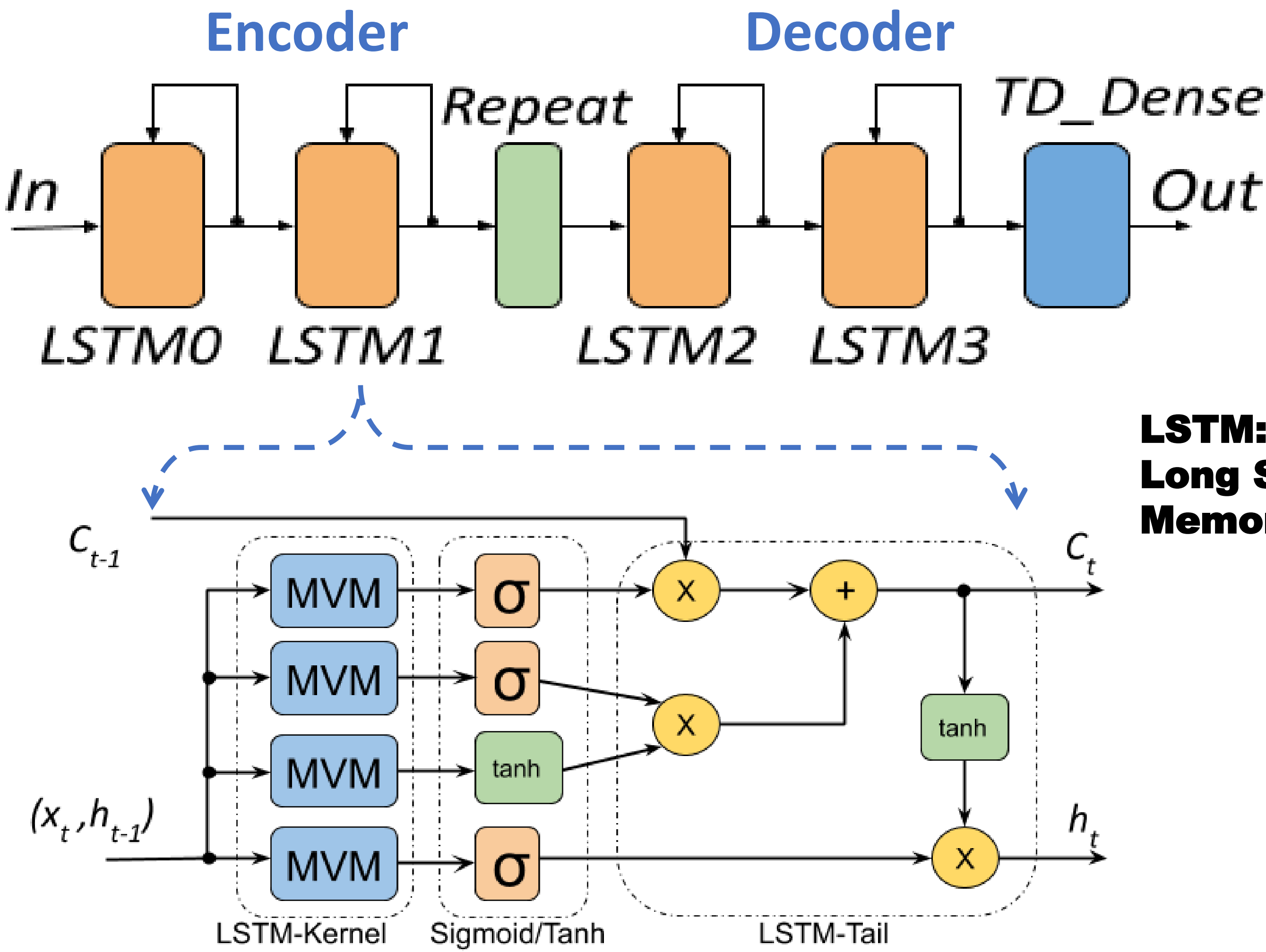
Accelerating Recurrent Neural Networks for Gravitational Wave Experiments

Zhiqiang (Walkie) Que, Wayne Luk
z.que@imperial.ac.uk
Imperial College London

Acknowledgement: Erwei Wang, Umar Marika, Eric Moreno, Jennifer Ngadiuba, Hamza Javed, Bartłomiej Borzyszkowski, Thea Aarrestad, Vladimir Loncar, Sioni Summers, Maurizio Pierini, Peter Y Cheung

LSTM-based autoencoder



LSTM:
Long Short Term Memory

Matrix-Vector Multiplication

Activation Functions

Element-wise Addition / Multiplication

Motivations: Challenges

- Develop a fast and efficient deep RNN inference
  - on FPGAs

- C1: Unbalanced Initiation Interval (II) : control re-use
  - multi-layer neural network inference system on FPGAs
  - long latency and low hardware efficiency

- C2: Need HLS-based LSTM template
  - scalable and low latency

Achievements

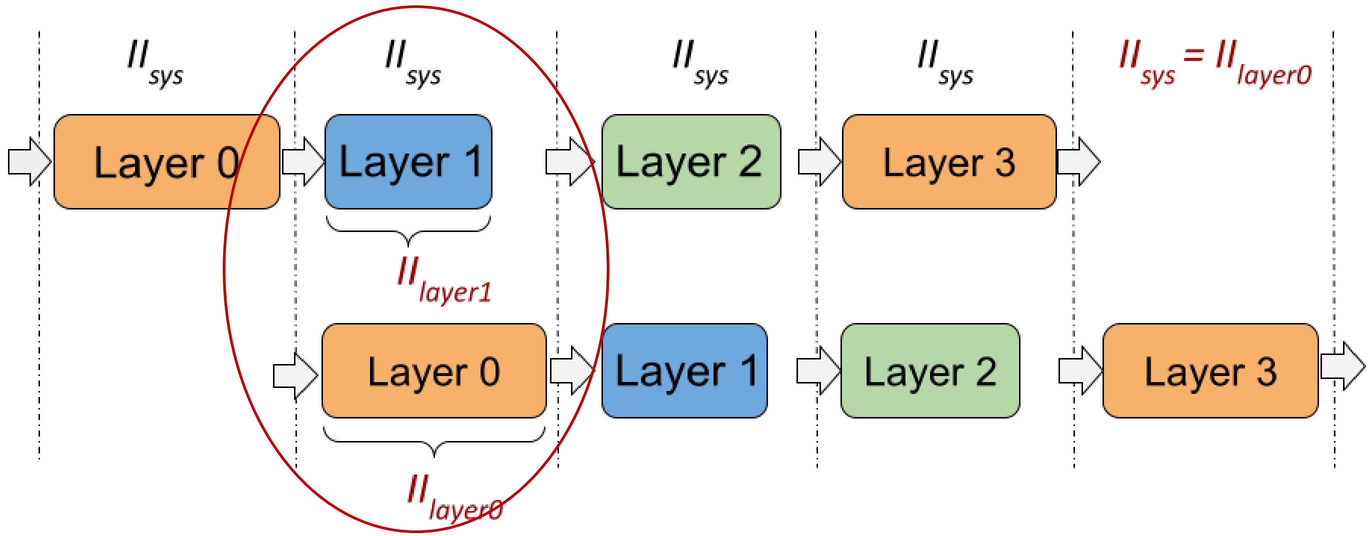
- A1: novel way to balance II of multi-layer LSTM inference
  - identify reuse factors for each LSTM layer: improve performance
  - balance the hardware resources: improve hardware efficiency
  - address challenge C1

- A2: A scalable and low latency HLS-based LSTM template
  - enable the generation of low-latency FPGA designs
  - open source: https://github.com/walkieq/RNN_HLS
  - address challenge C2

A1: balance Initiation Interval (II)

- For a model with multiple layers using HLS Dataflow, the system initiation interval (II):

$$II_{sys} = MAX\ (II_{layer0}\ ,\ II_{layer1}\ ,\ \ldots\ ,\ II_{layerN-1}, II_{layerN})$$

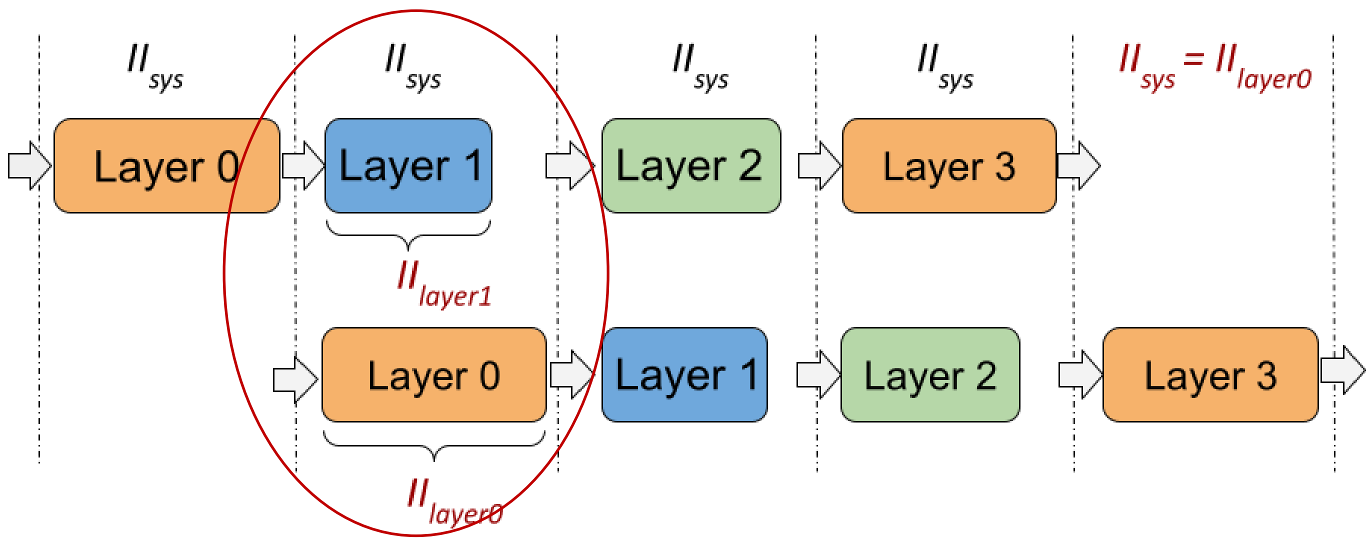


- Initiation interval (II): number of cycles until a unit can accept new inputs
- Control resource re-use: low II, less re-use, more resources/parallelism, faster
- Max efficiency: balance IIs

A1: balance Initiation Interval (II)

- For a model with multiple layers using HLS Dataflow, the system initiation interval (II):

$$II_{sys} = \mathrm{MAX}\,(II_{layer0}\,, II_{layer1}\,, \ldots, II_{layerN-1}, II_{layerN})$$

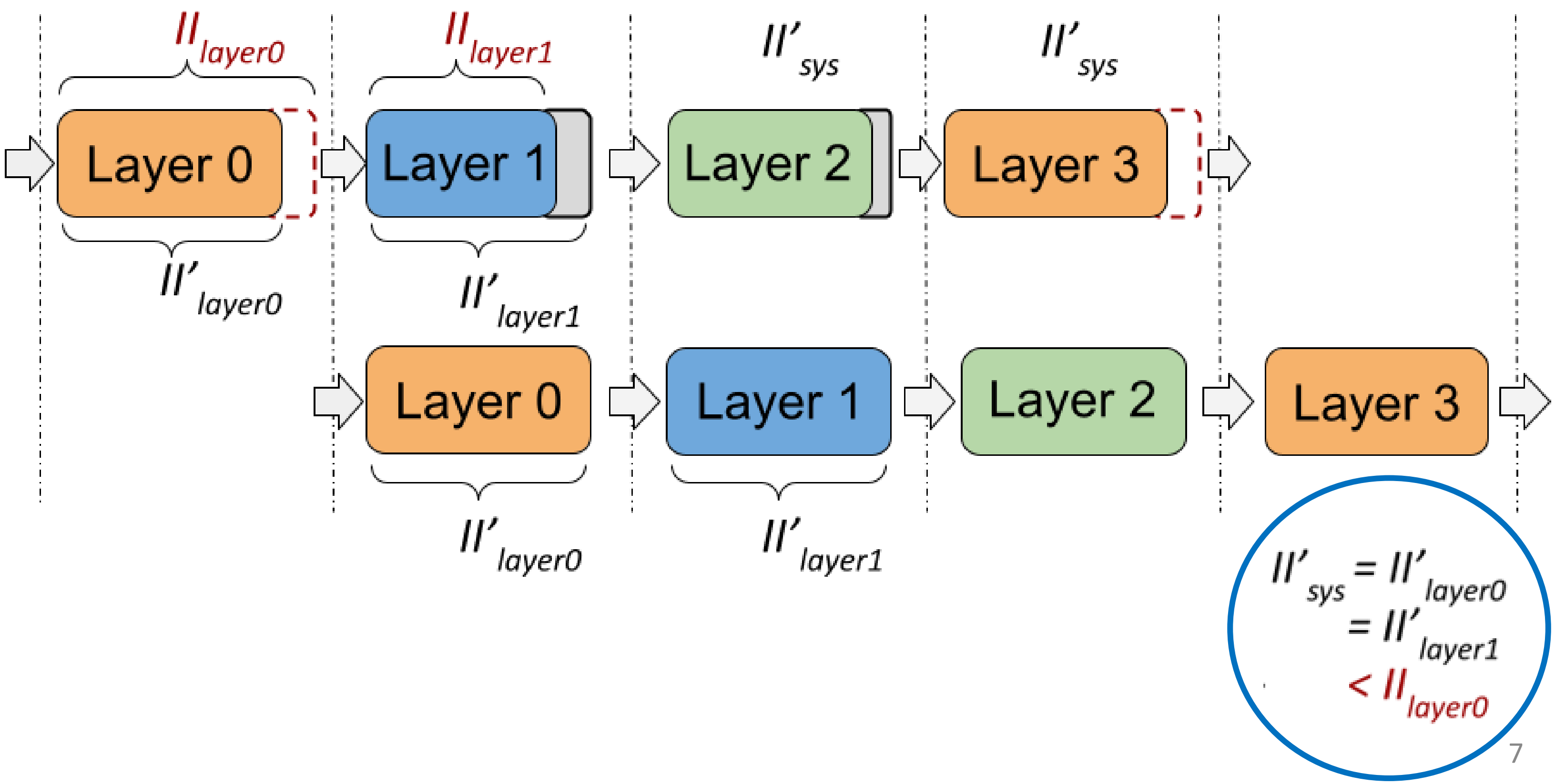


- The optimal case is all the layers have the same initiation internal :

$$II_{sys} = II_{layer1} = II_{layer2} = \ldots = II_{layerN-1} = II_{layerN}$$

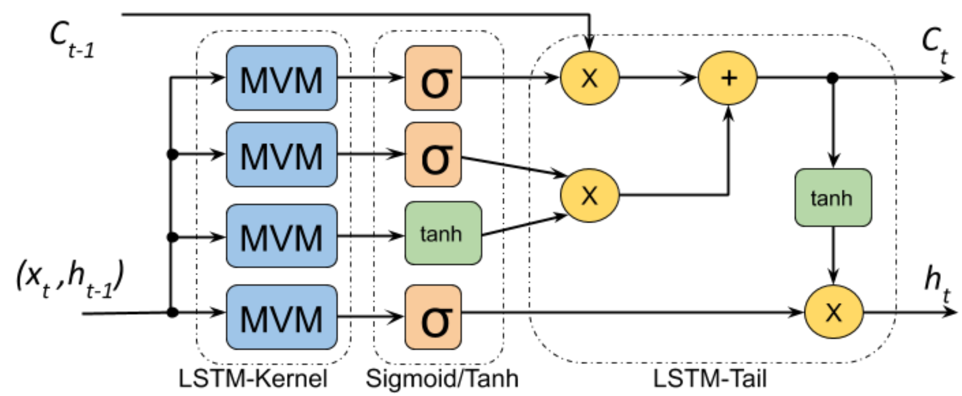
A1: balance Initiation Interval (II)

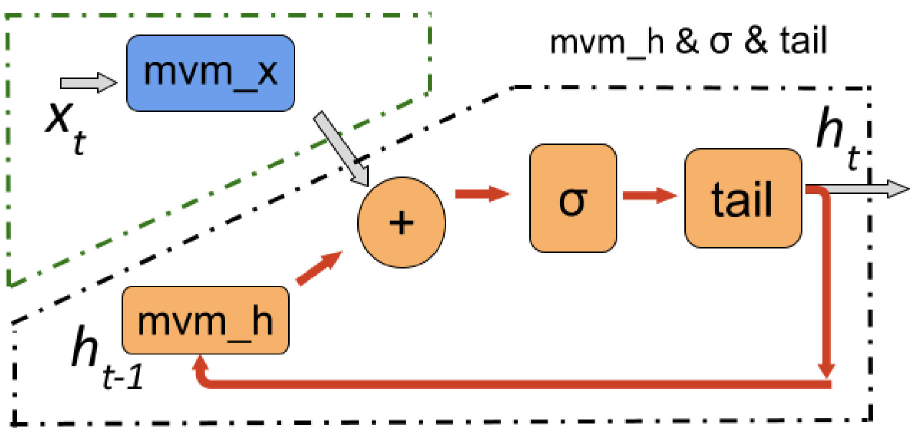
- Balance the IIs among layers

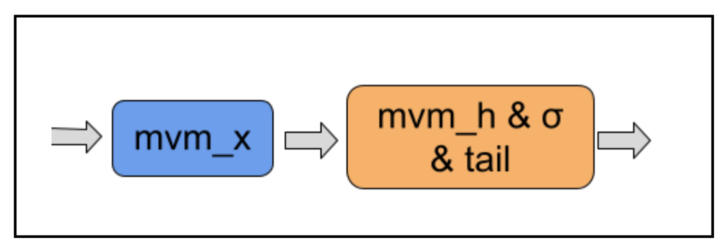


A2: The II of a single LSTM Layer

For(i=0; i<Timesteps; i++)
{
    Acc = mvm_x(Wx, b, x_t);
    Acc = mvm_h(Wh, Acc, h_{t-1});
    Acc = sigmoid_tanh(Acc);
    h_t = lstm_tail(Acc);
}

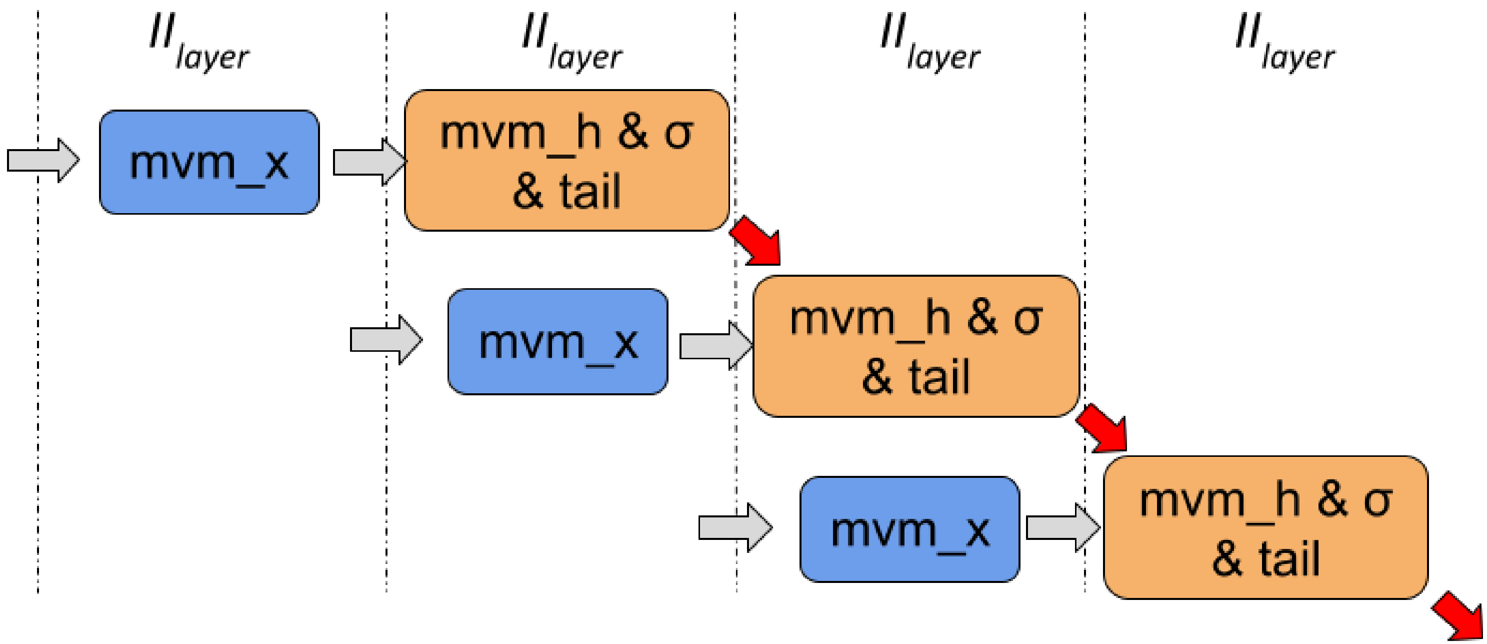






A2: The II of a single LSTM Layer

Depth_{mvm_x} Depth_{h_tail}



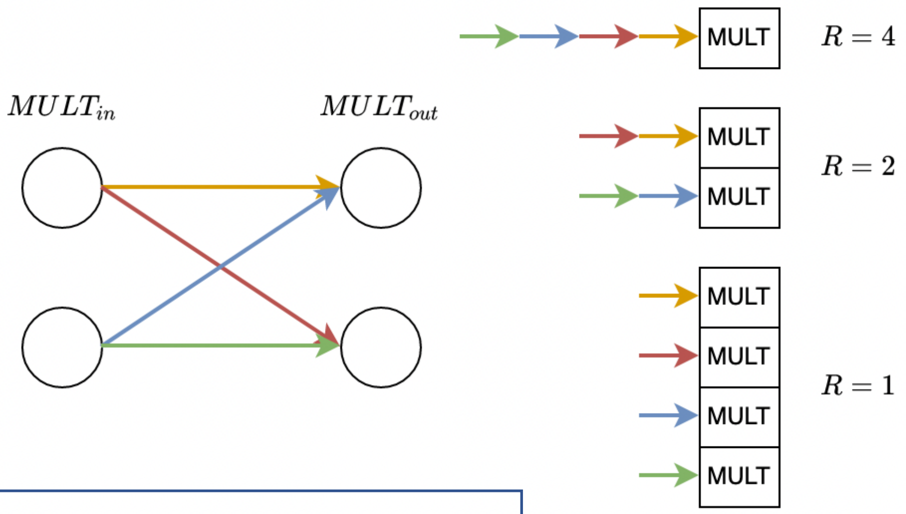
$II_{ts} = \text{MAX}\,(\text{Depth}_{mvm\_x}, \text{Depth}_{h\_tail})$

$II_{layer} = II_{ts} * \text{Timesteps}$

$II_{ts} = \text{Depth}_{mvm\_x} = \text{Depth}_{h\_tail}$

A2: The II of a single LSTM Layer

- R_x : Reuse factor [j18] for MVM involving x
- R_h: Reuse factor for MVM involving h



$$LT_{mvm\_x} = LT_{mvm\_h} + LT_{\sigma} + LT_{tail}$$

$$\Rightarrow R_x = R_h + LT_{\sigma} + LT_{tail}$$

higher R:
- more re-use
- less resources
- less parallelism

[j18]: J. Duarte, S. Han, P. Harris, S. Jindariani, E. Kreinar, B. Kreis, J. Ngadiuba, M. Pierini, R. Rivera, N. Tran et al., "Fast inference of deep neural networks in FPGAs for particle physics," Journal of Instrumentation, vol. 13, no. 07, p.P07027, 2018.

DSP usage

mvm x

mvm h

LSTM tail

$$DSP\ per\ Layer = \frac{4L_xL_h}{R_x} + \frac{4L_h^2}{R_h} + 4L_h$$

$$DSP\ per\ Model = \frac{\Sigma 4L_xL_h}{R_x} + \frac{\Sigma 4L_h^2}{R_h} + \Sigma\ 4L_h$$

$$DSP\ per\ Model \leq DSPs\_on\_FPGA$$

Example 1

- AutoEncoder for LIGO (4 LSTM layers)
- XCKU115 (5520 DSP)

```
def autoencoder_LSTM(X):
    inputs = Input(shape=(X.shape[1], X.shape[2]))
    L1 = LSTM(32, activation='relu', return_sequences=True,
              kernel_regularizer=regularizers.l2(0.00))(inputs)
    L2 = LSTM(8, activation='relu', return_sequences=False)(L1)
    L3 = RepeatVector(X.shape[1])(L2)
    L4 = LSTM(8, activation='relu', return_sequences=True)(L3)
    L5 = LSTM(32, activation='relu', return_sequences=True)(L4)
    output = TimeDistributed(Dense(X.shape[2]))(L5)
    model = Model(inputs=inputs, outputs=output)
    return model
```

1

2

$$\Rightarrow \frac{\Sigma 4 L_x L_h}{R_h + 8} + \frac{\Sigma 4 L_h^2}{R_h} + \Sigma\, 4 L_h \leq 5520$$

$$\Rightarrow R_h^2 + 5.858 R_h - 13.391 \geq 0$$

$$\Rightarrow R_h \geq 1.76$$

$\Rightarrow$ Best parameter: $\begin{cases} R_h = 2 \\ R_x = 10 \end{cases}$

Example 2

- AutoEncoder for LIGO (4 LSTM layers)
- XCKU115 (5520 DSP)

```
def autoencoder_LSTM(X):
    inputs = Input(shape=(X.shape[1], X.shape[2]))
    L1 = LSTM(32, activation='relu', return_sequences=True,
              kernel_regularizer=regularizers.l2(0.00))(inputs)
    L2 = LSTM(8, activation='relu', return_sequences=False)(L1)
    L3 = RepeatVector(X.shape[1])(L2)
    L4 = LSTM(8, activation='relu', return_sequences=True)(L3)
    L5 = LSTM(32, activation='relu', return_sequences=True)(L4)
    output = TimeDistributed(Dense(X.shape[2]))(L5)
    model = Model(inputs=inputs, outputs=output)
    return model
```

1
2

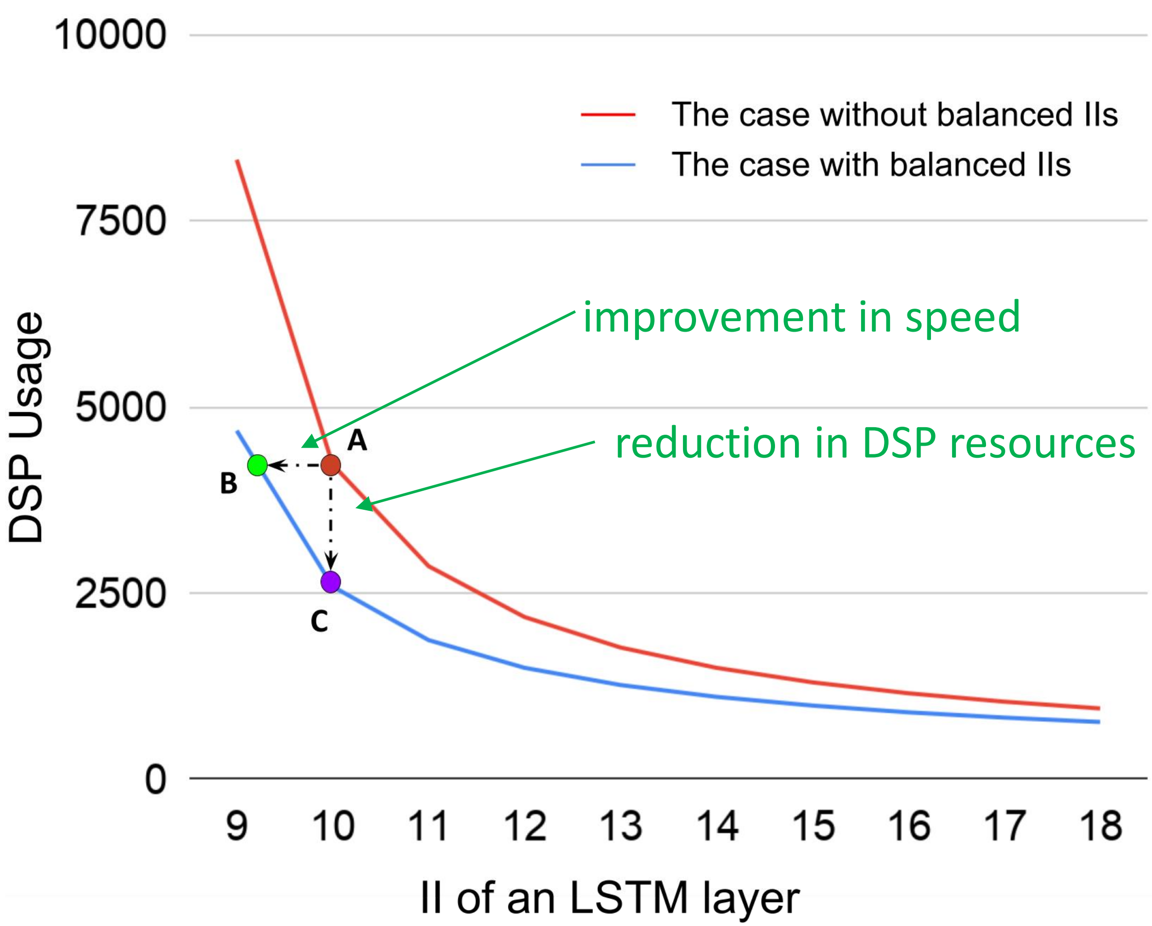
$$\Rightarrow \frac{\Sigma 4 L_x L_h}{R_h + 8} + \frac{\Sigma 4 L_h^2}{R_h} + \Sigma\, 4 L_h \leq 12288$$

$$\Rightarrow R_h^2 + 7.070 R_h - 5.847 \geq 0$$

$$\Rightarrow R_h \geq 0.75$$

$\Rightarrow$ Best parameter: $\begin{cases} R_h = 1 \\ R_x = 9 \end{cases}$

Pareto frontier



lower R, II:
- less re-use
- more resources
- more parallelism
- faster

A3. Comparison of the FPGA designs

Timestep loop initiation interval

	Z1	Z2	Z3	U1	U2	U3
FPGA	Zynq 7045			U250		
DSP total	900			12,288		
R_h	1	2	1	1	1	4
R_x	1	2	9	1	9	12
LUT used	45k (21%)	45k (21%)	43k (20%)	449k (26%)	463k (27%)	516k (30%)
DSP used	1,058 (118%)	578 (64%)	744 (83%)	11,123 (91%)	9,021 (73%)	2,713 (22%)
ii_{layer} cycles	9	10	9	12	12	13
II_{layer} cycles	72	80	72	96	96	104

Z1: max parallelism, but does not fit

Z3: lowest, same as Z1

A3. Comparison of the FPGA designs

	Z1	Z2	Z3	U1	U2	U3
FPGA	Zynq 7045			U250		
DSP total	900			12,288		
R_h	1	2	1	1	1	4
R_x	1	2	9	1	9	12
LUT used	45k (21%)	45k (21%)	43k (20%)	449k (26%)	463k (27%)	516k (30%)
DSP used	1,058 (118%)	578 (64%)	744 (83%)	11,123 (91%)	9,021 (73%)	2,713 (22%)
ii_{layer} cycles	9	10	9	12	12	13
II_{layer} cycles	72	80	72	96	96	104

U3: increase II by 8.3%, reduce DSP by 75%

U1: max parallelism, still fit

Summary

- A1: novel way to balance II of multi-layer LSTM inference
  - identify reuse factors for each LSTM layer: improve performance
  - balance the hardware resources: improve hardware efficiency

- A2: A scalable and low latency HLS-based LSTM template
  - enable the generation of low-latency FPGA designs
  - open source: https://github.com/walkieq/RNN_HLS