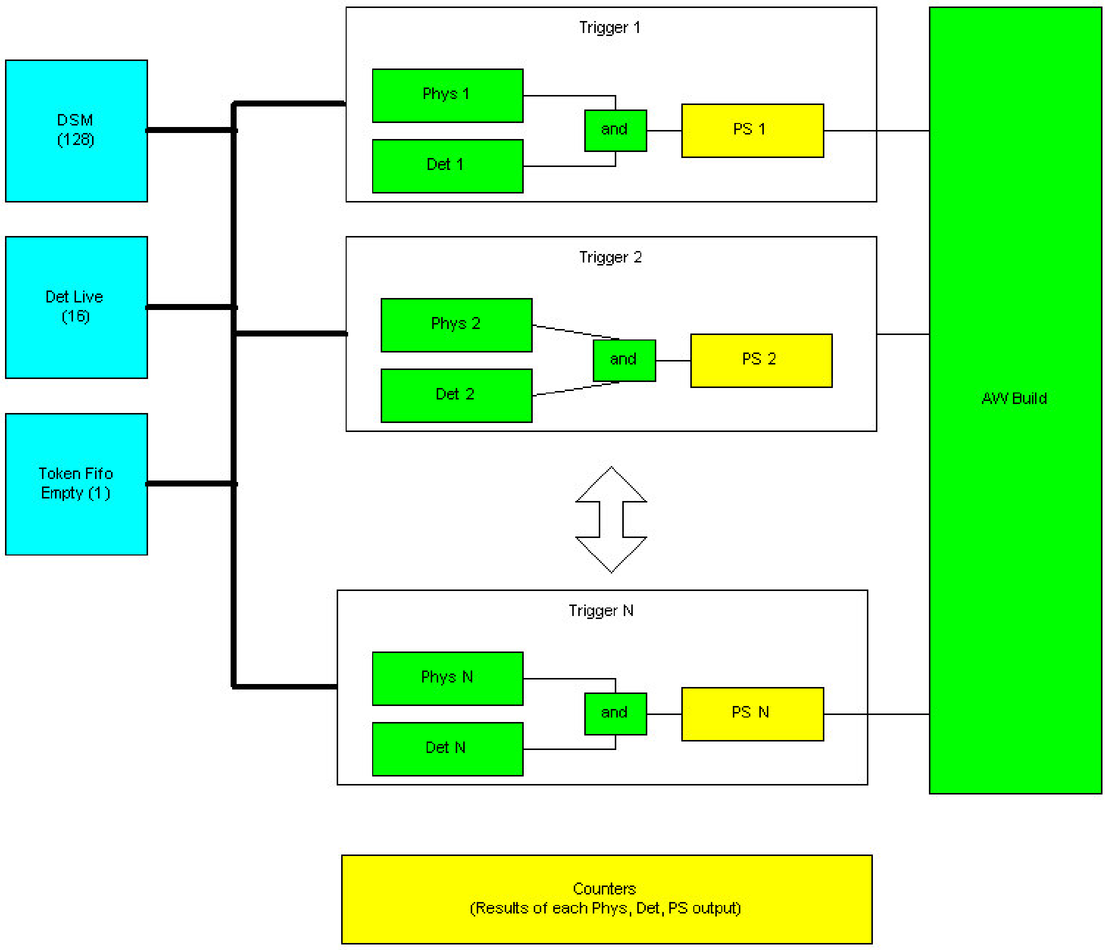
The logic separates naturally by trigger. It is natural, though not necessary, for the maximum number of triggers to be the same as the number of DSM bits.

The “Phys” logic compares the DSM input bits to the requirement specified in the run control. This requirement, for each trigger, is specified by 4 conditions which each consist of a 128 bit mask for on and off bits as specified by the run control. The output is a single bit which is on if the physics is satisfied & off if not. The output is routed to a 40 bit counter, and it is “anded” with the det result before being passed to the PS component.

The “Det” logic compares the 16 detector live bits to the two 16 bit masks for detector live and dead as specified by the run control. The output is a single bit which is routed to a 40 bit counter, as well as “anded” with the phys result before being passed to the PS component.

The “PS” component decrements the prescale counters. Its output is a single bit, which is routed to a 40 bit counter as well as to the “AW Build” component.

The “AW build” component constructs the action word. It calculates the logical “or” of all the detector requests, trigger commands & DAQ commands from all satisfied triggers.



“Phys” component

This component determines whether the DSM bits from this bunch satisfy the trigger. There is one for each trigger. I use the following indexes:

> “bit” denotes the DSM input bit [0-127]
> “trg” denotes the trigger [0-127]
> “cond” denotes the trigger condition [0-3

The DSM bit values I denote:

```
DSM(bit)
```

This component uses 4 configuration parameters:

```
TE(trg)                 // Is the trigger configured?
CE(cond)(trg)           // Condition enabled
ON(trg)(cond)(bit)      // condition onbits
OFF(trg)(cond)(bit)     // condition offbits
```

Then the logic for this component is:

```
PHYS(trg) = TE(trg) & CS(trg)(0) &
             CS(trg)(1) & CS(trg)(2) & CS(trg)(3)
```

Where CS stands for “Condition Satisfied” and is defined as:

```
CS(trg)(x) = !CE(trg)(cond) |
              ( ( !ON(trg)(x)(0) | DSM(0) ) &
                ( !ON(trg)(x)(1) | DSM(1) ) &
                  ...  &
                ( !ON(trg)(x)(127) | DSM_(127) ) &
               ( !OFF(trg)(x)(0) | !DET(0) ) &
               ( !OFF(trg)(x)(1) | !DET(1)) &
                ...  &
               ( !OFF(trg)(x)(127) | !DET(127) ) )
```

“Det” Component

There is one “DET” component for each trigger. The results of this component indicate whether the detector live/dead request is satisfied by the trigger. This component requires the additional index:

> “det” denotes the detector bit [0-16]

The live inputs are denoted

```
DET(det)              // detector live bits
TKN_EMPTY             // is the token fifo empty?
```

The configuration parameters are:

```
DON(trg)(det)
DOFF(trg)(det)
```

The logic is:

```
DET(trg) = TKN_EMPTY &
             ( !DON(trg)(0) | DET(0) ) &
             ( !DON(trg)(1) | DET(1) ) &
                 ... &
             ( !DON(trg)(15) | DET(15) ) &
             ( !DOFF(trg)(0) | !DET(0) ) &
             ( !DOFF(trg)(1) | !DET(1) ) &
                 ... &
             ( !DOFF(trg)(15) | !DET(15) )
```

"PS" component

There is one PS component for each trigger. The input to the PS component is the "AND" of the "PHYS" and "DET" components. If it is set, the trigger is satisfied. The PS component applies the prescale. The prescale value needs to be large enough to scale the rhic clock to a sub-hertz rate. We have used in L1 the concept of a floating point pre-scale, which is good for fine-tuning the bandwidth for many triggers. I propose implementing these in the hardware design as well. However, a simple pre-scale big enough to scale the rhic clock would be a vast improvement. The input bit I denote:

```
TS(trg)         // Trigger Satisfied
```

The output bit:

```
FIRE(trg)
```

The configuration parameters used are:

```
PS(trg)         // 32 bit register
PS_frac(trg)      // 8 bit register
```

Additionally, 2 temporary storage registers are needed,

```
PS_ctr(trg)    // 32 bit register
PS_ctr_frac(trg)   // 8 bit register
```

These must be initialized to the values of PS & PS_frac at the beginning of each run. The logic of the PS component is as follows:

```
if(TS(trg)) {
      if(PS_ctr(trg) == 0) {
            PS_ctr(trg) = PS(trg)-1;
            PS_ctr_frac(trg) += PS_frac(trg);
            if(PS_ctr_frac(trg) >= 100) {
                  PS_ctr(trg)++;
                  PS_ctr_frac(trg) -= 100;
            }
      }
      FIRE(trg) = true;
}
else {
      FIRE(trg) = false;
}

PS_ctr(trg)--;
```

“AW Build” Component

There is a single “AW” Build component. The inputs to this component are the outputs of the PS components:

FIRE(trg)

The configuration parameters are as follows:

```
DREQ(trg)(det)     // Detector fire requests
TCREQ(trg)(x)      // requested trgcmd (x=0-3)
DCREQ(trg)(x)      // requested daqcmd (x=0-3)
```

For simplicity I’ll combine these into a single 24-bit parameter for each trigger:

```
TREQ(trg)(x)        // x = (0-23)
```

The output of this component is a bit saying whether a trigger should be fired, as well as the action word request (actually only the detector request, trgcmd & daqcmd which can no be combined with the token to make the action word)

```
ISSUE_TRIGGER      // Should a trigger be issued
REQ(x)             // 24 bit piece of AW
```

Issue Trigger is simple to construct:

ISSUE_TRIGGER = FIRE(0) | FIRE(1) | ... | FIRE(127)

The action word is obtained from the logical or of each triggers request

```
REQ(x) = ( FIRE(0) & TREQ(trg)(x) ) |
          ( FIRE(1) & TREQ(trg)(x) ) |
               ... |
         ( FIRE(127) & TREQ(127)(x) )
```

This obtains the desired result for the detector mask. It does not give the right result for the trigger command if the triggers request different trigger commands. This would be a configuration error that must be handled by the configuration software.

Counters

The counters are a work in progress, but here are my initial thoughts.

1. Be able to readout counters periodically (~5-10 second intervals) for monitoring.
2. Counters should be 40 bits long, so we don’t worry about overflow.
3. Have a minimum set of:
    a. bunch crossings (1 counter)
    b. detector lives for each detector (16 counters)
    c. physics satisfied for each trigger (128 counters)
    d. combined detector live for each trigger (128 counters)

    e. events triggered for each trigger (128 counters)

4. To get full trigger overlaps is impossible... we would need 2^128 counters. But the first level might be useful.   This would be the 128x128 array of trigger coincidences for physics/det live/trigger issued.