CMC Configure Your Research Platform

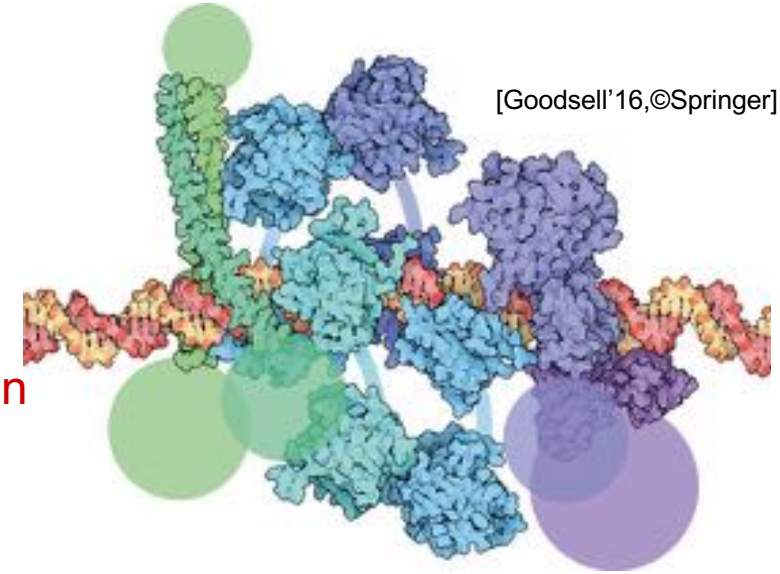
Accelerating Front-End Bioinformatics

Sebastian Magierowski

Electrical Engineering & Computer Science (EECS)
York University
Toronto, Canada

"Front-End" Bioinformatics

- **Bioinformatics**
  - computers + biological data (NIH)
- more narrowly…
  - analysis of biomolecules
    - their make-up, structure, and function
    - proteins, DNA, RNA, etc.


[Goodsell'16,©Springer]

- **"Front-End"**
  - computations done close to the raw sample measurements
    - often with real-time preference

"Accelerating"

- Making bioinformatics find solutions faster
  - of course
- With specialized computing hardware
  - our goal is to build platforms

[CC BY-SA 3.0]

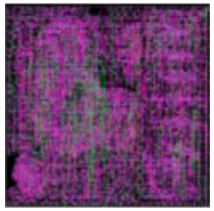
  - Edico Genome DRAGEN bioinformatics processor
    - on Amazon EC2 F1 (Xilinx VU9P Ultrascale+)

Our Acceleration Ambitions

- Interested in custom ASICs
  - following a top-down route to get there
- GPUs
  - application-level

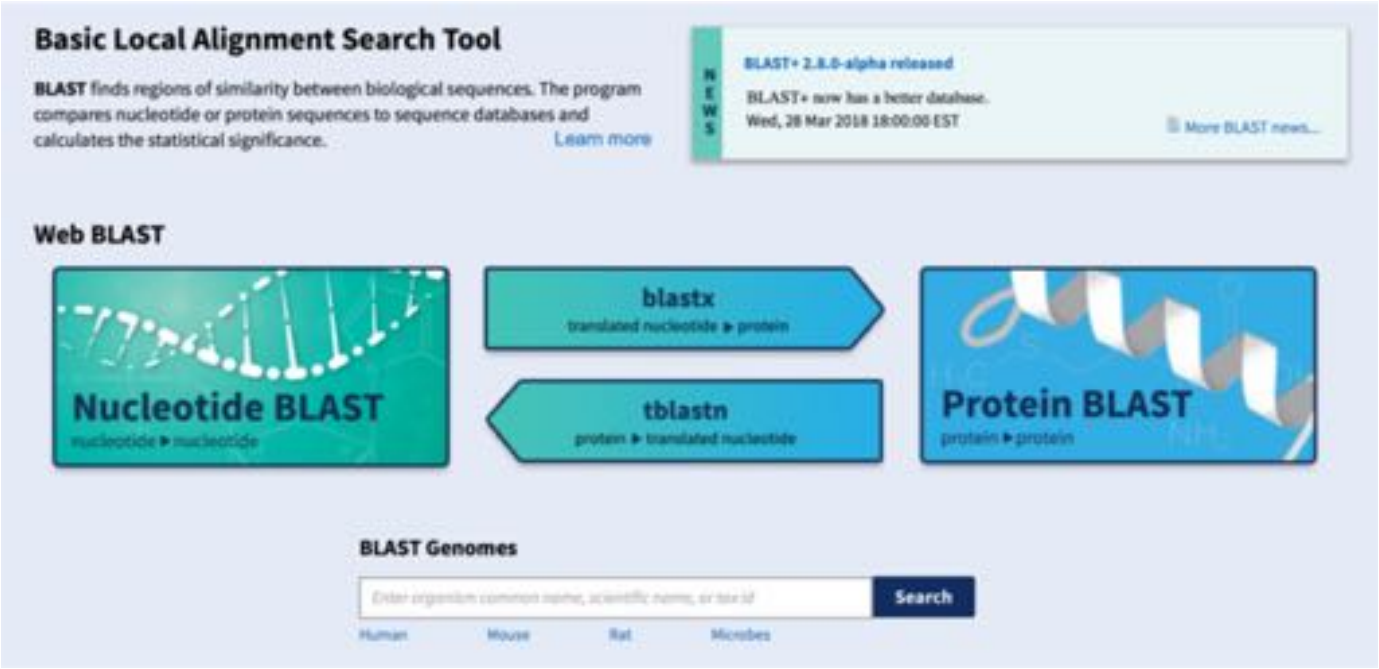
- FPGAs
  - kernel-level
  - RIFFA+PCIe

- SoCs
  - kernel/ISA-level
  - RISC-V+RoCC



Bioinformatics "Solutions"

- Finding similarities between databases
  - sequence database homology searching



For Example...Query a Protein

- *query*: ~100 character sequence (from alphabet of 20)
  - *target*: >25M recorded sequences
    - > 8G characters (amino acids)
- "Solutions" come back in seconds–minutes
  - list of sequences adhering to some matching criteria

PREDICTED: probable insulin-like peptide 3 [Drosophila suzukii]

Sequence ID: XP_016934436.1 Length: 130 Number of Matches: 1

Range 1: 33 to 124 GenPept Graphics ▼ Next Match ▲ Previous Match

Score	Expect	Method	Identities	Positives	Gaps
40.0 bits(92)	0.001	Compositional matrix adjust.	28/95(29%)	44/95(46%)	14/95(14%)

```
Query  26   VNQHLCGSHLVEALYLVCGERGFFYTPKTRREAEDLQVGQVELG----GGPGAGSLQPLA  81
             +  LCGS L EAL  +C    + +   T+R  + +    +E G    G      L+ L
Sbjct  33   ASMKLCGSKLPEALSRLCV---YGFNAMTKRTLDPMNFNLIEAGSLDLGFDDRSLLERLF  89

Query  82   LEGSLQ-------KRGIVEQCCTSICSLYQLENYC  109
            L+GS Q       + G+ ++CC   CS+ +L  YC
Sbjct  90   LDGSAQMLKTRRLREGVFDECCLKSCSMDELLRYC  124
```

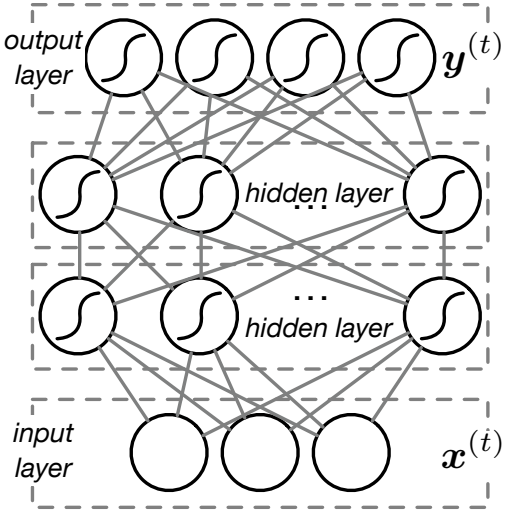
Bioinformatics "Answers"

- Then apply biological criteria to develop insight
  - computational biology
- Examples…
  - What other proteins are closely related?
  - What genes are responsible for the protein?
  - What proteins exhibit distant relations?
  - What protein domains are shared?

- These insights may be used to arrive at scientific/clinical insight
  - Evolutionary history
  - Identify disease
  - Design drugs

Common Algorithmic Patterns

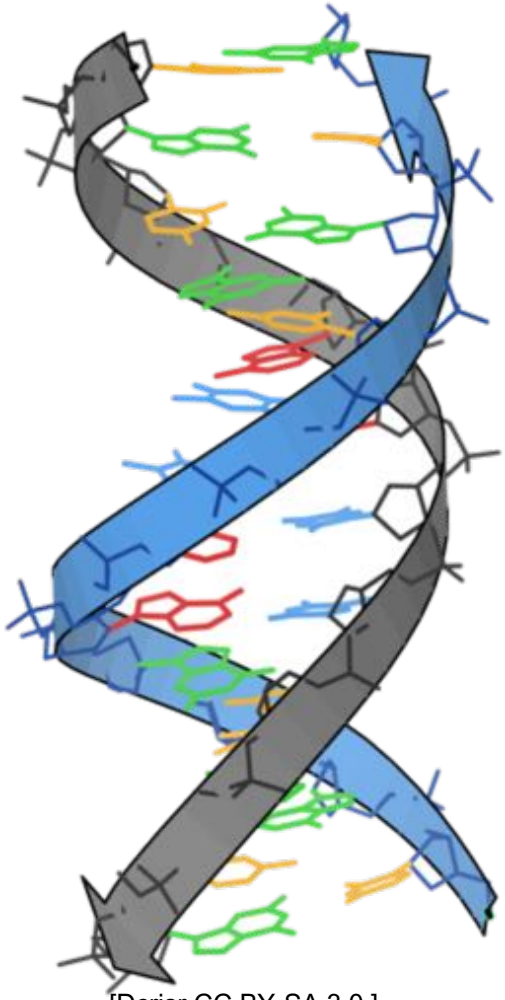
- dynamic programming
  - sequence comparison
- search
  - look for sequence patterns
- sort
  - transform one string to another
- combinatorics
  - find sub-string combinations that match other strings
- graph algorithms
  - sequence assembly
- clustering
  - molecular evolution
- classification and inference
  - Bayesian networks
  - neural networks



Front-End Sequencing

- Focus on DNA sequencing where…
  - …measurement has gotten **very fast**
  - …hardware has gotten **very small**

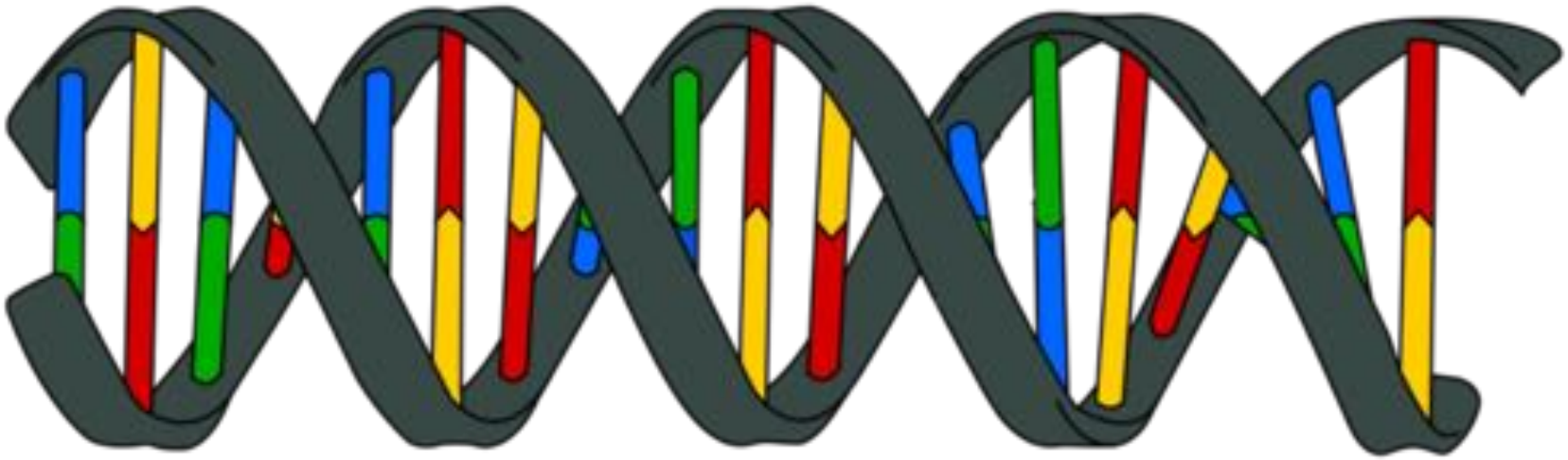
- Benefits from high-speed embedded computing
  - at least in part



[Dcrjsr CC BY-SA 3.0]

DNA Sequencing...a quick reminder

- Sequencing
  - Take given DNA sample…



  - …and figure out its particular base sequence

G T G T G A T C C A T G C A T G G A

DNA Sequencing Pipeline

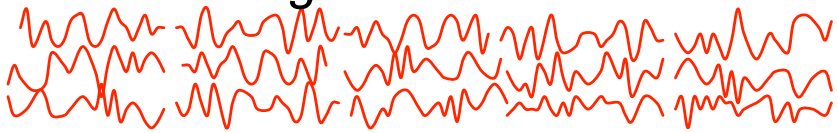
- This translation is just one step of a process

1. DNA isolation

2. Fragmentation

3. DNA amplification

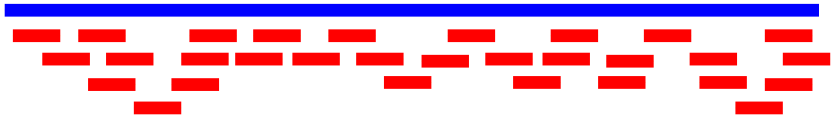
4. DNA-to-signal transduction



5. Basecalling

```
ACCTGTCGT GCAAAAATC TCAAAACGG CAAATGCGC ACGGACGGT
CGCACATAA AGTGCAACC CCAATTTAC CTAGATTAC CCTTGAGAA
TCTAGTCTA GCCTAATGC TCTCCCGAG CTGTGTCAT TTTCCGCAC
```

6. Alignment



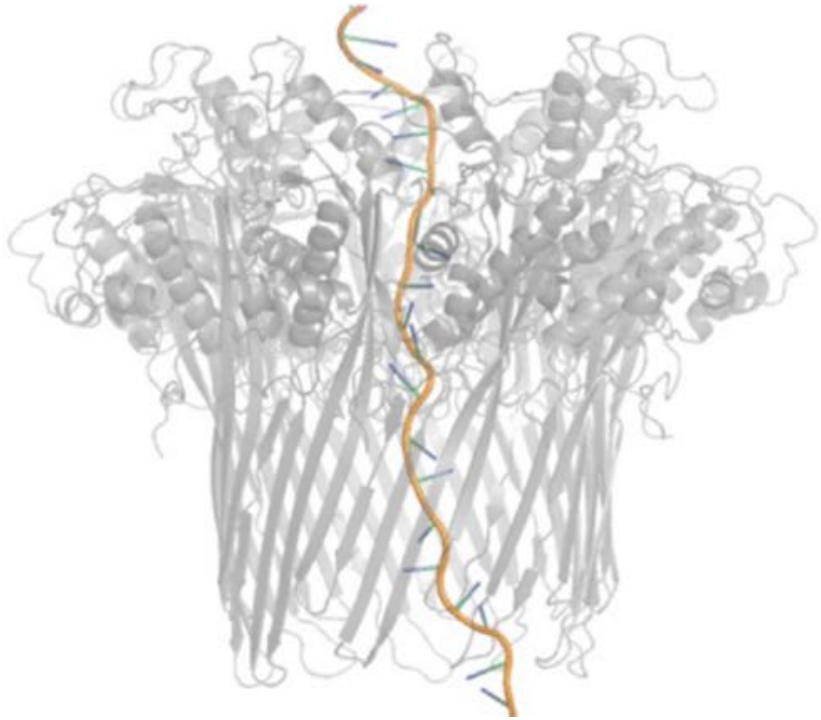
7. Sorting, 8. De-duplication
9. Local align, 10. Quality score

11. Variant calling

Sequencing Trend

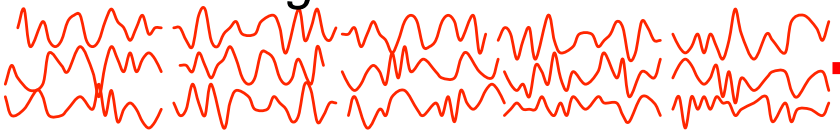
- Sequencer miniaturization
  - 100.0 kg
  - 000.1 kg
  - ~1000X smaller
    - ~10-20X slower
    - ~10% less accurate

Nanopore Sequencing



- Nanopore sequencing
  - Small hole (nanopore)
  - DNA passes through nanopore
  - Generates small current
  - Convert signal to text

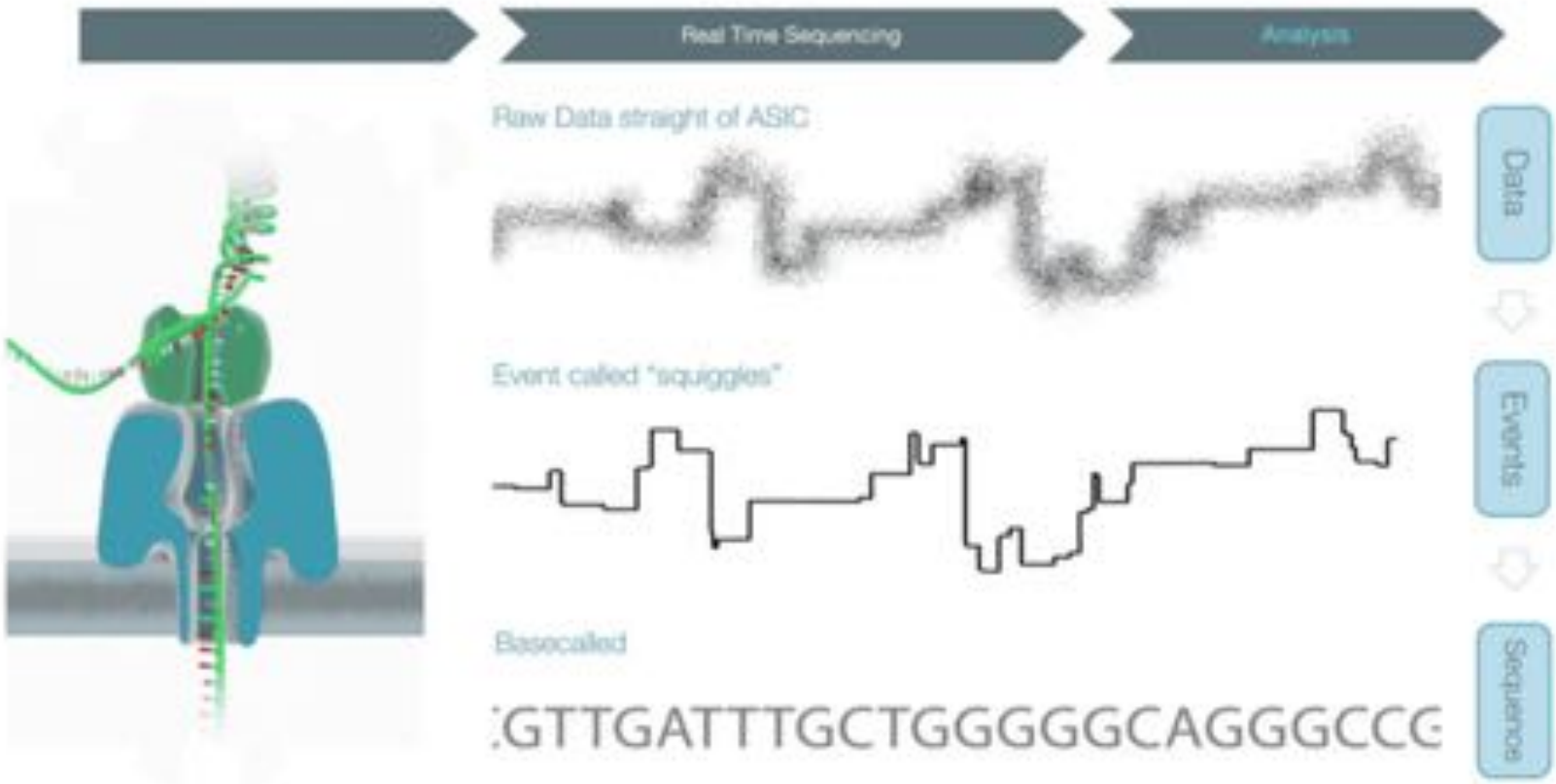
4. DNA-to-signal transduction



5. Basecalling

```
ACCTGTCGT GCAAAAATC TCAAAACGG CAAATGCGC ACGGACGGT
CGCACATAA AGTGCAACC CCAATTTAC CTAGATTAC CCTTGAGAA
TCTAGTCTA GCCTAATGC TCTCCCGAG CTGTGTCAT TTTCCGCAC
```

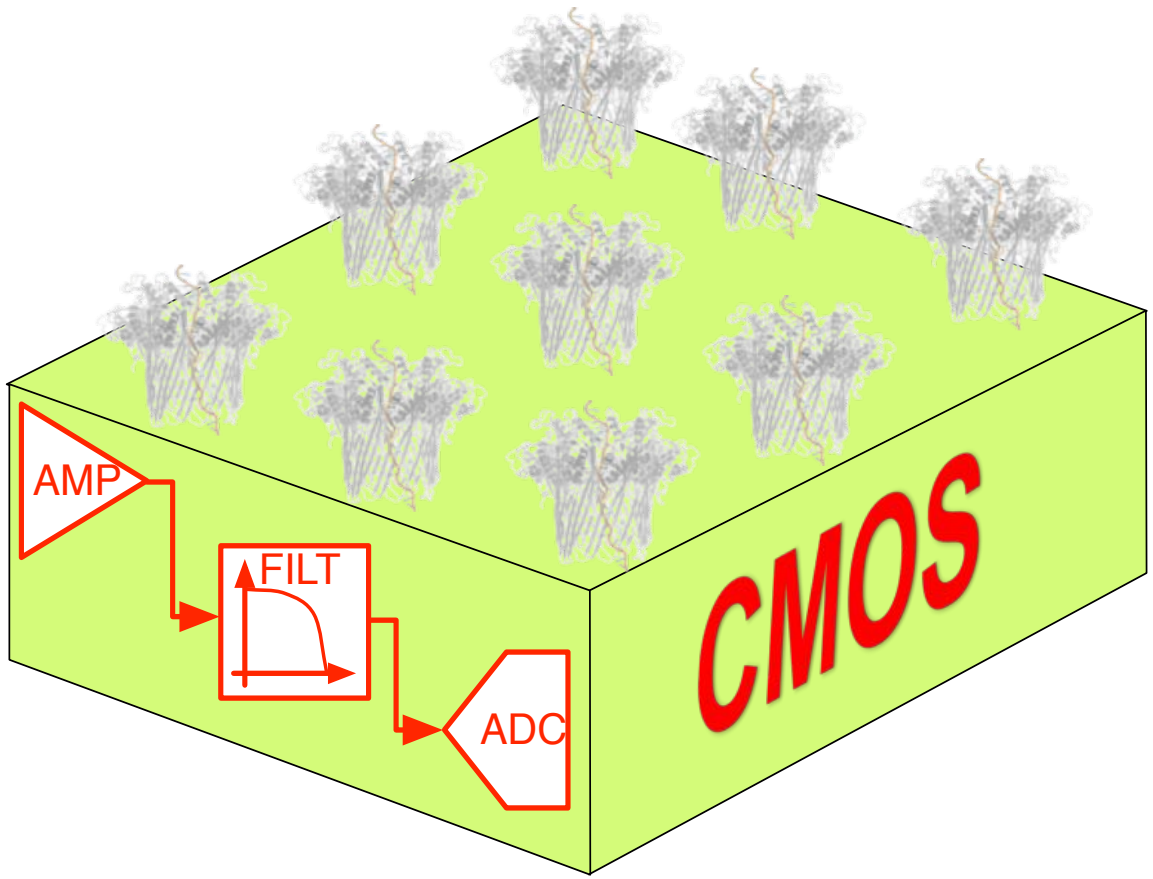
Nanopore Front-End Signals



[©Oxford Nanopore Tech]

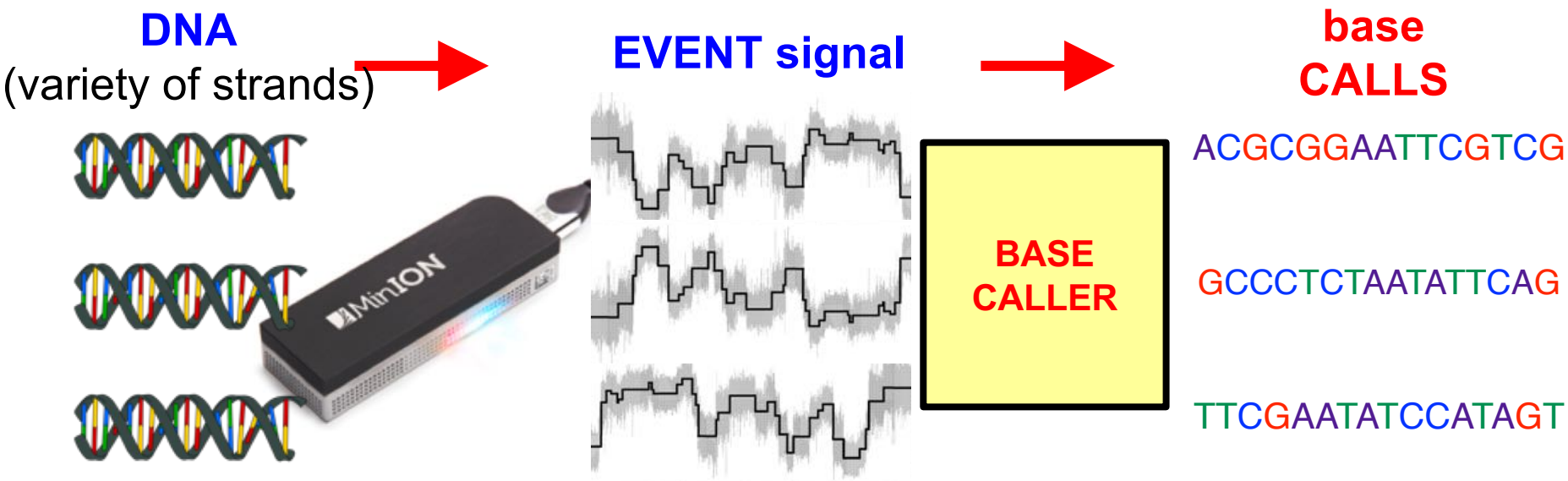
Nanopores + CMOS

- Not just nanopores
- A successful blend of…
  - ***nano***tech
    - sensors
  - ***micro***tech
    - mixed-signal CMOS
    - microfluidics



[©Oxford Nanopore Tech]

Computational Burden



- ~500 DNA bases/sec./channel
- ~500 channels (1 cm^2)
- 500 × 500 = 250,000 bases/sec.
  - 1 human genome / 3.5 hours

- ~1000 DNA bases/sec./core
- 250,000/1000 = 250 cores needed
  - **~25** W / core
- **~25** × 250 ~ 6,000 W
  - for real-time operation

Basecalling Algorithm

```
for: L = 0 to # of models/DNA (1 – 3)
  for: i = 0 to # of events/DNA strand ~(10^3 – 10^6)
    for: j = 0 to # of states/model ~(10^2 – 10^4)
      for: k = 0 to # transitions ~(4 – 10^2)
        load T(k)
        calc E(k,event(i)) ← event sample
        calc P(k) = T(k) × E(k,event(i))
      end
      calc P(j) = max{P(k)}
    end
    calc max{P(j)}
  end
end
```

170B iterations / sec. → 3000 GOPS

GPU Basecalling: Loop Unrolling

```
for: L
 for: i
  for: j
   for: k
    load T(k)
    calc E(event(i))
    calc P(k) = T(k) × E(event(i))
   end
   calc P(j) = max{P(k)}
  end
  calc max{P(j)}
 end
end
```

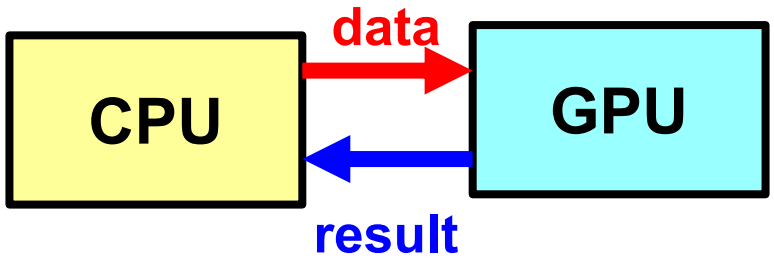
GPU Basecalling: Internal Loop Unrolling

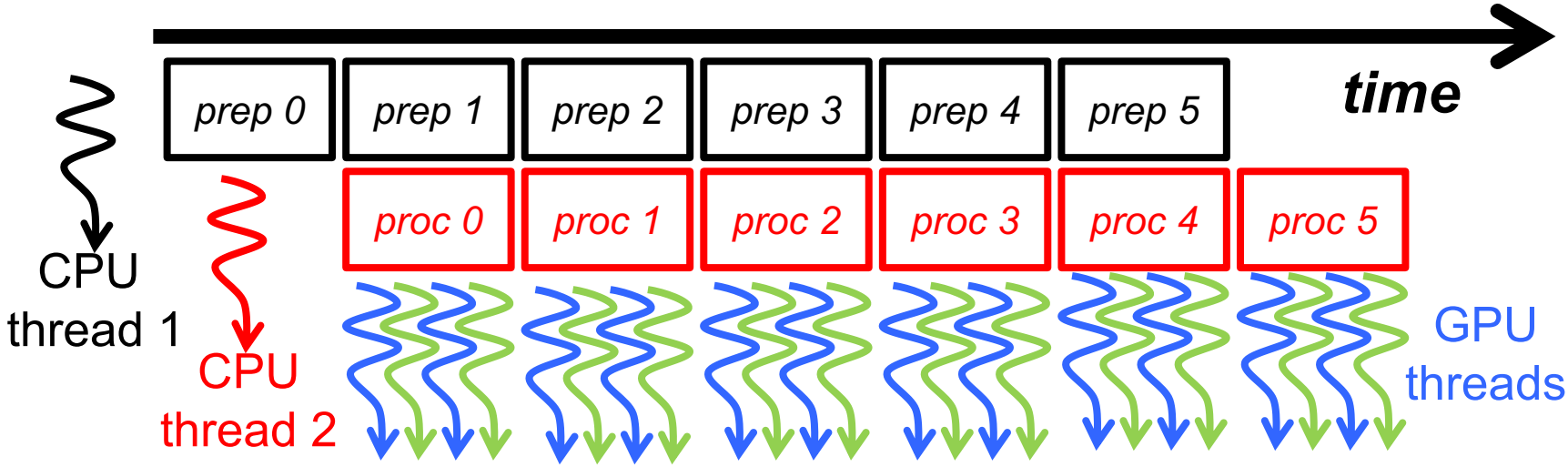
- 4096 threads
  - 128 threads per block
  - 32 blocks
- Threads assigned to
  - inner loop calculations
  - intermediate sorting
- Asynchronous convergence to local maximum

```
for: L
 for: i
  for: j
   for: k
    load T(k)
    calc E(i)
    calc P(k) = T(k) × E(i)
   end
   calc P(j) = max{P(k)}
  end
  calc max{P(j)}
 end
end
```

Streamline Communications

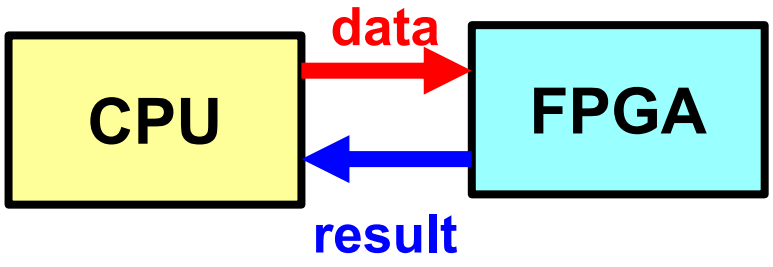
- Organizing dataflow
  - data preparation
  - data processing
  - interleave data sent to GPU
- Overall GPU gave ~6X speed-up in this case
  - GTX 680





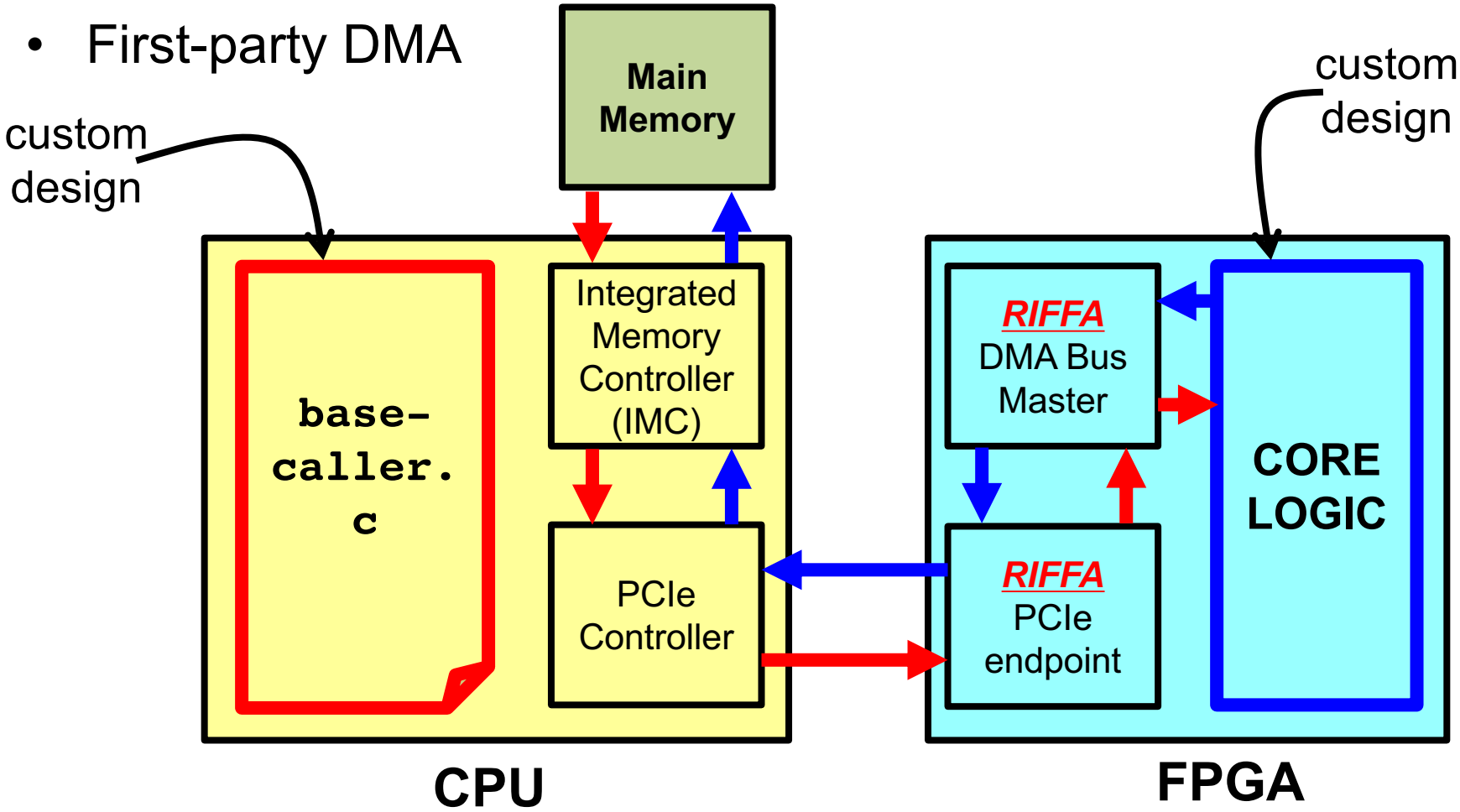
FPGA Acceleration

- For finer algorithm-to-hardware mapping
- RIFFA
  - Reusable Integration Framework for FPGA accelerators
    - from UCSD
  - open-source comms between FPGA core and CPU
  - runs over PCIe
- In the process of implementing basecalling



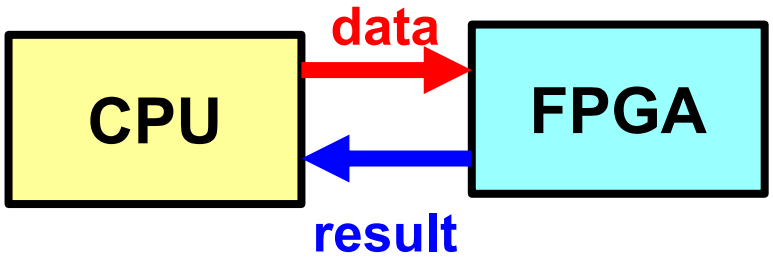
RIFFA Enabled CPU-FPGA Acceleration

- First-party DMA



RIFFA Hardware/Software Interface

- In the FPGA: Hardware Interface
  - a simple handshaking protocol
    - val/rdy settings, etc.
  - core can implement a simple controller to handle it
- In the CPU: Software Interface
  - a simple data transfer API is available
    - C/C++, Python, Java, Matlab
  - **`fpga_send()`**: CPU → FPGA
  - **`fpga_recv()`**: CPU ← FPGA
  - duplex comms possible with multithreading



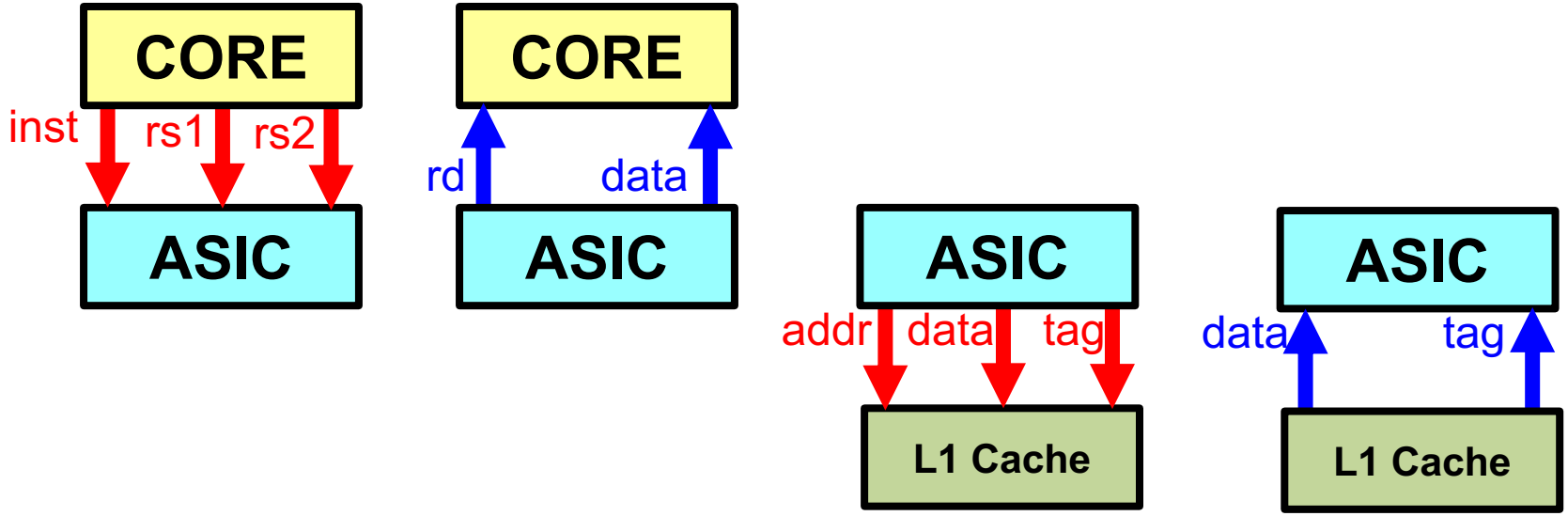
Performance Potentials

- RIFFA: ~ 800 MB/s transfer bandwidth
  - PCIe 3.0 (1 lane), ~80% of peak
  - downstream (CPU-to-FPGA)
- ~ 10% of resources consumed
  - Virtex-7 (28-nm CMOS)
- Core basecaller implementation
  - 100-MHz clock, Virtex-7
  - 40% FPGA and software overhead
  - 170,000 bp/s (1 human genome per 5 hours)
  - 5 W

SoCs

- Tighter integration in SoC form: CORE+ASIC
  - RISC-V (Rocket) + Rocket Custom Coprocessor (RoCC)
  - Custom 32-bit instructions
    - facilitate CPU/ASIC/Memory communications

31 – 25	24 – 20	19 – 15	14	13	12	11 – 7	6 – 0
funct7	rs2	rs1	xd	xs1	xs2	rd	opcode
7	5	5	1	1	1	5	7



The End

- Thanks for your attention
- Graduate student acknowledgments
  - Zhongpan Wu
  - Karim Hammad
  - Roksana Hussain
  - Robinson Mittmann
  - Xiaoyong Zhong
  - Sumaia Atiwa
  - Mahdieh Abbaszadegan
  - Chengjie Wang
  - Yiyun Huang
- Thanks to CMC!