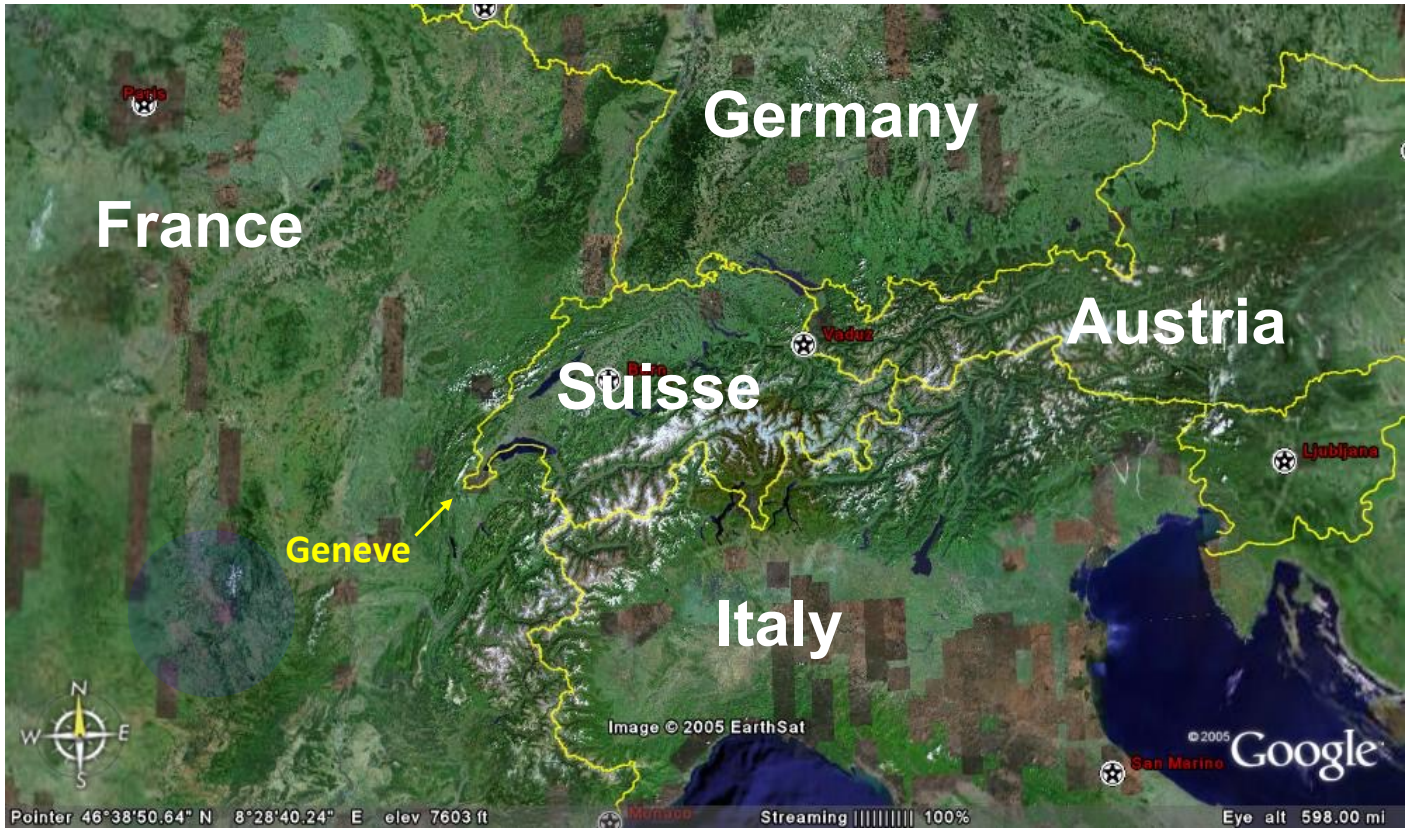
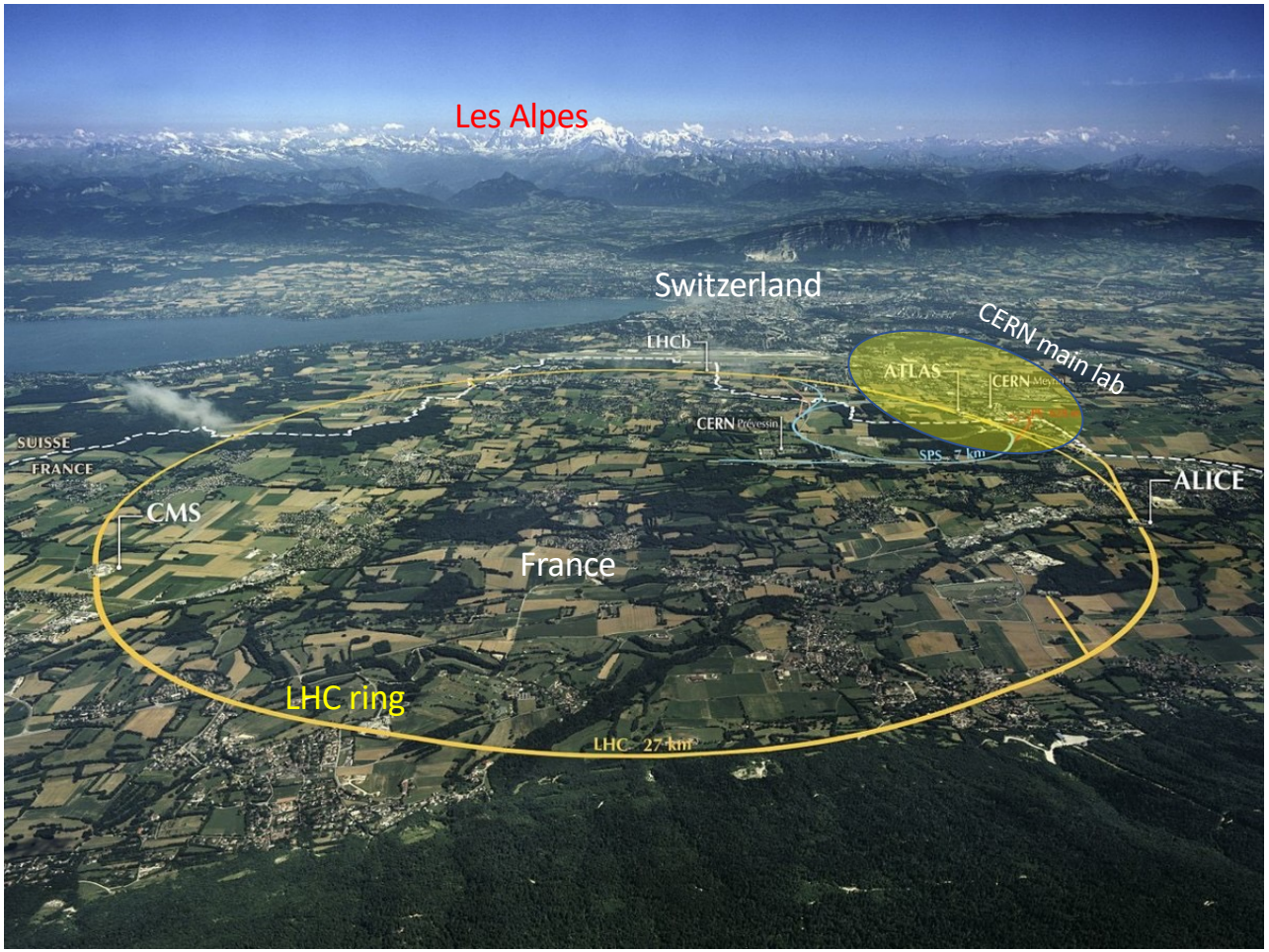
”Triggers” and FPGAs in Particle Physics

Drew Baden

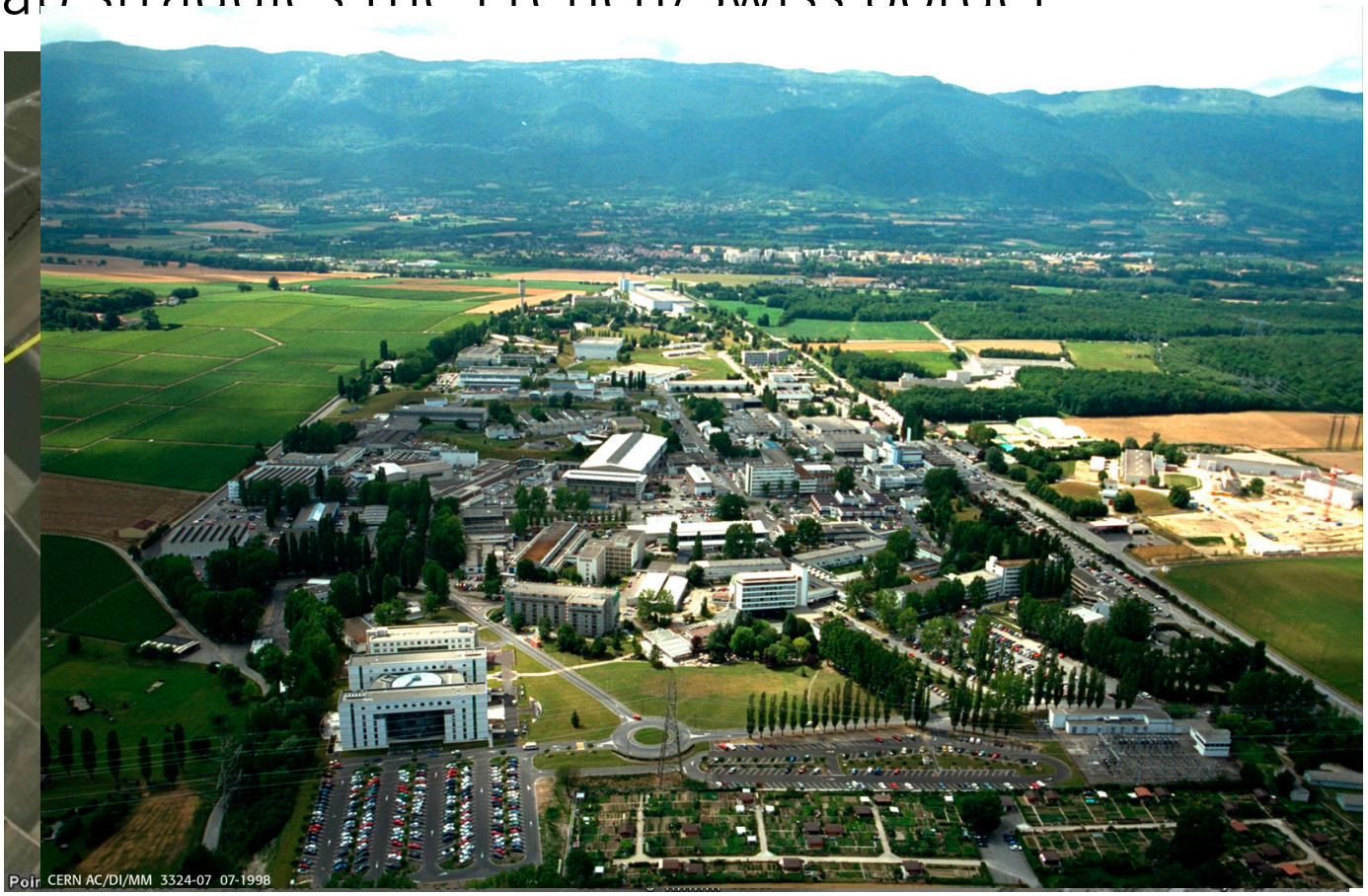
CERN: Conseil European pour la Reserche Nucleaire
or
European Council for Nuclear Research



CERN laboratory

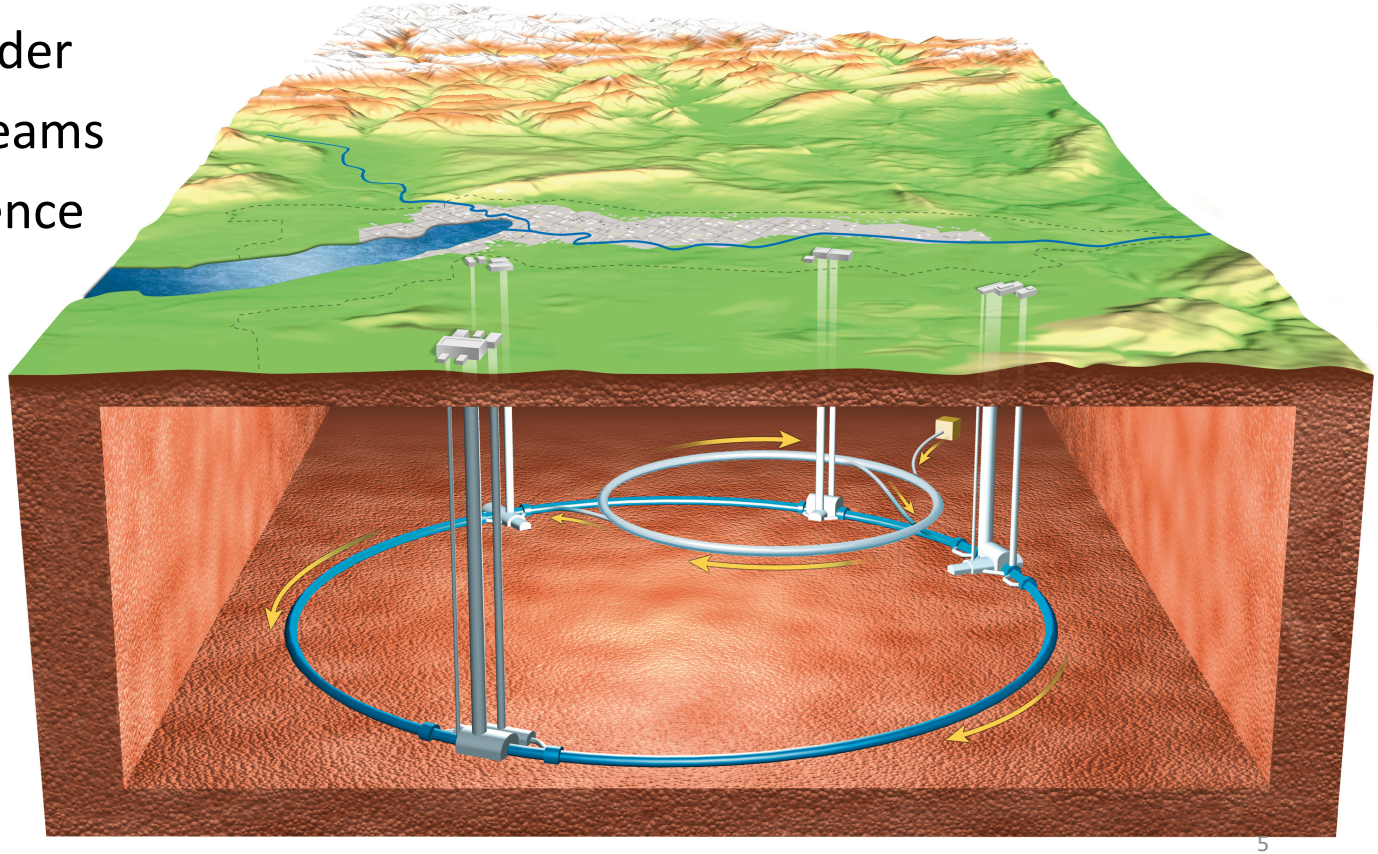


CERN Lab straddles the French/Swiss border



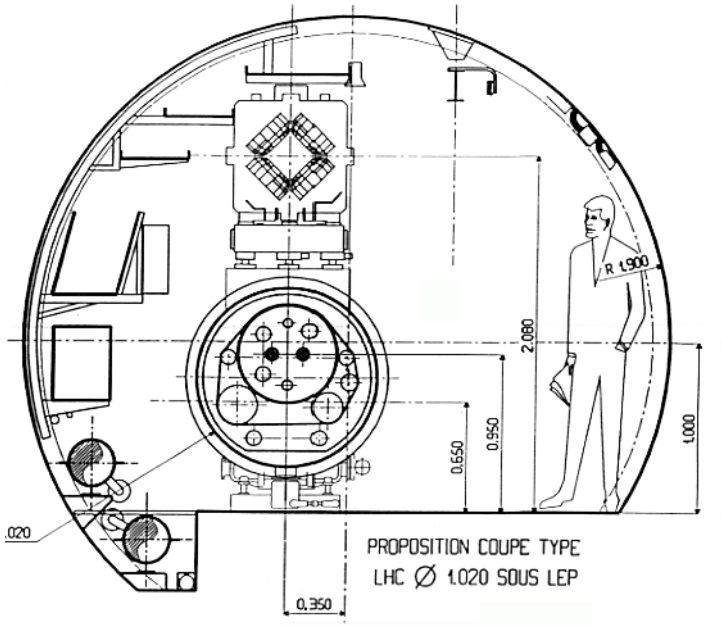
Large Hadron Collider

- Proton-proton collider
- Counter rotating beams
- 27km in circumference



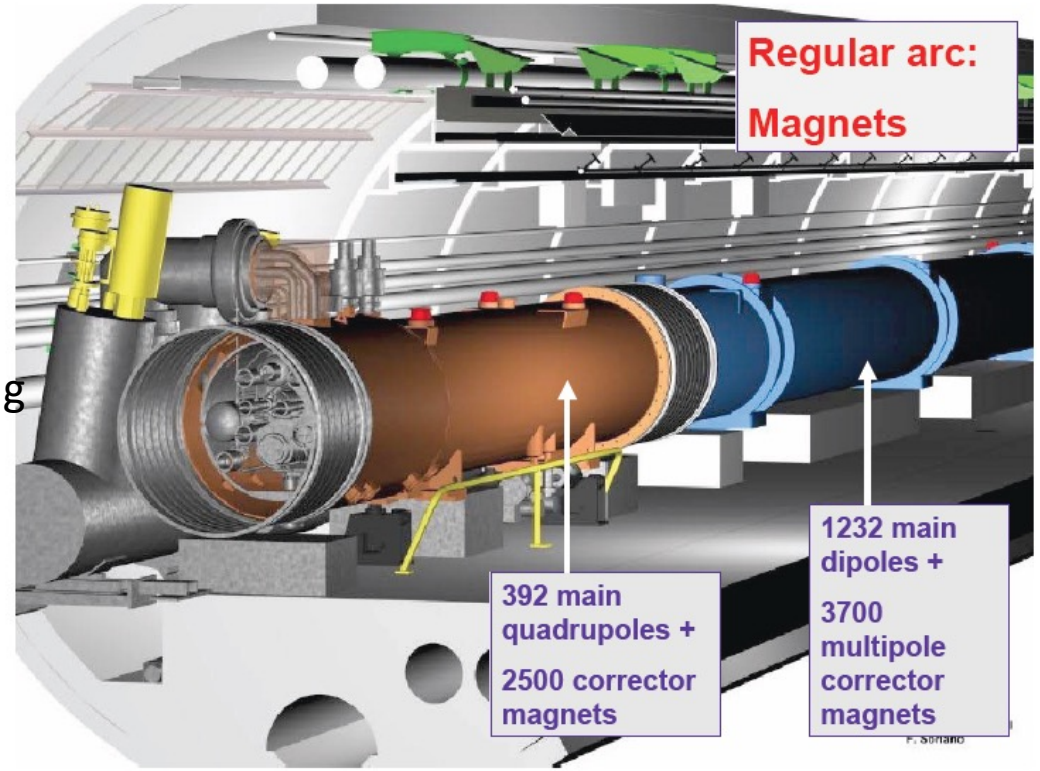
LHC Tunnel

- 27km long bored deep underground tunnel
  - Boring is more stable than cut/fill or blasted tunnels
  - 3km are actually under the Jura mountains
- Diameter 4 - 6m
- Depth 50 - 175m depending on location
- 1.4 x 10^6 m^3 ~ $(100m)^3$ soil extracted from digging



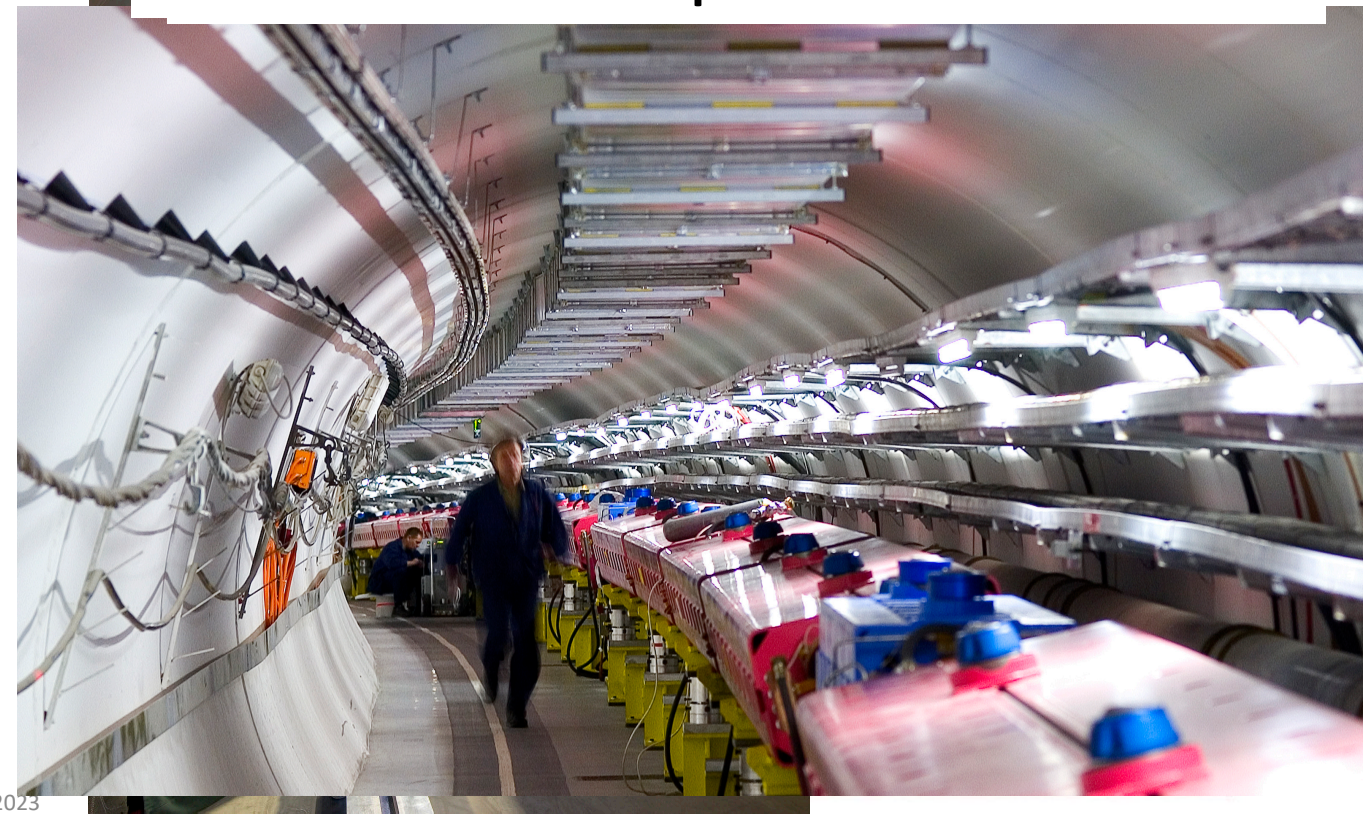
LHC Stats

- 2 counter-circulating proton beams
- Each proton has E = 7 TeV
  - $\beta = 1 - 9.0 \times 10^{-9}$
- DC beam current ~ 0.5 Amp
- 120 MW power consumption when running
- 1250 superconducting 1.9K NbTi dipole magnets around the ring
  - B ~ 8.3 T at full beam energy
- Stored beam energy ~500 MJoules



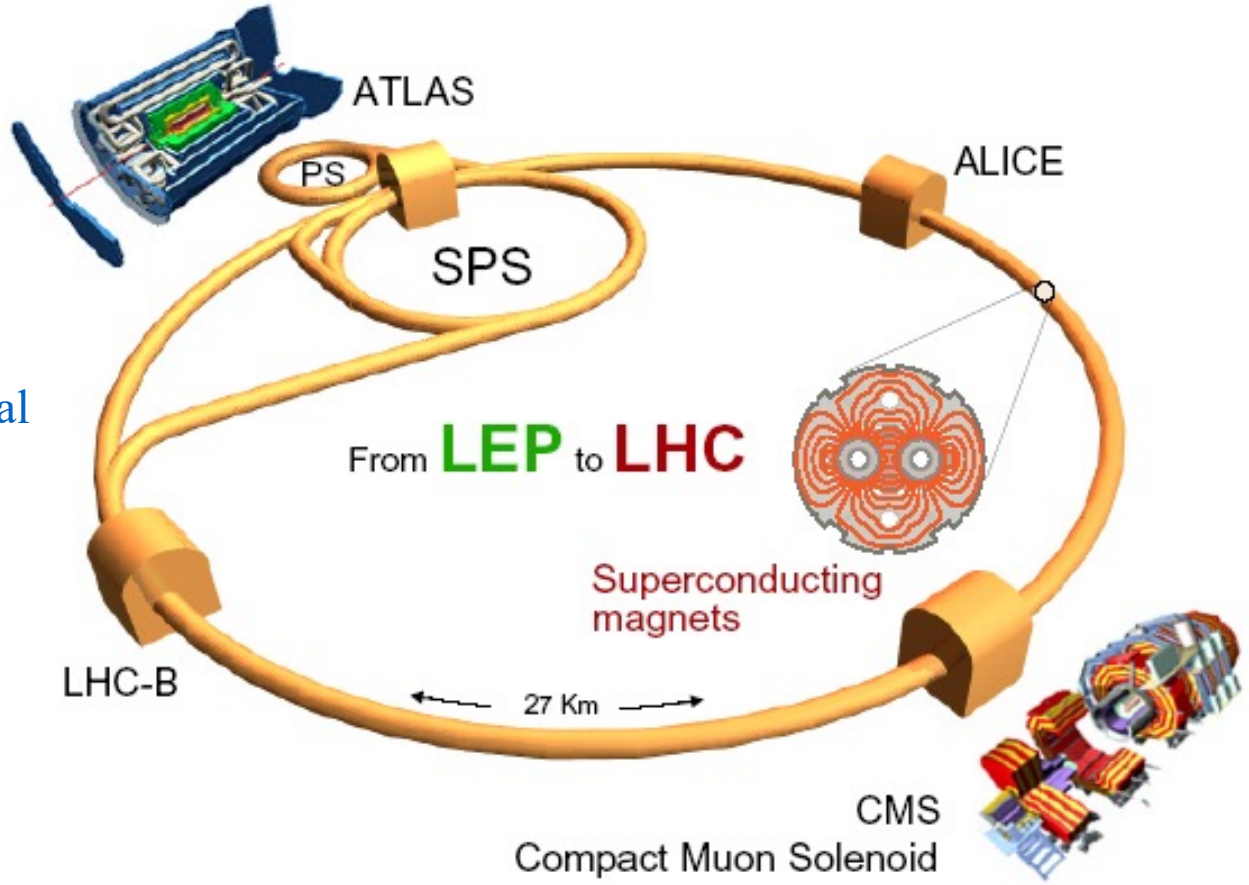
LHC infrastructure

…27 km of dipoles…whew!



Experiments

- ATLAS & CMS
  - All high p_T physics, hermetic, large, general purpose
- LHCb & Alice
  - Smaller in size and physics scope

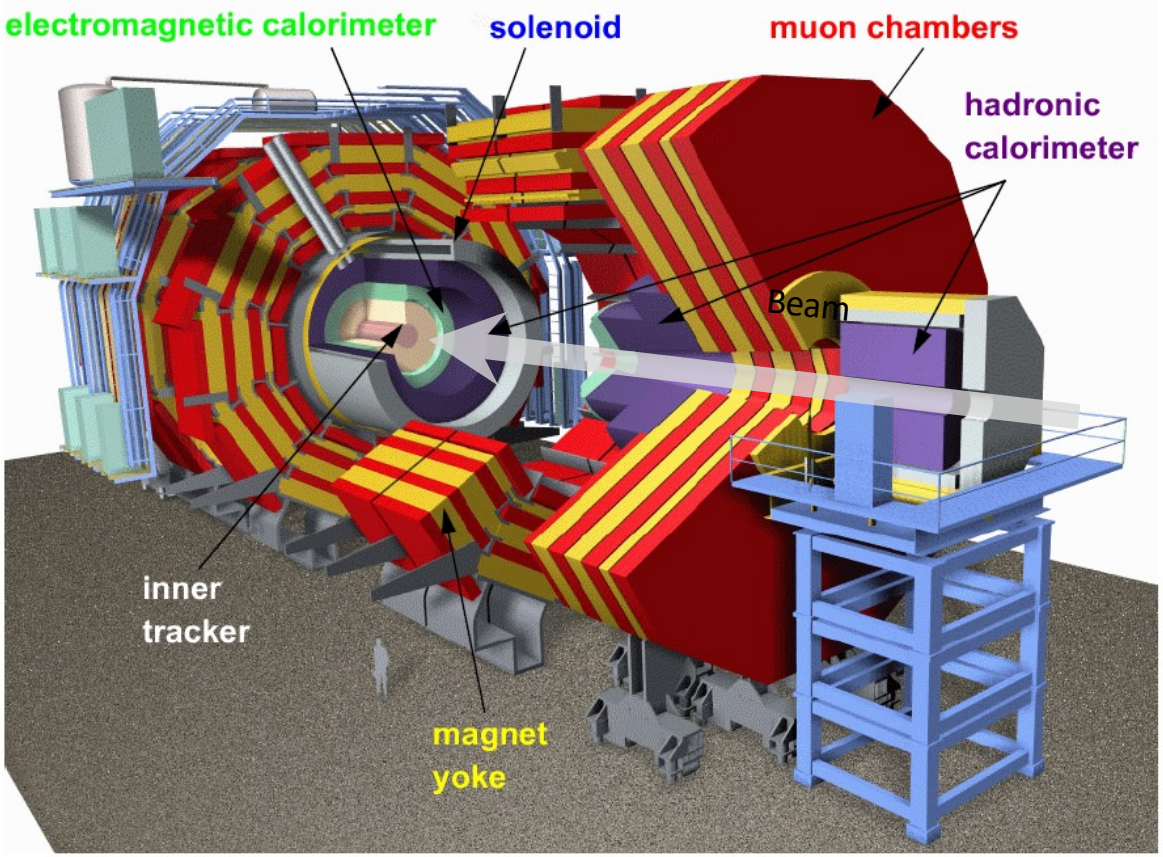


LHC Factoids

- Superconducting niobium cable
  - Combined strands would go around the equator 6.8 times.
  - Filaments of the strands together would stretch to the sun and back 5 times
  - LHC cryogenics will need
    - 40,000 leak-tight pipe junctions,
    - 12 million liters of liquid nitrogen vaporized during the initial cooldown
    - 700,000 liters of liquid helium
- Tunnel
  - 27km long circular tunnel, at excavation the two ends missed by 1 cm
- Beams
  - Proton beam energy equivalent to small car travelling at ~1900 mph

CMS: Compact Muon Solenoid

- CMS is to ~ a “can”
- Beams come in along cylindrical axis
- Collides in the center
- Resulting particles are measured by various detector components
-

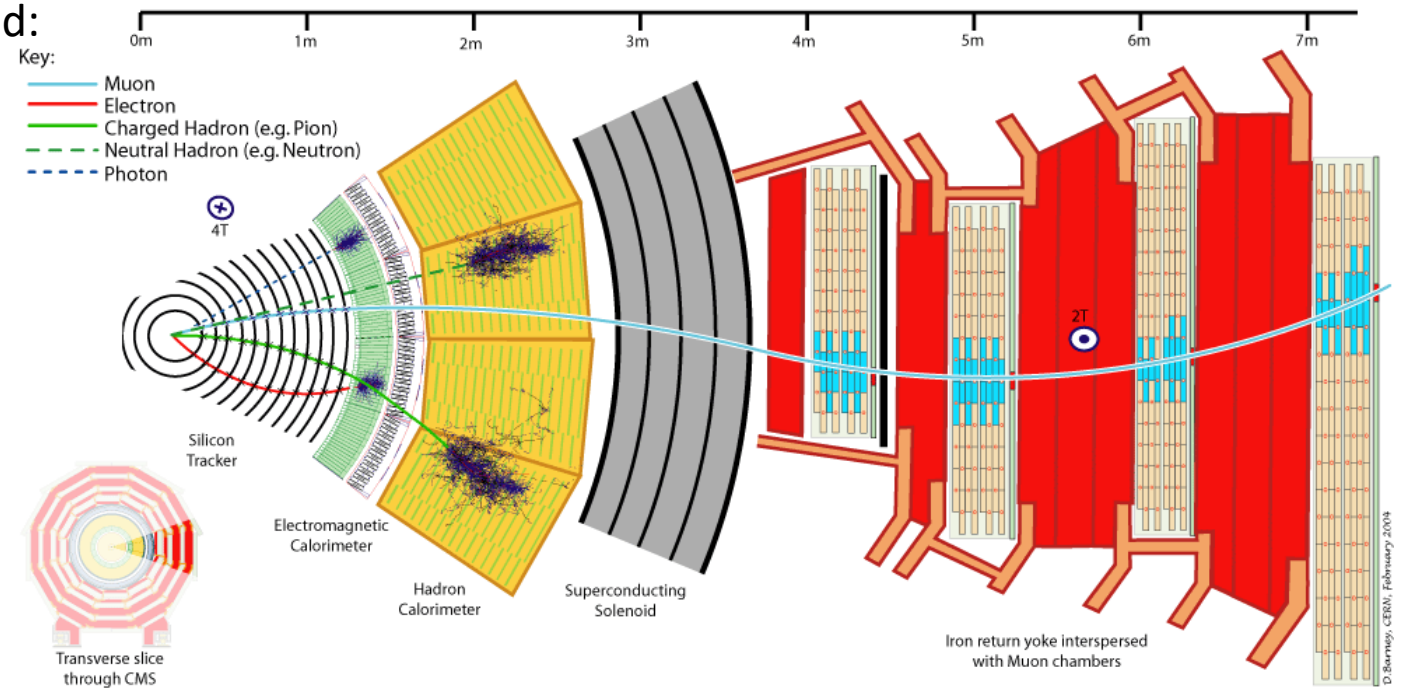


CMS Factoids

- Some of the calorimeters will see particles/cm^2 equivalent to Hiroshima bomb
- Data pours off the detector equivalent to ~100 million cell phone conversations
- Largest cold solenoid ever built
  - 4T, 20kA, 2.7GJ stored energy
  - 230 tons cold mass
  - 10k ton iron return yoke ~ Eiffel tower
  - 12.5m long x 6m diameter
  - Superconducting cable

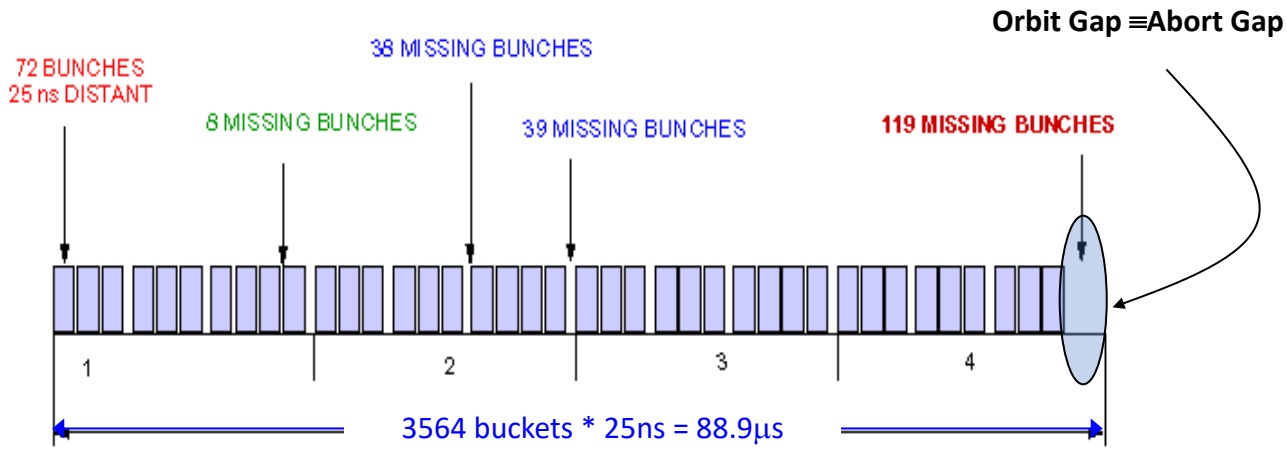
Measuring Particles

- $E^2 = m^2 + p^2$ so to know "m" you need to measure "E" and "p"
- Energy (E) is measured using calorimeters
- Momentum (p) measured in magnetic field using position detectors
- Stable particles measured:
  - Electrons
  - Photons
  - Muons
  - Protons
  - Pions
  - "other"



LHC beam structure

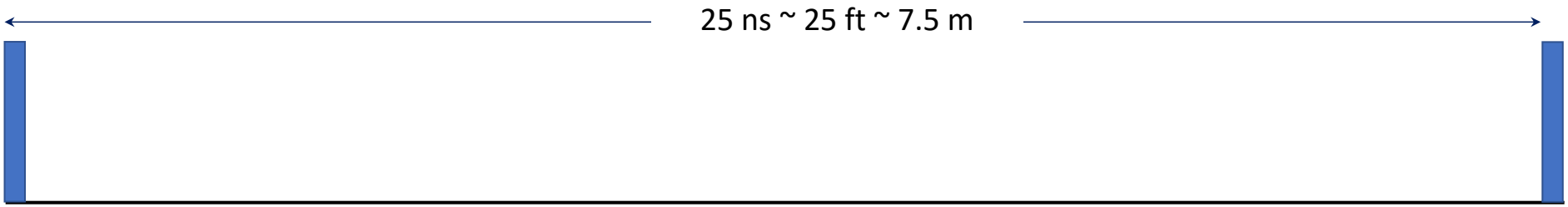
- At 7 TeV, proton takes 88.924 μs per orbit
- Orbit is divided up into 3564 “buckets”
- 88.924μs/3564 = 25ns between buckets



- Each bucket is used. Some hold protons, others are purposely left empty
  - E.g. injection, abort gap, etc

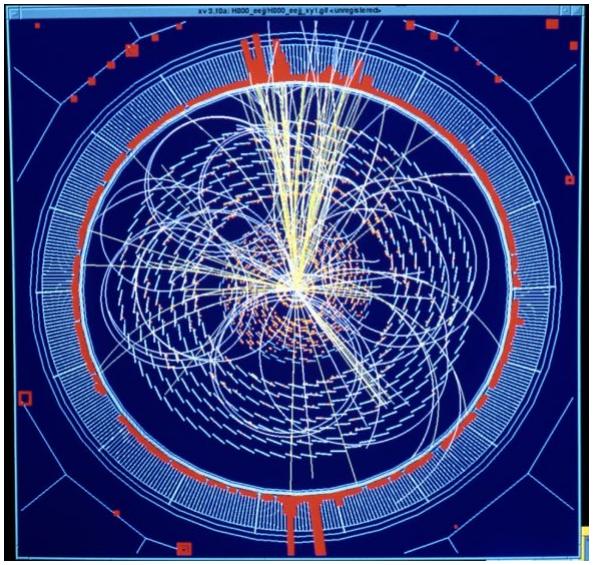
Beam collisions

- Speed of light 0.3m per ns, so distance between buckets is 7.5m
- There are 2808 buckets with protons, called “bunches”
- Each bunch has ~10^{11} protons and is around 0.1m = 10cm long
  - 0.1/7.5= 1.3% of the distance between bunches
- Counter rotating beams are made to collide at center of each experiment, every 25ns= 40MHz collision rate

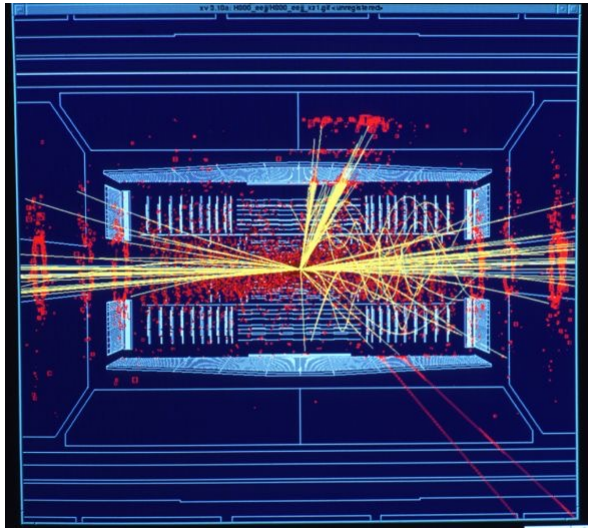


“Typical” Event – lots of data, complex reconstruction!

Looking down the beam direction…



Looking sideways



Data generated per collision

- Each proton-proton collision will provide ~ 1Mbyte data
- Each bunch-bunch interaction will produce on average ~50 proton-proton collisions
- Total data rate:

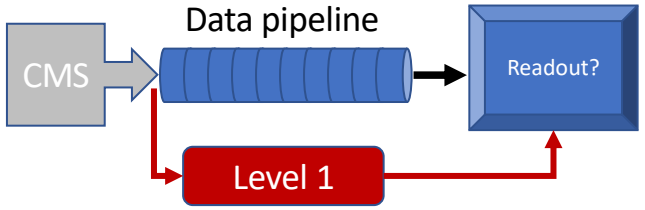
$$R = \frac{40M\ bunch\ collisions}{sec} \times \frac{50\ pp\ collisions}{bunch\ collision} \times \frac{1\ Mbyte}{pp\ collision} = 2000\ TB/sec$$

- If you read this out at 10 TB/sec then it would take 200 sec per event to readout
- While you are reading out, the detector has to be "dead" so as not to overwrite data buffers
- In 200 sec, you would miss 200sec/25ns ~ 8 x 10^9 collisions, efficiency of 1.25x10^{-8}%
- Of the events read, only ~1 per million or less are "interesting"
  - So most of the events you read out will be useless

“Trigger”

- No country or collection of countries will spend $10B to build an accelerator if you can’t read out more than 1 in a few billion events, and most of those events are useless.
- Need a new way to think about readout. This is what the TRIGGER does.

Trigger “pipeline” basic idea



- Digitize data every 25ns crossing and store in memory “pipeline”
- Memory is finite so will fill up fast
  - Let’s say we have 128 memory addresses per data word
  - Memory will be full after 128 x 25ns = 3.2 μs
- While memory is filling up…
  - Send subset of the data to fast digital “Level 1” processors to see if anything is interesting
- Processor has only 3.2 μs to communicate to pipeline whether to save the data before it will get overwritten
- If Level 1 says yes, then data is read out
- CMS might have 10^6 channels of electronics with maybe 2 bytes each. If each has 128 addresses that’s 128 x 2 x 10^6 = 256 Mbytes trigger pipeline
- Want to design a pipeline processor that can process an event in less time it takes to over write all the memory
- This will depend heavily on what you think you will want to calculate in the processor: Monte Carlo!

Trigger readout time

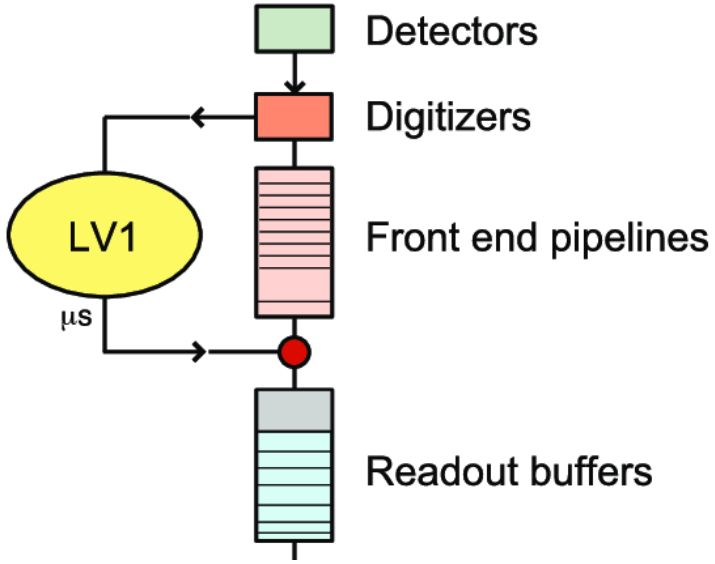
- An event deemed worthy of keeping has then passed the Level 1 trigger.
- Next, read all data from the event and send to next level. This takes time T_{read} and during this readout time, the detector pipeline will be "dead"
- Next, estimate using Monte Carlo calculations the rate at for seeing "interesting" events
  - Top quarks, Multi jets, Higgs bosons, etc.
- Let's say that we estimate the rate (events/sec) of anything interesting to be R_{keep}
  - Then time between interesting events will be $T_{keep} = 1/R_{keep}$
- We want $T_{keep} > T_{read}$ so that we only miss interesting events while reading out if there's a fluctuation in arrival of such events:

$$T_{keep} > T_{read} \text{ or } T_{read} < {}^{1}/_{R_{keep}}$$

- This tells us how fast we have to design the readout to be in order to be sure that we won't miss most of the interesting events

Level 1 Trigger: Pipeline processor

- Level 1 “accept” causes data to be read out from from front-end pipelines
- This data is read into “readout buffers” for further processing

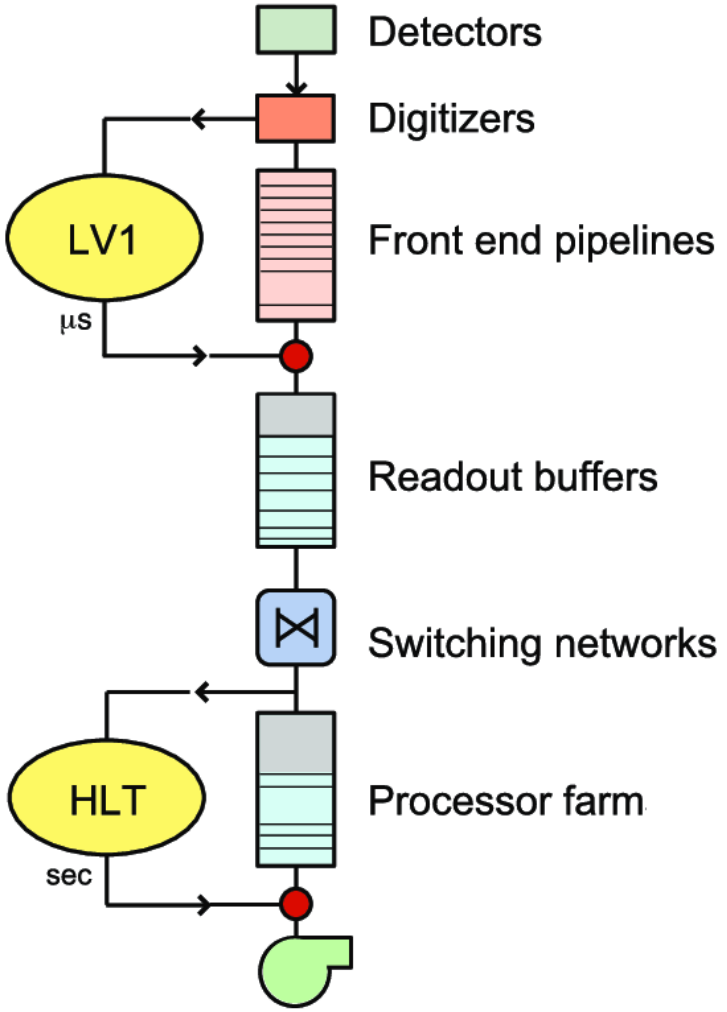


Level 2 Trigger

- Level 1 is a conservative guess at what might be interesting using a subset of the data
- Resolution of Level 1 determination is not too precise!
  - There could be many events that turn out to actually not be that interesting
- Writing uninteresting events to permanent storage is expensive and complex
- So we have a 2nd level trigger that consists of massively parallel computers looking at each event in detail

Level 2 – aka High Level Trigger (HLT)

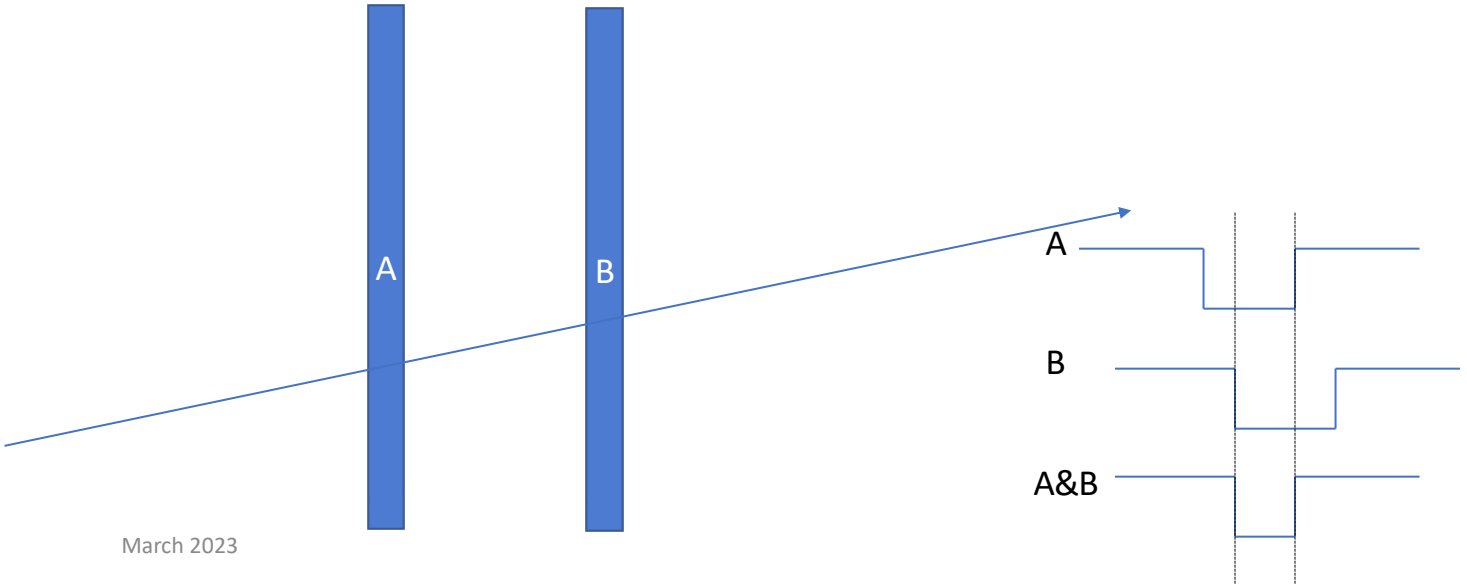
- Data sent from readout buffers through network to processor farm
- CPUs in farm have full access to all data and all reconstruction codes



How do we implement Level 1, Pipelines, Readout, …?

Using logic for science experiments

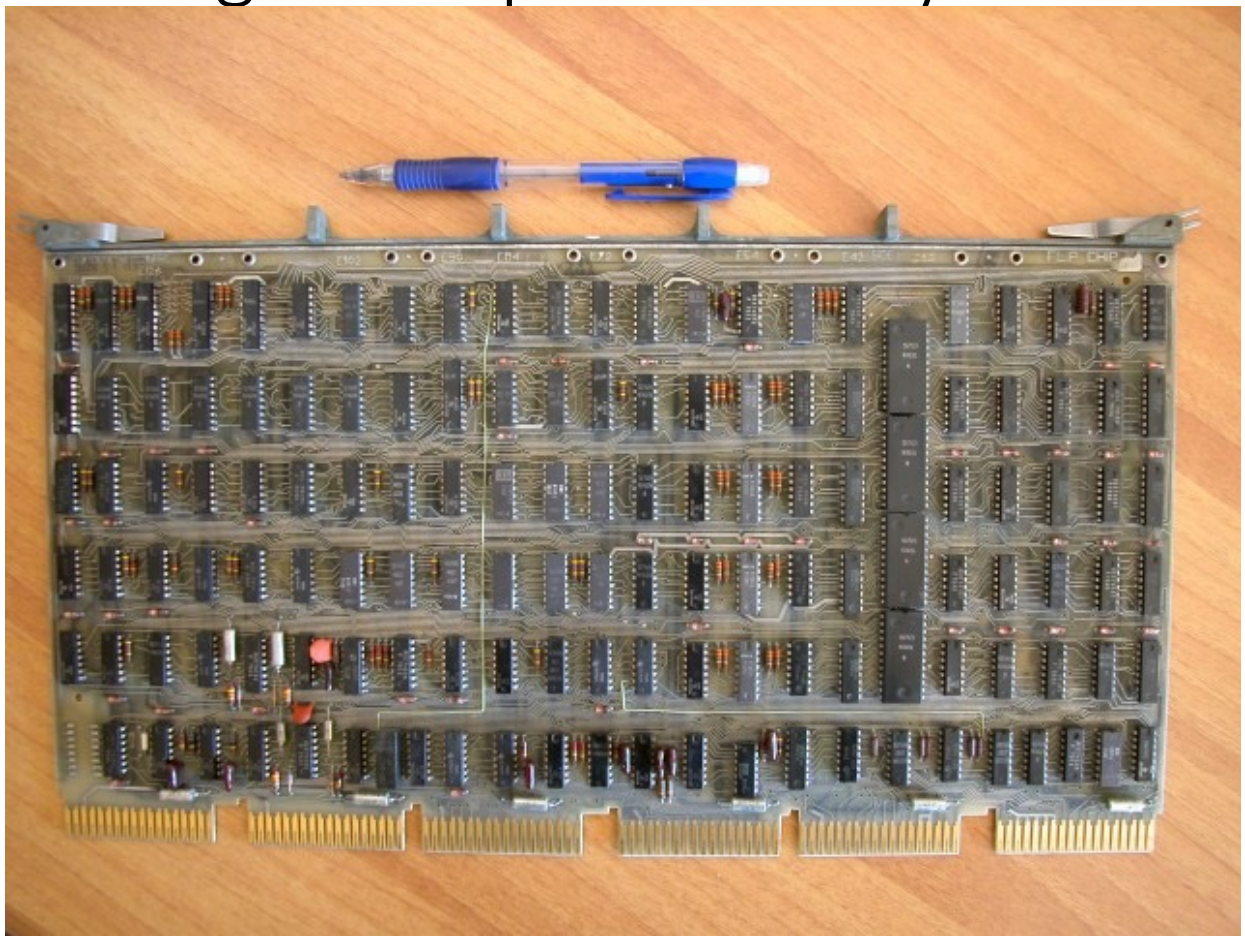
- Quite common to need logic to constrain experimental conditions.
- e.g 2 scintillator planes….each gives a pulse when a muon traverses
- Want to read out the data only when the 2 signals coincides. Need to implement logic:
  - Trigger = A ”AND” B



Translation to large particle physics experiments

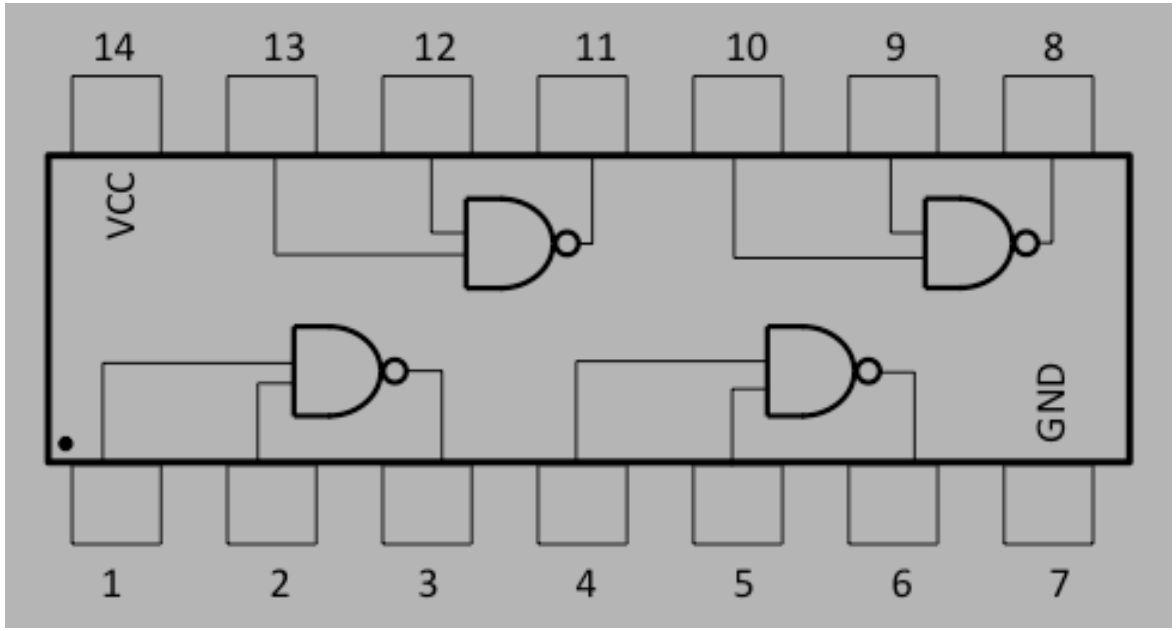
- How to know if event was “interesting” enough for Level 1 to keep?
  - For example: add all data in sections of the calorimeter.
    - If sum of the data is > some threshold, good chance there was something interesting
- Level 1 pipeline processing: can we use CPUs?
  - Millions of channels, lots of sums, hard to get all of that data into 1 CPU and CPU would only have 3.2μs to do its thing
- Alternative: custom logic that does the same thing as a CPU, but without overhead instructions, and with a flexible “topology”
  - We implement this with what is called “programmable logic”

This could get complicated very fast!



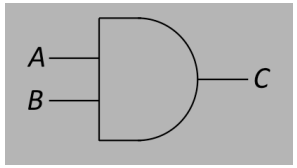
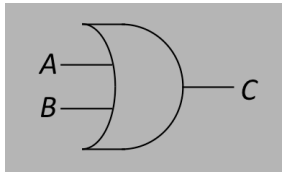
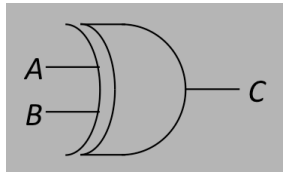
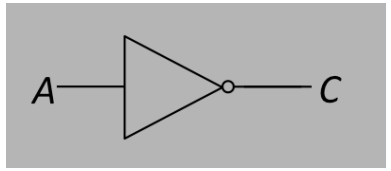
Logic chips

- Each logic chip consists of lots of AND, OR, etc gates



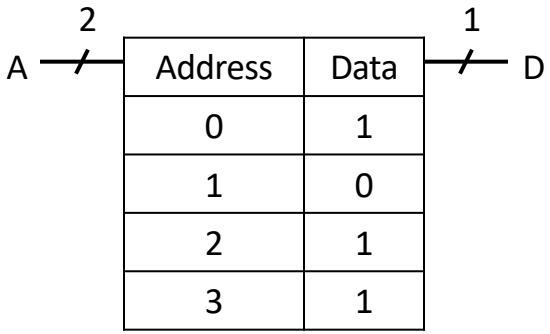
Logic Gates

- Implementation:

	AND	OR	XOR	NOT
Gate				
Truth table	AB C 00 0 01 0 10 0 11 1	AB C 00 0 01 1 10 1 11 1	AB C 00 0 01 1 10 1 11 0	A C 0 1 1 0

Look-up Table (LUT) memory

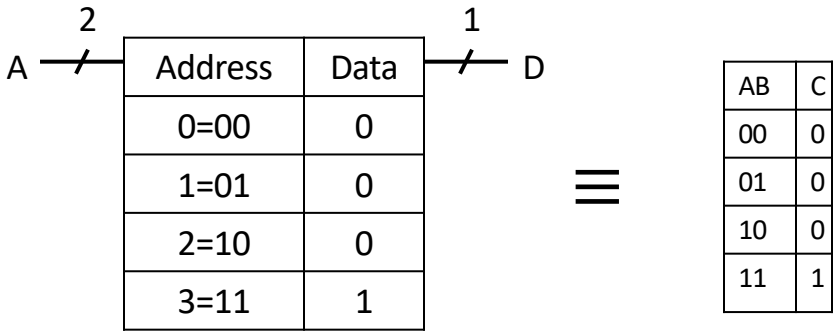
- Primitive memory, 2-bit address (A), 1-bit data value (D)



Address	Data
0	1
1	0
2	1
3	1

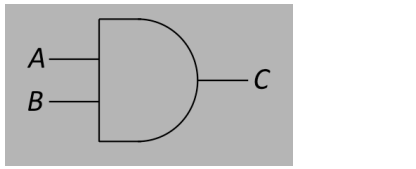
Necessity is the mother of invention

- 2-bit addressable LUT w/1 bit data looks like it will implement the truth table for an AND gate



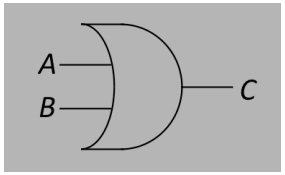
Programmable logic

- Implement logic gates using LUTs



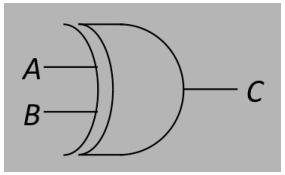
2 A — 1 D

Address	Data
0=00	0
1=01	0
2=10	0
3=11	1



2 A — 1 D

Address	Data
0=00	0
1=01	1
2=10	1
3=11	1

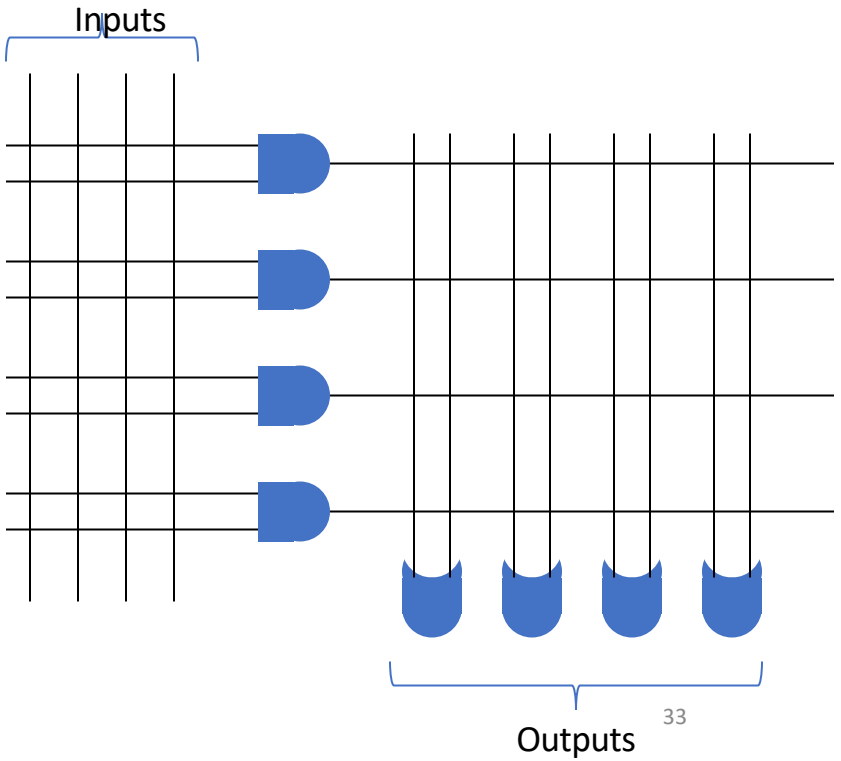


2 A — 1 D

Address	Data
0=00	0
1=01	1
2=10	1
3=11	0

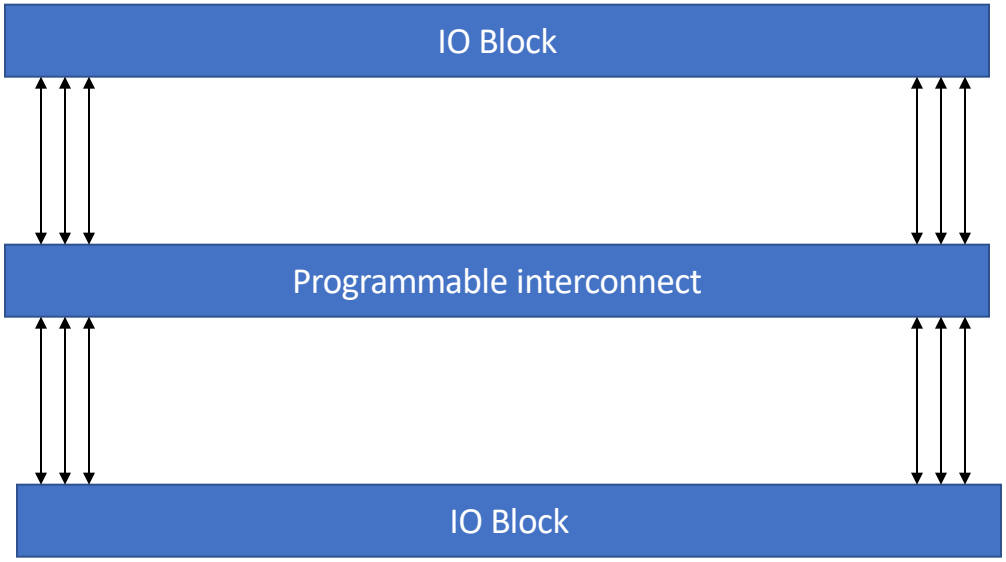
Programmable logic array

- Any input can be connected to any AND gate by connecting the vertical input line to the horizontal gate input lines with FETs
  - Programmable by turning FETs on or off
- Any AND output can be connected to any OR input also with FETs
- This is a “logic cell”
- Build up array of such cells on a chip
- This allows “functional logic”. All you need to do is decide which FETs to turn on to implement the logic
- Can implement each AND and OR gate with a LUT
  - Program the LUT to turn it into an AND or and OR ad hoc



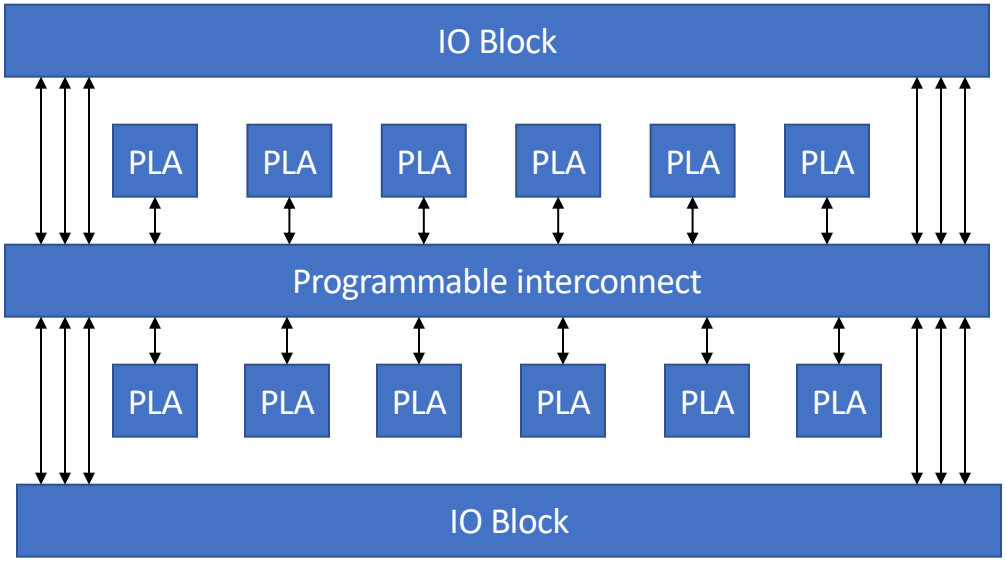
PLA on a chip: The “Gate Array”

- Start with
  - IO Block: signals into and out of the chip, connect to:
  - Programmable interconnect: traces with FETs to connect things together
    - Probably some kind of “mesh” network



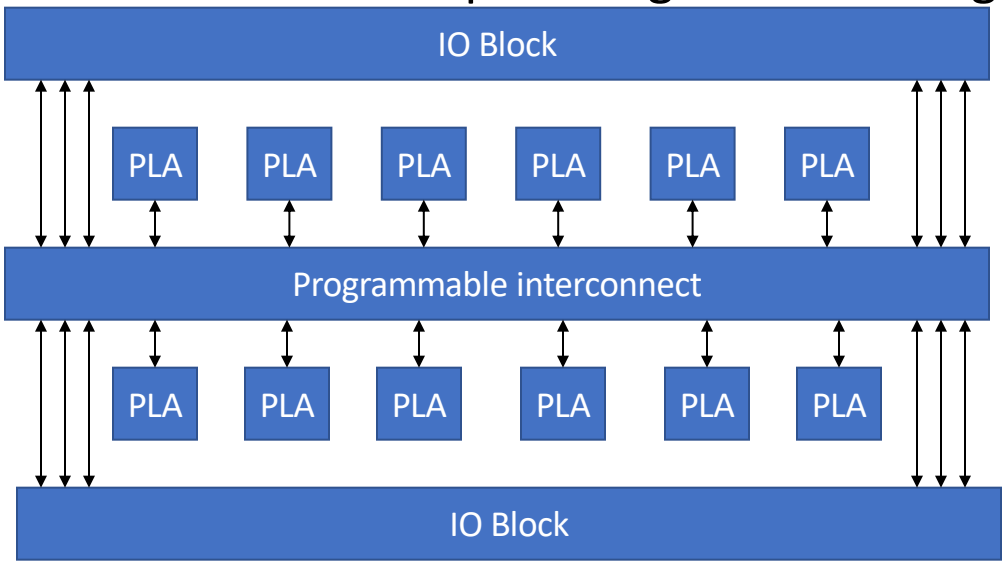
PLA on a chip

- Add as many PLA's as you can and hook up to the interconnect to be able to send signals into the PLAs



PLA on a chip

- This is a bit of an oversimplification, there's a lot more circuitry to deal with tristate, signal polarity control, etc. But this is the basic topology
- Also, PLAs can be PROM (1-time) or EEPROM (programmable)
- These are sometimes called Complex Programmable Logic Devices (CPLD)



Field Programmable Gate Arrays (FPGA)

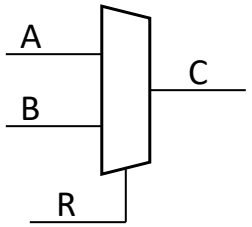
- The “Field” means that you can program it “on the fly”
  - But no one really does that. It’s an old term that is out of date
- The architecture of FPGAs is much more complex than a CPLD and much more powerful
- Instead of a PLA, FPGAs use what are called “Logic Cells” or “Logic Blocks”
- Two biggest manufacturers:
  - Xilinx
    - Emphasis was on maximizing number of logic gates
  - Altera (bought by Intel)
    - Emphasis was on maximizing IO
  - Both are very powerful and functional. Almost doesn’t make a difference anymore which one you choose

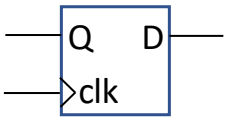
Xilinx FPGA Logic Block

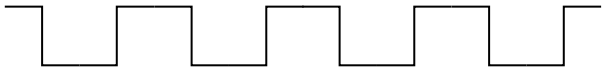
- Configurable Logic Blocks (CLB)
- Provides both logic and storage functionalities
- Can contain:
  - LUTs, multiplexors, flip-flops, hardware floating point, memory, even processors!

Xilinx Logic Block

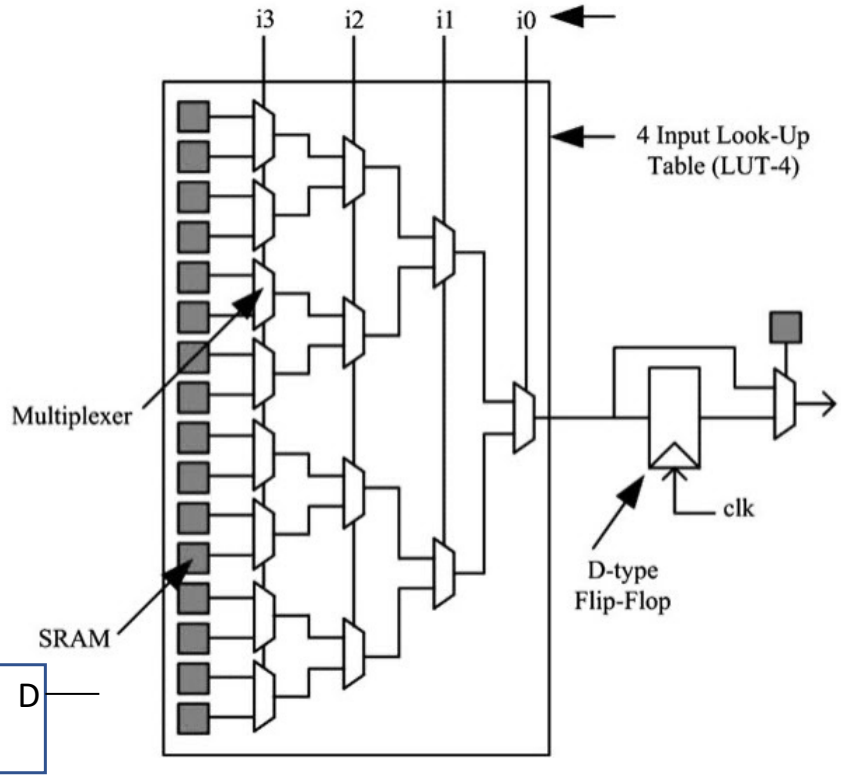
- 4-bit LUTs (instead of 2). Gives flexibility
- SRAM is configured to implement any logic
- Mux's allow flexibility
  - C = R? A : B conditional
- Clock input
  - For synchronous logic
  - Usually on a special line to allow it to be fast
- D-flip-flop
  - At rising edge of the clock, Q is copied to D





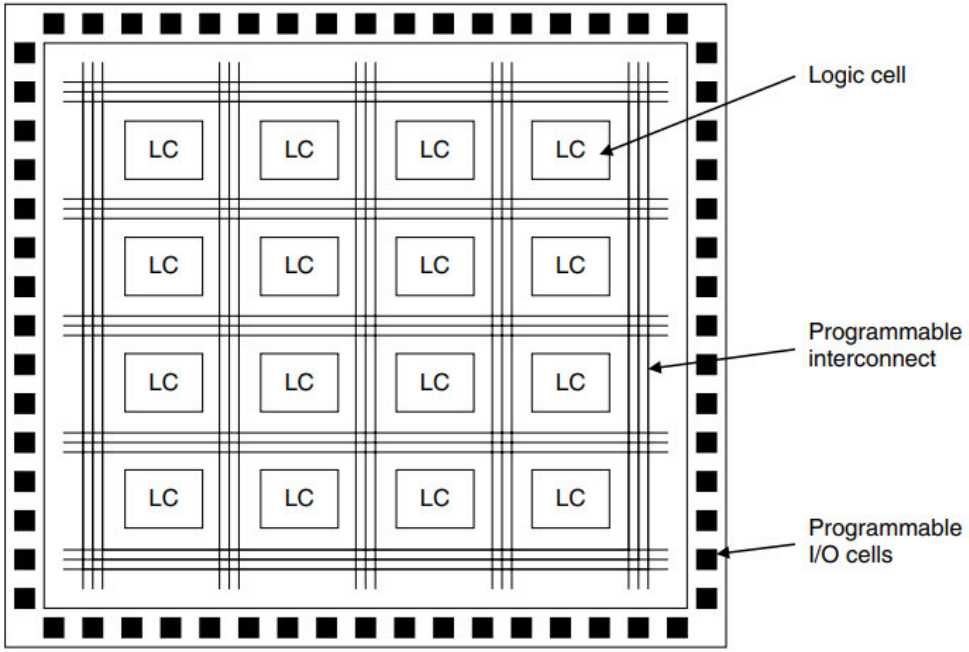


"clock"



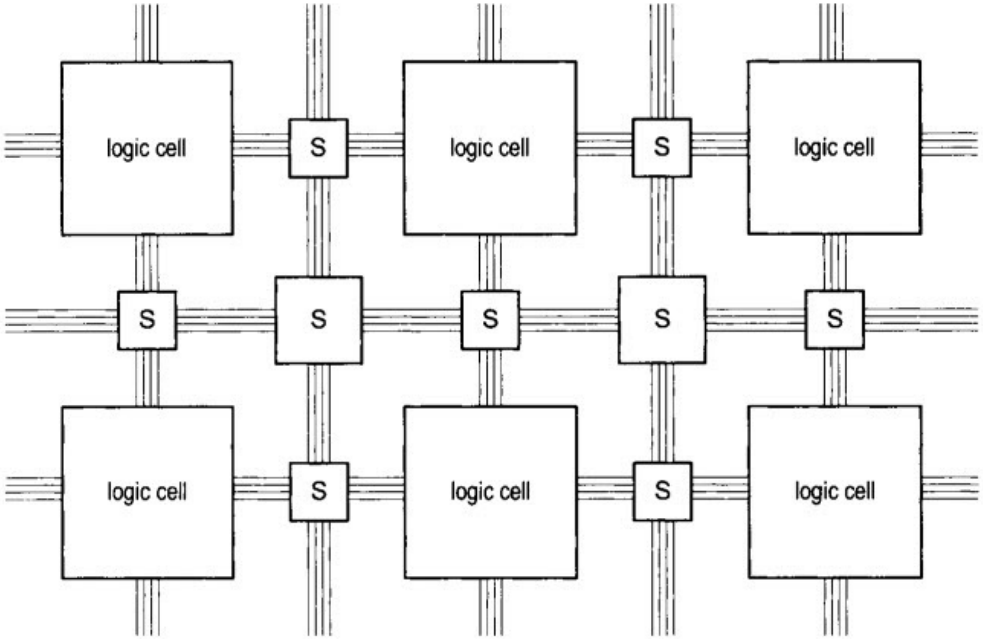
FPGA structure

- LC is “Logic Cell”



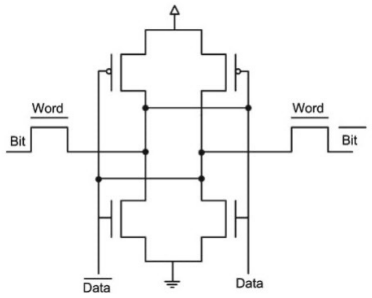
FPGA structure

- Zoom in….
- ”S” are interconnects, usually in a mesh
- At this point, we just use it and don’t worry too much about how it works!

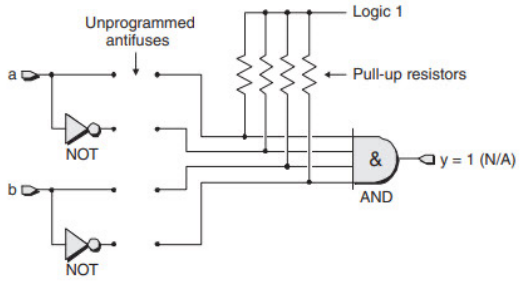


Types of FPGA

- SRAM
  - Static RAM. Value loaded will remain until changed, or power is removed
  - All CMOS transistor (no capacitor as in DRAM)



- EEPROM/Flash
  - Non-volatile
  - But has a finite number of times a bit can be changed, small compared to SRAM
  - Can be used in high rad environments, although still subject to single event upsets (SEU == bit flipping when a particle goes though and ionizes the wrong place on the transistors!)
- Anti-Fuse
  - Physical memory. Very useful in high radiation environments. Not subject to SEU
  - But slower. And once it's programmed, you cannot reprogram it

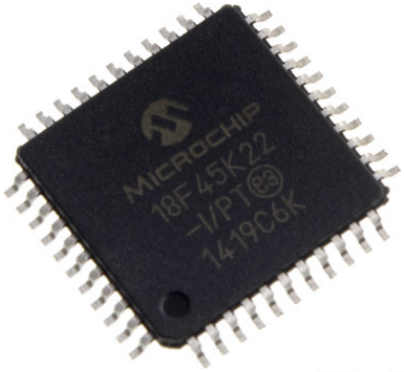


Cost and capability?

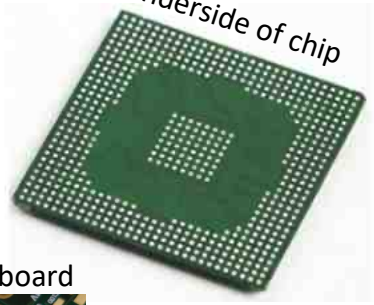
- These things are cheap, and powerful
  - e.g. Xilinx XC3S200AN-4FTG256C
    - SRAM
    - 4032 logic blocks
    - 256 pins, 195 are IO pins
    - Core voltage ~1.1V
    - Clock frequency 250 MHz, with:
      - digital clock management to be able to control clock phase
      - to be able to do clock multiplication and divide, to get any clock rate
    - RAM bits: 288k
  - $25.06 per 1, cheaper for higher quantities
  - This chip is state of the art 10-20 years ago

FPGA packaging

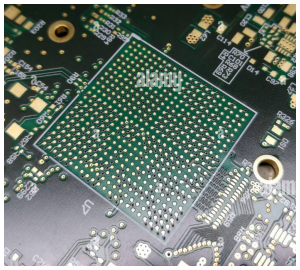
- Surface mount chips
  - Brings pins out to the edge, solder onto the board
  - Hard to do if you need ~200 or more IO pins!

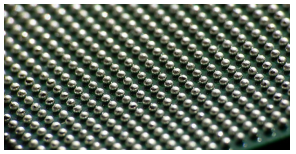


- Ball Grid Array (BGA)
  - Bring IO pins to the bottom side of the chip
  - Mount solder balls
    - Pitch is 20/inch (1.27mm), sometimes finer
  - Pads on the board that match
  - BGA is placed carefully so the balls and pads line up
  - Increase temp, balls melt, voila – soldered!



What's on the board





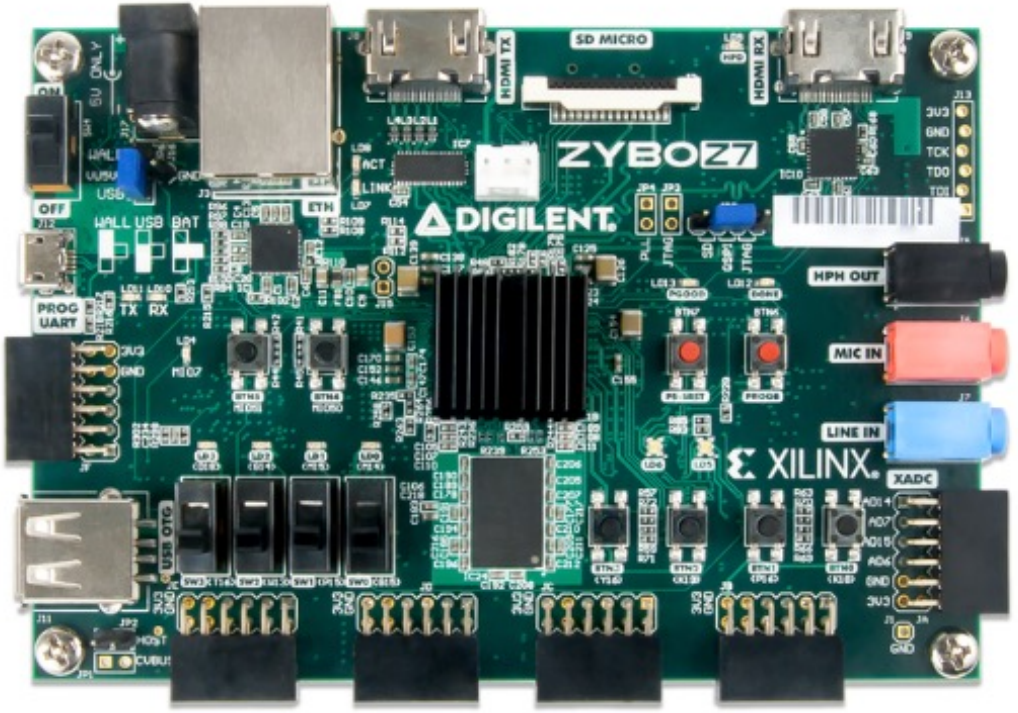
closeup

“Development kits”

- Xilinx and Altera (or contractors) mount chips on development boards so that you can ”try it out”
- Often these boards have huge functionality. And are subsidized
  - This is part of their business strategy
- Inexpensive “dev boards” ~ $100
  - Has USB, IO pins, maybe an ADC, LEDs, etc
- Expensive “dev boards” ~$10,000
  - Huge IO, O(1000), fast ADCs (12 bit GHz), fancy clock management (can use GHz clocks), processors…

Example: Digilent, Inc. Zybo Z7

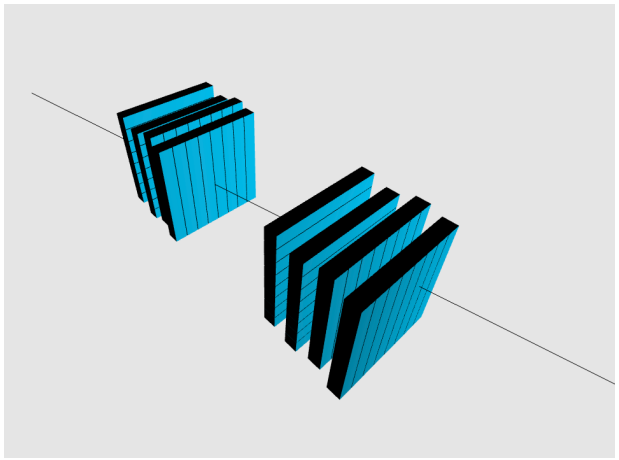
- Cost ~$200
- Xilinx Zynq 7000 chip
  - Dual-core 667MHz ARM processor
  - Xilinx 7-series FPGA
- 1G ethernet, USB 2.0, SDIO
- SPI, UART, I2C (slow serial IO)
- HDMI input and HDMI output
- 1G DDR3L external memory
- microSD slot
- 6 Connectors, 12 pins each for
- Switches, push button, LEDs

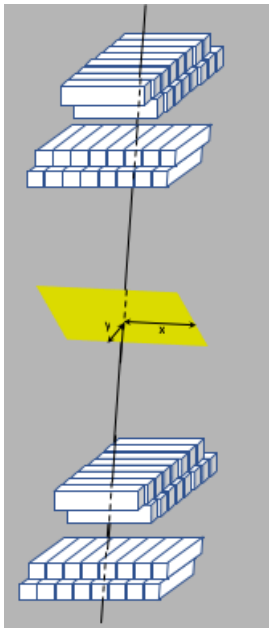


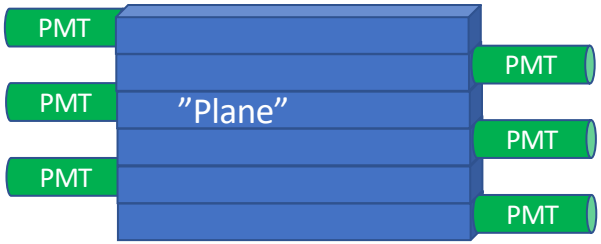
BUT you have to learn to program it!!!!

Muon telescope

- Very simple telescope to measure 3D muon track from cosmics
  - 8 planes, 6 tubes per plane (can have 8 if I get more PMTs)
  - 2 halves consisting of 4 planes per half
  - Each half:
    - 2 planes along x, 2 along y
    - This gives a 3d space-point for the μ
  - 48 total PMTs





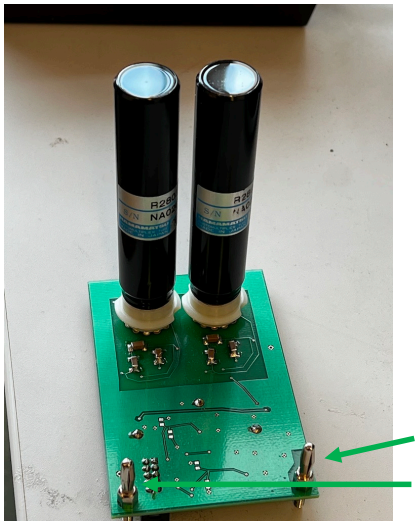


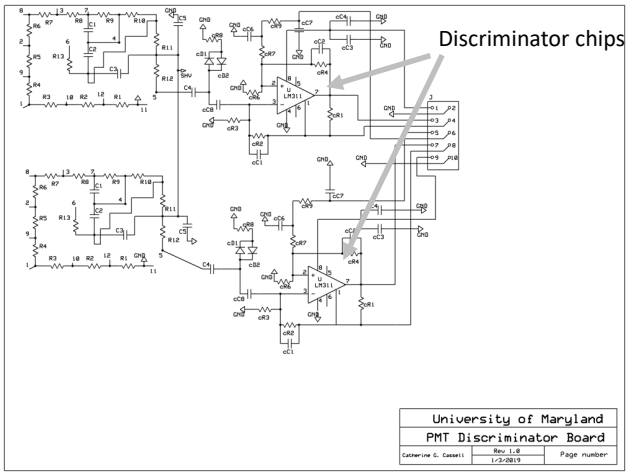
PMT

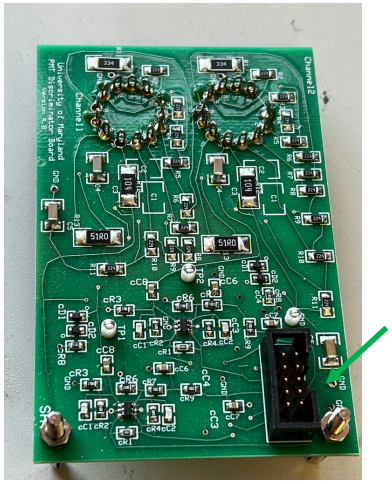
- Hamamatsu R2801
- 10 stage photomultiplier
- Bialkali photocathode
- Peak response is at around 375nm
- Needs ~1500 volts

PMT voltage and readout

- PMT board custom made by the undergrads (PC Express)
- Each board services 2 PMTs
- Input: HV for PMT and 5V for discriminator and output driver
- Output:
  - Each tube output goes into discriminator
  - Digital (LVDS, differential) signal sent to FPGA to say if there's a hit
- Connectors:
  - HV to banana plug
  - 1 LV in and 2 digital out via 10-pin connector
- 48 tubes x 3 pins/tube = 148 wires!
- Probably cost a few $100 total

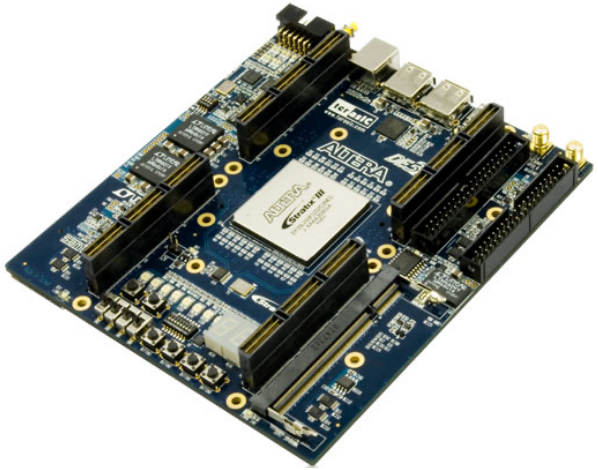


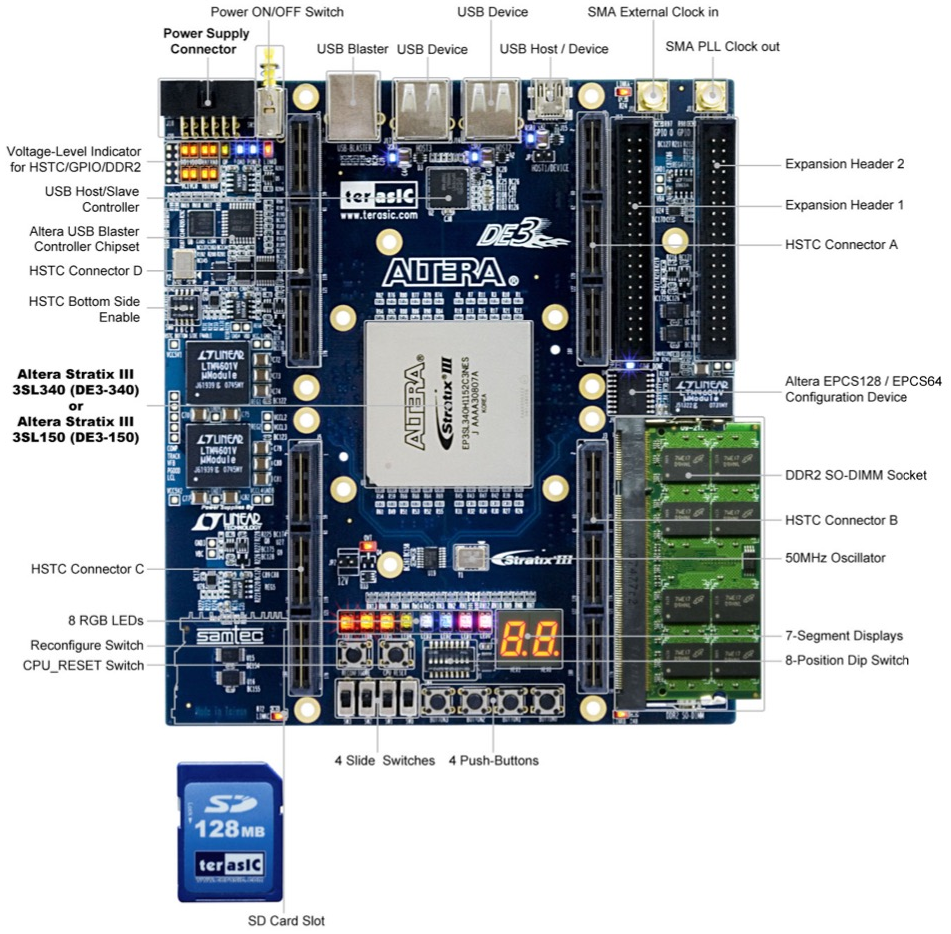




FPGA development

- Altera DE3
  - Stratix 3 chip
  - 4 HSTC connectors
    - High Speed Terasic Connector
    - Each HSTC has 121 signal pins
  - USB, push button, LEDs

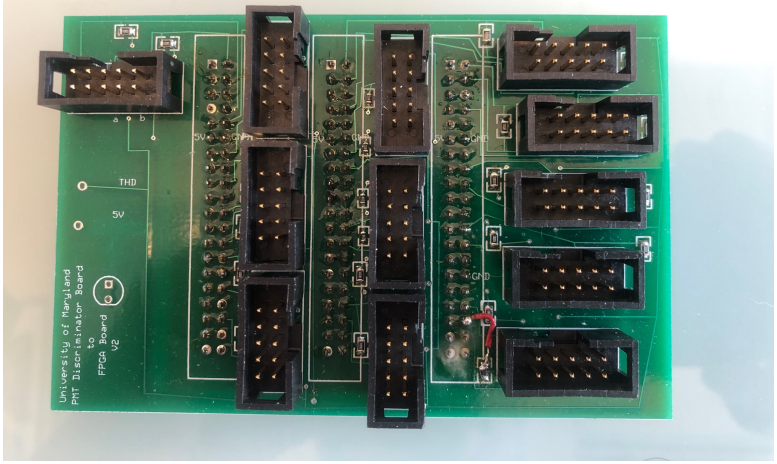




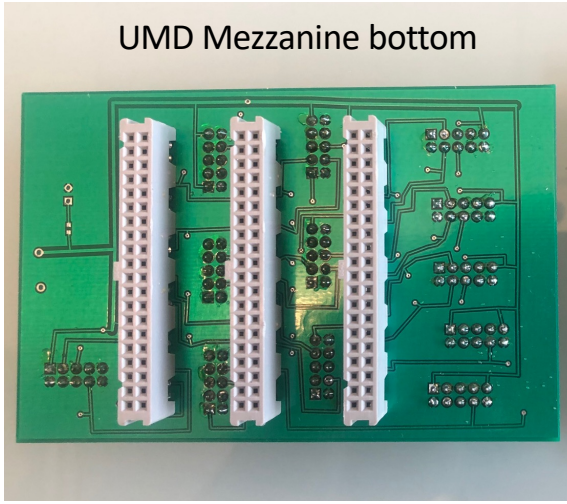
Mezzanine card

- Built to interface between cables from PMT boards and Dev Board's HSTC connectors

UMD Mezzanine top



UMD Mezzanine bottom

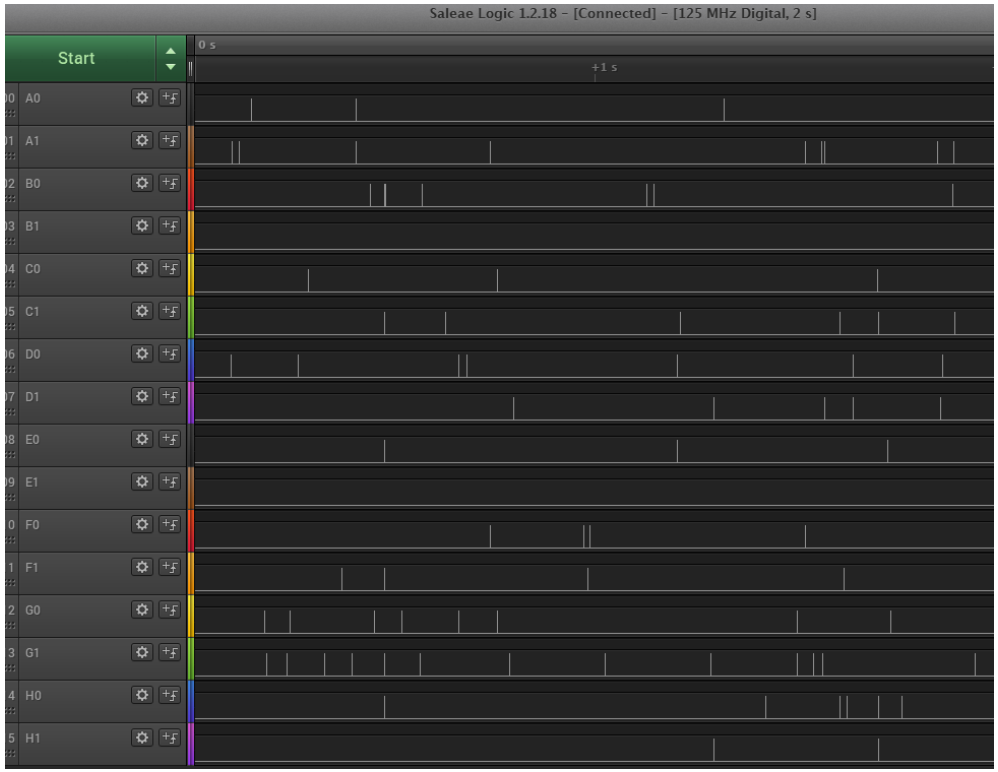


HSTC adapter board

What does the FPGA do?

- Converts differential signals to single ended to run around inside FPGA chip
- FPGA job:
  - Buffers (stores) all PMT signals
    - PMT signal is probably 20ns wide. FPGA has to be able to respond that fast
  - Reads in all
  - Require at least 1 tube in each plane for a plane hit
  - Require at least 1 of 2 planes per “x” and 1 of 2 per “y” direction on the top
  - Require a signal in TOP and a signal in BOT
  - Set a bit in a status register (FPGA memory) if the above is satisfied
  - Connect FPGA to RaspberryPI processor over serial interface (1Mbps)
  - Write Python code in RaspberryPI to read status register. If there’s a hit, then read out the data (1 bit per tube)
  - Reconstruct the muon track and display on the computer terminal

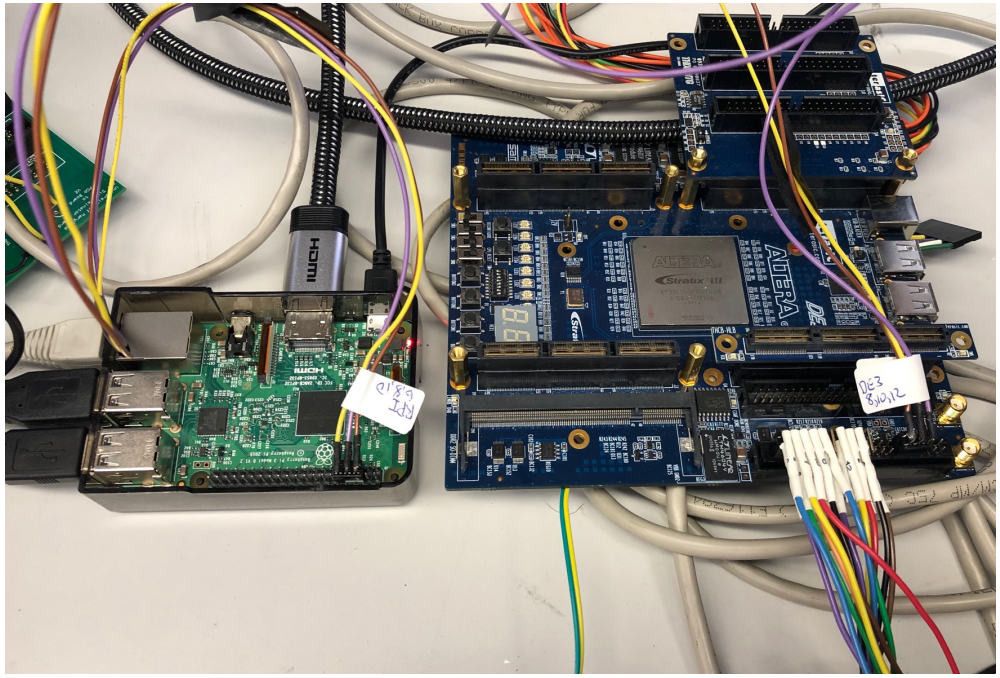
Data looks like this on a logic analyzer



Time

FPGA and Raspberry PI

- Serial data over a few wires to let the RPI talk to the FPGA
- RPI has dedicated “UART” pins and Python libraries. Makes it very easy!



Underconstruction!