MoGURA update

Benda XU

2012-03-15 Thu

Outline

the MoGURA ambition

Record what happens after a muon event, embrace it, study it, exploit it.
There are still many blocks in front:

dead time MoGURA

- Performance is not enough to record all the data we need

dead time MoGURA

- Performance is not enough to record all the data we need
- In other words, MoGURA has **dead time**, no different than FBE.

dead time MoGURA

- Performance is not enough to record all the data we need
- In other words, MoGURA has **dead time**, no different than FBE.
- It is shameful that MoGURA is designed to be dead time free.

time to reflect

TWO YEARS! We still couldn't make it dead time free!
Two ways if we want to keep going:

- hire another one who can not only code but also understand our physics motivation
- master the fancy technology in a reduced, easier way

I will cover the second point today.

index

1. Port Linux to
    - Virtex 2 Pro (user FPGA), powerpc405
    - Spartan 3/E (Front-End FPGA, System FPGA, Trigger Logic/System FPGA), microblaze
2. Reduce the hard task to a hierarchy of easy ones.
3. Use linux and software, starting from what is familiar. Test and explore the possibilities.
    - performance critical part: wrap current VHDL IP cores and manage them from CPU (and software running upon it)

      e.g. making frames, compress data, single trigger
    - complex house keeping part: software

      e.g. mode switch, register configuration, gluing IP cores

results: Linux on Virtex 2 Pro

- summary

ISE	10.1
Linux	3.0
toolchain	compiled
gcc	4.5.3
binutils	2.22
glibc	2.12
rootfs	nfs
CPU	PowerPC405
clock	100Mhz
console	RS232

- figure

Potential usage

- user FPGA on main board
- software level on-board control
- read data from system FPGA via board circuit and push out by Ethernets
- preliminary speed test
    - 100Mbps on board connected to 1Gbps host
    - standard FTP protocol
    - 629 KB/s, bottleneck being CPU

Results: Linux on Spartan 3E

- summary

ISE	13.2
Linux	3.0
toolchain	downloaded
gcc	4.1.2
binutils	2.16.1
glibc	2.3.6
rootfs	ramdisk
CPU	MicroBlaze
clock	50Mhz
console	RS232

- figure

Potential Usage

- system/front-end FPGA on main board
  - replace non time critical and complex housekeeping function with software
- FPGA on trigger board
  - trigger logic/system FPGA on trigger board
  - implement smart trigger logic (e.g. FFT, DWT) with software

experiment motivation

- FEF buffer becomes full when recording after muon events

experiment motivation

- FEF buffer becomes full when recording after muon events
- 1 system FPGA ~ 6 FEF, synchronized reading, bottlenect is obvious

experiment motivation

- FEF buffer becomes full when recording after muon events
- 1 system FPGA ˜ 6 FEF, synchronized reading, bottlenect is obvious
- In the test by Ishikawa (*Study of buffer-full on MoGURA board*, 2012, this meeting), it turns out that 1 system ˜ 2 FEF would be a balance

experiment motivation

- FEF buffer becomes full when recording after muon events
- 1 system FPGA ˜ 6 FEF, synchronized reading, bottlenect is obvious
- In the test by Ishikawa (*Study of buffer-full on MoGURA board*, 2012, this meeting), it turns out that 1 system ˜ 2 FEF would be a balance
- How about just enabling 2 FEF/board to record all the waveform? And reconstruct the events using one third of the PMTs – only energy resolution will suffer.

experiment motivation

- FEF buffer becomes full when recording after muon events
- 1 system FPGA ~ 6 FEF, synchronized reading, bottlenect is obvious
- In the test by Ishikawa (*Study of buffer-full on MoGURA board*, 2012, this meeting), it turns out that 1 system ~ 2 FEF would be a balance
- How about just enabling 2 FEF/board to record all the waveform? And reconstruct the events using one third of the PMTs – only energy resolution will suffer.
- We can also apply wavelet compression on a whole waveform to evaluate its performance

experiment settings (run mog 2259)

- main board

ChannelEnable	**0x01e**
ChannelHitEnable	**0xffe**
DiscriminatorSelect	1
EventWindowLength	24
MaximumSignalLength	24
TriggerLatency	94
FifoAlmostFullValue	**400**

- trigger

HitsumLatency	0
TriggerLatency	48
TriggerPrecedingLength	36
HitWindowLength	6
TriggerWindowLength	24
SingleThreshold	**70**
PrescaledThreshold	50
LaunchThreshold	**950**
SuccessiveInterval	24
SuccessiveWindowLength	200

BANG!! No LaunchTrigger Captured!

- Errr. Another bug is found! (ok, it is a feature)

BANG!! No LaunchTrigger Captured!

- Errr. Another bug is found! (ok, it is a feature)
- Even if we enabled all the channels for hit, it seems that only 2 FEFs was used for hit! ChannelEnable overwrites ChannelHitEnable?!

BANG!! No LaunchTrigger Captured!

- Errr. Another bug is found! (ok, it is a feature)
- Even if we enabled all the channels for hit, it seems that only 2 FEFs was used for hit! ChannelEnable overwrites ChannelHitEnable?!
- Evidence: for run mog 2256, ChannelEnable = all. mean clocks between two normal event segments: 7×10^5 run 2259: 23×10^5

BANG!! No LaunchTrigger Captured!

- Errr. Another bug is found! (ok, it is a feature)
- Even if we enabled all the channels for hit, it seems that only 2 FEFs was used for hit! ChannelEnable overwrites ChannelHitEnable?!
- Evidence: for run mog 2256, ChannelEnable = all. mean clocks between two normal event segments: 7×10^5 run 2259: 23×10^5
- Roughly 3 times difference, that's 6 FEF vs 2 FEF

BANG!! No LaunchTrigger Captured!

- Errr. Another bug is found! (ok, it is a feature)
- Even if we enabled all the channels for hit, it seems that only 2 FEFs was used for hit! ChannelEnable overwrites ChannelHitEnable?!
- Evidence: for run mog 2256, ChannelEnable = all. mean clocks between two normal event segments: 7×10^5 run 2259: 23×10^5
- Roughly 3 times difference, that's 6 FEF vs 2 FEF
- What to do next? Spend *one month* to debug this or try to find a dirty workaround? Or just do something else? It's hard to decide and soon we get exhausted of indecision.

further impove history trigger

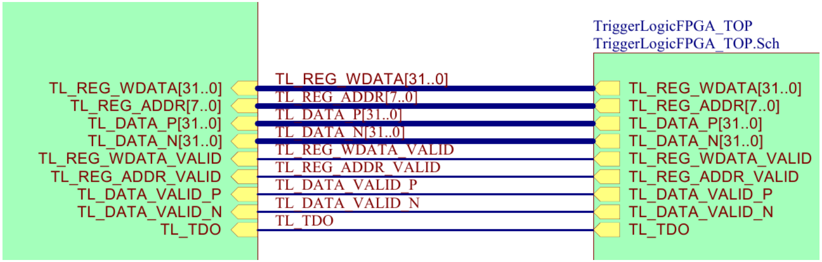

Figure: src: trigger board schematics rev 1.01 by inrevium

- present scheme: TSF read from 32 pins of TLF in every clock and store them in SDRAM

further impove history trigger (cont.)

- hitsum and trigger records overlap
- buffer them separately
- hardware lz77 compression to record more data[1]

Example (src: TLF specification by Sanshiro)

```
when Trigger Window End, write
    1: x"01" & TRIGGER_COMMAND
    2: x"02" & TRIGGER_FLAGS_ALL
    3: x"03" & NHITS_MAX
    4: x"04" & TIMESTAMP(47 downto 24)
    5: x"05" & TIMESTAMP(23 downto 0)
```

[1] http://opencores.org/project,deflatecore

wavelet

- analyze hitsum history
- similar to a low pass filter (either difference or substract moving average)

ethernet out

- MoGURA do not have ethernet built in, only 10 pins from user FPGA are available

ethernet out

- MoGURA do not have ethernet built in, only 10 pins from user FPGA are available
- the motivation is to circumvent VME readout noise, observed during $CdWO_4$ runs and reported by Sanshiro-san (*MoGURA Peripherals*, 2011, Tuscalusa Meeting)

ethernet out

- MoGURA do not have ethernet built in, only 10 pins from user FPGA are available
- the motivation is to circumvent VME readout noise, observed during $CdWO_4$ runs and reported by Sanshiro-san (*MoGURA Peripherals*, 2011, Tuscalusa Meeting)
- idea is immature, we don't know the nature of this noise yet. What if it resides in reading SDRAM? SDRAM readout noise instead...