An Automated, Efficient and Static Bit-width Optimization Methodology Towards Maximum Bit-width-to-Error Tradeoff with Affine Arithmetic Model

Yu Pu [1,2]
Pu_Yu@nus.edu.sg

Yajun Ha [1]
elehy@nus.edu.sg

National University of Singapore[1]
Technische Universiteit Eindhoven[2]

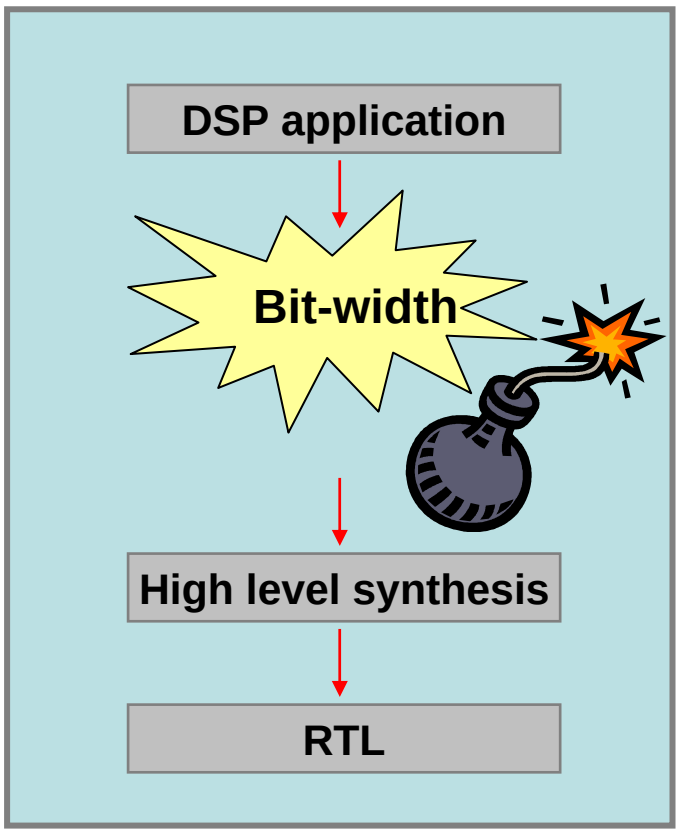
Outline

- **Introduction**
- **AA Bit-width analysis methodology**
- **Experimental results and verification**
- **Summary & Conclusions**

Outline

- Introduction
- AA Bit-width analysis methodology
- Experimental results and verification
- Summary & Conclusions

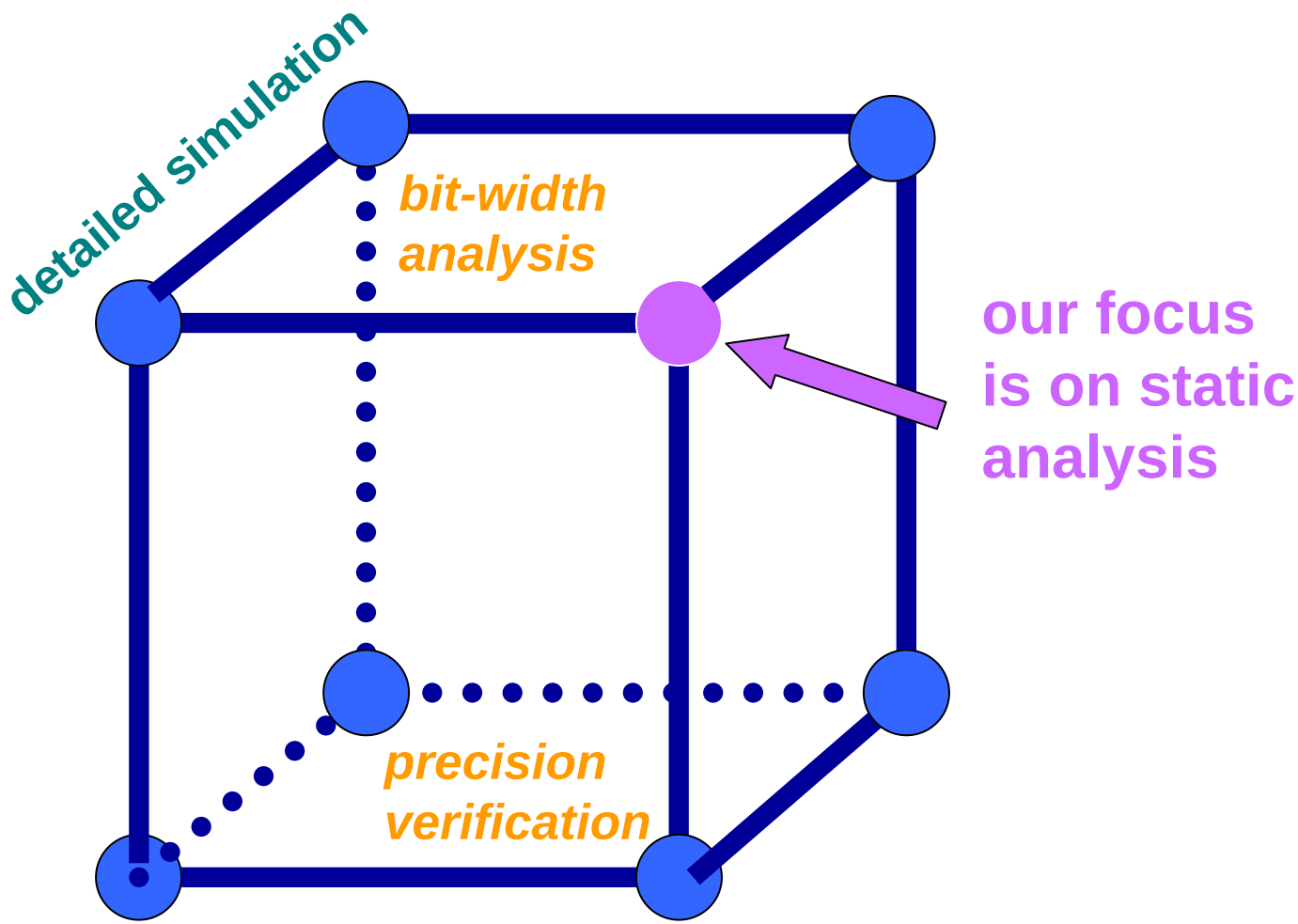
Bit-width analysis is important!



- **Bit-width analysis guarantees the precision of computation results.**

- **Bit-width analysis trades-offs precision with hardware resources, such as silicon area, power, etc.**

We focus on static bit-width analysis



Simulation based analysis vs. Static analysis

Simulation based analysis

- Monte-Carlo style simulation, iterative trial
- Close to optimal bit-width results
- Often huge searching space with full coverage of input vectors, thus running time inefficient

Static analysis

- Infer bit-width for integer by value range propagation (forward, backward propagation)
- Infer bit-width for fraction by precision analysis
- Sub-optimal bit-width results
- Much shorter running time and iteration reduced

Interval Arithmetic overestimates bit-width

- Existing static bit-width analysis methods are IA based.
- What is Interval Arithmetic (IA)?

An uncertain variable $\overline{x}$ is expressed as $\overline{x} = [x_{\min}, x_{\max}]$

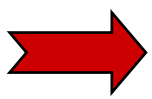
Take addition for an example:

$$\overline{z} = \overline{x} + \overline{y} = [x_{\min} + y_{\min}, x_{\max} + y_{\max}]$$

- An Example

Program

```
b = a - a;
Where a is 16-bit long
```



Interval Expressions

$b = [a_{min} - a_{max}, a_{max} + a_{max}]$

Results:

b is 17-bit long

- IA overestimates bit-width, enabling fairly pessimistic results.

Affine Arithmetic estimates bit-width better

- **What is Affine Arithmetic (AA)**

An uncertain variable $\overline{x}$ is expressed as

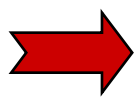
$$\hat{x} = x_0 + x_1\varepsilon_1 + x_2\varepsilon_2 + \cdots + x_n\varepsilon_n, -1 \le \varepsilon_i \le 1$$

central value coefficient noise symbol

- **Previous Example**

Program

```
b = a - a;
a is 16-bit long
```



Affine Expressions

$$a = a_0 + a_1\varepsilon_a$$
$$b = (a_0 + a_1\varepsilon_a) - (a_0 + a_1\varepsilon_a);$$
$$= 0;$$

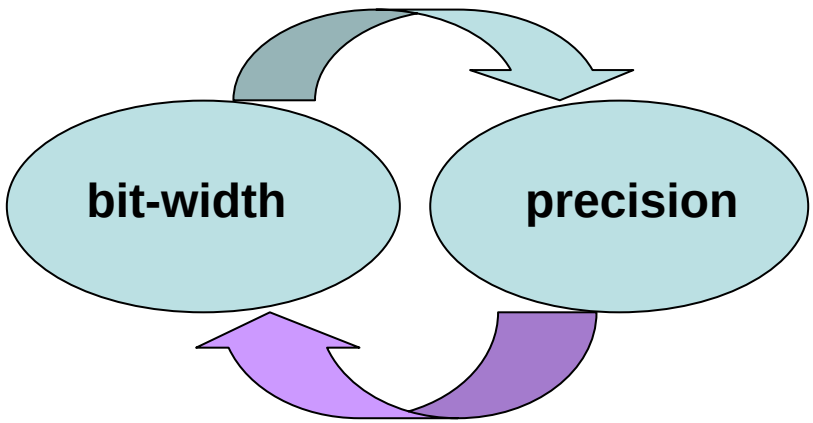
Results:

```
b is 1-bit long
```

- **AA models correlation between variables, enables tighter range propagation through cancelling some uncertainties along data-path.**

We extend AA for bit-width analysis

- **Fang et. al. (CMU) have introduced AA model into the verification of the finite-precision effects in DSP applications. Detailed mathematical reasoning has been explained in [1] [2].**
- **We extend AA model for bit-width analysis.**



[1] C. F. Fang, R. A. Rutenbar, M.Puschel and T. Chen, "Towards efficient static analysis of finite precision effects in DSP applications via affine arithmetic modeling," in *Proceedings of 40th Design Automation Conference*, pp.496 – 501, 2003.

[2] C. F. Fang, R. A. Rutenbar, M. Puschel and T. Chen, "Fast, accurate static analysis for fixed-point finite-precision effects in DSP designs," in *Proceedings of the International Conference on Computer Aided Design (ICCAD'03)*, pp.275- 282, San Jose, California, USA, 2003.

Closely related work

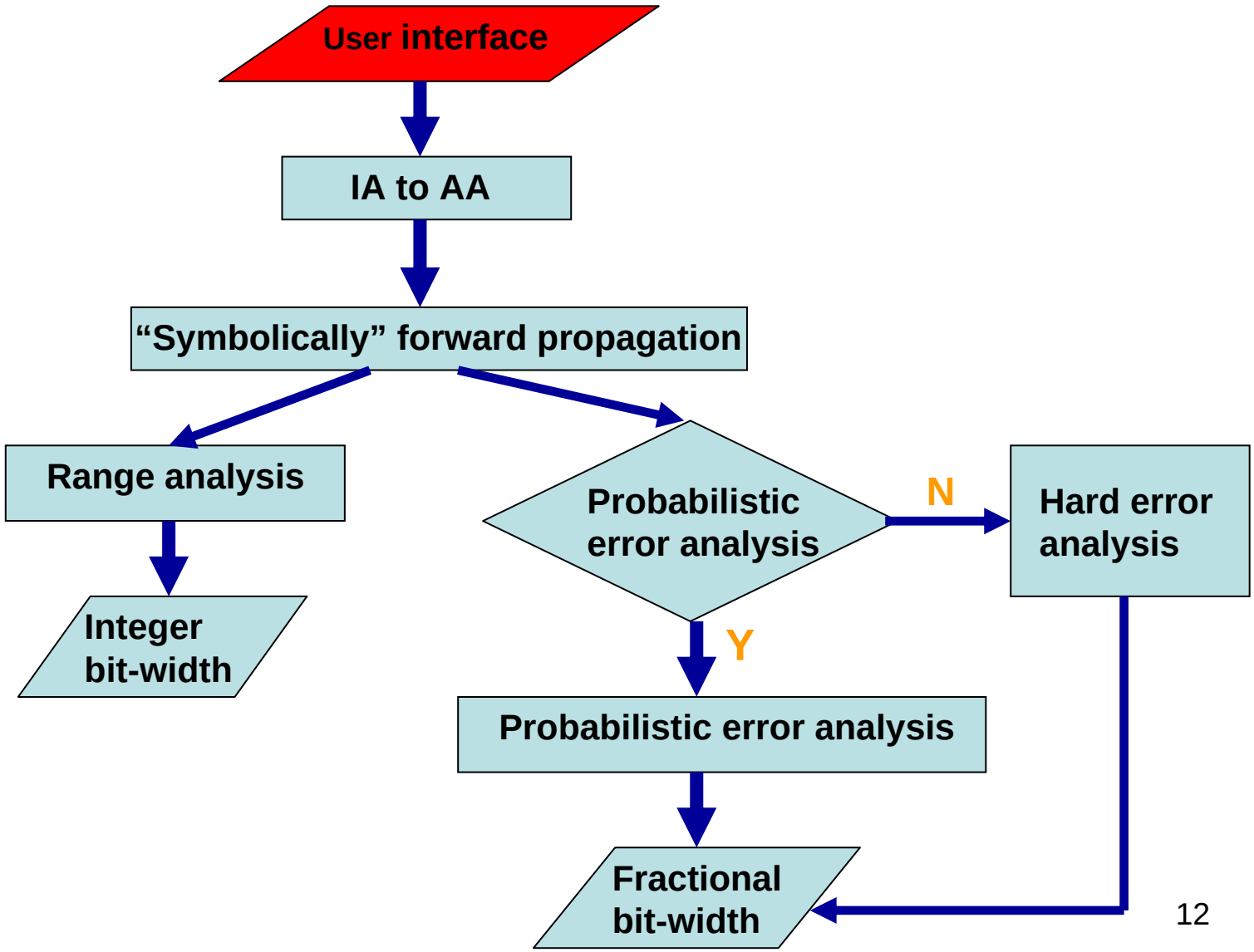
A similar research was conducted by Dr. Lee [3] (imperial college). They applied AA together with the adaptive simulated annealing algorithm to find the optimal fractional bits.

[3] D-U. Lee, A. A. Gaffar, O. Mencer, W. Luk, "MiniBit: Bit-width optimization via affine arithmetic," in *Proceedings of the 42nd annual conference on Design automation*, June, 2005.

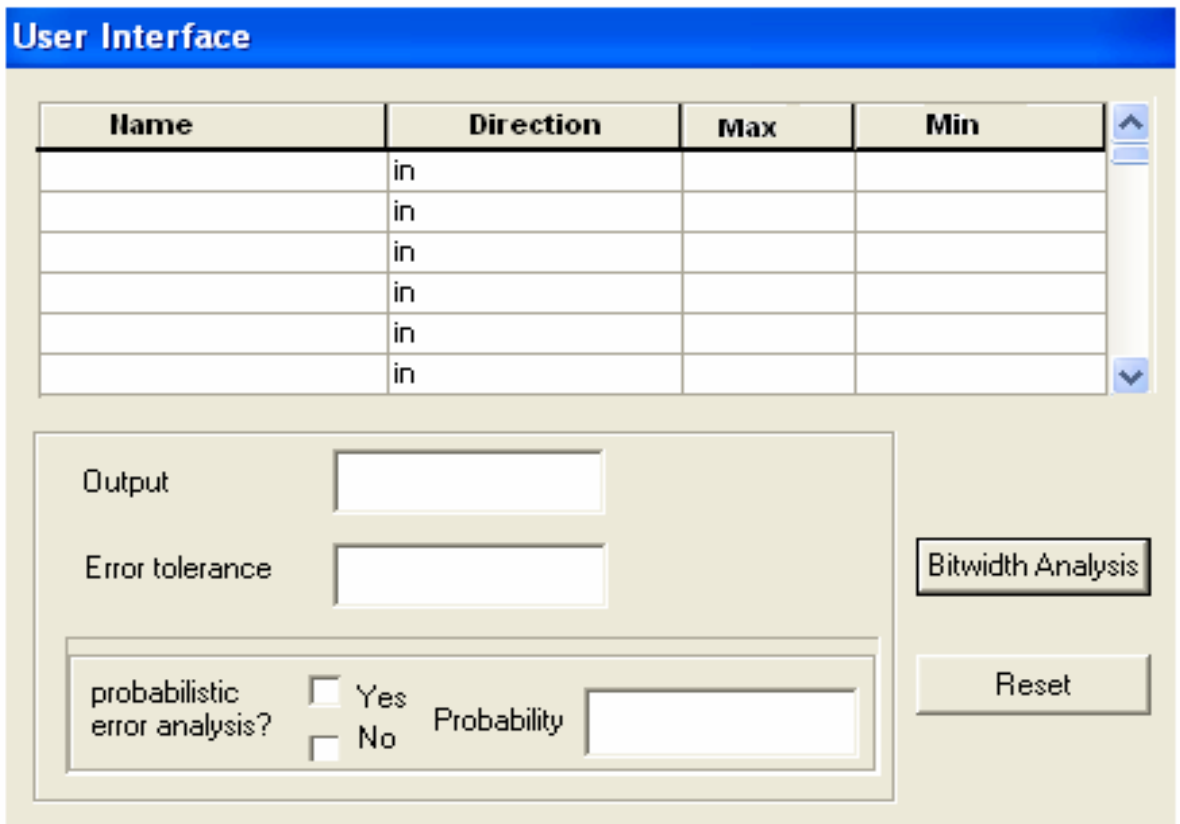
Outline

- **Introduction**
- **AA Bit-width analysis methodology**
- **Experimental results and verification**
- **Summary & Conclusions**

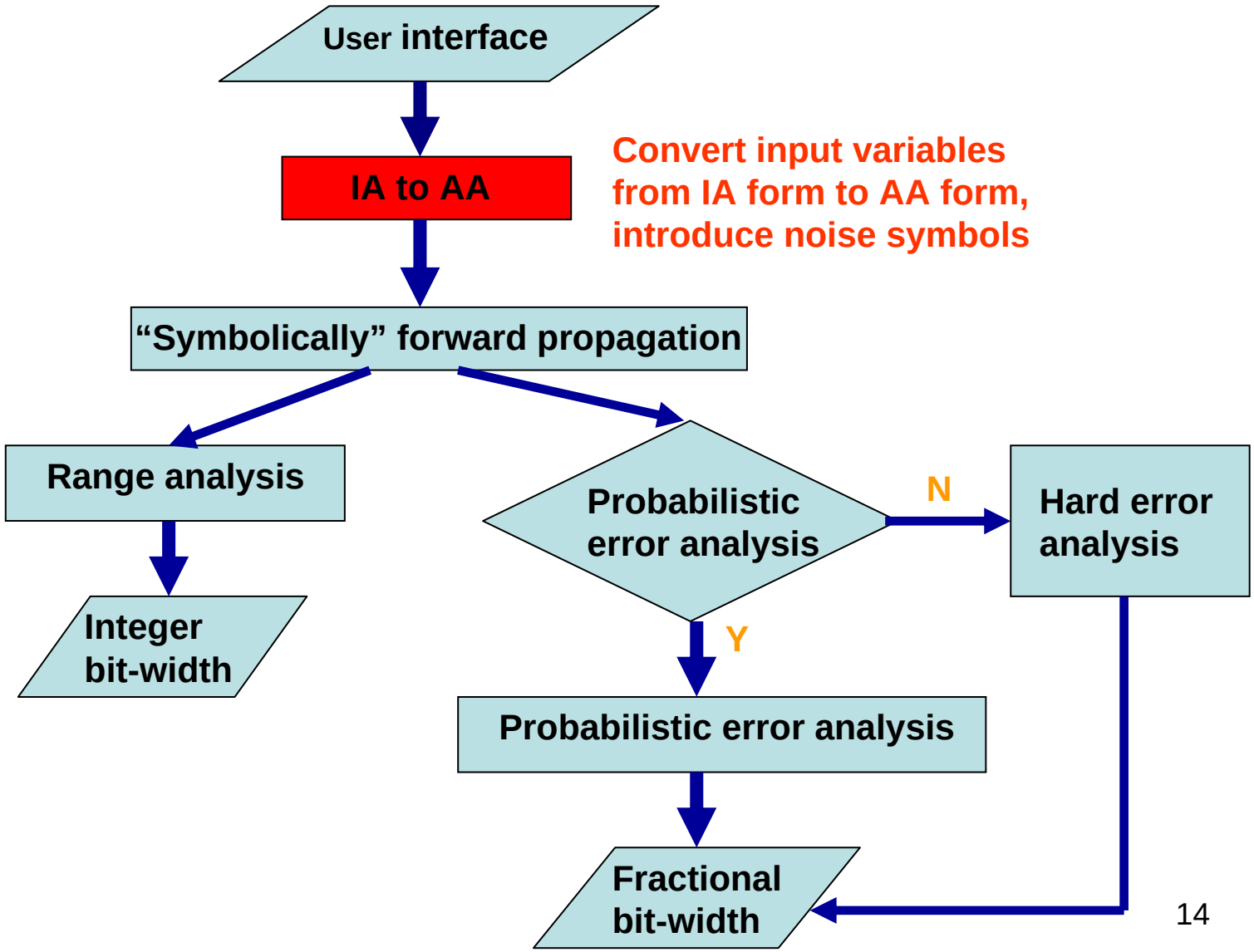
Methodology overview



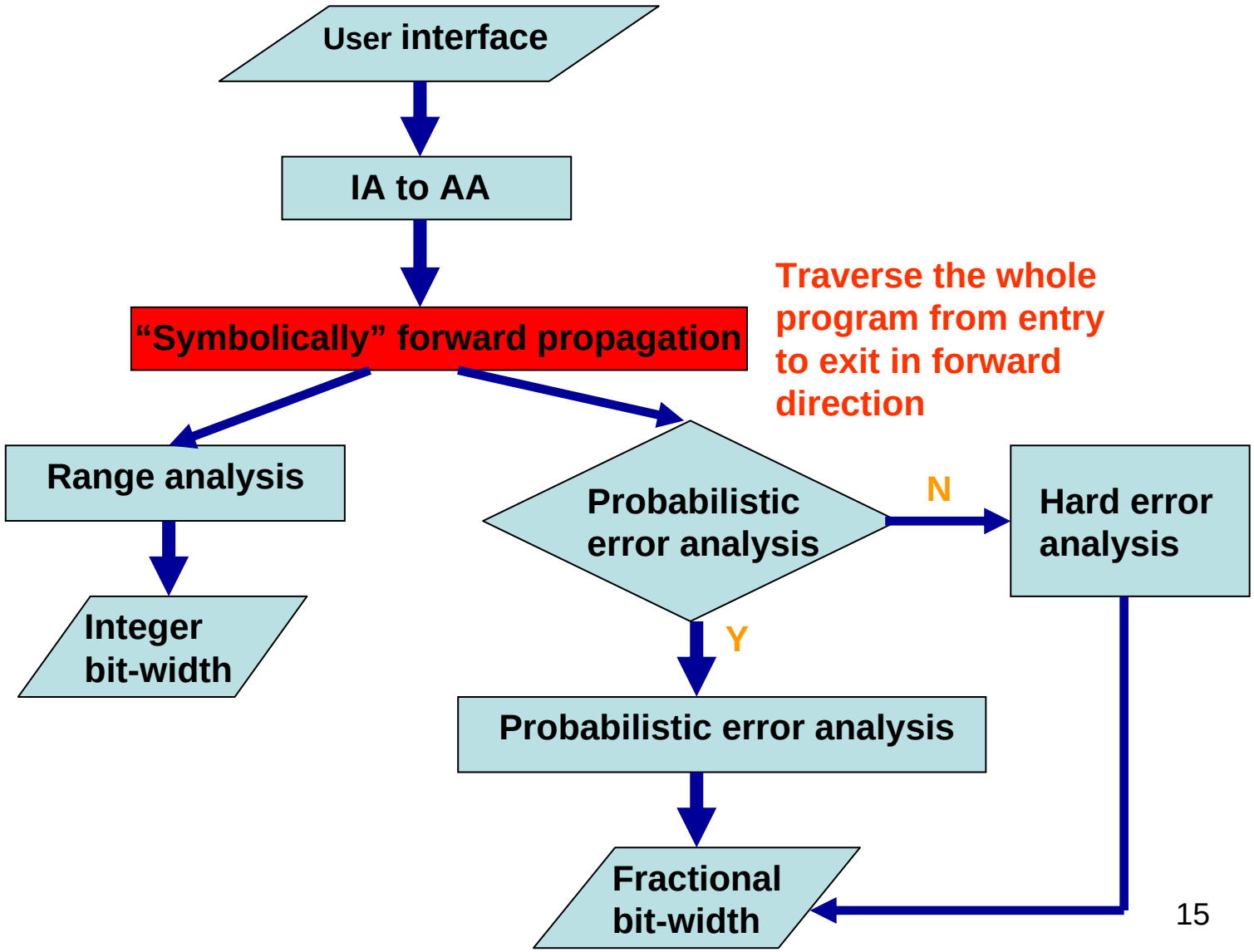
User interface for MISO system



Methodology overview



Methodology overview



Variable expression

Assume that there are totally N floating-point variables whose upper quantization error bounds are expressed as

$$|\Delta_1|, |\Delta_2|, |\Delta_3| \cdots |\Delta_N|$$

When they are transformed to fixed-point, the actual quantization errors incurred at each of them are expressed in affine forms as

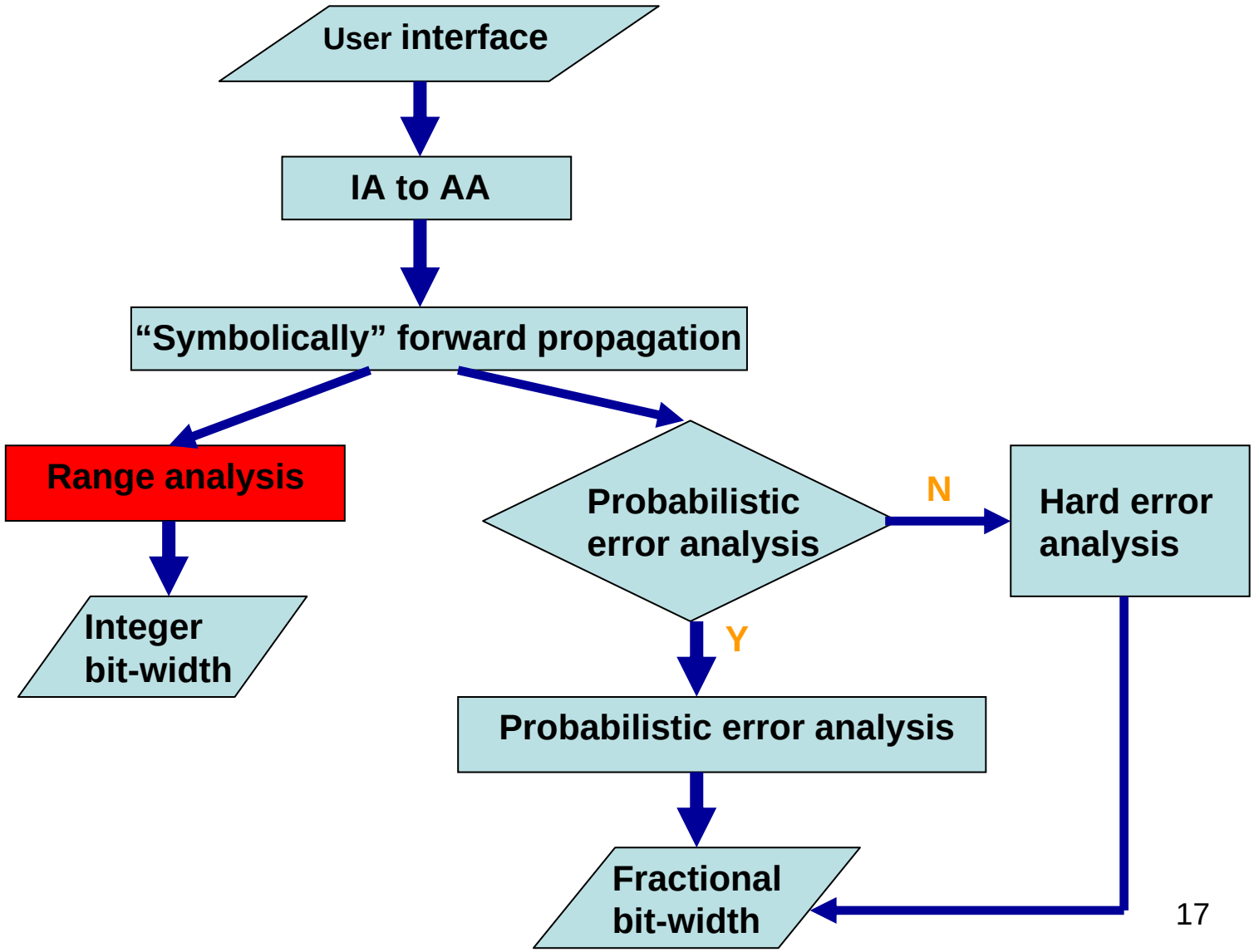
$$|\Delta_1|\varepsilon_1, |\Delta_2|\varepsilon_2, |\Delta_3|\varepsilon_3, \cdots |\Delta_N|\varepsilon_N$$

The intermediate and output variable is expressed as

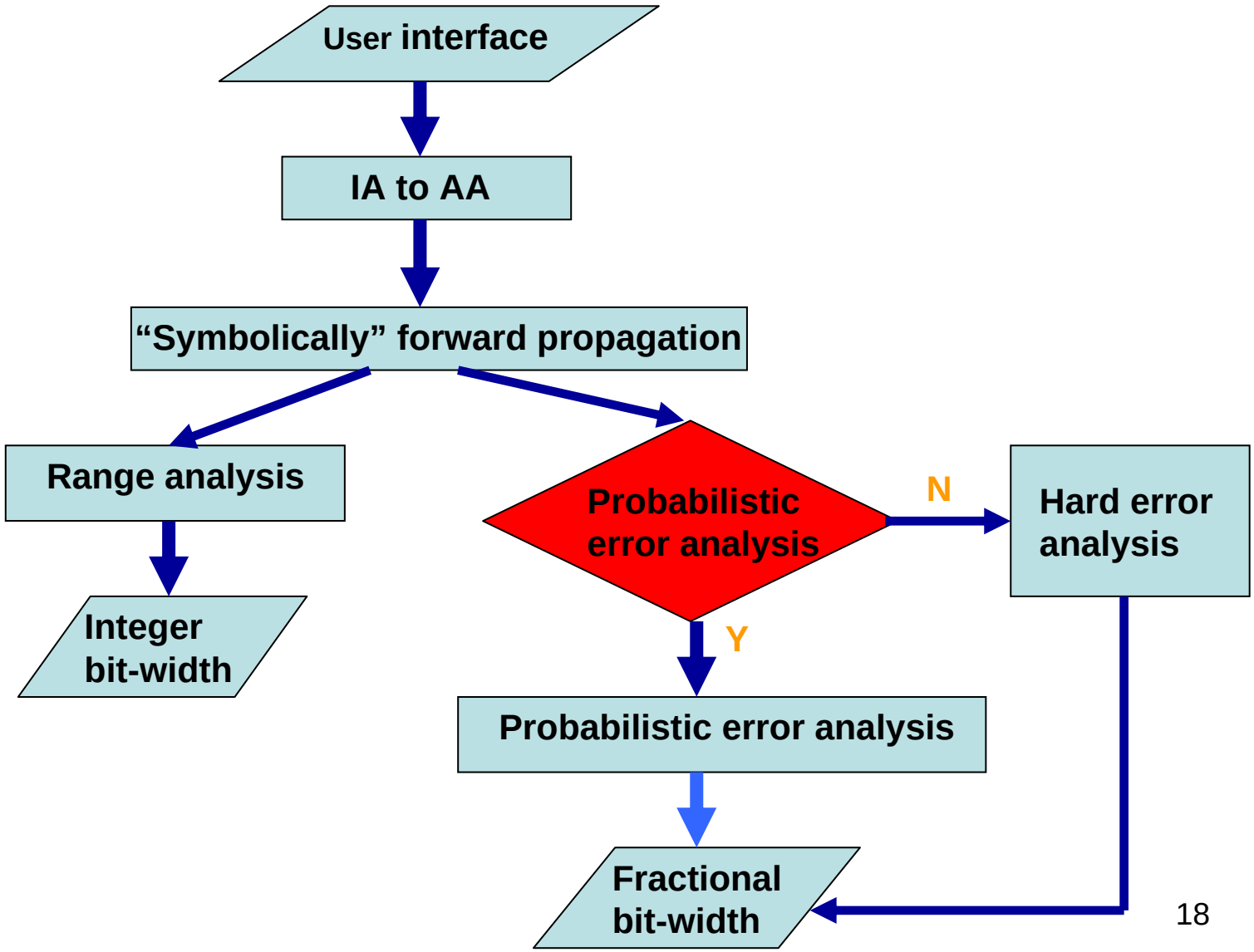
$$Variable = Const + \Delta Variable = Const + |A_1||\Delta_1|\varepsilon_1 + |A_2||\Delta_2|\varepsilon_2 + |A_3||\Delta_3|\varepsilon_3 + \cdots + |A_N||\Delta_N|\varepsilon_N$$

central value **error term**

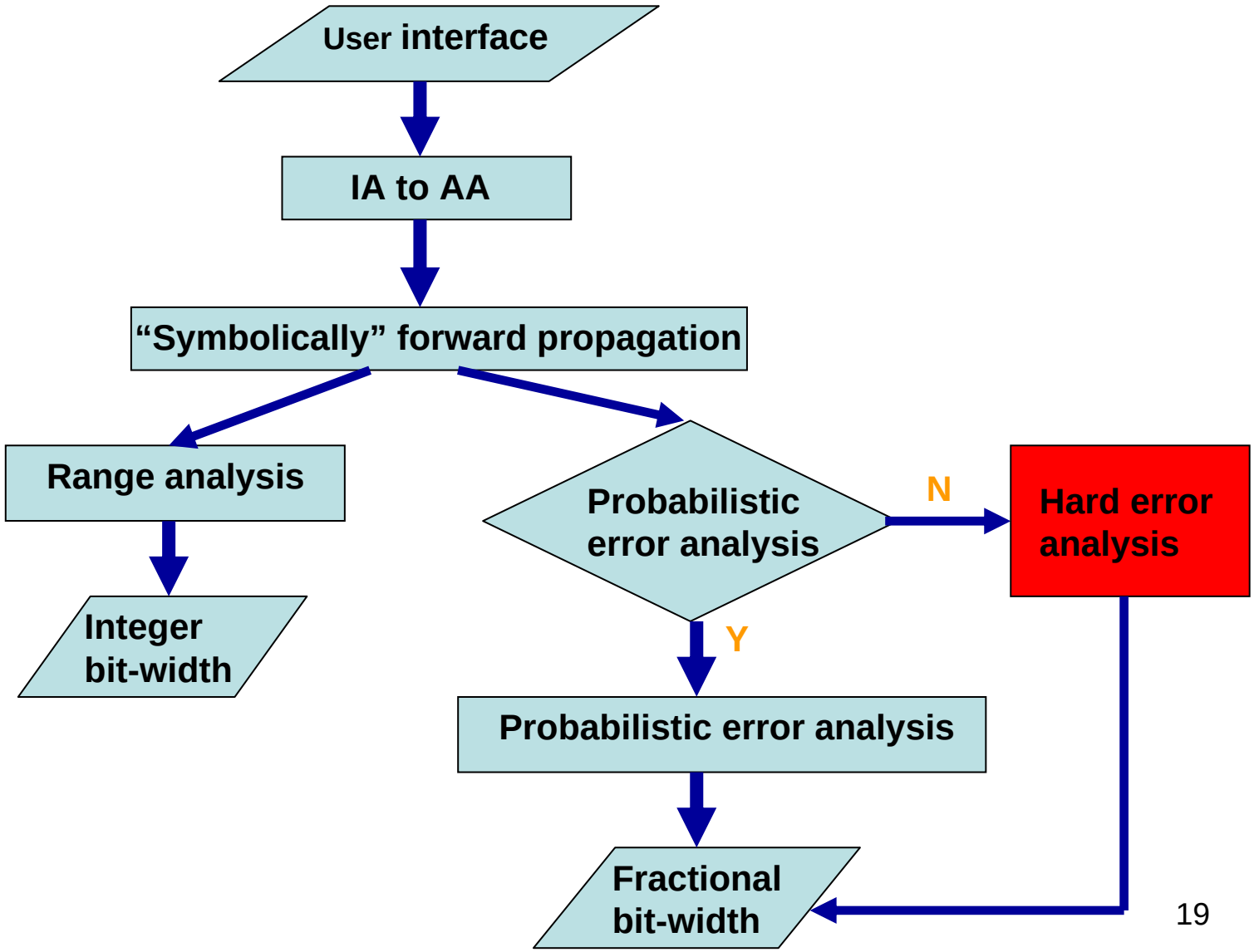
Methodology overview



Methodology overview



Methodology overview



Hard error analysis

The designer-specified output error tolerance E_{spec} sets the upper bound of Δ_{output}

$$|A_1||\Delta_1|\varepsilon_1 + |A_2||\Delta_2|\varepsilon_2 + |A_3||\Delta_3|\varepsilon_3 + \cdots + |A_N||\Delta_N|\varepsilon_N = \Delta_{output} \le Espec$$

To fully insures the output error not to exceed the designer-specified error tolerance, we take the extreme scenario, i.e.,

$\varepsilon_1 = \varepsilon_2 = \varepsilon_3 = \cdots = \varepsilon_N = 1$ **Therefore, we have**

$$|A_1||\Delta_1| + |A_2||\Delta_2| + |A_3||\Delta_3| + \cdots + |A_N||\Delta_N| \le Espec$$

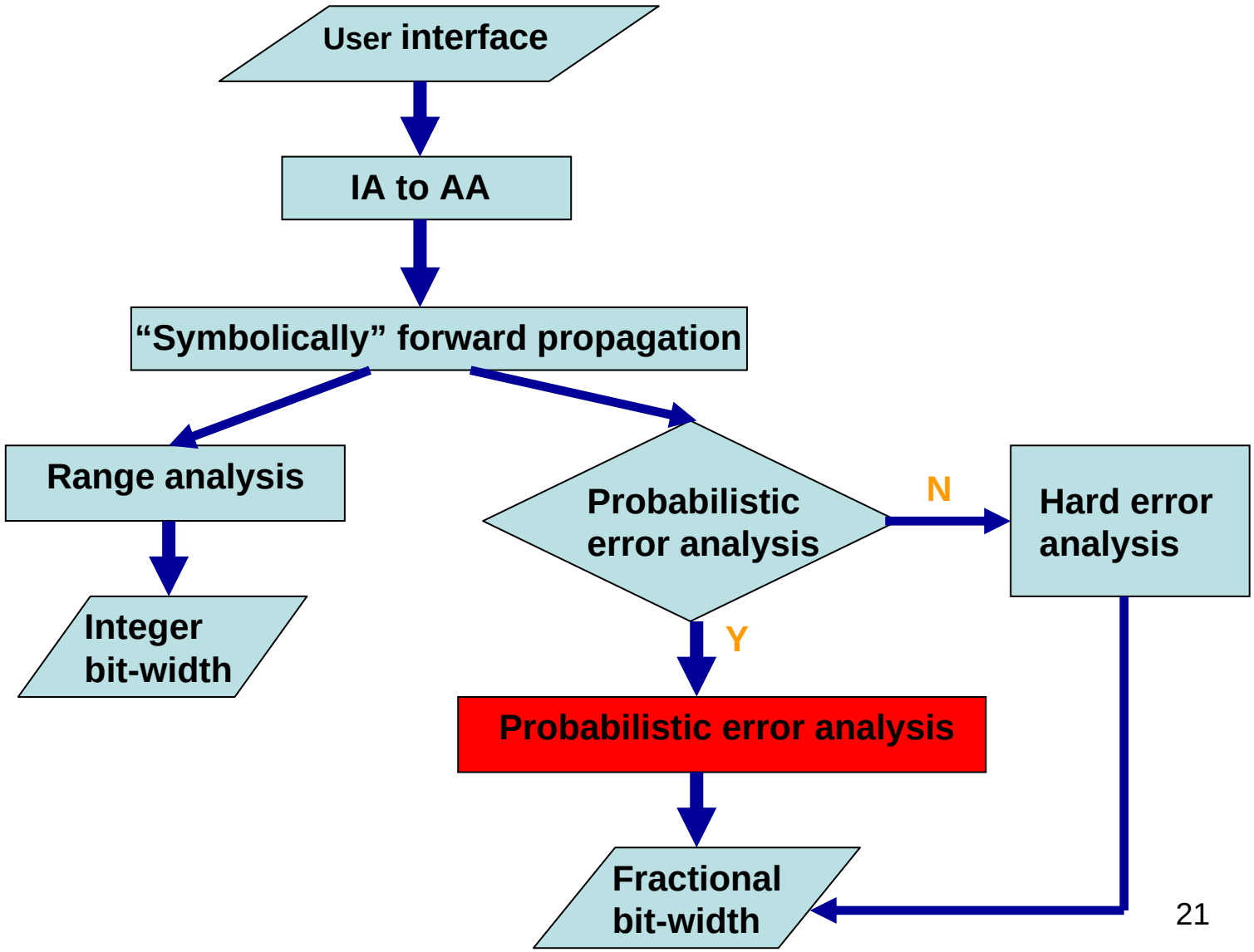
Assign non-negative weights to each term of the equation,

$$\sum_{i=1}^{N} W_i |A_i||\Delta_i| = Espec$$

$$\sum_{i=1}^{N} W_i = 1$$

In our research, we assign each term with same weights $W_i = 1/N$

Methodology overview



Probabilistic error analysis

In most of the DSP designs, the designer allows certain degree of error rate, which enables a larger bit-width-to-error tradeoff than using hard error analysis. Our probabilistic error analysis almost insures that the probability for the output error to lie in the specified error tolerance is higher than the designer specified parameter λ

$$|A_1||\Delta_1|\varepsilon_1 + |A_2||\Delta_2|\varepsilon_2 + |A_3||\Delta_3|\varepsilon_3 + \cdots + |A_N||\Delta_N|\varepsilon_N = \Delta_{output} \le Espec$$

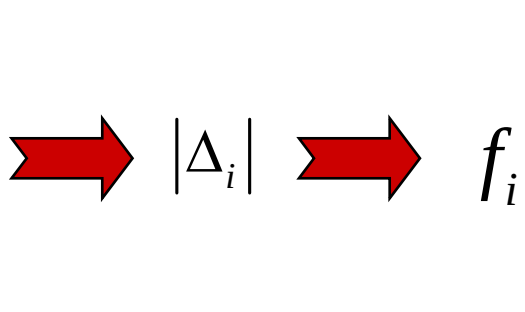
Let $|A_i||\Delta_i| = |A_i||\Delta_i| = \cdots = |A_i||\Delta_i| = k$,

the equations depicts a sum of many statistically independent and identical distributed terms. By the central limit theorem, the equation approaches a Gaussian CDF.

$$\frac{\Delta_{output}}{\sqrt{N}\sqrt{Variance}} \to \mathrm{N}(0,1)$$

$$Variance = k^2/3 = (|A_i||\Delta_i|)^2/3$$

$$prob(|\Delta_{output}| \le E_{spec}) \ge \lambda$$



Outline

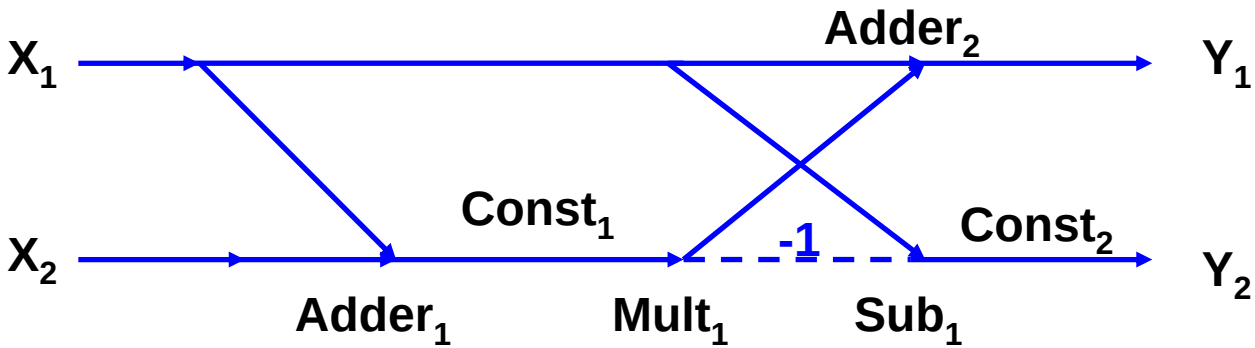
- **Introduction**
- **Motivation**
- **AA Bit-width analysis methodology**
- **Experimental results and verification**
- **Summary & Conclusions**

Experimental Results

A butterfly part in IDCT

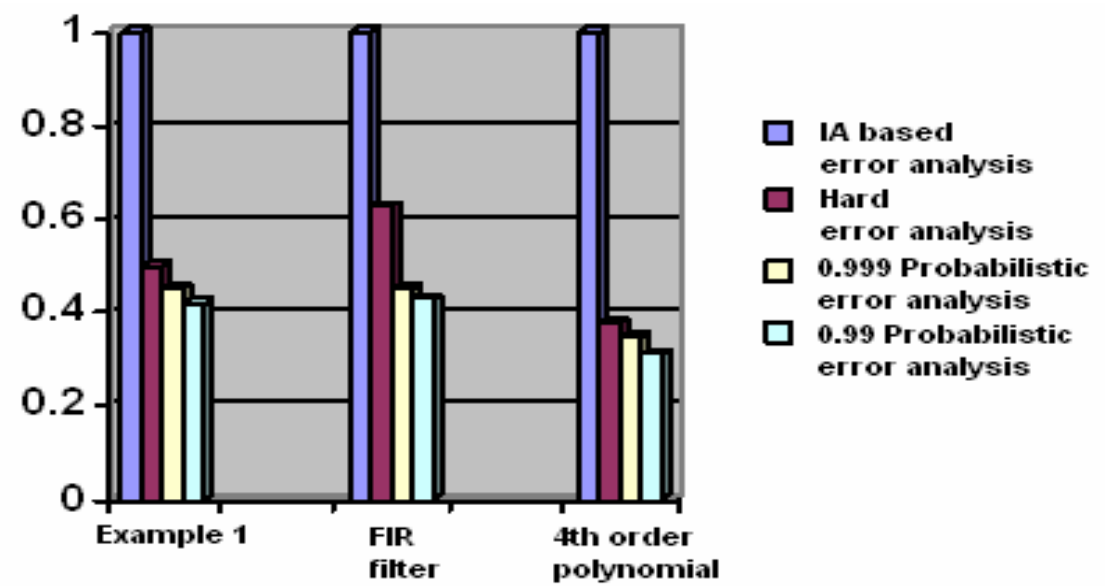
$X_1 \in [-128,127]$ $X_2 \in [-128,127]$ **Error tolerance at Y_2 is 1.0**

In probabilistic error analysis, the probability $\lambda = 0.999$.



Node		X_1	X_2	$Adder_1$	$Const_1$	$Mult_1$	Sub_1	$Const_2$	Y_2
Integer	IA	8	8	9	1	9	10	1	9
	AA	8	8	9	1	9	8	1	8
Fractional	IA	8	8	8	8	8	8	8	8
	Hard	0	3	3	9	3	3	8	3
	Prob.	0	2	2	10	2	2	9	2^{24}

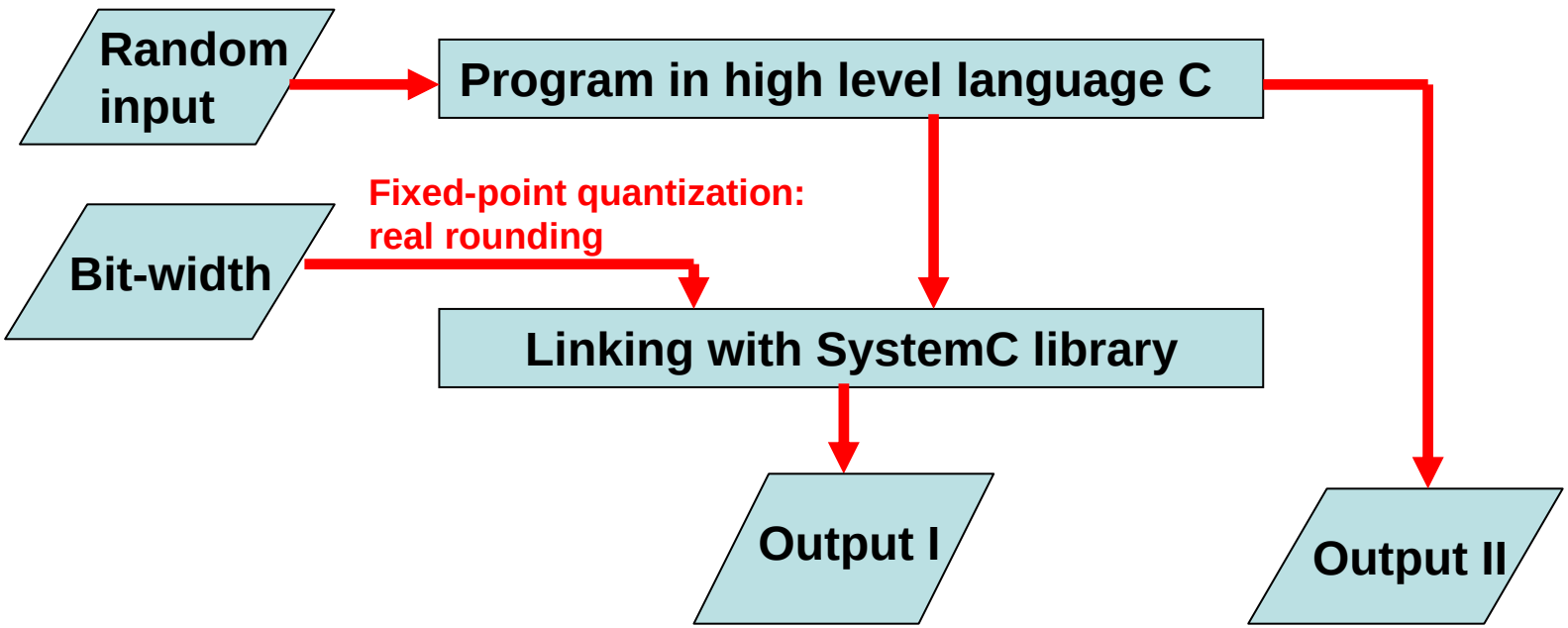
Experimental Results (Continued)



- Bit-width is normalized with respect to the IA based fractional bit-width.
- Hard error analysis → more than 30% fractional bit-width reduction
- Probabilistic error analysis → 50% reduction
- Tradeoff goes up as the relaxing the probabilistic restriction

Result Verification

- For the hard error analysis, whether the maximum output error lies within the specified tolerance after implementation?
- For the probabilistic error analysis, whether the probability for the output to lie in the specified error tolerance can be higher than the specified probability parameter?



Result Comparison and Analysis

Benchmark	Bit-width Analysis Method	Simulated output error	Specified Probability	Simulated Probability
Example I	IA based	0.02	N.A.	1
	AA based	0.22	N.A.	1
		0.47	0.99	1
		0.33	0.999	1
FIR filter	IA based	0.09	N.A.	1
	AA based	0.15	N.A.	1
		0.58	0.99	1
		0.58	0.999	1
4^{th} order polynomial	IA based	0.88	N.A.	1
	AA based	0.93	N.A.	1
		1.10	0.99	0.999
		0.95	0.999	1

Outline

- Introduction
- Motivation
- AA Bit-width analysis methodology
- Experimental results and verification
- Summary & Conclusions

Summary & Conclusions

- An automated and efficient static bit-width analysis methodology based on AA model is presented.
- The proposed probabilistic error analysis can further shorten the bit-width and explore a larger bit-width-to-error tradeoff.

Limitations:

- Our method lies in algorithm level which does not consider the hardware cost function.
- We use Gaussian approximation during analysis, theoretically it is difficult to fully guarantee the error probability to be bounded. Therefore, we suggest that the specified probability should be flexibly restricted.

Thank you!