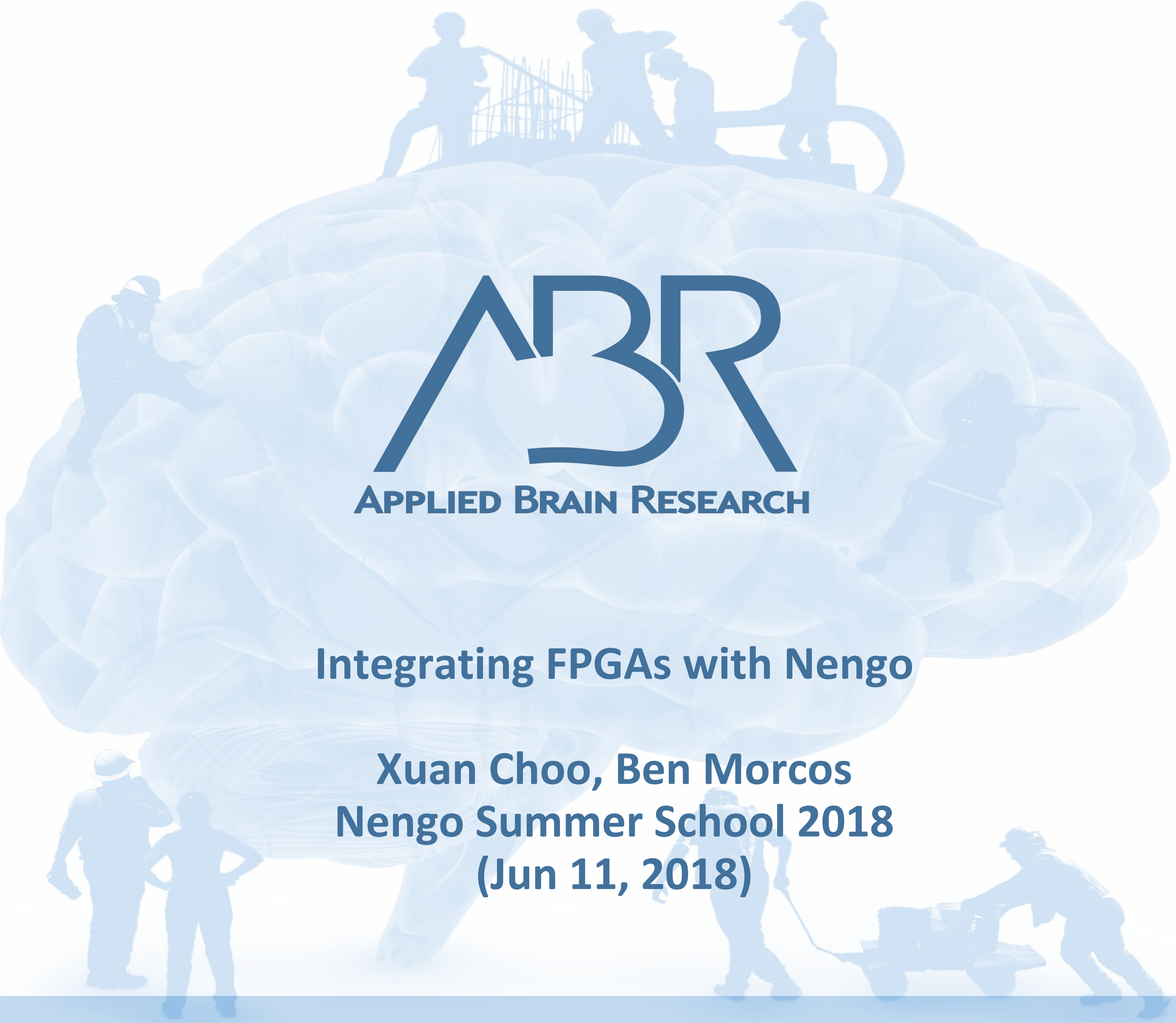


ABR

APPLIED BRAIN RESEARCH

Integrating FPGAs with Nengo

Xuan Choo, Ben Morcos

Nengo Summer School 2018

(Jun 11, 2018)

FPGA Basics

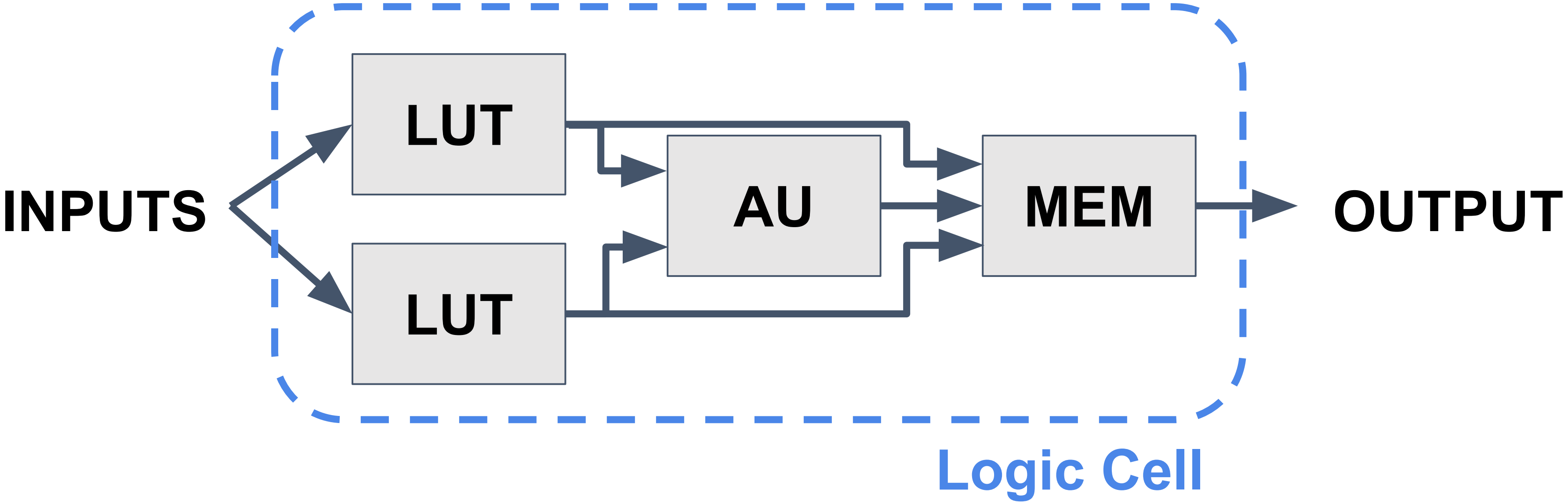
- **What is an FPGA?**

FPGA Basics

- **What is an FPGA?**
- **Field programmable gate array**

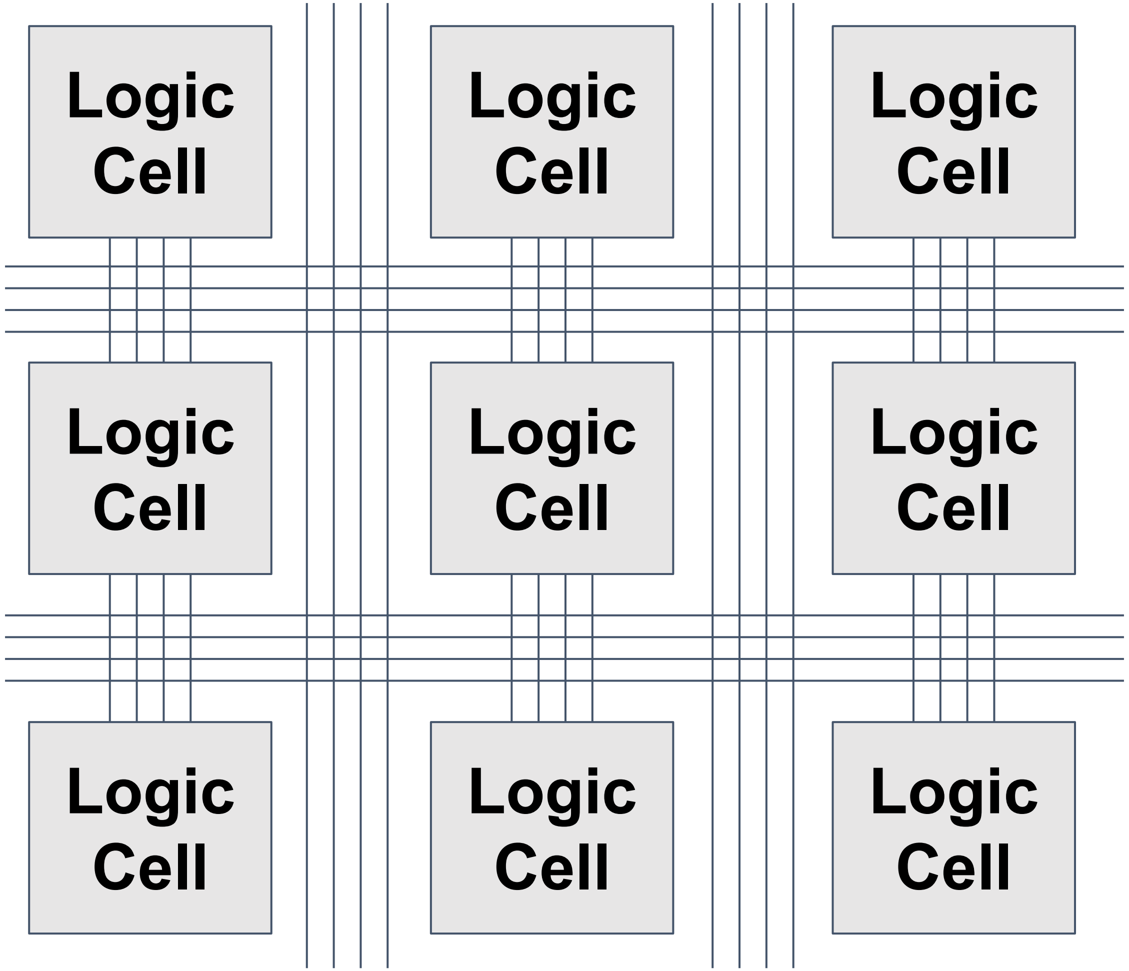
FPGA Basics

- **What is an FPGA?**
- **Field programmable gate array**



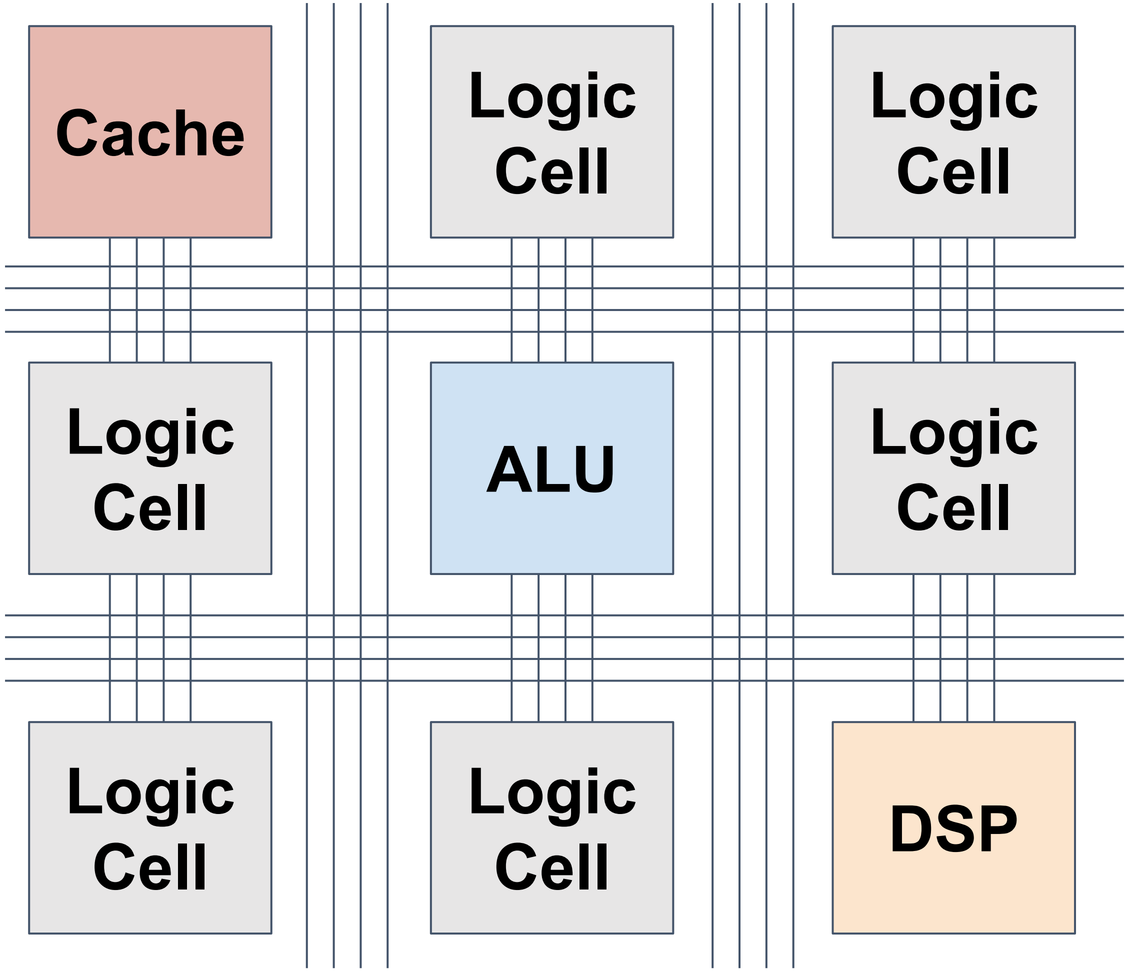
FPGA Basics

- **What is an FPGA?**
- **Field programmable gate array**



FPGA Basics

- **What is an FPGA?**
- **Field programmable gate array**



Big Picture

- **Why use FPGAs?**
  - Quick to prototype
  - Low power
  - Low latency
  - Potential for higher performance
  - Stepping stone between CPU/GPU and expensive/unavailable neuromorphics

Big Picture

- **We already have multiple backends for nengo**
  - CPU, GPU, Neuromorphics

- **Ideally fully featured seamless FPGA backend (Nengo FPGA)**
  - Large FPGAs (Intel Arria/Stratix, Xilinx Virtex/Kintex)
  - Still working on this, not ready for use yet

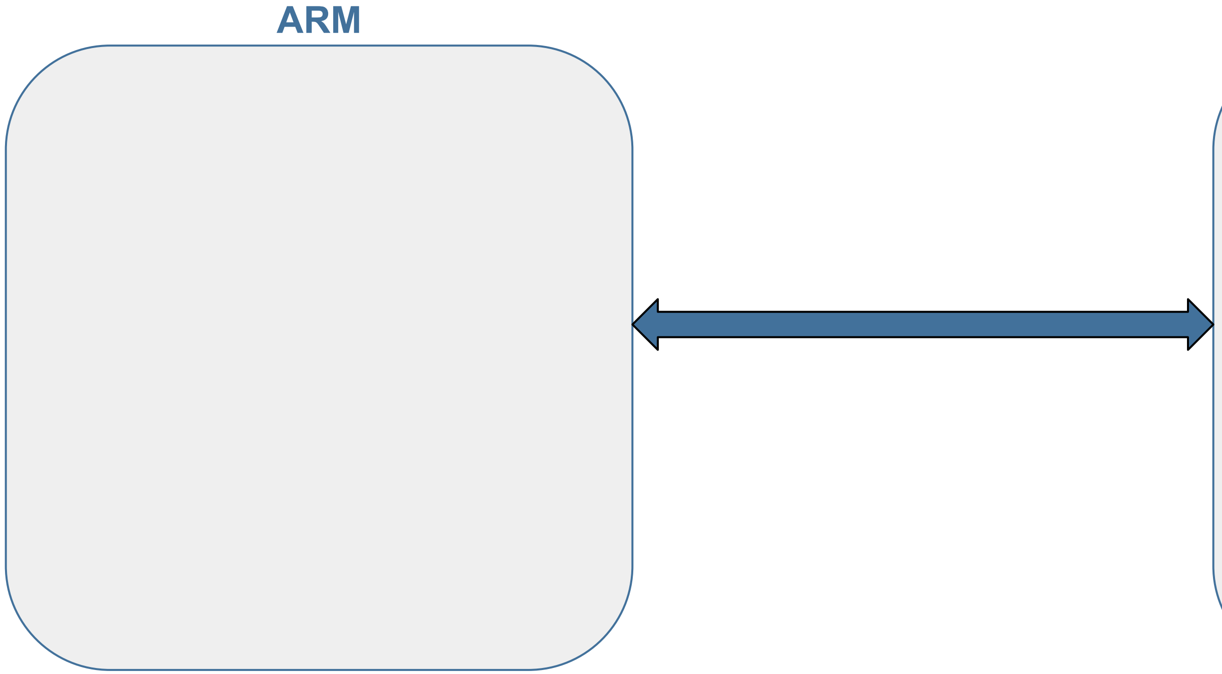
- **Right now we have smaller catalog implementations (Nengo Board)**
  - Xilinx PYNQ (Digilent)
  - Intel DE1-SoC (Terasic)

Nengo Board

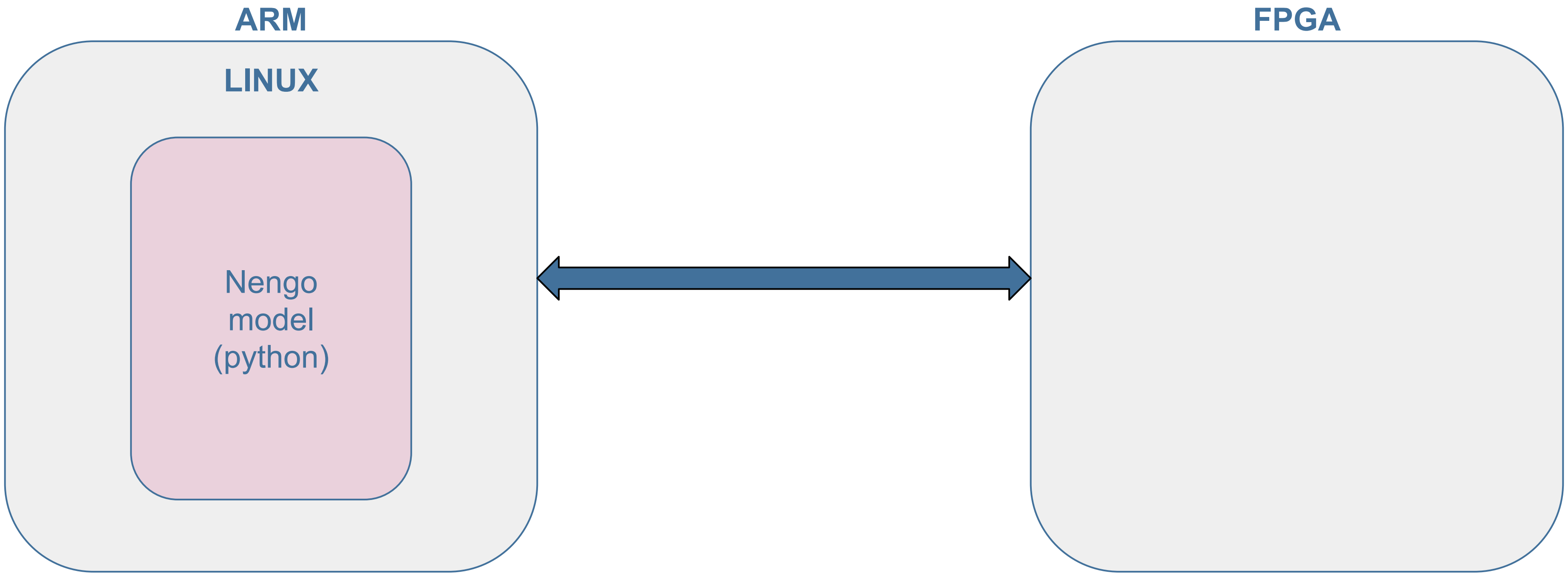
Board	PYNQ	DE1
Cost (USD)	$199	$249
PCB colour	Pink	Blue
Chip	Xilinx ZYNQ XC7Z020-1CLG400C SoC	Intel Cyclone V SoC 5CSEMA5F31C6
Processor	ARM Cortex A9 Dual Core	ARM Cortex A9 Dual Core
Logic Elements	85K	85K
BRAM (Mb)	4.9	4.45
DSP*	220	261

* DSPs are not equivalent

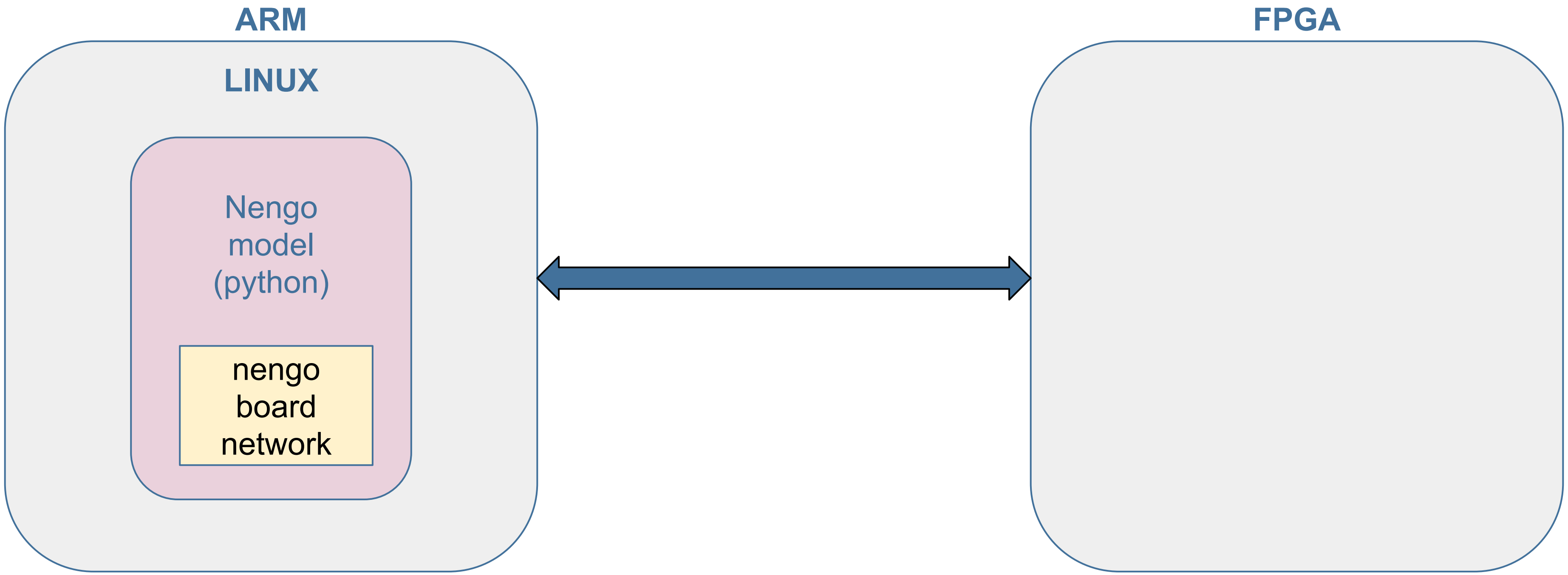
Nengo Board - Implementation



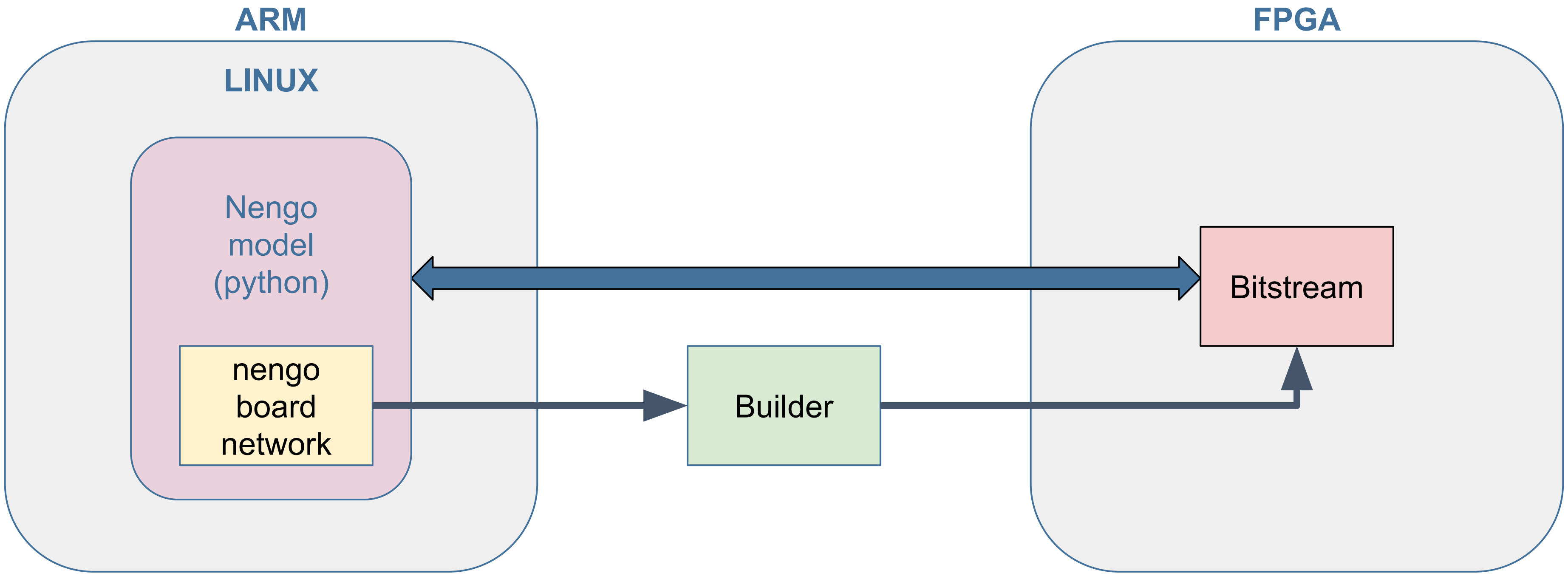
Nengo Board - Implementation



Nengo Board - Implementation



Nengo Board - Implementation

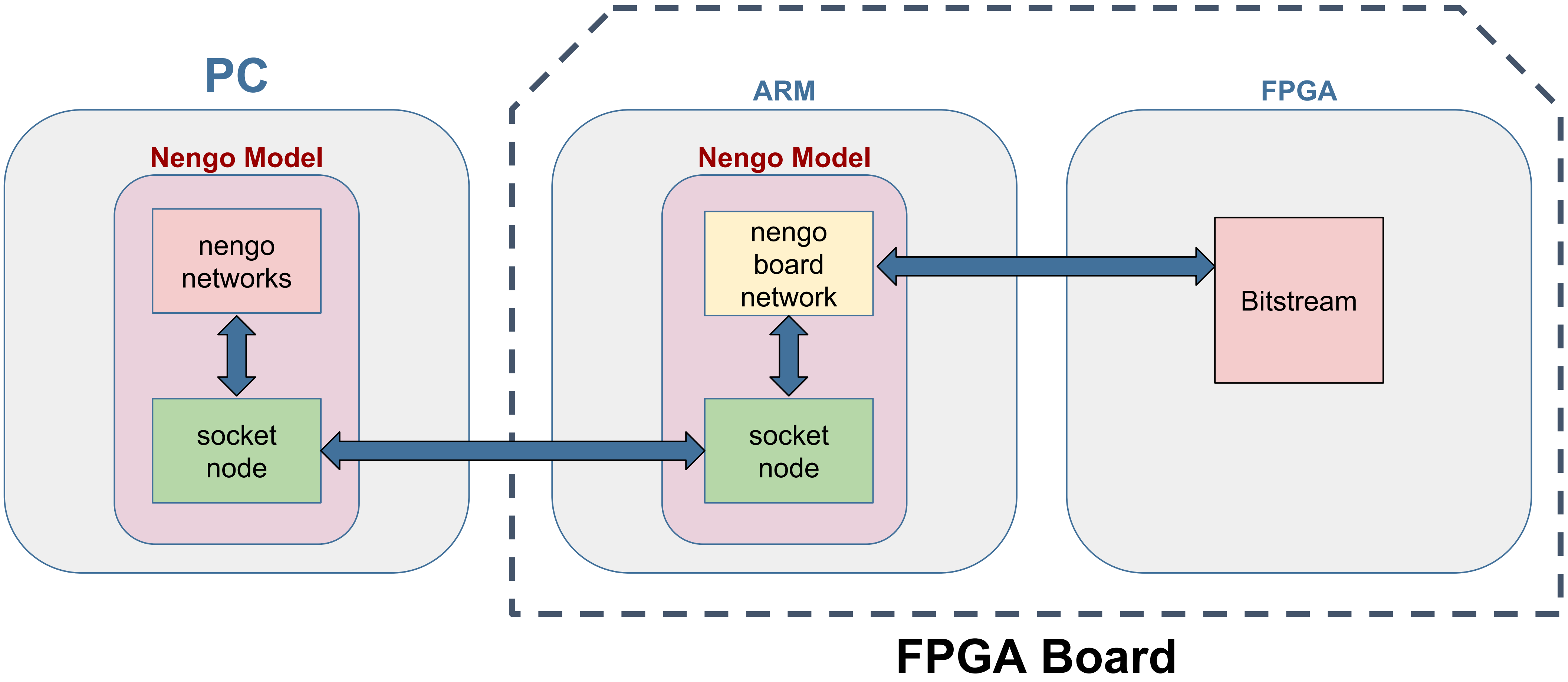


Nengo Board - Limitations

- **Topology catalog, not full backend**
  - Single ensemble with PES learning rule
  - Possible to expand catalog
- **Only Rectified Linear (ReLU) neurons implemented**
  - Rate neurons on PYNQ
  - Spiking and Rate on DE1
  - Possible to expand catalog
- **Network size limitations**
  - Limited by amount of memory on the boards
  - Solving for decoders on the ARM only advised for small ensembles

Parameter	PYNQ	DE1
n_neurons	16k	16k
max(input_dim, output_dim)	1k	1k
n_neurons * max(dim)	32k	16k

Nengo Board - Extended Implementation



Nengo Board - Usage

- **Documentation (WIP) can be found on the `nengo_board` branch of the summerschool git repo.**
- **Basic setup steps:**
  - Connect to board with UART (Serial connection over USB cable)
    - 115200; 8 data; 1 stop; no parity; no flow control
  - Check network settings on the board (see documentation)
    - Change the board IP if necessary
  - Set your PC network settings to match the subnet of the board
    - Easier with router, no need to bridge network interfaces to get internet access to the board.
  - Test connection to board by using SSH
    - Port 22
    - DE1: root, no password
    - Pynq: xilinx, xilinx

Nengo Board - Usage

- **Setting up the Nengo -- Nengo_board interface**
  - Checkout the `nengo_board` branch from the summerschool repo
  - Install `nengo_board` with `python setup.py develop`
  - Check `board_config` settings (in the `nengo_board` folder)

- **Running example scripts**
  - Navigate to `nengo_board/examples/automated`
  - Run an example with `nengo -b nengo_board <example_script>`

Nengo Board - Usage

- **Example communication channel script**

```
import numpy as np
import nengo
from nengo_board.networks import RemotePESEnsembleNetwork

nengo.utils.logging.log('info')

def input_func(t):
    return [np.sin(t * 10), np.cos(t * 10)]

with nengo.Network() as model:
    # Reference signal
    input_node = nengo.Node(input_func, label='input signal')

    # FPGA neural ensemble
    pes_ens = RemotePESEnsembleNetwork(
        ‘de1’, input_dimensions=2, input_synapse=None,
        learn_rate=0, n_neurons=50, label='ensemble')
    nengo.Connection(input_node, pes_ens.input)
```

Nengo Board - Usage

- **Example learned communication channel script**

```
import numpy as np
import nengo
from nengo_board.networks import RemotePESEnsembleNetwork
...
with nengo.Network() as model:
    # Reference signal
    input_node = nengo.Node(input_func, label='input signal')

    # Adaptive neural ensemble (run on the FPGA)
    pes_ens = RemotePESEnsembleNetwork(
        'cyclonev', input_dimensions=1, input_synapse=None,
        learn_rate=5e-5, n_neurons=100, ens_args={'radius': 1},
        output_conn_args={'function': lambda x: [0]}, label='pes ensemble')
    nengo.Connection(input_node, pes_ens.input)

    # Error signal computation
    error = nengo.Ensemble(50, 1)

    # Compute the error (error = actual - target = post - pre)
    nengo.Connection(pes_ens.input, error, transform=-1)
    nengo.Connection(pes_ens.output, error)

    # Project the error to the adaptive neural ensemble
    nengo.Connection(error, pes_ens.error)
```