

The 'Magic' Behind

# High-Speed Digital Communications

---

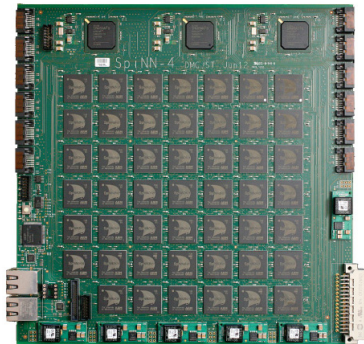
Jonathan Heathcote

# Motivation

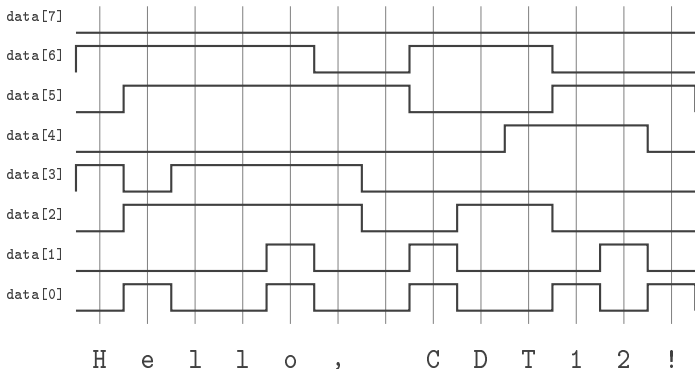
- Computers are **fast**
- Can process **huge amounts of data** per second
- SpiNNaker is (or will be...) a **million-core computer**
- ...so **really huge** amounts of data!
- Taster project **building a high-speed link** to communicate with it

## SpiNNaker: A Brain Emulator

- 1,200 boards, 1,036,800 cores
- 12 Gigabits/sec between boards
- 120 Gigabits/sec within board
- 100 Megabits/sec Ethernet too slow
- Difficult to debug
- Difficult to connect to sensors
- Need more speed!

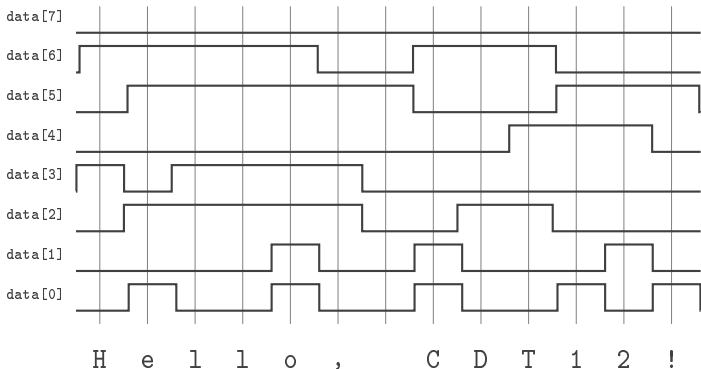


# Parallel Communications



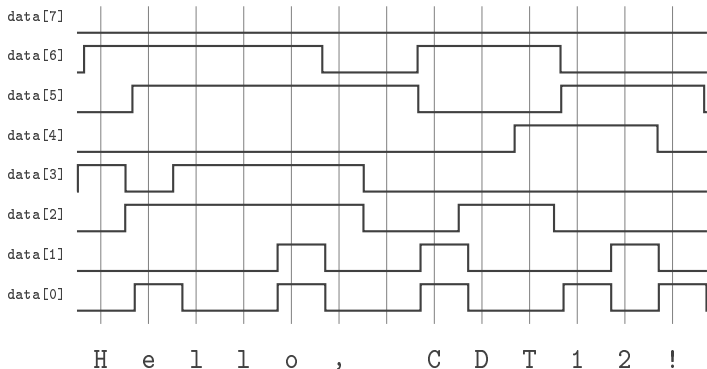
\*\*\*\*

# Parallel Communications



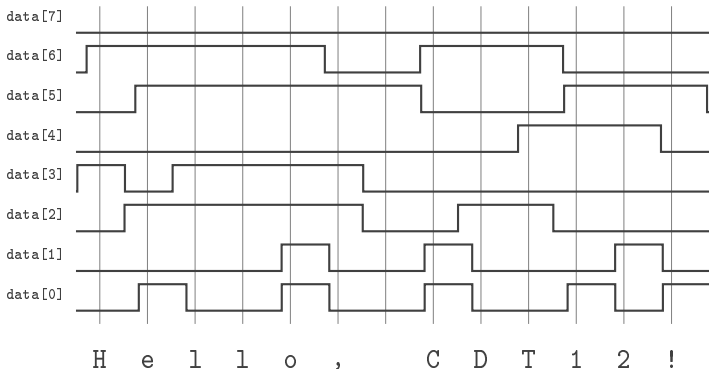
\*\*\*

# Parallel Communications



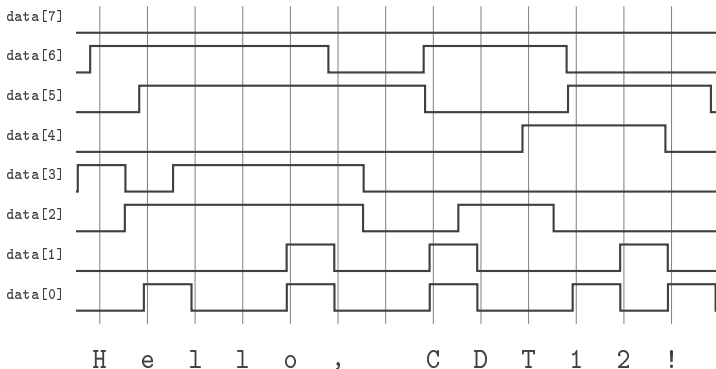
\*\*

# Parallel Communications



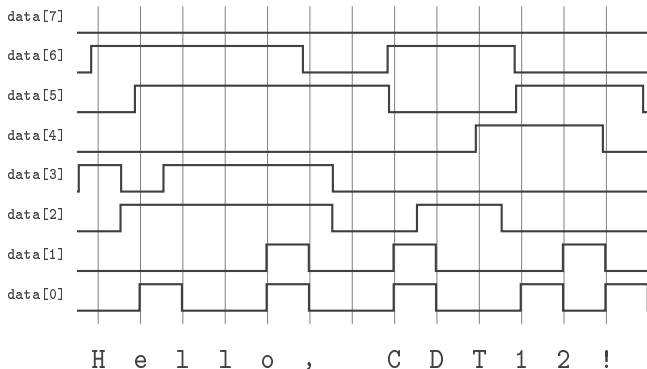
\*

## Parallel Communications



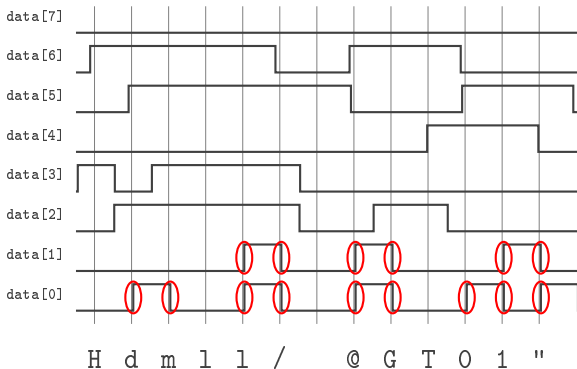


# Parallel Communications



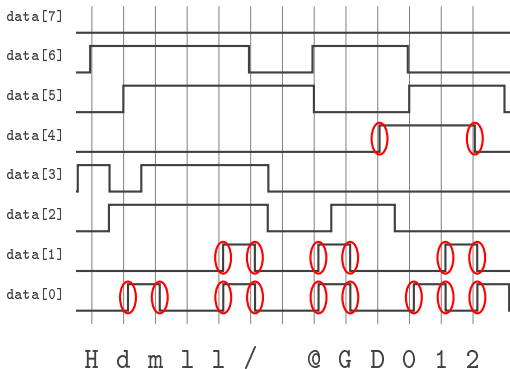
\*\*\*

# Parallel Communications



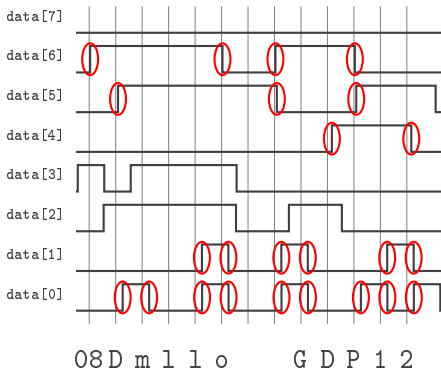
\*\*

# Parallel Communications



\*

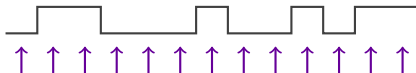
# Parallel Communications



# Serial Transmission

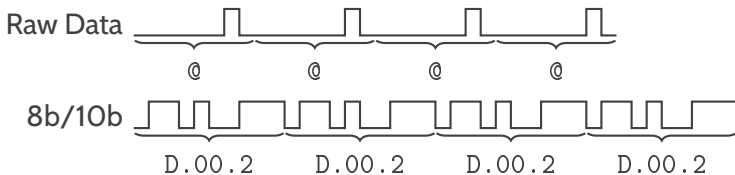


## Sampling the Signal



- **Midpoint** between transitions
- **Phase detector** can find this...
- ...but requires **frequent transitions**

## 8b/10b Coding



- Encodes 8-bit bytes into 10-bit symbols
- Transitions at least every 5 bits
- 256 'data' + 8 'special' codes = 264 valid values

## From Bits to Bytes

```

...011101000101010011101101100010110
110101011000101110101000111010010011
10100111011011010101010110000110001110
100111000011010011001101110001011110
100110000110110001011000101001100010
111010100011101001001110101000100100
010101011011100111001010010011001101
110000110100111010001010100111011011
000101101101010110001011101010001110
100100111010011101101101010101100001
100011101001110000110100110011011...

```



```

...0xfd0xdc??o0xc70x00?Ux
0xf20xf6P???0x7fo0xc70x00
?0xf5?0xa2P0xf60xfd0xdc??
o0xc70x00?Ux0xf20xf6P???0
x7fo0xc70x00?0xf5?0xa2P0x
f60xfd0xdc??o0xc70x00?Ux0
xf20xf6P???0x7fo0xc70x00?
0xf5?0xa2P0xf60xfd0xdc??o...

```

- Where does one byte start and another begin?



## From Bits to Bytes

```

...011101000101010011101101100010110
110101011000101110101000111010010011
1010011101101101010101010000110001110
100111000011010011001101110001011110
100110000110110001010100111101100010
100110001011101010001110100100111010
100010010001010101101110011100101001
001100110111000011010011101000101010
011101101100010110110101011000101110
101000111010010011101001110110110101
01011000011001010011100111010011...
    
```



```

...0xfd0xdc??o0xc70x00?Ux
0xf20xf6P???>0x7fo0xc70x0
0?0xf5?0xa2P0xf60xfd0xdc?
?o0xc70x00?Ux>0xf20xf6P??
?0x7fo0xc70x00?0xf5?0xa2P
0xf60xfd0xdc??>o0xc70x00?
Ux0xf20xf6P???0x7fo0xc70x
00?0xf5?0xa2>P0xf60xfd0xd...
    
```

- Where does one byte start and another begin?
- Special **'comma'** code in 8b/10b inserted periodically

## From Bits to Bytes

<pre> ...011101000101010011101101100010110 110101011000101110101000111010010011 101001110110101010101010000110001110 100111000011010011001101110001011110 10011000011011000101010011101100010 100110001011101010001110100100111010 100010010001010101101110011100101001 001100110111000011010011101000101010 011101101100010110110101011000101110 101000111010010011101001110110110101 01011000011001010011100111010011...                 </pre>	<p>→</p>	<pre> ...0xfd0xdc??o0xc70x00?Ux 0xf20xf6P??DT12! Hello CDT12! Hello CDT12! Hello CDT12! Hello CDT12! Hello CDT12! Hello CDT12!...                 </pre>
---	----------	--

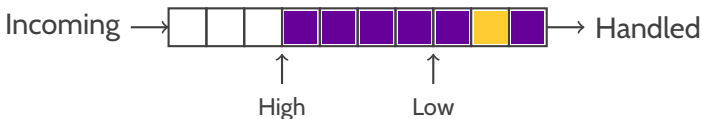
- Where does one byte start and another begin?
- Special 'comma' code in 8b/10b inserted periodically
- When a comma is found, align bits

## Differing Transmission Rates

- Clock accuracy  $\approx \pm 300$  parts-per-million (ppm)
- Off-by-one every 1666 bits
- At 16 GHz, up to **1MB extra sent/received per second**
- Solved by inserting extra 'skip packets'

## Skip Packets Example: Sender Faster

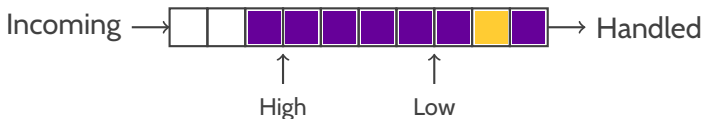
- Fills faster than empties
- Eventually falls above 'high' mark
- Skip packets **skipped to empty buffer faster**



- Data Packet
- Skip Packet

## Skip Packets Example: Sender Faster

- Fills faster than empties
- Eventually falls above 'high' mark
- Skip packets **skipped to empty buffer faster**



- Data Packet
- Skip Packet

## Skip Packets Example: Sender Faster

- Fills faster than empties
- Eventually falls above 'high' mark
- Skip packets **skipped to empty buffer faster**



- Data Packet
- Skip Packet

## Skip Packets Example: Sender Faster

- Fills faster than empties
- Eventually falls above 'high' mark
- Skip packets **skipped to empty buffer faster**



- Data Packet
- Skip Packet

## Skip Packets Example: Sender Faster

- Fills faster than empties
- Eventually falls above 'high' mark
- Skip packets **skipped to empty buffer faster**

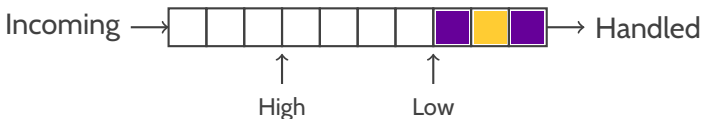


- Data Packet
- Skip Packet



## Skip Packets Example: Receiver Faster

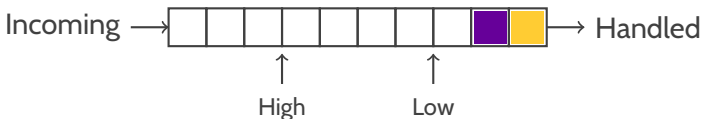
- Empties faster than fills
- Eventually falls below 'low' mark
- Skip packets **duplicated to refill buffer**



- Data Packet
- Skip Packet

## Skip Packets Example: Receiver Faster

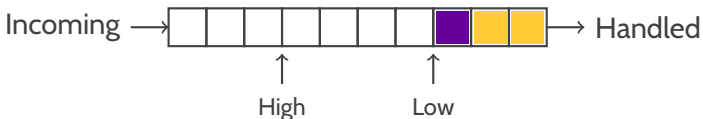
- Empties faster than fills
- Eventually falls below 'low' mark
- Skip packets **duplicated to refill buffer**



- Data Packet
- Skip Packet

## Skip Packets Example: Receiver Faster

- Empties faster than fills
- Eventually falls below 'low' mark
- Skip packets **duplicated to refill buffer**



- Data Packet
- Skip Packet

## Skip Packets Example: Receiver Faster

- Empties faster than fills
- Eventually falls below 'low' mark
- Skip packets **duplicated to refill buffer**



- Data Packet
- Skip Packet

## Skip Packets Example: Receiver Faster

- Empties faster than fills
- Eventually falls below 'low' mark
- Skip packets **duplicated to refill buffer**



- Data Packet
- Skip Packet

## Skip Packets Example: Receiver Faster

- Empties faster than fills
- Eventually falls below 'low' mark
- Skip packets **duplicated to refill buffer**



- Data Packet
- Skip Packet

## Skip Packets Example: Receiver Faster

- Empties faster than fills
- Eventually falls below 'low' mark
- Skip packets **duplicated to refill buffer**



- Data Packet
- Skip Packet

## Skip Packets Example: Receiver Faster

- Empties faster than fills
- Eventually falls below 'low' mark
- Skip packets **duplicated to refill buffer**



- Data Packet
- Skip Packet



## Skip Packets Example: Receiver Faster

- Empties faster than fills
- Eventually falls below 'low' mark
- Skip packets **duplicated to refill buffer**



- Data Packet
- Skip Packet

## Skip Packets Example: Receiver Faster

- Empties faster than fills
- Eventually falls below 'low' mark
- Skip packets **duplicated to refill buffer**



- Data Packet
- Skip Packet

## Skip Packets Example: Receiver Faster

- Empties faster than fills
- Eventually falls below 'low' mark
- Skip packets **duplicated to refill buffer**



- Data Packet
- Skip Packet

# Where Is All This Going?

- No Idea.

## Where Is All This Going?

- So... We have a fast link, now what?
- Take some ideas forward to 'SpiNNaker 2'

## Summary

- SpiNNaker needs a fast link to the outside world
- Simple parallel links limit speed due to skew
- Avoid skew problems using **serial**
- Frequent transitions for **phase detection** with **8b/10b**
- Find the start of bytes using **commas**
- Compensate for speed differences using **skip packets**





A digital signal waveform is shown above the text. It consists of a series of rectangular pulses. The first three pulses are wide and low, corresponding to the letters 'A', 'n', and 'y'. This is followed by a gap, then a series of many narrow, high-frequency pulses corresponding to the letters 'Q', 'u', 'e', 's', 't', 'i', 'o', 'n', 's'. The final pulse is a single wide, low pulse corresponding to the question mark '?'.

A n y            Q u e s t i o n s ?