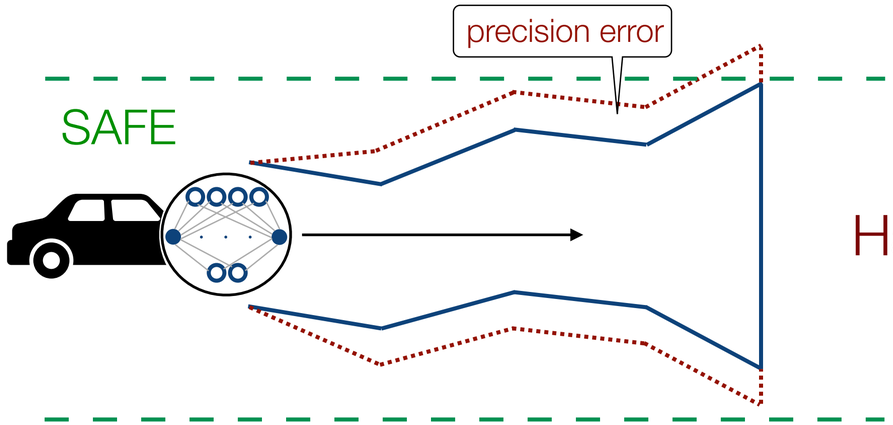
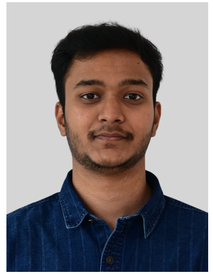
Towards Precision-Aware Safe Neural-Controlled Cyber-Physical Systems

EMSOFT 2024



Is the controller still safe with finite precision errors?



Harikishan Thevendhriya, Sumana Ghosh, **Debasmita Lohar**

STATISTICAL INDIAN INSTITUTE UNITY IN DIVERSITY

Karlsruhe Institute of Technology

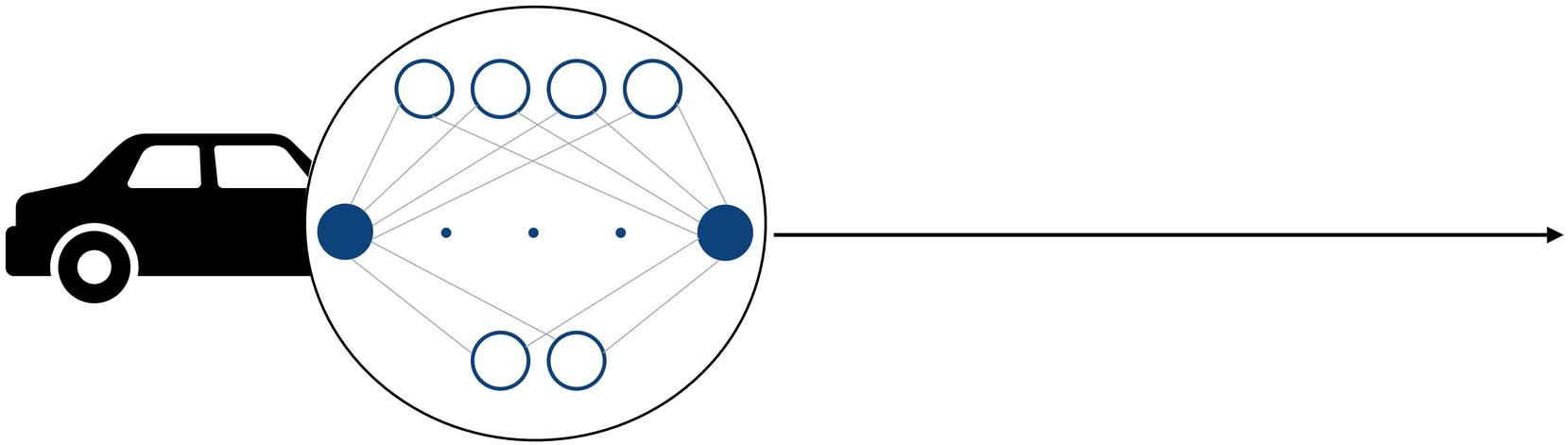
Is the system safe until 4 s?

initial conditions

$l_1 <= in_1 <= u_1$

...

$l_m <= in_m <= u_m$



Is the system safe until 4 s?

- Polar-Express
- NNV
- ReachNN
- …

real-valued arithmetic

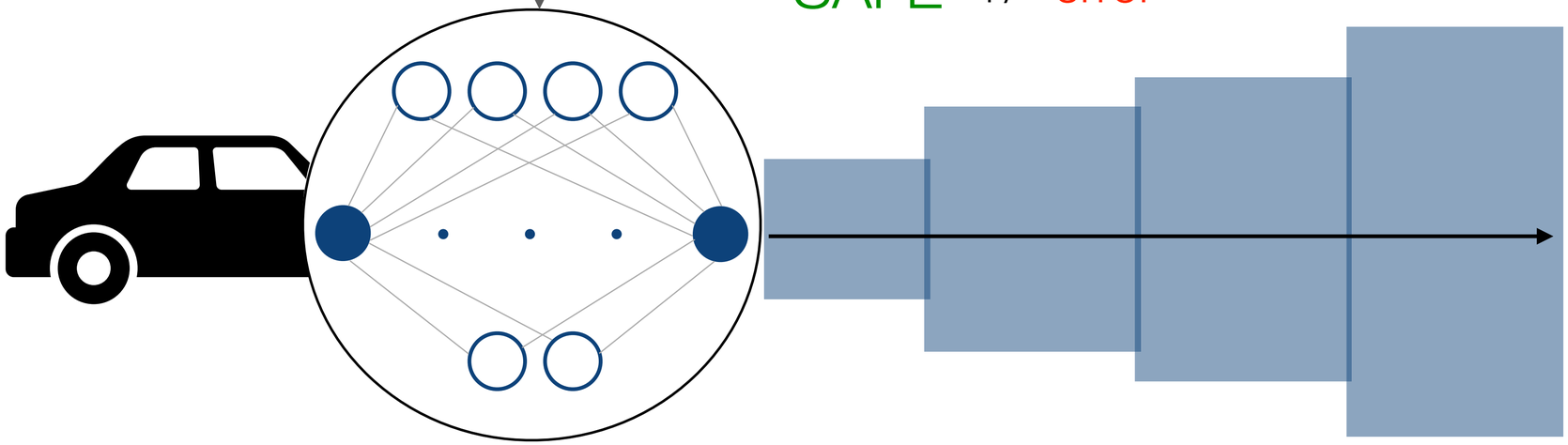
SAFE +/- error

initial conditions

$l_1 <= in_1 <= u_1$

...

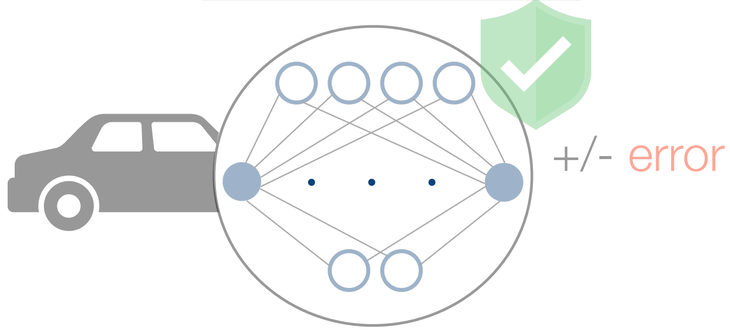
$l_m <= in_m <= u_m$



Polar-Express: Efficient and precise formal reachability analysis of neural-network controlled systems, Y. Wang et al., TCAD 2023
NNV: The Neural Network Verification Tool for Deep Neural Networks and Learning-Enabled Cyber-Physical Systems, H.Tran et al., CAV 2020
ReachNN: Reachability Analysis of Neural-Network Controlled Systems, C. Huang et al., TECS 2019

Is the finite-precision implementation still safe?

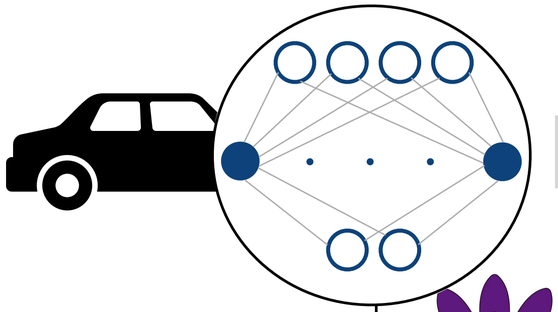
$l_1 <= in_1 <= u_1$
...
$l_m <= in_m <= u_m$



+/- error

Safety analysis
without finite-precision errors

$l_1 <= in_1 <= u_1$
...
$l_m <= in_m <= u_m$



predefined error

```
res +/- 1e-5
```

quantization

mixed-precision fixed-point code

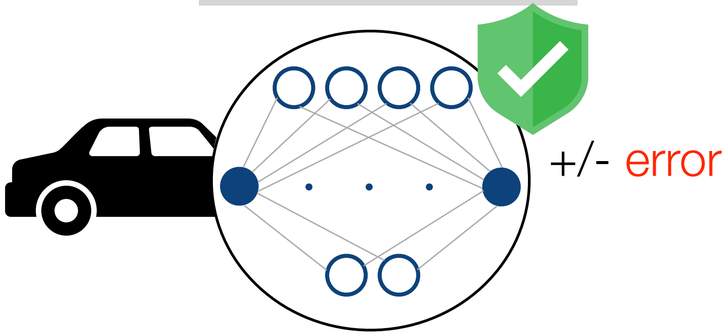
```
#include <math.h>
#include <ap_fixed.h>
#include <hls_math.h>
#include <ap_fixed.h>

void nn1(ap_fixed<24,5> x_0, ap_fixed<24,4> x_1, ap_fixed<24,3>
x_2, ap_fixed<24,2> x_3, ap_fixed<27,8> _result[2]) {
  ap_fixed<21,2> weights1[20][4] = {{-0.036691424, ...}...}
  ap_fixed<17,1> bias1[20] = {0.027661305, ...}
  ...
  for (int i=0; i<2; i++) {
    ap_fixed<32,18> _dot_tmp = weights2[i][0] * layer1[0] + ...
    ...
  _result[1] = layer2[1];
}
```

XILINX

Aster: Sound Mixed Fixed-Point Quantization of Neural Networks, D. Lohar et al., EMSOFT-TECS 2023

This Paper: An End-To-End Solution

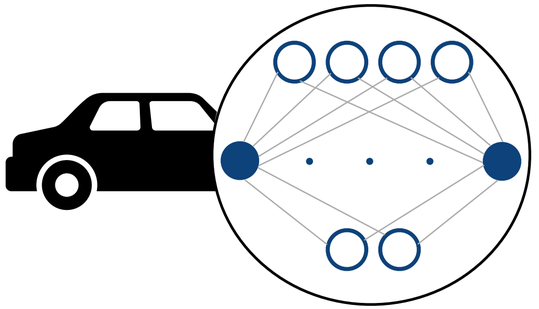
$l_1 <= in_1 <= u_1$
...
$l_m <= in_m <= u_m$



+/- error

Safety analysis
without finite-precision errors

$l_1 <= in_1 <= u_1$
...
$l_m <= in_m <= u_m$



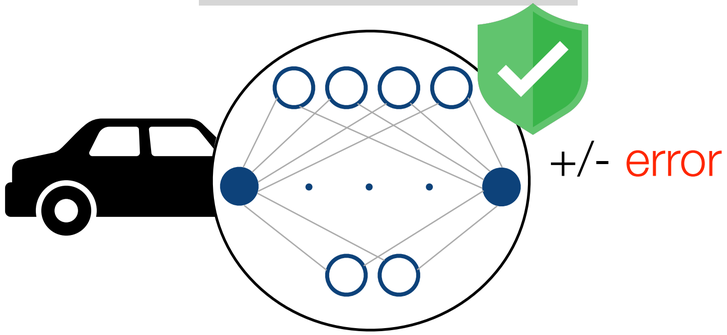
res +/- 1e-5

```
#include <math.h>
#include <ap_fixed.h>
#include <hls_math.h>
#include <ap_fixed.h>

void nn1(ap_fixed<24,5> x_0, ap_fixed<24,4> x_1, ap_fixed<24,3> x_2,
ap_fixed<24,2> x_3, ap_fixed<27,8> _result[2]) {
  ap_fixed<21,2> weights1[20][4] = {{-0.036691424, ...}...}
  ap_fixed<17,1> bias1[20] = {0.027661305, ...}
  ...
  for (int i=0; i<2; i++) {
    ap_fixed<32,18> _dot_tmp = weights2[i][0] * layer1[0] + ...
    ...
  _result[1] = layer2[1];
}
```

Quantization
without safety analysis

This Paper: An End-To-End Solution

$l_1 <= in_1 <= u_1$
$...$
$l_m <= in_m <= u_m$

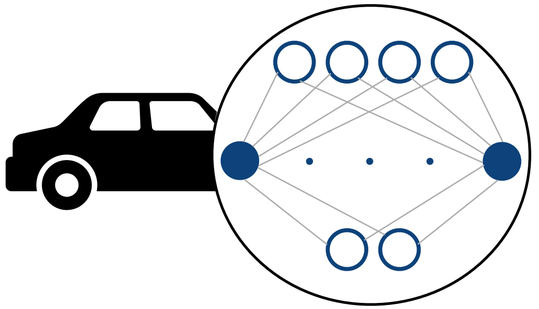


+/- error

Safety analysis
without finite-precision errors

+

$l_1 <= in_1 <= u_1$
$...$
$l_m <= in_m <= u_m$



res +/- 1e-5

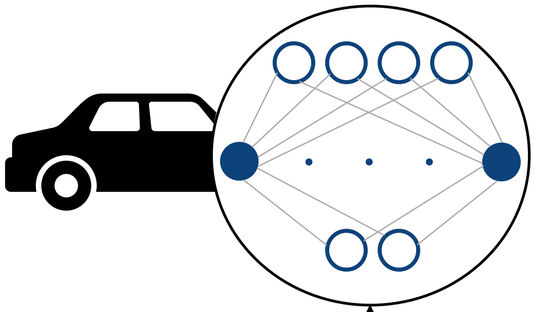
```
#include <math.h>
#include <ap_fixed.h>
#include <hls_math.h>
#include <ap_fixed.h>

void nn1(ap_fixed<24,5> x_0, ap_fixed<24,4> x_1, ap_fixed<24,3> x_2,
ap_fixed<24,2> x_3, ap_fixed<27,8> _result[2]) {
  ap_fixed<21,2> weights1[20][4] = {{-0.036691424, ...}...}
  ap_fixed<17,1> bias1[20] = {0.027661305, ...}
  ...
  for (int i=0; i<2; i++) {
    ap_fixed<32,18> _dot_tmp = weights2[i][0] * layer1[0] + ...
    ...
  _result[1] = layer2[1];
}
```

Quantization
without safety analysis

Our Contribution: A Framework for Closed-Loop Safety Verification and Sound, Efficient Quantization

Safety Verification with Precision Error

$l_1 <= in_1 <= u_1$
...
$l_m <= in_m <= u_m$

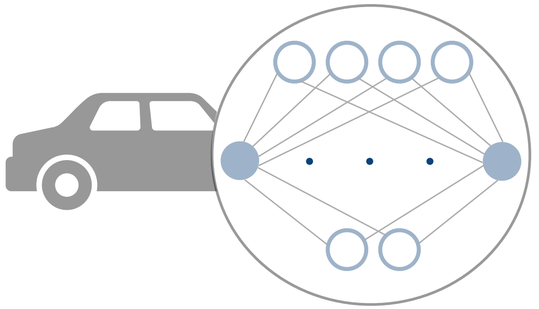


+/- error

`+/- 1e-5` for every iteration

Polar-Express

$l_1 <= in_1 <= u_1$
...
$l_m <= in_m <= u_m$



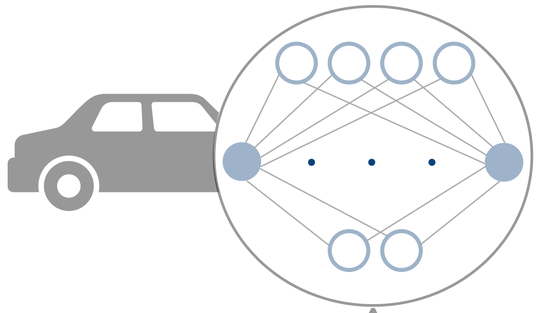
`res +/- 1e-5`

```
#include <math.h>
#include <ap_fixed.h>
#include <hls_math.h>
#include <ap_fixed.h>

void nn1(ap_fixed<24,5> x_0, ap_fixed<24,4> x_1, ap_fixed<24,3> x_2,
ap_fixed<24,2> x_3, ap_fixed<27,8> _result[2]) {
  ap_fixed<21,2> weights1[20][4] = {{-0.036691424, ...}...}
  ap_fixed<17,1> bias1[20] = {0.027661305, ...}
  ...
  for (int i=0; i<2; i++) {
    ap_fixed<32,18> _dot_tmp = weights2[i][0] * layer1[0] + ...
    ...
  _result[1] = layer2[1];
}
```

Nexus https://github.com/harikishants/Nexus

Sound Quantization of NN Controllers

$l_1 <= in_1 <= u_1$
...
$l_m <= in_m <= u_m$



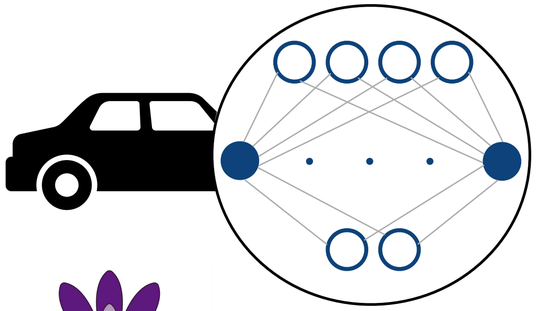
+/- error

+/- 1e-5

Polar-Express

SAFE

$l_1 <= in_1 <= u_1$
...
$l_m <= in_m <= u_m$



Aster

res +/- 1e-5

```
#include <math.h>
#include <ap_fixed.h>
#include <hls_math.h>
#include <ap_fixed.h>

void nn1(ap_fixed<24,5> x_0, ap_fixed<24,4> x_1, ap_fixed<24,3> x_2,
ap_fixed<24,2> x_3, ap_fixed<27,8> _result[2]) {
  ap_fixed<21,2> weights1[20][4] = {{-0.036691424, ...}...}
  ap_fixed<17,1> bias1[20] = {0.027661305, ...}
  ...
  for (int i=0; i<2; i++) {
    ap_fixed<32,18> _dot_tmp = weights2[i][0] * layer1[0] + ...
    ...
  _result[1] = layer2[1];
}
```

XILINX

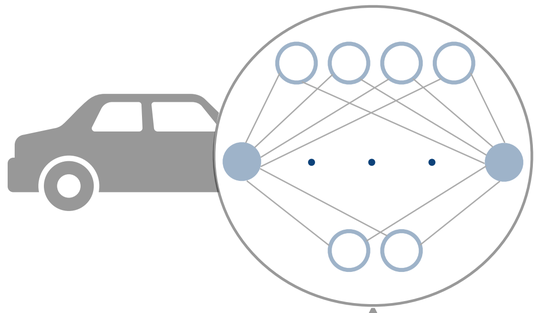
latency = 107 cycles

Nexus

https://github.com/harikishants/Nexus

Sound Quantization of NN Controllers

$l_1 <= in_1 <= u_1$
...
$l_m <= in_m <= u_m$

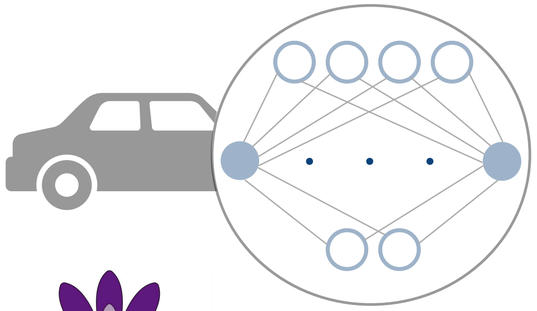


+/- error

+/- 1e-5

Polar-Express

$l_1 <= in_1 <= u_1$
...
$l_m <= in_m <= u_m$



Aster

res +/- 1e-5

serial code

```
#include <math.h>
#in...<ap_fixed.h>
#...clude <...s_math.h>
#include <a..._fixed.h>

void nn1(a..._fixed<24,5> x_0, ap_fixed<24,4> x_1, ap_fixed<24,3> x_2,
ap_...2> x_3, ap_fixed<27,8> _result[2]) {
  ap_fixed...,2> weights1[20][4] = {{-0.036691424, ...}...}
  ap_fixed<...> bias1[20] = {0.027661305, ...}
  ...
  for (int i=0; i<2; i++) {
    ap_fixed<32,18> _dot_tmp = weights2[i][0] * layer1[0] + ...
    ...
  _result[1] = layer2[1];
}
```

XILINX

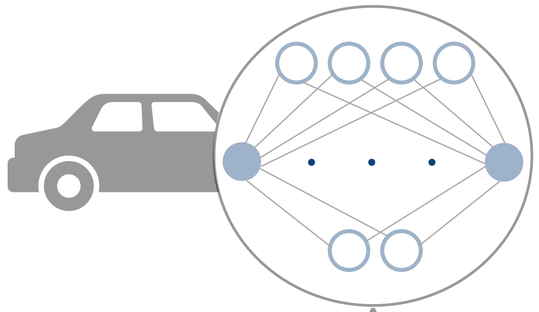
latency = 107 cycles

Nexus

https://github.com/harikishants/Nexus

Extended Code Generation

$l_1 <= in_1 <= u_1$
$...$
$l_m <= in_m <= u_m$



+/- error

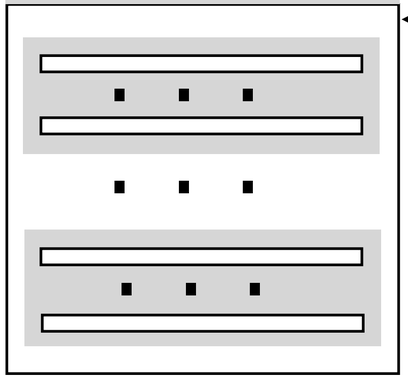
+/- 1e-5

Polar-Express

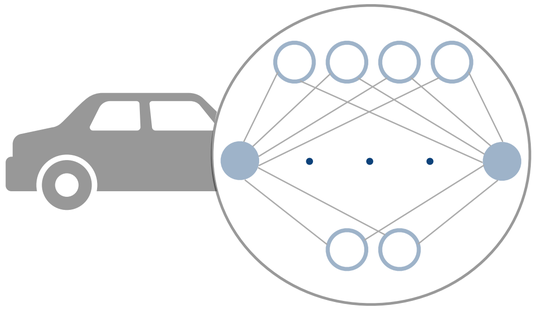
+

extended code generation

parallel constructs



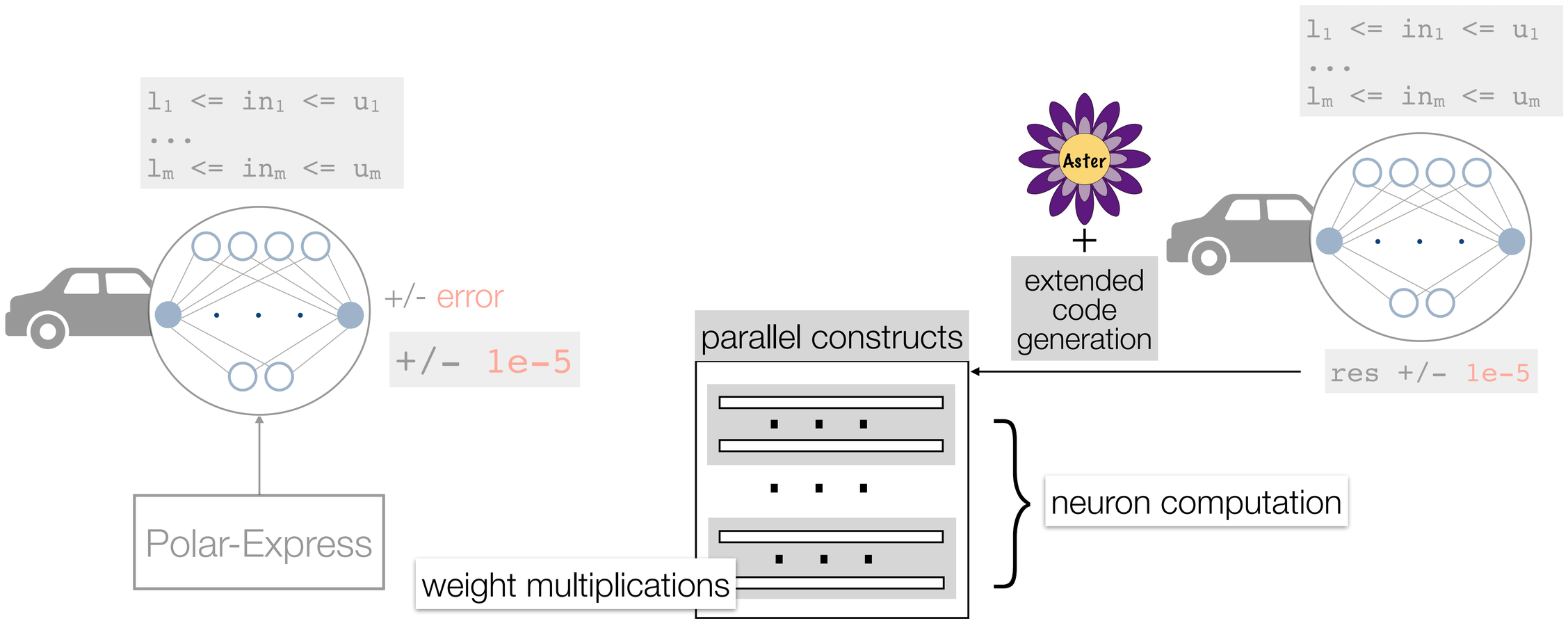
$l_1 <= in_1 <= u_1$
$...$
$l_m <= in_m <= u_m$



res +/- 1e-5

Nexus https://github.com/harikishants/Nexus

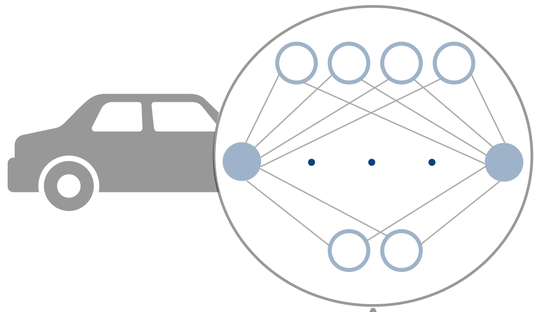
Extended Code Generation



Nexus https://github.com/harikishants/Nexus

Extended Code Generation

$l_1 <= in_1 <= u_1$
$...$
$l_m <= in_m <= u_m$

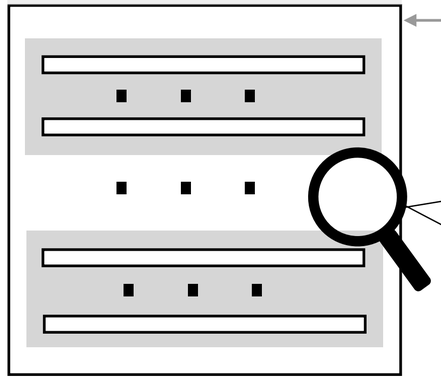


+/- error

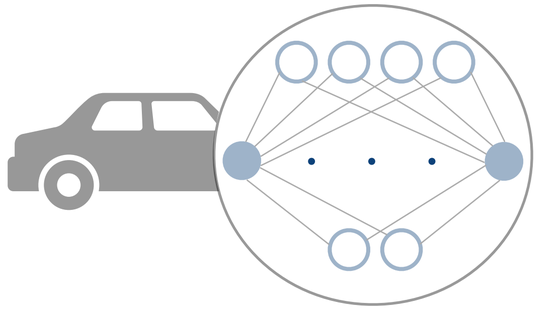
+/- 1e-5

Polar-Express

parallel constructs



$l_1 <= in_1 <= u_1$
$...$
$l_m <= in_m <= u_m$



res +/- 1e-5

```
#include <math.h>
#include <ap_fixed.h>
#include <hls_math.h>
#include <ap_fixed.h>
void nn1(ap_fixed<24,5> x_0, ap_fixed<24,4> x_1, ap_fixed<24,3> x_2,
ap_fixed<24,2> x_3, ap_fixed<27,8> _result[2]) {
#pragma HLS PIPELINE
#pragma HLS ARRAY_PARTITION variable=_result complete
  ap_fixed<21,2> weights1[500][4] = {{-0.036691424, ...}...}
  ...
  for (int i=0; i<20; i++) {
    ap_fixed<32,18> _dot_tmp = weights2[i][0] * layer1[0] + ...
    ...
  _result[1] = layer2[1];
}
```

XILINX

latency = 63 cycles

Nexus https://github.com/harikishants/Nexus

An Evaluation of Nexus

benchmarks	#plant-vars	ctrl-step	#params	safety
InvPend	6	0.05	60	✓
MountCar	3	1.00	336	✓
SglPend	4	0.05	775	✓
DblPend	7	0.02	825	✓
ACC5	10	0.10	1,820	✓
Unicyle	7	0.20	3,500	✗
Airplane	19	0.10	13,540	✗
TORA	5	1.00	20,800	✗

Safety analysis and sound code generation considering target error 1e-5, ✓:safe, ✗:unsafe, ✗: reachability analysis fails

Nexus vs Aster in terms of latencies of the implementations

benchmarks	#plant-vars	ctrl-step	#params	safety	latency		design syn-time (s)	
					Nexus	Aster	Nexus	Aster
InvPend	6	0.05	60	✓	**14**	18	**24.37**	24.85
MountCar	3	1.00	336	✓	**25**	38	31.32	**28.33**
SglPend	4	0.05	775	✓	**27**	47	46.16	**35.30**
DblPend	7	0.02	825	✓	**28**	51	43.21	**36.65**
ACC5	10	0.10	1,820	✓	**63**	107	98.23	**50.34**
Unicyle	7	0.20	3,500	✗	-	-	-	-
Airplane	19	0.10	13,540	✗	-	-	-	-
TORA	5	1.00	20,800	✗	-	-	-	-

Safety analysis and sound code generation considering target error 1e-5, ✓:safe, ✗:unsafe, ✗: reachability analysis fails

Nexus vs Aster in terms of latencies of the implementations

benchmarks	#plant-vars	ctrl-step	#params	safety	latency		design syn-time (s)	
					Nexus	Aster	Nexus	Aster
InvPend	6	0.05	60	✓	**14**	18	**24.37**	24.85
MountCar	3	1.00	336	✓	**25**	38	31.32	**28.33**
SglPend	4	0.05	775	✓	**27**	47	46.16	**35.30**
DblPend	7	0.02	825	✓	**28**	51	43.21	**36.65**
ACC5	10	0.10	1,820	✓	**63**	107	98.23	**50.34**
Unicyle	7	0.20	3,500	✗	-	-	-	-
Airplane	19	0.10	13,540	✗	-	-	-	-
TORA	5	1.00	20,800	✗	-	-	-	-

Safety analysis and sound code generation considering target error 1e-5, ✓:safe, ✗:unsafe, ✗: reachability analysis fails

Nexus integrates safety verification and quantization, and also improves implementations' latencies through parallelization!