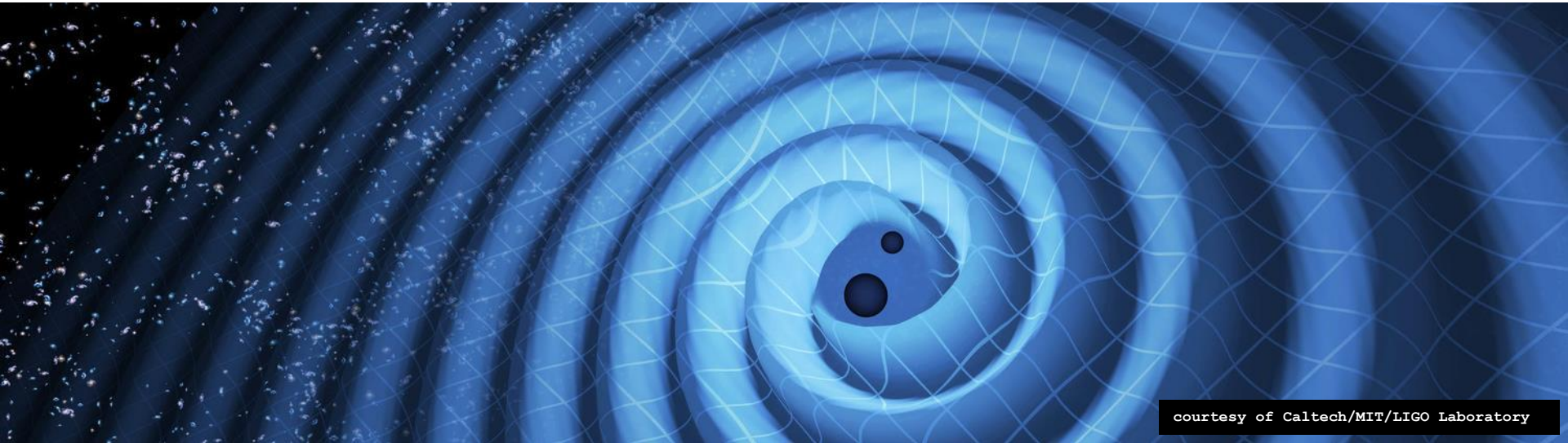
Detecting Binary Black Hole Mergers through LIGO Gravity Wave Measurements with Ultra96

AVNET
Reach Further™



Gravity primer

What is Gravity?

Sir Isaac Newton

$$F = \frac{Gm_1m_2}{r^2}$$

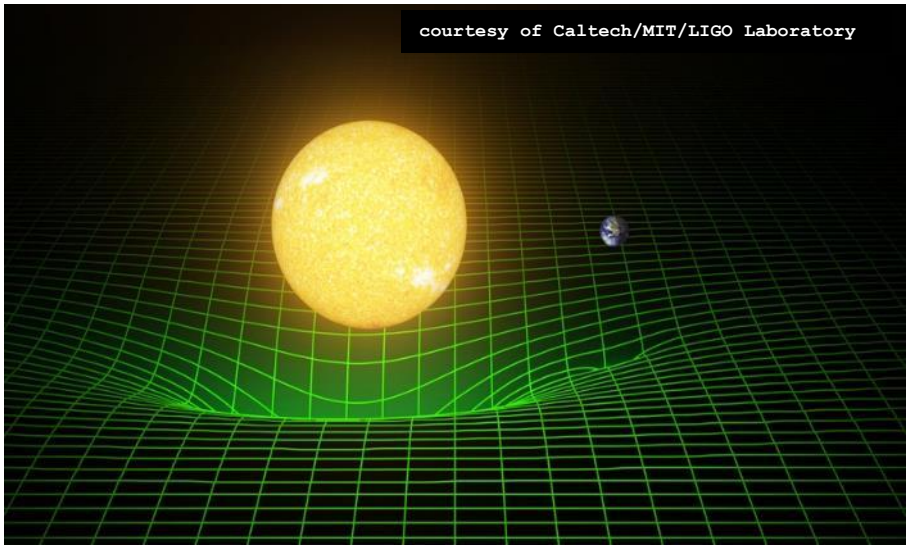
Painting by Sir Godfrey Kneller 1689: http://www.newton.ac.uk/about/art-artefacts/newton-portrait

What are Gravity Waves (GW) ?

- Einstein said (paraphrased):

$$h_{\mu v} \approx \frac{1}{r}\frac{G}{c^4} I_{\mu v}$$

Gravitational waves are *propagating mathematical solutions* to the Field Equations of General Relativity that show space and time are dynamically coupled

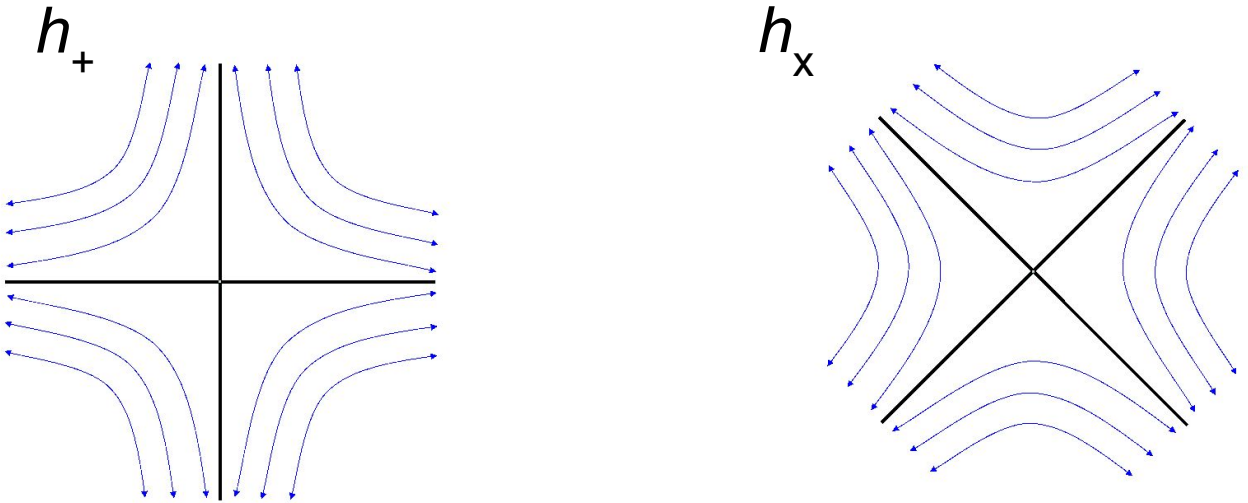


What if you are not Einstein?

- Just accept that GW are physical ripples in the fabric of space-time ☺

AVNET

A strain through space and time:



2 polarizations of *strain*

courtesy of Caltech/MIT/LIGO Laboratory

AVNET

courtesy of Caltech/MIT/LIGO Laboratory

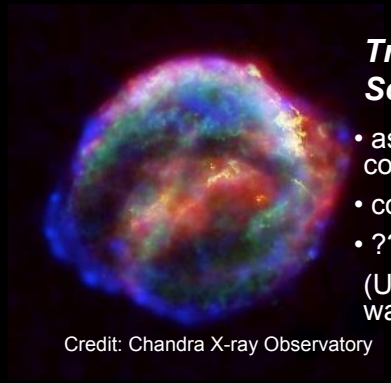
The Astrophysical Gravitational-Wave Source Catalog

Credit: Bohn, Hébert, Throwe, SXS

Coalescing Binary Systems

- Black hole – black hole
- Black hole – neutron star
- Neutron star – neutron star

(modeled waveform)

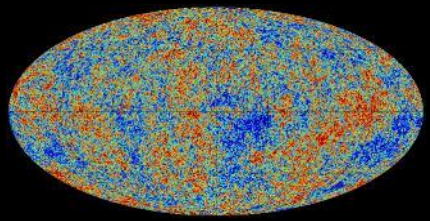


Credit: Chandra X-ray Observatory

Transient'Burst' Sources

- asymmetric core collapse supernovae
- cosmic strings
- ???

(Unmodeled waveform)

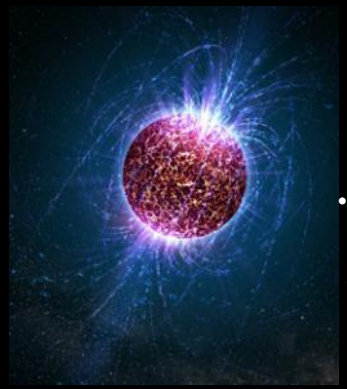


Credit: Planck Collaboration

Stochastic Background

- residue of the Big Bang
- incoherent sum of unresolved 'point' sources

(stochastic, incoherent noise background)



Credit: Casey Reed, Penn State

Continuous Sources

- Spinning neutron stars

(monotone waveform)

The Gravitational Wave Spectrum

Sources

Quantum fluctuations in early universe

Binary Supermassive Black Holes in galactic nuclei

Compact Binaries in our Galaxy & beyond

Compact objects captured by Supermassive Black Holes

Rotating NS, Supernovae

courtesy of Caltech/MIT/LIGO Laboratory

wave period

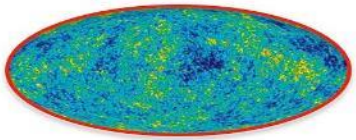
age of universe | years | hours | sec | ms

log(frequency)

-16 -14 -12 -10 -8 -6 -4 -2 0 +2

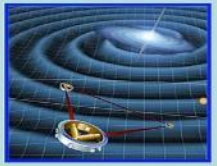
Detectors

Cosmic microwave background polarization



Pulsar Timing

Space Interferometers

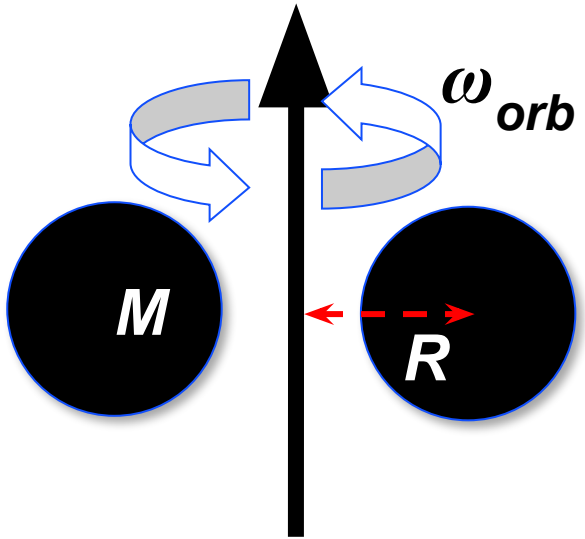


Terrestrial interferometers

Gravity waves and binary black holes:

10M$_\bullet$ Binary Black Hole System

$$h \approx \frac{8GMR^2\omega_{orb}^2}{rc^4} \sim 10^{-21}$$

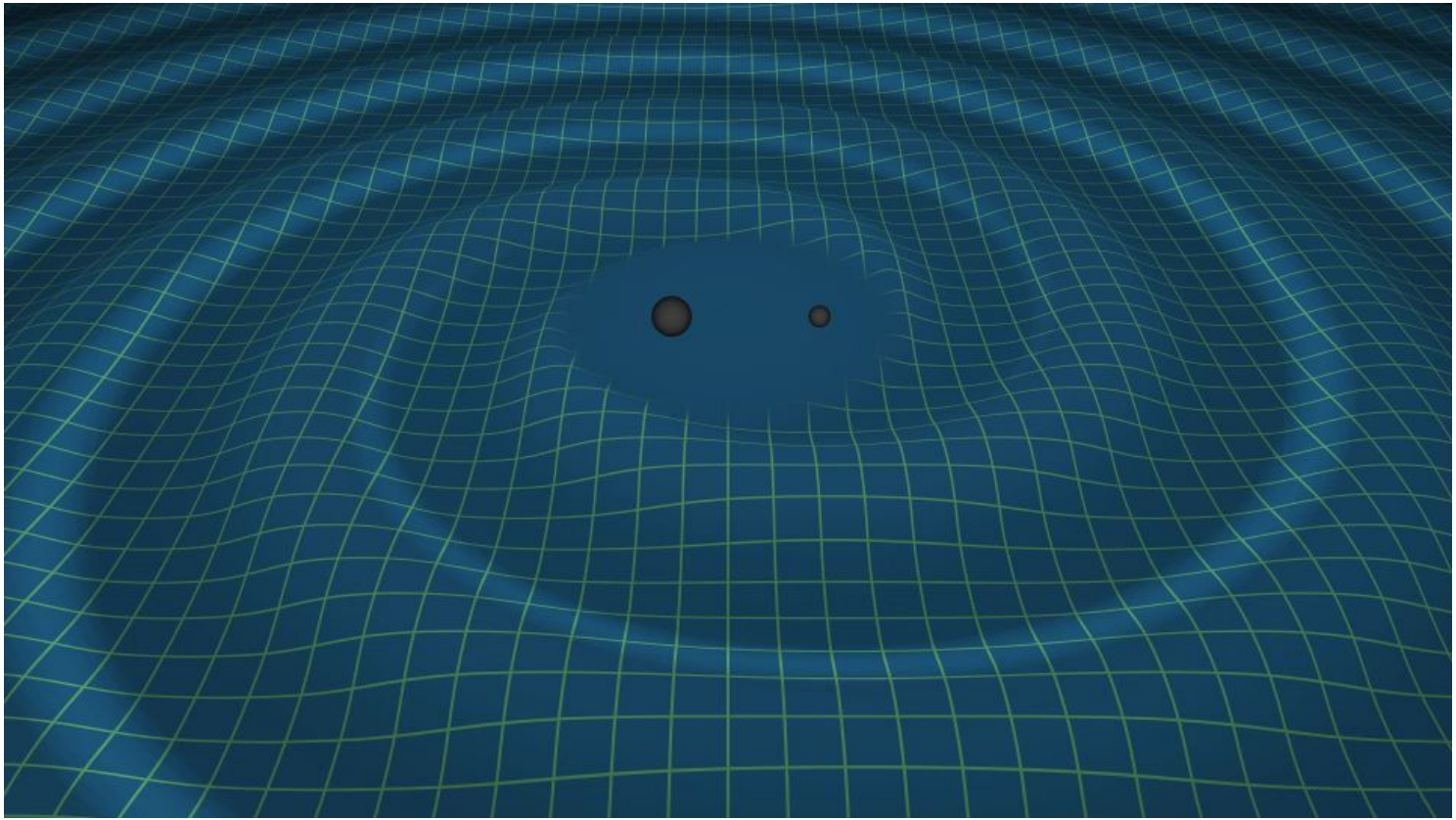


Physically, *h* (a *strain*): $\Delta L/L$

courtesy of Caltech/MIT/LIGO Laboratory

AVNET

Binary black hole merger simulation:



courtesy of Caltech/MIT/LIGO Laboratory

AVNET

LIGO and Laser Interferometry

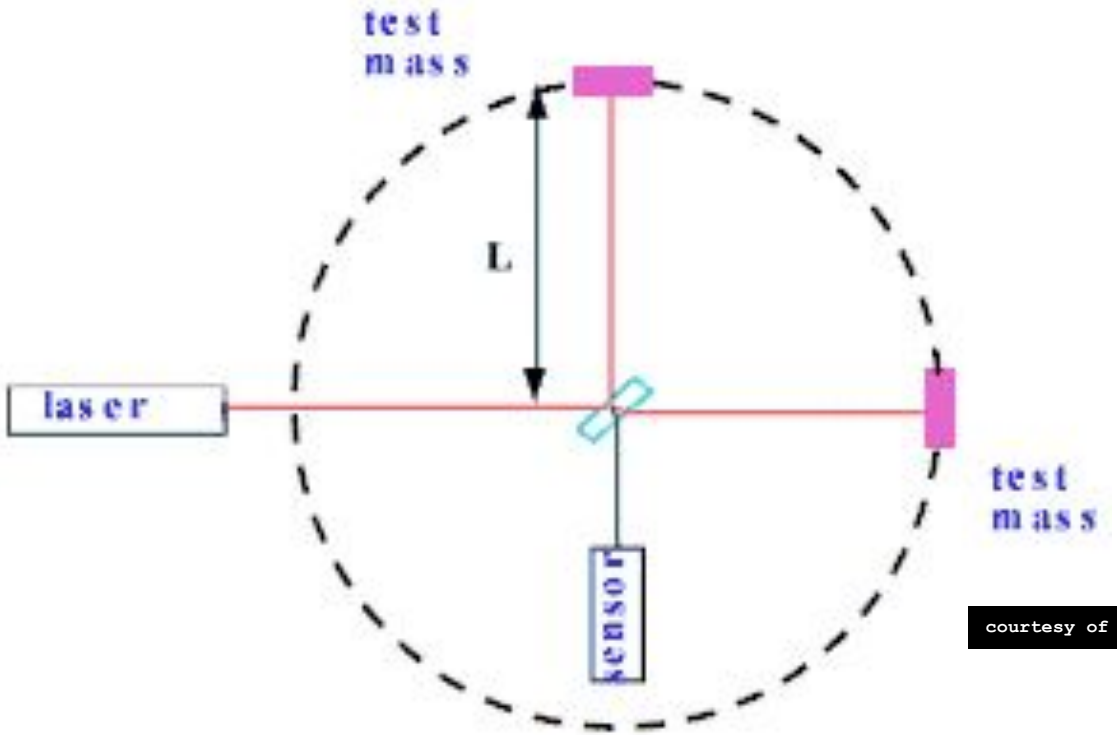
LIGO = Laser Interferometer Gravitational-wave Observatory:

Hansford, Washington

(Each vacuum tube is ~4km)

AVNET

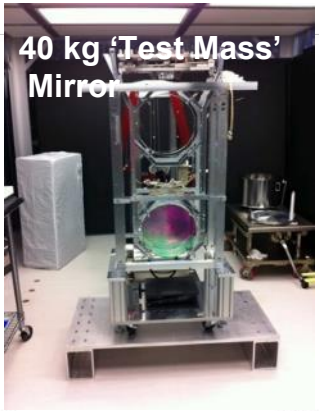
Gravity wave effects on a laser interferometer:

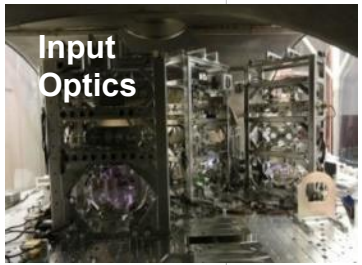


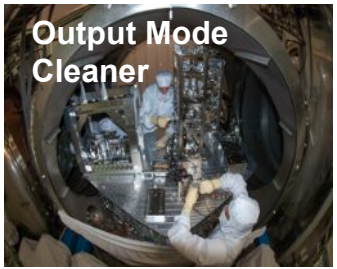
courtesy of Caltech/MIT/LIGO Laboratory

AVNET

LIGO = Laser Interferometer Gravitational-wave Observatory:







courtesy of Caltech/MIT/LIGO Laboratory

AVNET

Noise floor determines detection limitations (receiver sensitivity):

Fundamental Noises:

I. Displacement Noises

→ $\Delta L(f)$

- Seismic noise
- Radiation Pressure
- Thermal noise
  - Suspensions
  - Optics

II. Sensing Noises

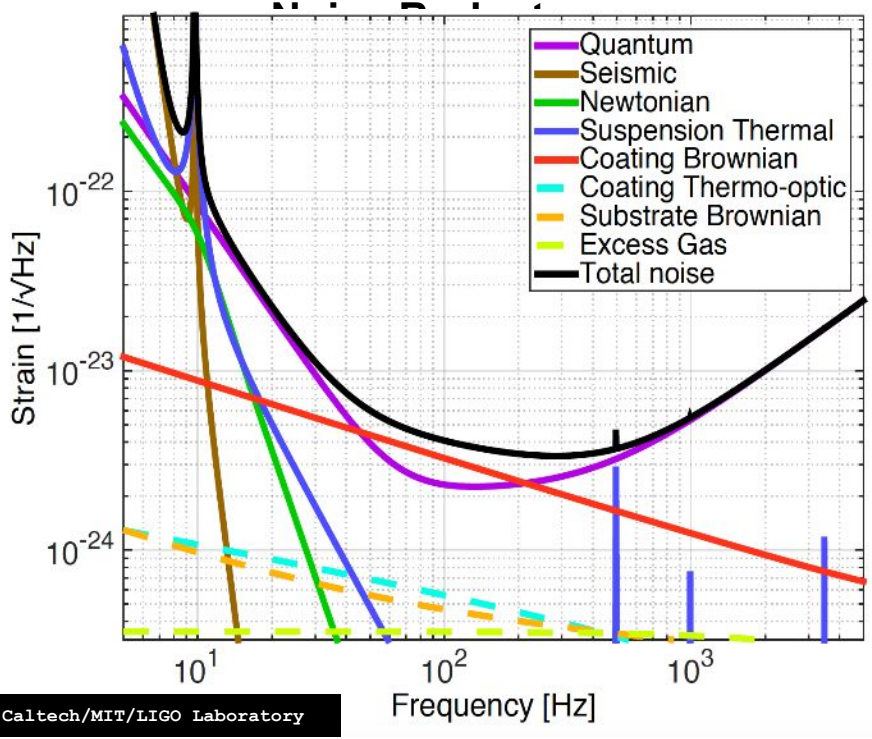
→ $\Delta t_{photon}(f)$

- Shot Noise
- Residual Gas

Technical Noises:

→ ***Hundreds of them…***

Advanced LIGO Design

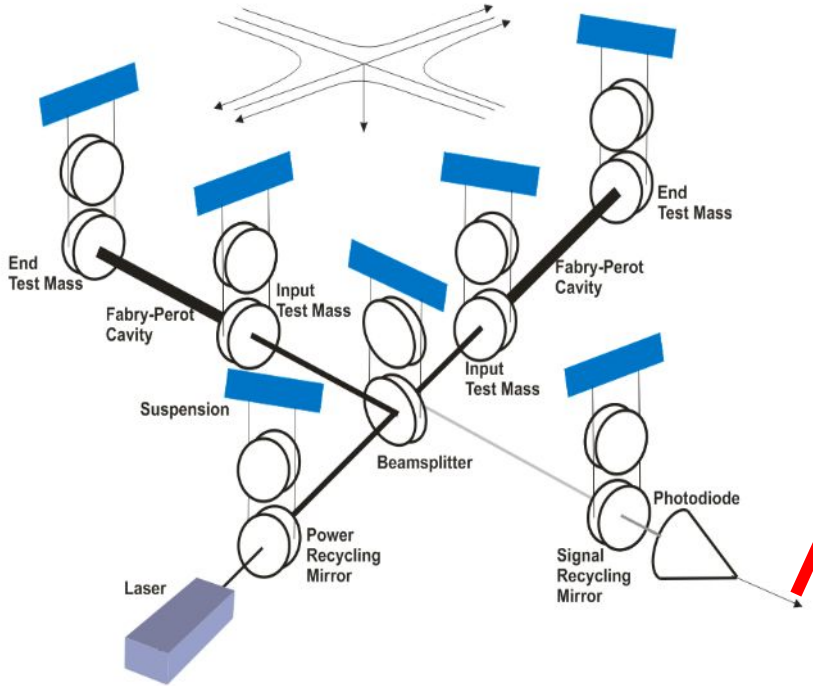


courtesy of Caltech/MIT/LIGO Laboratory

AVNET

LIGO signals, data and Event catalog

LIGO = Laser Interferometer Gravitational-wave Observatory:



courtesy of Caltech/MIT/LIGO Laboratory

***h*(*t*): 16384 Hz rate**
LOSC *h*(*t*): 4096 Hz rate

Saved on the LIGO servers in HDF5 file format

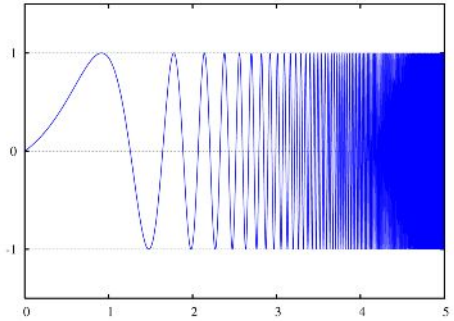
Analyzed and processed, GW Events cataloged

AVNET

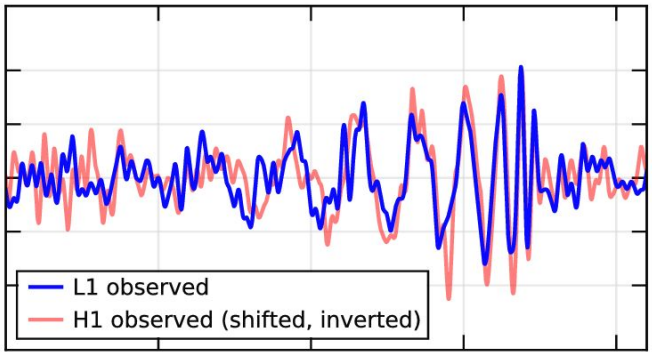
What signal are they looking for?

After all the signal processing what is the confirming signal for a binary merger: Chirp!

Constant envelope exponential chirp:



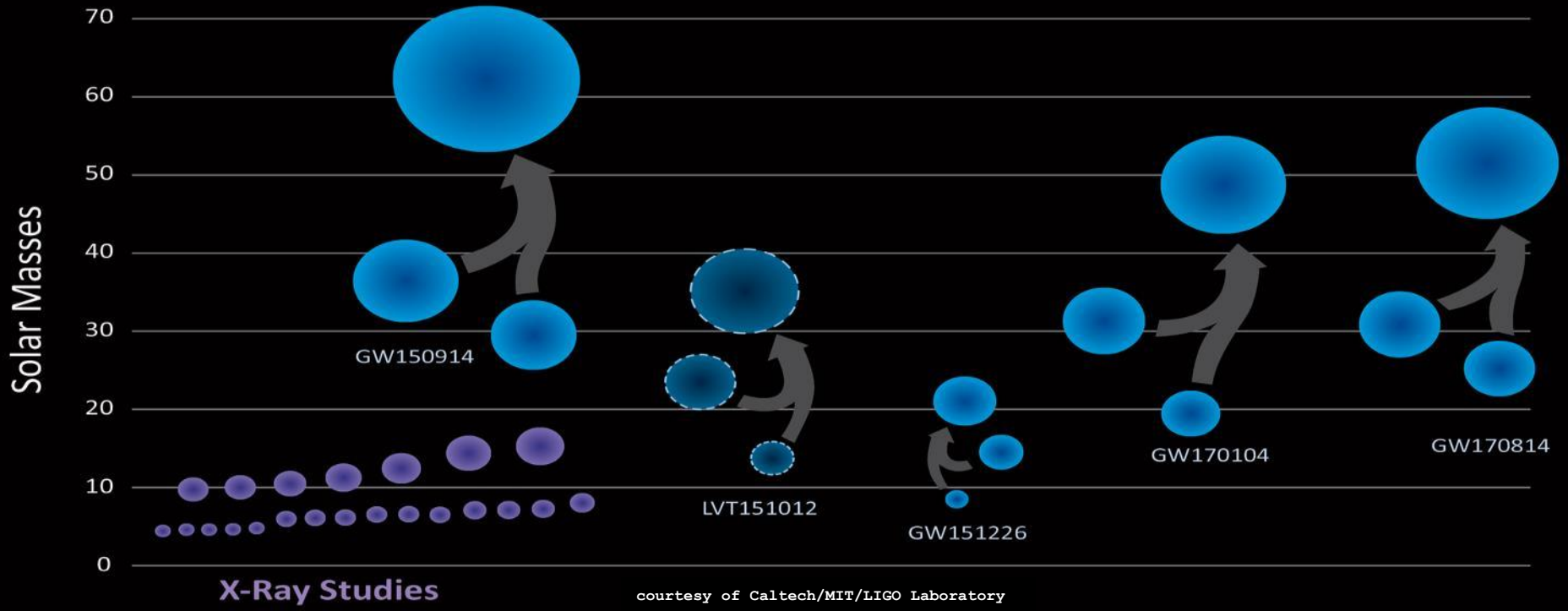
Actual measured strain signal:



courtesy of Caltech/MIT/LIGO Laboratory

AVNET

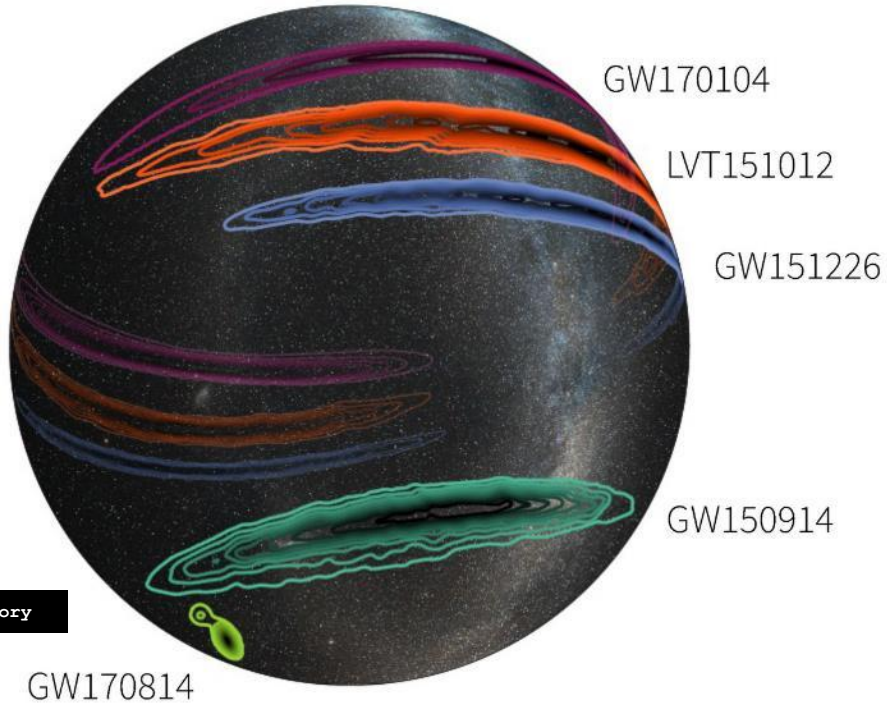
Black Holes of Known Mass



courtesy of Caltech/MIT/LIGO Laboratory

LIGO/VIRGO

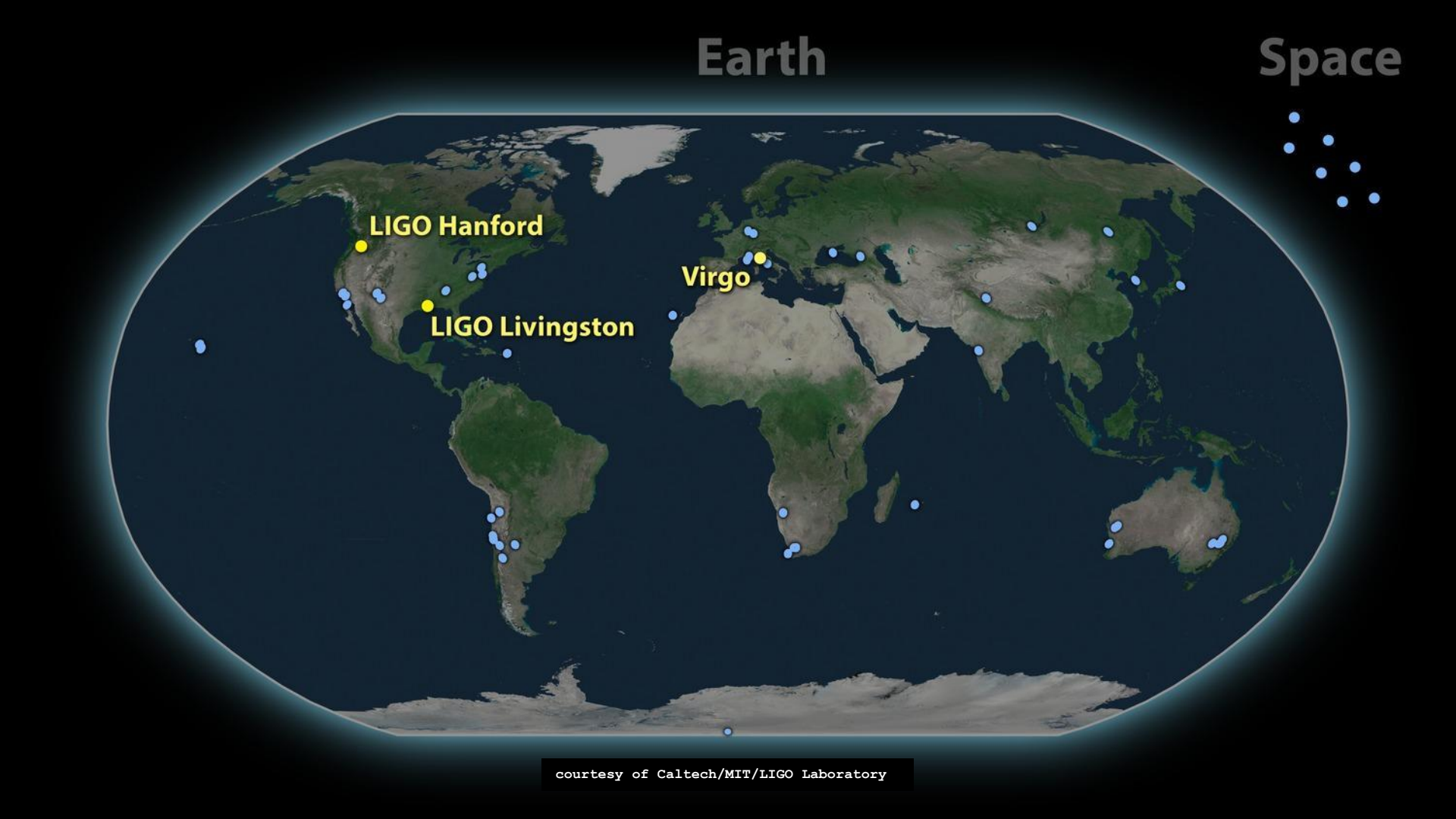
Locating the origin using multiple detectors and TDOA:



courtesy of Caltech/MIT/LIGO Laboratory

(Skymap projection onto a sphere
of detected events as of Sept 2017)

AVNET


Earth
Space
LIGO Hanford
LIGO Livingston
Virgo
courtesy of Caltech/MIT/LIGO Laboratory

Event case study: GW150914

- Black hole masses involved: $36M_{\odot}$ $29M_{\odot} \rightarrow 62M_{\odot}$
- Distance of origin from Earth: over 1 BILLION Light Years
- LIGO's 4km mass movement:
  - Approx. 1000 times less than width of a proton (2e-18 Meters)
  - Proportionally equivalent to changing the distance to the nearest star (Proxima Centauri) 4.3 light years away by one hair's width

- General event notes:
  - LVT = LIGO Virgo Trigger which is a candidate that has not been scrutinized enough to not just be a false alarm
  - Other event examples and details found here: https://losc.ligo.org/events/
  - Notation decoder:
    - **GW** = Gravity Wave detected with great certainty (The false alarm rate for this event is statistically estimated to be less than 1 event per 203,000 years)
    - **15** = Year, **09** = Month, **14** = Day

AVNET

Using Ultra96 for gravity wave detection and the case for PYNQ™

Accessing LIGO data yourself:

- Data (https://losc.ligo.org/data/)
  - Public
    - Confirmed “events”, some triggers and partial timelines with reliability metrics (mostly 4kHz) from multiple detectors
    - More will be added in the future
  - LIGO members (access to full timeline and 16kHz)
- Easy way is to use Python (or C or Matlab)
  - Use readligo.py (https://losc.ligo.org/tutorial_gps/)

Data wisdom:

- Download .hdf5 from the server then easy peasy with readligo.py:

  strain, time, dq = rl.loaddata('ligo_data/H-H1_LOSC_4_V1-842653696-4096.hdf5', 'H1')

- Read the instructions! They explain all the strings used above (like 'H1' is Handon, Wa and 'L1' is Livingston, La, etc.)!

- When searching for your own events learn where injection (test) samples are located!

- Use the data quality flags to mask out invalid samples that occur due to system maintenance!

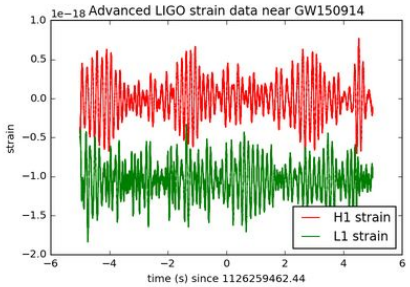
- For a manipulatable example study the Jupyter Notebook!

LIGO Jupyter Notebook for Cataloged Events on Ultra96:

- All notebooks (including for download to run on Ultra96*): https://losc.ligo.org/tutorials/
- Direct link to Notebook in an Azure binder: https://notebooks.azure.com/losc/libraries/tutorials/html/LOSC_Event_tutorial.ipynb

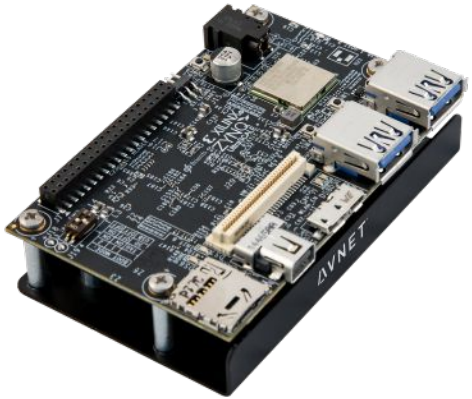
In [8]:
```
# plot +- deltat seconds around the event:
# index into the strain time series for this time interval:
deltat = 5
indxt = np.where((time >= tevent-deltat) & (time < tevent+deltat))
print(tevent)

if make_plots:
    plt.figure()
    plt.plot(time[indxt]-tevent,strain_H1[indxt],'r',label='H1 strain')
    plt.plot(time[indxt]-tevent,strain_L1[indxt],'g',label='L1 strain')
    plt.xlabel('time (s) since '+str(tevent))
    plt.ylabel('strain')
    plt.legend(loc='lower right')
    plt.title('Advanced LIGO strain data near '+eventname)
    plt.savefig(eventname+'_strain.'+plottype)
```

```
1126259462.44
```



courtesy of Caltech/MIT/LIGO Laboratory

* Xilinx's PYNQ™ for Ultra96 framework comes with Jupyter installed on the target



AVNET

Xilinx PYNQ™ for Ultra96

- **What is PYNQ?**

An open source software **framework** designed to make Ultra96 more Python friendly and easier for Python to interact with the PL in embedded systems. It is comprised of:

- PetaLinux (aarch64 kernel)
- Ubuntu Bionic root file-system
- Full Python (as opposed to Micro Python)
- Jupyter Notebooks
- Python libraries for using the Xilinx PS and PL

- **Why would I want to use it?**

In combination with using PL it has the ability to make some of your slow Python programs run FAST, really really FAST and allows Python to control hardware that other platforms could only dream about. It can also dramatically reduce design time and effort!

AVNET

When will Xilinx PYNQ™ be available for Ultra96?

- PYNQ for Ultra96 is coming soon, expected 1st week of Oct. 2018!!!

http://www.pynq.io & https://github.com/Xilinx/PYNQ
http://zedboard.org/product/ultra96

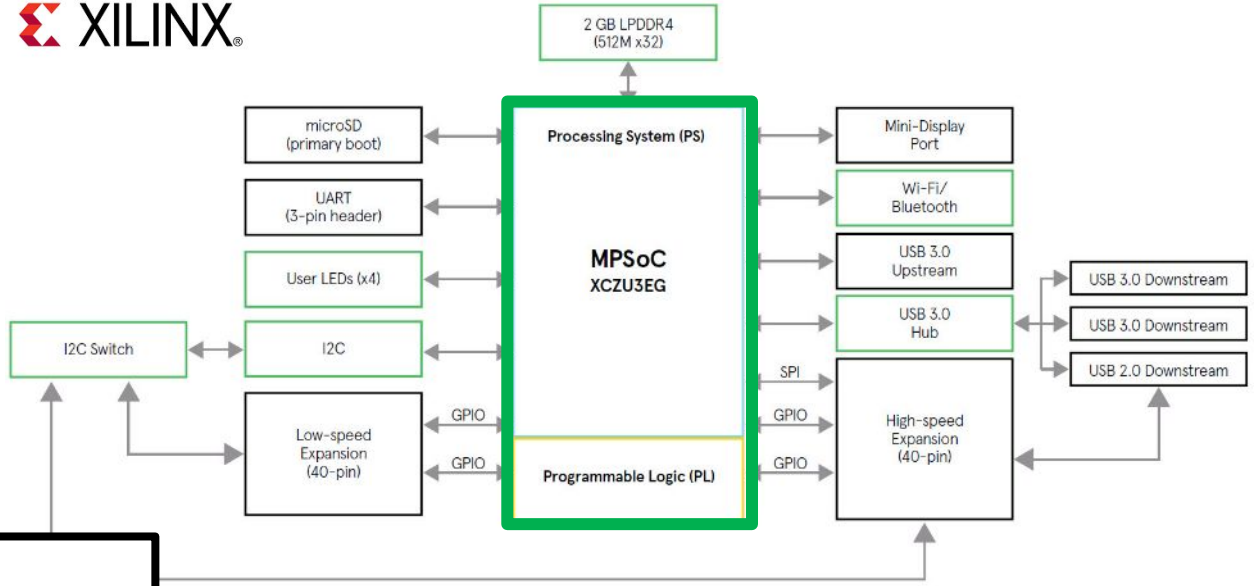
Ultra96 PYNQ platform will be hosted on Avnet's github:
http://github.com/Avnet

Also come to my 2nd Linaro Connect presentation later today: "A Call to Action: Accelerating Python with FPGAs"

AVNET

Accelerating the search for Black Holes with Ultra96:

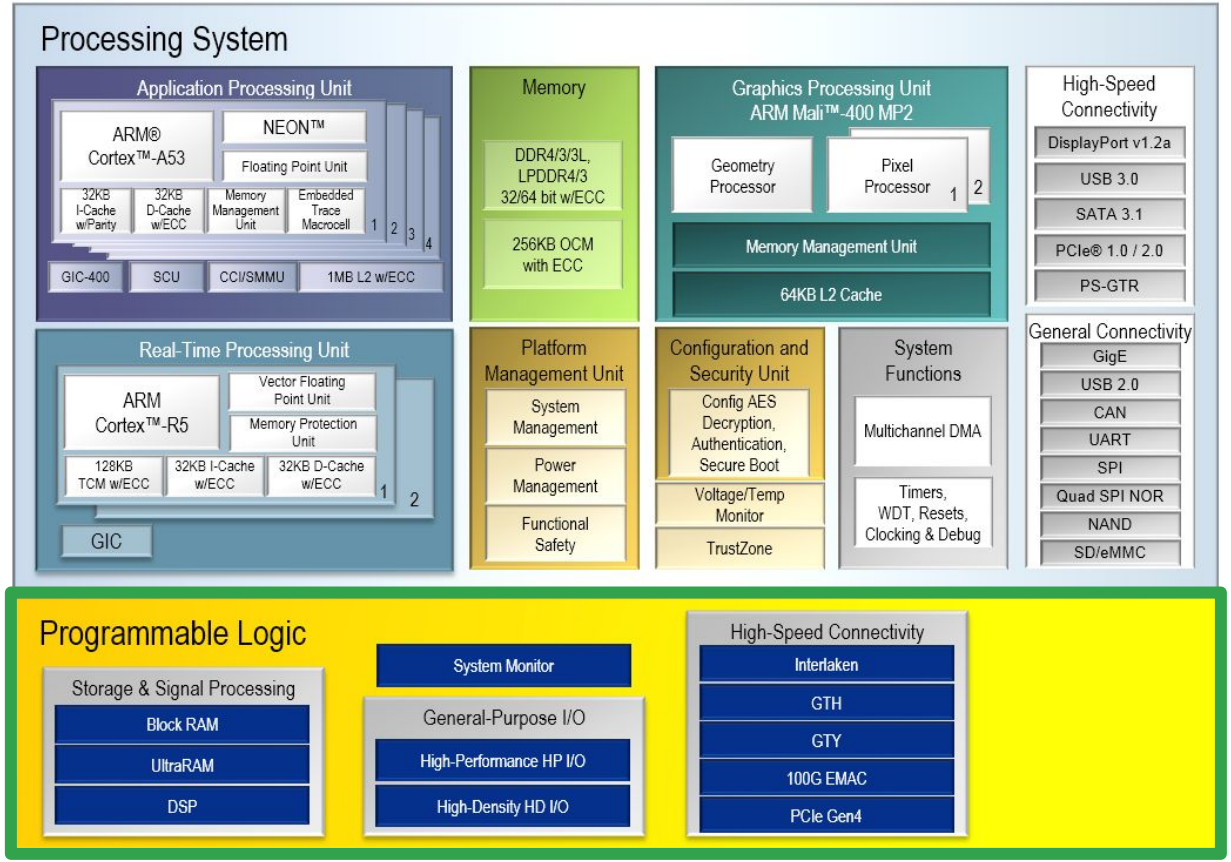
XILINX



Important terms:
PS = CPUs/Processing System
PL = Programmable Logic/Hardware,
(FPGA and some Xilinx hard IP)

AVNET

Ultra96 PS + PL all in one MPSoC ZYNQ:



Ultra96's ZU3EG
ZYNQ UltraScale+ MPSoC

PS = Processing System
PL = Programmable Logic (FPGA)

AVNET

Compelling reasons to use the PL vs PS:

1. For many operations like convolution (time-domain filtering) and the FFT the PL will be orders of magnitude faster mainly due to parallelism and the PL's built-in DSP slices (MAC) hardware!

2. Precision timing (picoseconds of jitter accuracy) for control of other hardware

3. Determinism of algorithmic execution (no cache or interrupts if you design it that way)

4. You can program the PL to do anything, even design your own CPU/GPU. For example see Xilinx's MicroBlaze™ for PYNQ: https://pynq.readthedocs.io/en/v2.0/pynq_libraries/pynq_microblaze_subsystem.html

5. The art of designing hardware with software can be rewarding and enjoyable!

Notes:
My 2nd presentation later today "A Call to Action: Accelerating Python with FPGAs" will go into more details.

AVNET

Compelling reasons to use the PL vs PS redux:

1. The PL hardware has 100's and 1000's of built-in primitives that excel at bit manipulations, modulo arithmetic, multiplication and logical operations as well as memory. Many physical I/O pins for interfacing with external hardware. All of which can be combined to accomplish just about anything.

2. The hardware can exploit true parallelism (analogous to SW threads and processes)

3. The hardware can maintain synchronous timing with small jitter in units of picoseconds

4. Because of the precise timing capabilities, reliable determinism can be designed in (in it's simplest sense you will know exactly when you will have your answer)

5. Some people enjoy the art of hardware design using software ☺

Challenges to overcome when accelerating with hardware:

1. Identifying appropriate functionality to move into the PL
2. Moving the data between Python on the PS and the PL
3. Understanding the sensitivity of algorithms to quantization error and required mathematical precision
4. You will also need to design the hardware itself to execute the algorithms

Using the PYNQ framework will help solve some of these obstacles more quickly.

Obstacle 1 – Identifying candidates to move into the PL

You must find the calculations that are slowing down Python and move the functionality into the PL.

A good example candidate from the LIGO Notebook: Numpy's Power Spectral Density function and FFT.

```
"mlab.psd(strain_H1, Fs = fs, NFFT = NFFT)"
```

Hints: Use your favorite Python package (cProfile, profile, hotshot, etc) and profile the code. Look at the source or use knowledge and intuition to identify code that will require a lot of mathematical computation. Look at code that is attempting to manipulate or compute in any Galois Field or doing binary bit manipulations or streaming or block operations on large amounts of data. These are typically excellent candidates! The PL can have some access to large amounts of moderately fast external memories. The PL has direct high speed access to smaller chunks of internal memories. Algorithms that require large amounts of high speed memory will be more work to move into hardware. An example of something that typically will not work well in the PL is searching and sorting large amounts of data. I don't mean impossible, rather the CPU (PS) is likely a better candidate for those tasks, it does depend on many factors.

Obstacle 2 - Understanding mathematical precision

You must understand the sensitivity of your algorithm to quantization error and mathematical precision requirements.

Hints: Built-in Python floating point numbers are double precision with an 11-bit exponent for dynamic range and 52-bits mantissa for quantization. Numpy offers 128-bit floating point precision. PL can implement floating point arithmetic but it will generally use more resources than fixed point. Typically PL signal processing math is done with signed or unsigned fixed point integers. In some cases hybrid solutions of floating point and fixed-point are applied to compromise accuracy, speed and resource usage. Depending on the algorithm, bit for bit compatible math may or may not have a detrimental effect. This aspect can cost additional development time.

Obstacle 3 - Moving the data between the PS and the PL

You will need to move data between the Python environment on the PS and the PL.

Hints: PYNQ can help you here! Study the PYNQ community examples. spooNN (ML CNN application) shows how to use the Python libraries for moving data with DMA. Alternatively you can use Python to C methodologies and utilize Xilinx's SDSoC tools. SDSoC allows one to write in C/C++ and through the use of proprietary compiler pragmas to convert the code into PL. SDSoC also handles moving the data between PS and PL for you. If this is your first time attempting to utilize the PL, using PYNQ is going to be easier.

Obstacle 4 – Designing the hardware to execute the algorithm

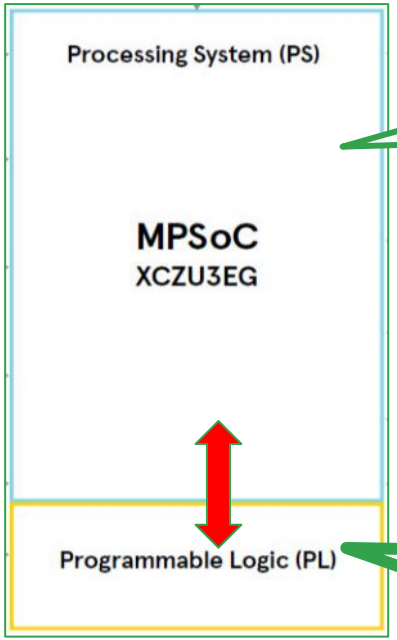
You will need to design the hardware that executes the algorithm

Hints: This is the largest obstacle to overcome. If you are completely new to this, you must peruse the existing PYNQ community examples and documentation. The troubles can become compounded by not having enough resources in the PL portion for more complicated algorithms. Execution speed is another practical factor that can complicate design completion. Using different larger parts that have more PL resources to allow for quicker development is not that uncommon in the industry. If you are stuck with a certain size part you may not be able to conveniently fit all of it into the PL, more effort and compromise will be required. You may have to design a hybrid system where the PS and PL each do part of the job. Due to many variables you will have to try and build it before you know whether it will work successfully; that should be considered in your schedule of work. If you are not versed in the typical hardware design languages (VHDL, Verilog), the highly recommended and superb alternatives from Xilinx for PYNQ are SDSoC and HLx. There is also an unrelated 3rd party project which allows hardware design to be accomplished with Python (see: http://www.myhdl.org). One downside to MyHDL at this time is that it is not directly integrated for you into PYNQ, you would have to figure that out yourself.

Verification is another part of hardware design. For people new to the process, for complex designs this can take a significant amount of time. Using PYNQ with HLx or SDSoC helps bypass a lot of the verification steps as well. The re-use of already verified libraries and IP are what help a lot here.

As far as low-cost tools for PL design, ones that work with Ultra96, Xilinx has kindly made them available. PYNQ is free open source. For SDSoC a free voucher is included with the purchase of an Ultra96 board. There are free Webpack tools available for download after registering with Xilinx. Sign-up for free through here: https://www.xilinx.com/products/design-tools/software-zone.html

Accelerating the search for Black Holes:

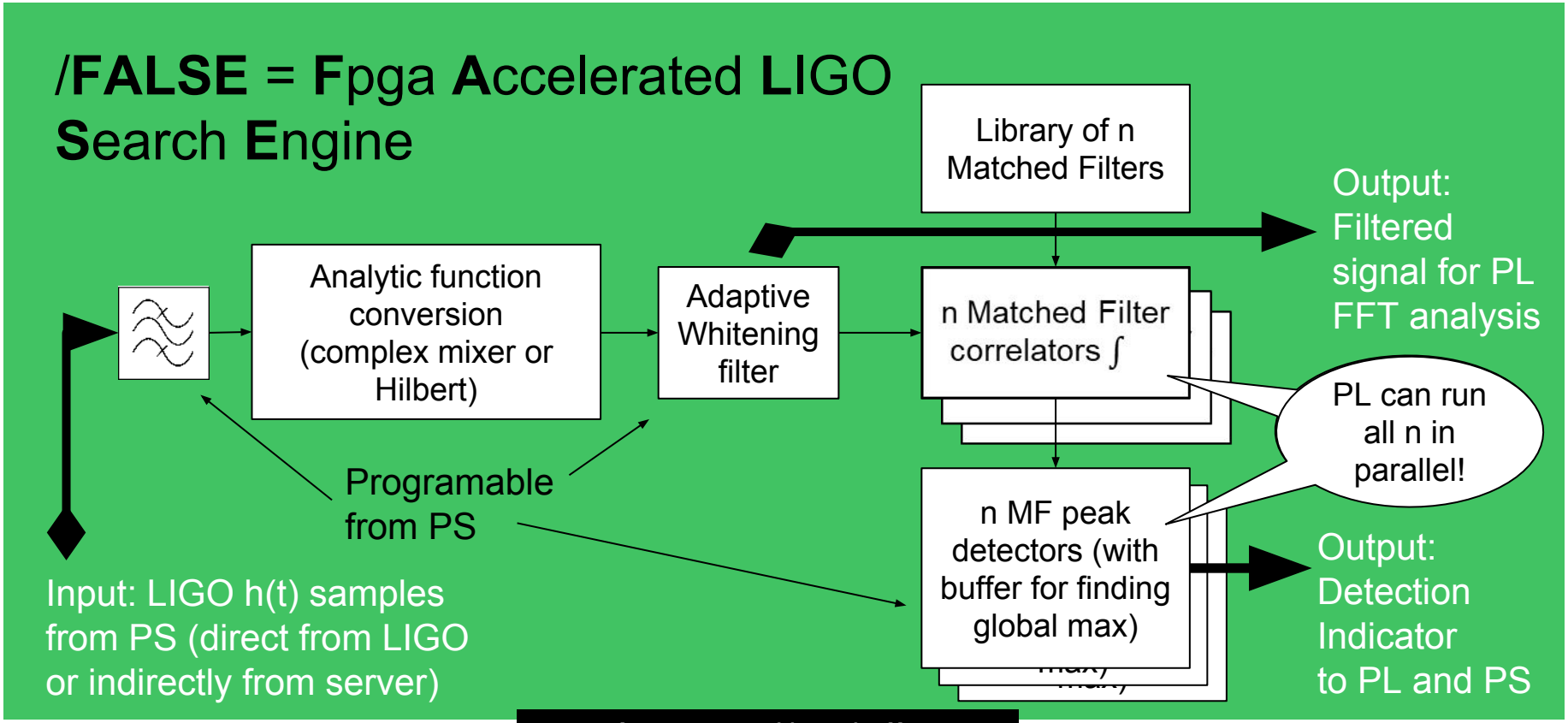


Without acceleration the CPUs (PS) run everything: Linux/Jupyter/Notebook /LIGO-search-PYTHON

Using PYNQ and various tools we can move the data and the algorithmic processing from the CPUs into the PL hardware = much faster execution = finding more black holes!

AVNET

PL LIGO processing concept design: TRUE

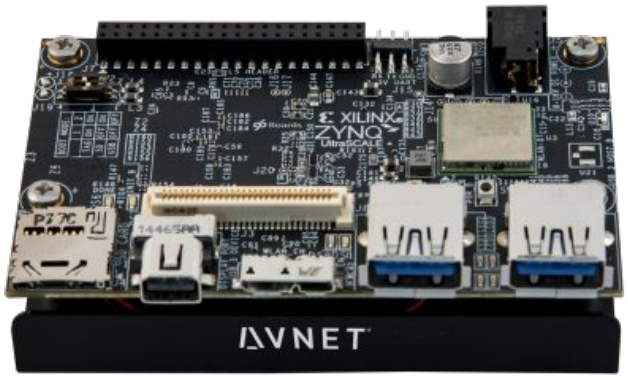


Conceptual content created by Fred Kellerman

AVNET

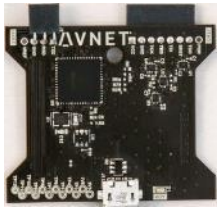
Acquire your own Ultra96 board for $249:

http://zedboard.org/product/ultra96 ◀ **Order here**



Includes:
- Ultra96 development board
- 16 GB pre-loaded MicroSD card + adapter
- Voucher for SDSoC license from Xilinx
- Quick-start instruction card

Does not include (but necessary):
- External 12V 2A 96boards adapter



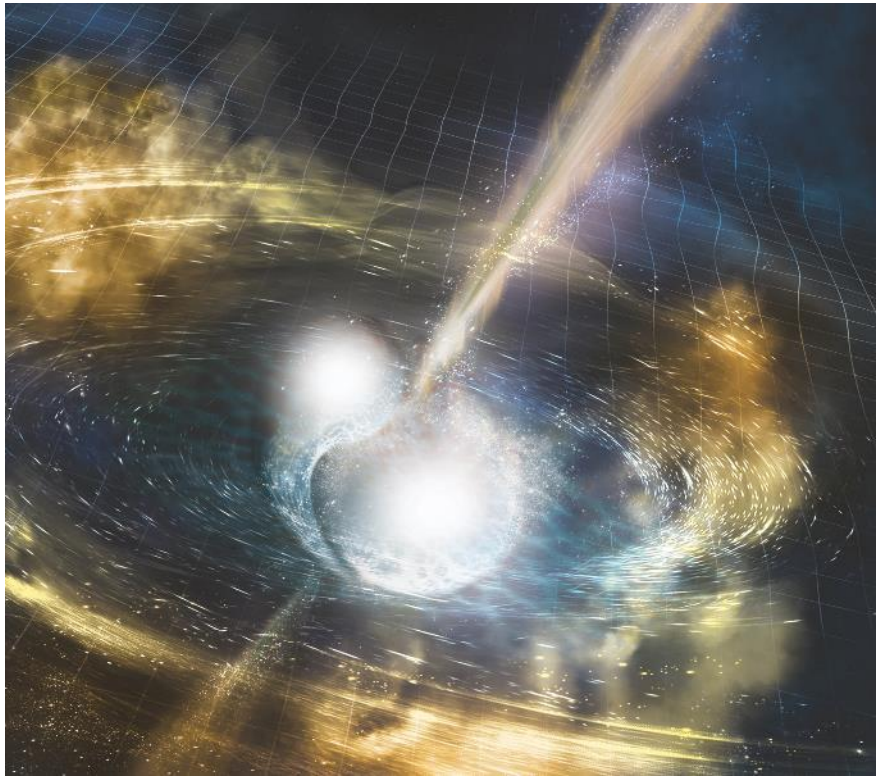
Optional Accessories:
- Seed Studios Grove Starter Kit for 96boards
- Other compatible accessories
- JTAG to USB adapter board

96Boards Partner

AVNET

Further proof:

Neutron star merger further authenticates the system!



courtesy of Caltech/MIT/LIGO Laboratory

GW170817
When neutron stars merge they emit electromagnetic energy also in the visual spectrum

For this GW event, both LIGO and Virgo measured gravity waves.

Moments later the light show occurred and 70 optical telescopes directed their gaze to the area of the sky that LIGO predicted and watched the fireworks!

AVNET

Acknowledgements:

THANK YOU for attending!

Much gratitude to my friend **Aron Khan** who weathered v1.0 of my presentation and offered very useful advice to improve it

A special thank you to Dr Ofek Birnholtz PhD GW Research Scientist at RIT for personally enlightening me about this fascinating topic!

And thank you to the following people who provided for the opportunity to share this with you:

Avnet: Kevin Keryk, Bryan Fletcher

AVNET

Materials credits:

Several slides (denoted on the slide itself) utilized imagery and materials from Caltech/MIT/LIGO Laboratory and are courtesy of Caltech/MIT/LIGO Laboratory.

The materials used from Caltech/MIT/LIGO Laboratory are in no manner intended to imply endorsement of any product or service. The usage here is for educational intent only. The Ultra96 board is simply executing the materials that were provided as-is.

All diagrams and information are provided as-is with no guarantee or liabilities. Any information utilized is at your own risk, none of it is intended for real-world usage, there may be many errors and flaws in this information.

Any additional diagrams, statements or imagery, unless noted were created and owned by the author.

AVNET

Get more help on PYNQ™ and Python for Ultra96:

PYNQ Workshop: https://github.com/Xilinx/PYNQ_Workshop

See other's examples (Ultra96 examples will be added soon):
http://www.pynq.io/community

See Avnet's Ultra96 tutorials (more on the way):
http://zedboard.org/support/design/24166/156

Join Xilinx's Developer Zone to access free tools:
https://www.xilinx.com/products/design-tools/software-zone.html

Did I mention my 2nd presentation at Linaro Connect later today:
"A Call to Action: Accelerating Python with FPGAs"

AVNET