Numerical Program Optimization for High-Level Synthesis

Xitong Gao, George A. Constantinides
xi.gao08@imperial.ac.uk, g.constantinides@imperial.ac.uk
Imperial College London

Introduction

Floating-point operations...

- are easy to use

  High dynamic range

- are ubiquitous

  Altera introduced new FPGAs (Arria 10 and Stratix 10) with hardened floating-point DSP elements

Introduction

Floating-point operations...

- are easy to use

  High dynamic range

- are ubiquitous

  Altera introduced new FPGAs (Arria 10 and Stratix 10) with hardened floating-point DSP elements

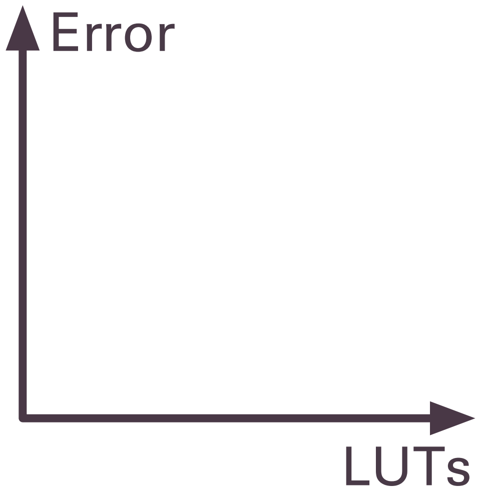
However, floating-point operations

- use a lot of resources
- have round-off errors

Introduction

Our tool exploits the rules of equivalence to **automatically** optimize the structure of numerical programs, for example:

- $(a + b) + c \equiv a + (b + c)$
- $(a + b) * c \equiv a * b + a * c$
- and many more



Introduction

GCC / LLVM /
Vivado HLS
-ffast-math

- Simple transformations
- What about accuracy?

Introduction

GCC / LLVM / Vivado HLS

-ffast-math

- Simple transformations
- What about accuracy?

SOAP

Arithmetic Expressions

- Deep transformations
- Resource usage & Accuracy!

Introduction

GCC / LLVM /
Vivado HLS
-ffast-math

- Simple transformations
- What about accuracy?

SOAP
Arithmetic
Expressions

- Deep transformations
- Resource usage & Accuracy!

SOAP2
Full
Programs

- Full program transformations

Example

Program

```
if (x < 20) {
    x = x + (y + 500);
} else {
    x = (x + y) + 500;
}
```

Example

Specification

```
float x, y;
assume(0 <= x <= 500);
assume(err(x) == 0);
assume(0 <= y <= 30);
assume(err(y) == 0);
```

Program

```
if (x < 20) {
    x = x + (y + 500);
} else {
    x = (x + y) + 500;
}
```

Example

Specification

```
float x, y;
assume(0 <= x <= 500);
assume(err(x) == 0);
assume(0 <= y <= 30);
assume(err(y) == 0);
```

Program

```
if (x < 20) {
    x = x + (y + 500);
} else {
    x = (x + y) + 500;
}
```

Example

Specification

```
float x, y;
assume(0 <= x <= 500);
assume(err(x) == 0);
assume(0 <= y <= 30);
assume(err(y) == 0);
```

Program

```
if (x < 20) {
    x = x + (y + 500);
} else {
    x = (x + y) + 500;
}
```

Example

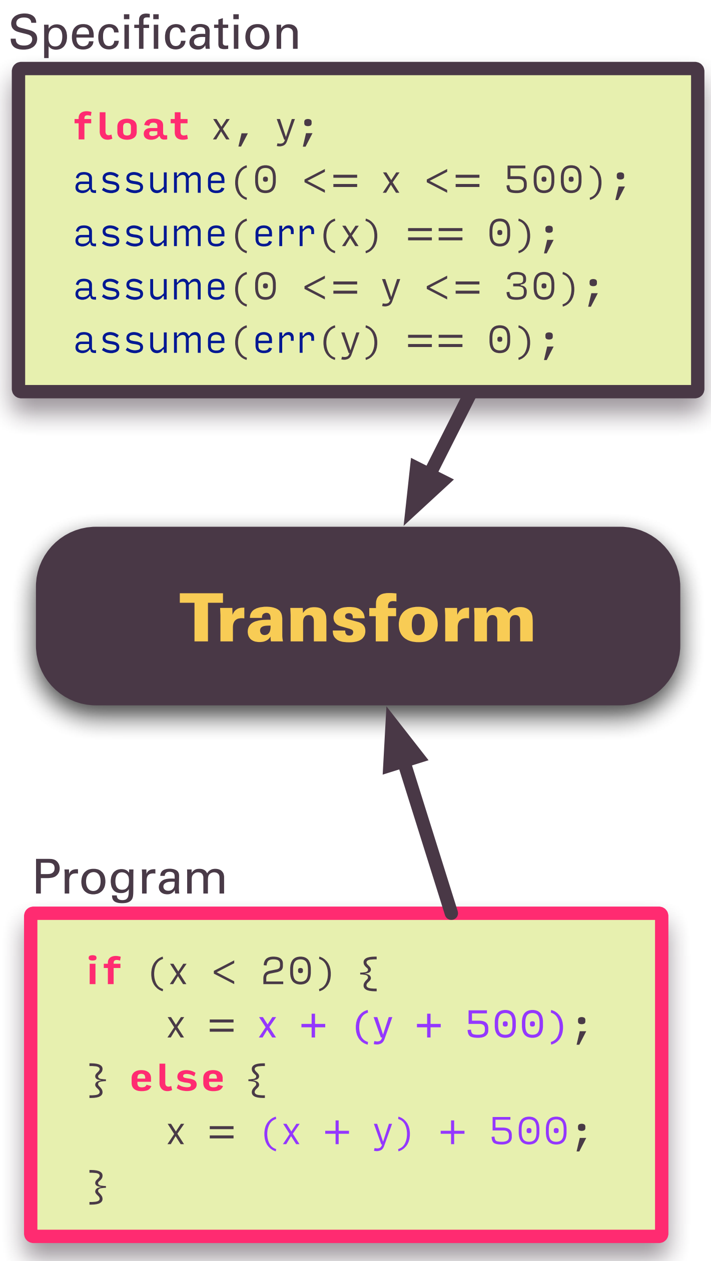
Specification

```
float x, y;
assume(0 <= x <= 500);
assume(err(x) == 0);
assume(0 <= y <= 30);
assume(err(y) == 0);
```

Program

```
if (x < 20) {
    x = x + (y + 500);
} else {
    x = (x + y) + 500;
}
```

Example



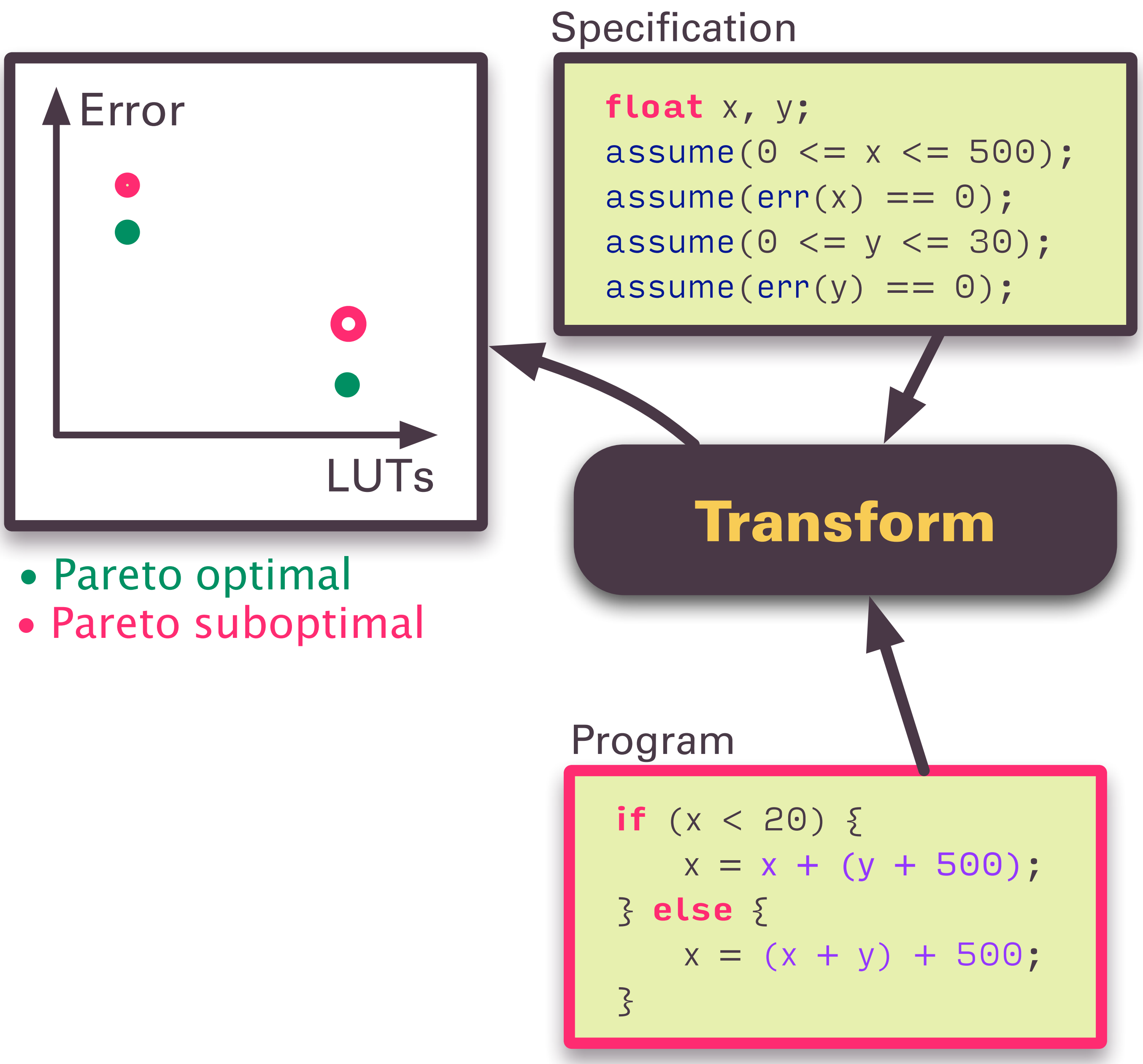
Specification

```
float x, y;
assume(0 <= x <= 500);
assume(err(x) == 0);
assume(0 <= y <= 30);
assume(err(y) == 0);
```

Program

```
if (x < 20) {
    x = x + (y + 500);
} else {
    x = (x + y) + 500;
}
```

Example



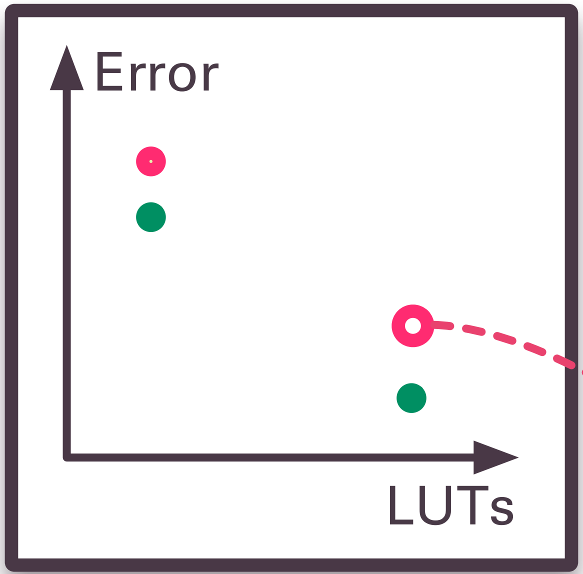
Specification

```
float x, y;
assume(0 <= x <= 500);
assume(err(x) == 0);
assume(0 <= y <= 30);
assume(err(y) == 0);
```

Program

```
if (x < 20) {
    x = x + (y + 500);
} else {
    x = (x + y) + 500;
}
```

Example



● Pareto optimal
● Pareto suboptimal

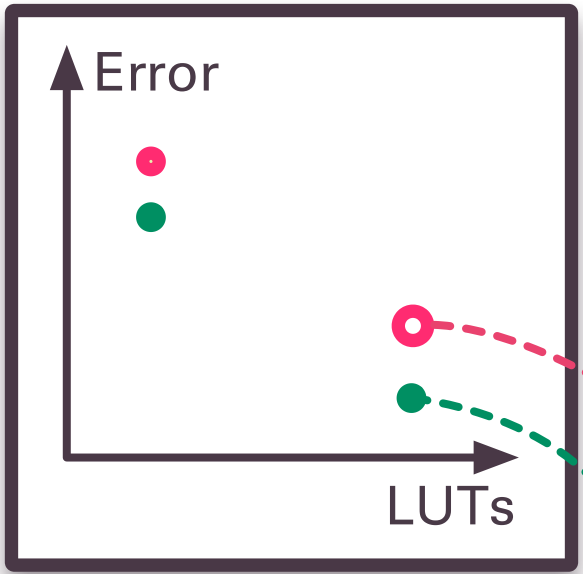
Specification

```
float x, y;
assume(0 <= x <= 500);
assume(err(x) == 0);
assume(0 <= y <= 30);
assume(err(y) == 0);
```

Program

```
if (x < 20) {
    x = x + (y + 500);
} else {
    x = (x + y) + 500;
}
```

Example



● Pareto optimal
● Pareto suboptimal

Specification

```
float x, y;
assume(0 <= x <= 500);
assume(err(x) == 0);
assume(0 <= y <= 30);
assume(err(y) == 0);
```

Most accurate

```
if (x < 20) {
    x = (x + y) + 500;
} else {
    x = x + (y + 500);
}
```

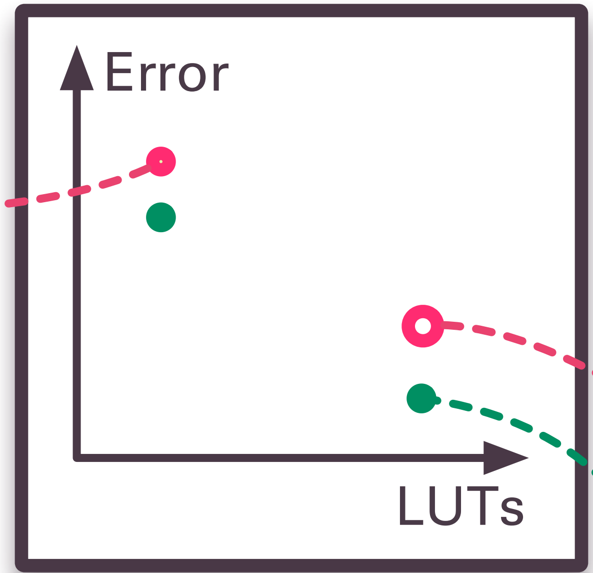
Program

```
if (x < 20) {
    x = x + (y + 500);
} else {
    x = (x + y) + 500;
}
```

Example

Fewest resources but less accurate

```
x = x + (y + 500);
```



● Pareto optimal
● Pareto suboptimal

Specification

```
float x, y;
assume(0 <= x <= 500);
assume(err(x) == 0);
assume(0 <= y <= 30);
assume(err(y) == 0);
```

Most accurate

```
if (x < 20) {
    x = (x + y) + 500;
} else {
    x = x + (y + 500);
}
```

Program

```
if (x < 20) {
    x = x + (y + 500);
} else {
    x = (x + y) + 500;
}
```

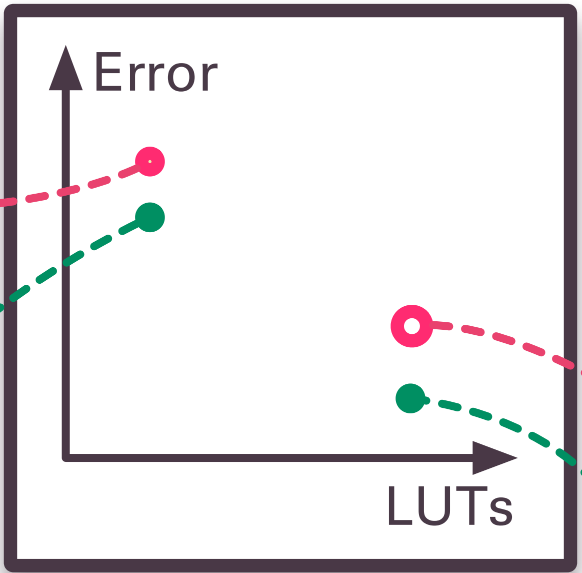
Example

Fewest resources but less accurate

```
x = x + (y + 500);
```

Fewest resources but more accurate

```
x = (x + y) + 500;
```



● Pareto optimal
● Pareto suboptimal

Specification

```
float x, y;
assume(0 <= x <= 500);
assume(err(x) == 0);
assume(0 <= y <= 30);
assume(err(y) == 0);
```

Most accurate

```
if (x < 20) {
    x = (x + y) + 500;
} else {
    x = x + (y + 500);
}
```

Program

```
if (x < 20) {
    x = x + (y + 500);
} else {
    x = (x + y) + 500;
}
```

And there is more...

There are a lot of things we did not cover:

- how we do that
  a more complex example program
- work flow
  how it fits in the traditional HLS work flow
- results
  ~60% better accuracy

All of these above are in the poster!

Thank you!

Join us in the poster session