Clash Formal

Synergizing Functional Programming, Proofs & Hardware Design

Dr. Felix Klein

October 6th, 2025 – **FOMSESS Annual Workshop**

QBayLogic.
The key to FPGA-ASIC design

QBayLogic: Our Expertise

Member of CHIPTECH TWENTE

- **FPGA based Development:**
  IP Design in VHDL, (System)Verilog, **Haskell/Clash**
- **ASIC Design:** RTL Development, Simulation
- **FPGA/ASIC Validation and Verification**
- **Systems-on-Chip:**
  QSys, IP-designer Vivado, IP-blocks, LiTex, RISC-V
- **Workflow Design & Setup:** CI / CD / CT / CV
- **(Research related) Product Development:**
  from the idea to the first prototype
- **Project Planning & Management**

Customers:

Google AIRBUS DEMCON AXIGN An mes Company esa Myrtle.ai Bronkhorst TNO innovation for life POSITRON

Haskell: more than just a programming language

- **Functional Language**
  - based on the lambda calculus
  - independent from any computer architecture specifics
- **Strongly Typed**
  - bugs can be catched even before running the program
  - properties can be expressed and ensured via the types
- **Research Use**
  - various papers, tools and research about (and using) the language
  - popular at universities (introductory courses to computer science)
- **Industrial Use**
  - *Meta, Microsoft, Standard Chartered, Tesla, Klarna, Galois, Serokell, ...*
  - popular for applications with strong safety and security requirements

Haskell: a versatile all-rounder

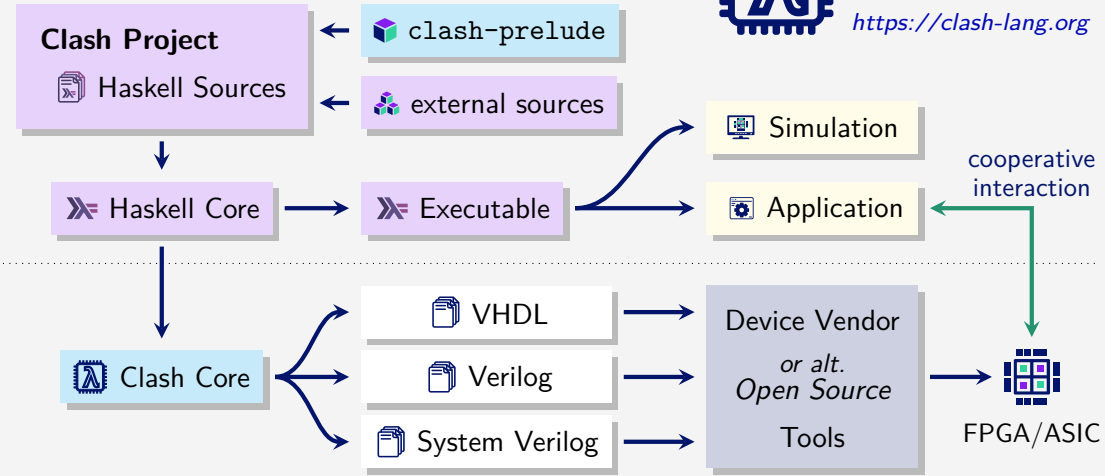
- **Web Frameworks:** IHP, Obelisk, Snap, Yesod, ...
- **Build & Package Management:** Cabal, Nix, Shake, ...
- **Embedded & Distributed Systems:** Ivory, Copilot, Cloud Haskell, sparkle, ...
- **Graphics & Music & Art:** Gloss, Diagrams, Haskore, Tidal Cycles, ...
- **Formal Verification**
  - Haskell has its own, built-in, type based constraint system
  - Correctnes Proofs / Proof Assistants: `hs-to-coq`, `agda2hs`, Haskabelle, ...
  - Refinement Types: Liquid Haskell
  - Automated Reasoning: `sbv`, `yices-painless`, ...
- **Hardware:** Bluespec, Lava, ForSyDe, **Clash**
- ...

Clash Compiler: Haskell → Hardware

Clash

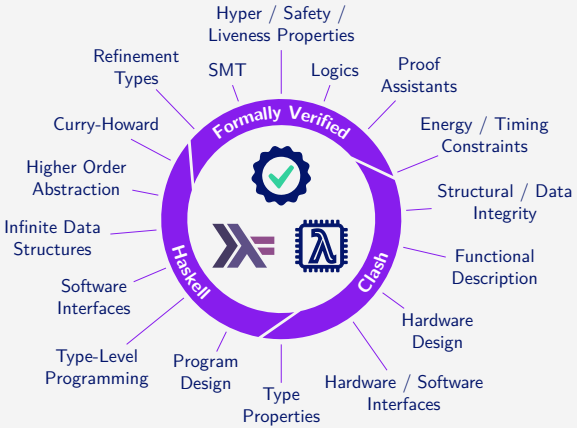
A modern, functional, open source hardware description language

https://clash-lang.org



Clash Formal

Ecosystem formally verifiable IT



➢ Proof Assistants

Agda ROCQ LEAN

➢ IFC / SMT / Refinement Types

- `seclib`: static information-flow security
- Liquid Haskell

➢ Temporal (Hyper-)Properties

- (Hyper) Temporal Stream Logic
- Functional Reative Programming

➢ Specification Frameworks

Sail Sail RISC-V CHERI

Clash Formal

- **Research Questions (50%):**
  - To which extend can we reuse the existing solutions for software verification in Haskell to also verify hardware designs in Clash?
  - To which extend can we utilize the type system of Haskell to describe the properties to be verified?
  - Can the properties that are specificed this way be shared and used in a cooperative manner between the hardware and the software, running on that particular hardware in the end?
- **Development (50%):**
  - Extension of the existing Clash compiler / ecosystem with user friendly tools for the construction of formally verified designs (open source).
  - Smart Card Demonstrator (open source)

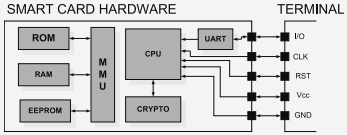
Smart Cards

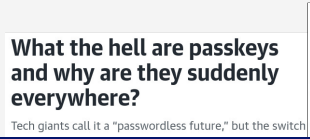
- dedicated hardware tokens for security relevant data management (passwords, identities, ...)
- hand-held mini-computer:
  CPU, memory, crypto core, custom operating system
- various applications & interfaces
- open standards:

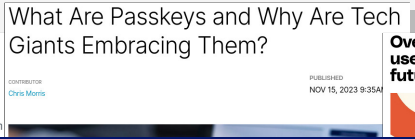
fido ALLIANCE

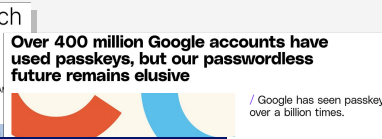
NIST NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY U.S. DEPARTMENT OF COMMERCE

- **Passkeys** will be the technology of the future!









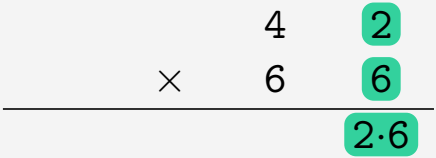
Karatsuba Multiplication in a Nutshell

$$\begin{array}{rcc} & 4 & 2 \\ \times & 6 & 6 \end{array}$$

Karatsuba Multiplication in a Nutshell

$$\begin{array}{rrr} & 4 & 2 \\ \times & 6 & 6 \\ \hline \end{array}$$

Karatsuba Multiplication in a Nutshell



Karatsuba Multiplication in a Nutshell

$$
\begin{array}{rcll}
 & & 4 & 2 \\
 & \times & 6 & 6 \\
\hline
 & & 4{\cdot}6 & 2{\cdot}6
\end{array}
$$

Karatsuba Multiplication in a Nutshell

$$
\begin{array}{rrrr}
 & & 4 & 2 \\
 & \times & 6 & 6 \\
\hline
 & & 4{\cdot}6 & 2{\cdot}6 \\
 & & 2{\cdot}6 &
\end{array}
$$

Karatsuba Multiplication in a Nutshell

$$
\begin{array}{rrll}
 & & 4 & 2 \\
 & \times & 6 & 6 \\
\hline
 & & 4{\cdot}6 & 2{\cdot}6 \\
 & 4{\cdot}6 & 2{\cdot}6 &
\end{array}
$$

Karatsuba Multiplication in a Nutshell

$$
\begin{array}{rrrr}
 & & 4 & 2 \\
 & \times & 6 & 6 \\
\hline
 & & 24 & 12 \\
 & 24 & 12 & \\
\end{array}
$$

Karatsuba Multiplication in a Nutshell

$$
\begin{array}{rrrr}
 & & 4 & 2 \\
 & \times & 6 & 6 \\
\hline
 & & 24 & 12 \\
 & 24 & 12 & \\
\hline
 & & & 2
\end{array}
$$

Karatsuba Multiplication in a Nutshell

$$
\begin{array}{rrrr}
 & & 4 & 2 \\
 & \times & 6 & 6 \\
\hline
 & & 24 & 12 \\
 & 24 & 12 & \\
 & & 1 & \\
\hline
 & & & 2
\end{array}
$$

Karatsuba Multiplication in a Nutshell

$$
\begin{array}{rrrr}
 & & 4 & 2 \\
 & \times & 6 & 6 \\
\hline
 & & 24 & 12 \\
 & 24 & 12 & \\
 & & 1 & \\
\hline
 & & 7 & 2
\end{array}
$$

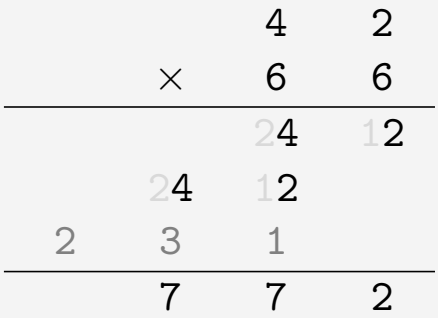
Karatsuba Multiplication in a Nutshell

$$
\begin{array}{rrr}
 & 4 & 2 \\
\times & 6 & 6 \\
\hline
 & 24 & 12 \\
24 & 12 & \\
3 & 1 & \\
\hline
 & 7 & 2
\end{array}
$$

Karatsuba Multiplication in a Nutshell

$$
\begin{array}{rrrr}
 & & 4 & 2 \\
 & \times & 6 & 6 \\
\hline
 & & 24 & 12 \\
 & 24 & 12 & \\
 & 3 & 1 & \\
\hline
 & 7 & 7 & 2
\end{array}
$$

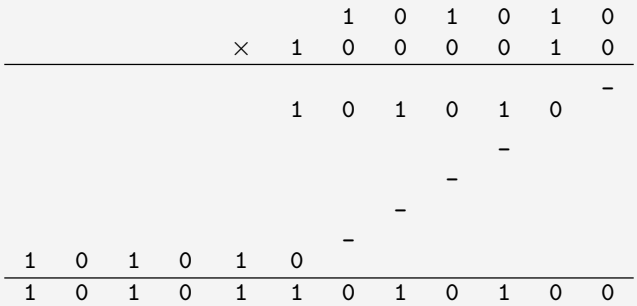
Karatsuba Multiplication in a Nutshell



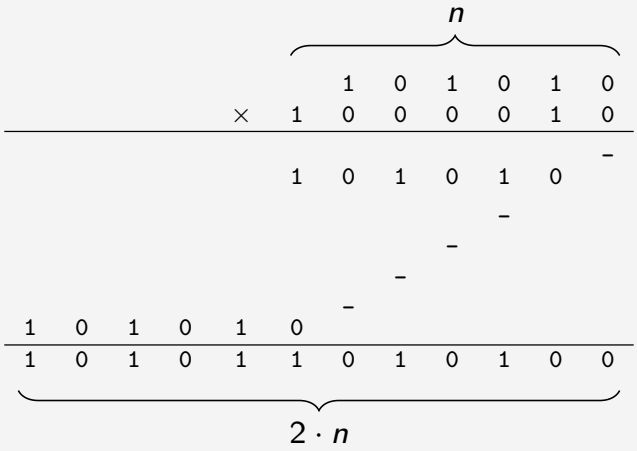
Karatsuba Multiplication in a Nutshell

$$
\begin{array}{cccc}
 & & 4 & 2 \\
 & \times & 6 & 6 \\
\hline
 & & 24 & 12 \\
 & 24 & 12 & \\
2 & 3 & 1 & \\
\hline
2 & 7 & 7 & 2
\end{array}
$$

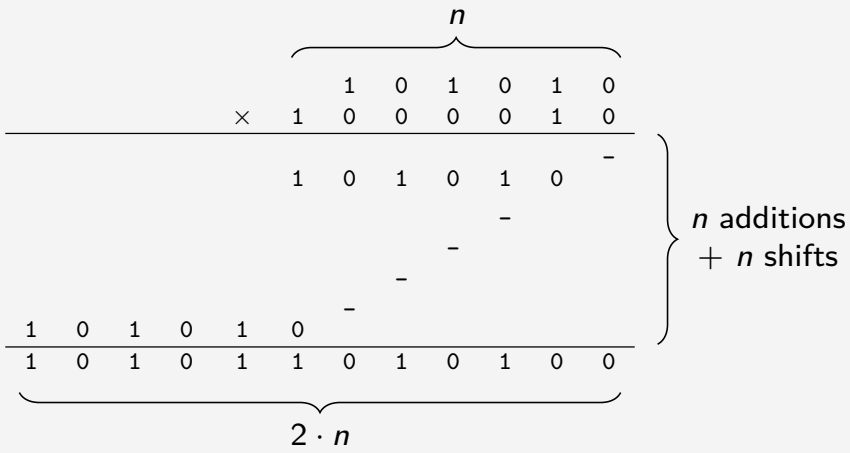
Karatsuba Multiplication in a Nutshell



Karatsuba Multiplication in a Nutshell

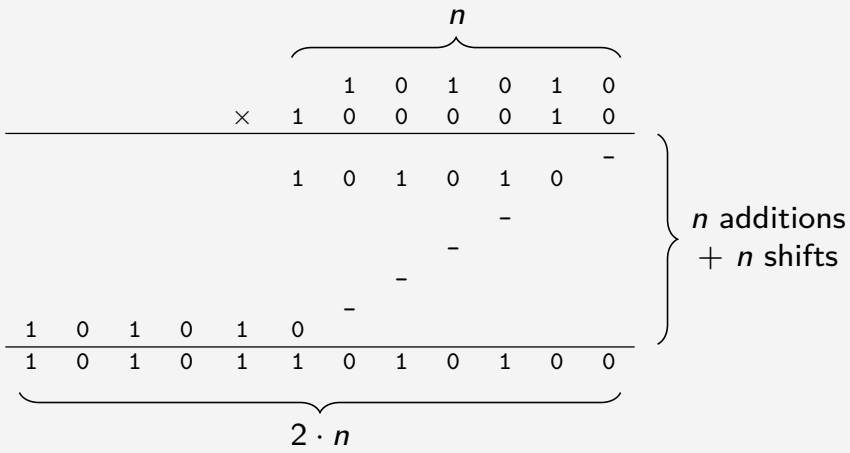


Karatsuba Multiplication in a Nutshell



Karatsuba Multiplication in a Nutshell

$\mathcal{O}(n^2)$



Karatsuba Multiplication in a Nutshell

$$0\ 1\ 0\ 1\ 0\ 1\ 0 \quad \times \quad 1\ 0\ 0\ 0\ 0\ 1\ 0$$

Karatsuba Multiplication in a Nutshell

$$\begin{pmatrix} & 0 & 1 & 0 & \_ & \_ & \_ & \_ \\ + & & & & 1 & 0 & 1 & 0 \end{pmatrix} \times \begin{pmatrix} & 1 & 0 & 0 & \_ & \_ & \_ & \_ \\ + & & & & 0 & 0 & 1 & 0 \end{pmatrix}$$

Karatsuba Multiplication in a Nutshell

$$\begin{pmatrix} & 0 & 1 & 0 & \_ & \_ & \_ & \_ \\ + & & & & 1 & 0 & 1 & 0 \end{pmatrix} \times \begin{pmatrix} & 1 & 0 & 0 & \_ & \_ & \_ & \_ \\ + & & & & 0 & 0 & 1 & 0 \end{pmatrix}$$

$$\begin{array}{ll} & (010 \times 100)\_\_\_\_\_\_\_\_ \\ + & \quad (1010 \times 100)\_\_\_\_ \\ + & \quad (010 \times 0010)\_\_\_\_ \\ + & \quad\quad (1010 \times 0010) \end{array}$$

Karatsuba Multiplication in a Nutshell

$$\begin{pmatrix} & 0 & 1 & 0 & \_ & \_ & \_ & \_ \\ + & & & & 1 & 0 & 1 & 0 \end{pmatrix} \times \begin{pmatrix} & 1 & 0 & 0 & \_ & \_ & \_ & \_ \\ + & & & & & 0 & 0 & 1 & 0 \end{pmatrix}$$

$$\begin{aligned} & (010 \times 100)\_\_\_\_\_\_\_ \\ + \quad & (1010 \times 100)\_\_\_\_ \\ + \quad & (010 \times 0010)\_\_\_\_ \\ + \quad & (1010 \times 0010) \end{aligned}$$

Multiply x and y:

$$\begin{aligned} & (x_{\mathtt{hi}} \times y_{\mathtt{hi}}) \cdot 2^{n} \\ + \quad & (x_{\mathtt{hi}} \times y_{\mathtt{lo}}) \cdot 2^{\frac{n}{2}} \\ + \quad & (x_{\mathtt{lo}} \times y_{\mathtt{hi}}) \cdot 2^{\frac{n}{2}} \\ + \quad & (x_{\mathtt{lo}} \times y_{\mathtt{lo}}) \end{aligned}$$

Karatsuba Multiplication in a Nutshell

$$(x_{\mathtt{lo}} + x_{\mathtt{hi}}) \cdot (y_{\mathtt{lo}} + y_{\mathtt{hi}}) = x_{\mathtt{lo}} \cdot y_{\mathtt{lo}} + x_{\mathtt{lo}} \cdot y_{\mathtt{hi}} + x_{\mathtt{hi}} \cdot y_{\mathtt{lo}} + x_{\mathtt{hi}} \cdot y_{\mathtt{hi}}$$

$$\begin{array}{ll} & (010 \times 100)\_\_\_\_\_ \\ + & \quad (1010 \times 100)\_\_\_\_ \\ + & \quad (010 \times 0010)\_\_\_\_ \\ + & \quad\quad (1010 \times 0010) \end{array}$$

Multiply x and y:

$$\begin{array}{ll} & (x_{\mathtt{hi}} \times y_{\mathtt{hi}}) \cdot 2^{n} \\ + & (x_{\mathtt{hi}} \times y_{\mathtt{lo}}) \cdot 2^{\frac{n}{2}} \\ + & (x_{\mathtt{lo}} \times y_{\mathtt{hi}}) \cdot 2^{\frac{n}{2}} \\ + & (x_{\mathtt{lo}} \times y_{\mathtt{lo}}) \end{array}$$

Karatsuba Multiplication in a Nutshell

$$(x_{\mathtt{lo}} + x_{\mathtt{hi}}) \cdot (y_{\mathtt{lo}} + y_{\mathtt{hi}}) \;=\; x_{\mathtt{lo}} \cdot y_{\mathtt{lo}} + x_{\mathtt{lo}} \cdot y_{\mathtt{hi}} + x_{\mathtt{hi}} \cdot y_{\mathtt{lo}} + x_{\mathtt{hi}} \cdot y_{\mathtt{hi}}$$

$$\begin{array}{ll} & (010 \times 100)\_\_\_\_\_\_\_\_ \\ + & \quad (1010 \times 100)\_\_\_\_ \\ + & \quad (010 \times 0010)\_\_\_\_ \\ + & \quad\quad (1010 \times 0010) \end{array}$$

Multiply x and y:

$$\begin{array}{ll} & (x_{\mathtt{hi}} \times y_{\mathtt{hi}}) \cdot 2^{n} \\ + & (x_{\mathtt{hi}} \times y_{\mathtt{lo}}) \cdot 2^{\frac{n}{2}} \\ + & (x_{\mathtt{lo}} \times y_{\mathtt{hi}}) \cdot 2^{\frac{n}{2}} \\ + & (x_{\mathtt{lo}} \times y_{\mathtt{lo}}) \end{array}$$

Karatsuba Multiplication in a Nutshell

$$(x_{\mathtt{lo}} + x_{\mathtt{hi}}) \cdot (y_{\mathtt{lo}} + y_{\mathtt{hi}}) \; - \; x_{\mathtt{lo}} \cdot y_{\mathtt{lo}} \; - \; x_{\mathtt{hi}} \cdot y_{\mathtt{hi}} \; = \; x_{\mathtt{lo}} \cdot y_{\mathtt{hi}} \; + \; x_{\mathtt{hi}} \cdot y_{\mathtt{lo}}$$

$$\begin{array}{ll} & (010 \times 100)\_\_\_\_\_\_\_\_ \\ + & \quad (1010 \times 100)\_\_\_\_ \\ + & \quad (010 \times 0010)\_\_\_\_ \\ + & \quad\quad (1010 \times 0010) \end{array}$$

Multiply x and y:

$$\begin{array}{ll} & (x_{\mathtt{hi}} \times y_{\mathtt{hi}}) \cdot 2^{n} \\ + & (x_{\mathtt{hi}} \times y_{\mathtt{lo}}) \cdot 2^{\frac{n}{2}} \\ + & (x_{\mathtt{lo}} \times y_{\mathtt{hi}}) \cdot 2^{\frac{n}{2}} \\ + & (x_{\mathtt{lo}} \times y_{\mathtt{lo}}) \end{array}$$

Karatsuba Multiplication in a Nutshell

$$(x_{\mathtt{lo}} + x_{\mathtt{hi}}) \cdot (y_{\mathtt{lo}} + y_{\mathtt{hi}}) \; - \; x_{\mathtt{lo}} \cdot y_{\mathtt{lo}} \; - \; x_{\mathtt{hi}} \cdot y_{\mathtt{hi}} \; = \; x_{\mathtt{lo}} \cdot y_{\mathtt{hi}} \; + \; x_{\mathtt{hi}} \cdot y_{\mathtt{lo}}$$

$$\begin{array}{rr} & (010 \times 100)\_\_\_\_\_\_\_\_ \\ + & \quad (1010 \times 100)\_\_\_\_ \\ + & \quad (010 \times 0010)\_\_\_\_ \\ + & \quad\quad (1010 \times 0010) \end{array}$$

Multiply x and y:

$$\begin{array}{rr} & (x_{\mathtt{hi}} \times y_{\mathtt{hi}}) \cdot 2^{n} \\ + & (x_{\mathtt{lo}} + x_{\mathtt{hi}}) \times (y_{\mathtt{lo}} + y_{\mathtt{hi}}) \cdot 2^{\frac{n}{2}} \\ - & (x_{\mathtt{lo}} \times y_{\mathtt{lo}}) \cdot 2^{\frac{n}{2}} \\ - & (x_{\mathtt{hi}} \times y_{\mathtt{hi}}) \cdot 2^{\frac{n}{2}} \\ + & (x_{\mathtt{lo}} \times y_{\mathtt{lo}}) \end{array}$$

Karatsuba Multiplication in a Nutshell

$$(x_{\mathtt{lo}} + x_{\mathtt{hi}}) \cdot (y_{\mathtt{lo}} + y_{\mathtt{hi}}) \; - \; x_{\mathtt{lo}} \cdot y_{\mathtt{lo}} \; - \; x_{\mathtt{hi}} \cdot y_{\mathtt{hi}} \; = \; x_{\mathtt{lo}} \cdot y_{\mathtt{hi}} \; + \; x_{\mathtt{hi}} \cdot y_{\mathtt{lo}}$$

$$\begin{array}{ll} & (010 \times 100)\_\_\_\_\_ \\ + & \quad (1010 \times 100)\_\_\_\_ \\ + & \quad (010 \times 0010)\_\_\_\_ \\ + & \quad\quad (1010 \times 0010) \end{array}$$

Multiply x and y:

$$\begin{array}{lr} & (x_{\mathtt{hi}} \times y_{\mathtt{hi}}) \cdot 2^{n} \\ + & (x_{\mathtt{lo}} + x_{\mathtt{hi}}) \times (y_{\mathtt{lo}} + y_{\mathtt{hi}}) \cdot 2^{\frac{n}{2}} \\ - & (x_{\mathtt{lo}} \times y_{\mathtt{lo}}) \cdot 2^{\frac{n}{2}} \\ - & (x_{\mathtt{hi}} \times y_{\mathtt{hi}}) \cdot 2^{\frac{n}{2}} \\ + & (x_{\mathtt{lo}} \times y_{\mathtt{lo}}) \end{array}$$

Karatsuba Multiplication in a Nutshell

$\mathcal{O}(n^{1.59})$

$$(x_{\mathtt{lo}} + x_{\mathtt{hi}}) \cdot (y_{\mathtt{lo}} + y_{\mathtt{hi}}) \; - \; x_{\mathtt{lo}} \cdot y_{\mathtt{lo}} \; - \; x_{\mathtt{hi}} \cdot y_{\mathtt{hi}} \; = \; x_{\mathtt{lo}} \cdot y_{\mathtt{hi}} \; + \; x_{\mathtt{hi}} \cdot y_{\mathtt{lo}}$$

$$\begin{array}{rr} & (010 \times 100)\_\_\_\_\_ \\ + & \quad (1010 \times 100)\_\_\_\_ \\ + & \quad (010 \times 0010)\_\_\_\_ \\ + & \quad\quad (1010 \times 0010) \end{array}$$

Multiply x and y:

$$\begin{array}{rr} & (x_{\mathtt{hi}} \times y_{\mathtt{hi}}) \cdot 2^{n} \\ + & (x_{\mathtt{lo}} + x_{\mathtt{hi}}) \times (y_{\mathtt{lo}} + y_{\mathtt{hi}}) \cdot 2^{\frac{n}{2}} \\ - & (x_{\mathtt{lo}} \times y_{\mathtt{lo}}) \cdot 2^{\frac{n}{2}} \\ - & (x_{\mathtt{hi}} \times y_{\mathtt{hi}}) \cdot 2^{\frac{n}{2}} \\ + & (x_{\mathtt{lo}} \times y_{\mathtt{lo}}) \end{array}$$

```
karatsuba ::

  Unsigned n → Unsigned m → Unsigned (n + m)
karatsuba
```

Multiply x and y:

$$
\begin{array}{ll}
 & (x_{\mathtt{hi}} \times y_{\mathtt{hi}}) \cdot 2^{n} \\
+ & (x_{\mathtt{lo}} + x_{\mathtt{hi}}) \times (y_{\mathtt{lo}} + y_{\mathtt{hi}}) \cdot 2^{\frac{n}{2}} \\
- & (x_{\mathtt{lo}} \times y_{\mathtt{lo}}) \cdot 2^{\frac{n}{2}} \\
- & (x_{\mathtt{hi}} \times y_{\mathtt{hi}}) \cdot 2^{\frac{n}{2}} \\
+ & (x_{\mathtt{lo}} \times y_{\mathtt{lo}})
\end{array}
$$

```
karatsuba ::

  Unsigned n → Unsigned m → Unsigned (n + m)
karatsuba
```

```
type Low  n = n `Div` 2
type High n = n - n `Div` 2
```

Multiply x and y:

$$
\begin{array}{ll}
 & (x_{\mathtt{hi}} \times y_{\mathtt{hi}}) \cdot 2^{n} \\
+ & (x_{\mathtt{lo}} + x_{\mathtt{hi}}) \times (y_{\mathtt{lo}} + y_{\mathtt{hi}}) \cdot 2^{\frac{n}{2}} \\
- & (x_{\mathtt{lo}} \times y_{\mathtt{lo}}) \cdot 2^{\frac{n}{2}} \\
- & (x_{\mathtt{hi}} \times y_{\mathtt{hi}}) \cdot 2^{\frac{n}{2}} \\
+ & (x_{\mathtt{lo}} \times y_{\mathtt{lo}})
\end{array}
$$

```
karatsuba ::

  Unsigned n → Unsigned m → Unsigned (n + m)
karatsuba
  | SNat :: SNat s ← SNat @(Max n m)
```

```
type Low  n = n `Div` 2
type High n = n - n `Div` 2
```

Multiply x and y:

$$
\begin{aligned}
& (x_{\mathrm{hi}} \times y_{\mathrm{hi}}) \cdot 2^{n} \\
+\ & (x_{\mathrm{lo}} + x_{\mathrm{hi}}) \times (y_{\mathrm{lo}} + y_{\mathrm{hi}}) \cdot 2^{\frac{n}{2}} \\
-\ & (x_{\mathrm{lo}} \times y_{\mathrm{lo}}) \cdot 2^{\frac{n}{2}} \\
-\ & (x_{\mathrm{hi}} \times y_{\mathrm{hi}}) \cdot 2^{\frac{n}{2}} \\
+\ & (x_{\mathrm{lo}} \times y_{\mathrm{lo}})
\end{aligned}
$$

```
karatsuba ::

  Unsigned n → Unsigned m → Unsigned (n + m)
karatsuba
  | SNat :: SNat s ← SNat @(Max n m)



 where
  xlo , ylo :: Unsigned (Low s)
  xhi , yhi :: Unsigned (High s)
  (xhi , xlo) = bitCoerce (resize x)
  (yhi , ylo) = bitCoerce (resize y)
```

```
type Low  n = n `Div` 2
type High n = n - n `Div` 2
```

Multiply x and y:

$$
\begin{aligned}
& (x_{\mathrm{hi}} \times y_{\mathrm{hi}}) \cdot 2^{n} \\
+\ & (x_{\mathrm{lo}} + x_{\mathrm{hi}}) \times (y_{\mathrm{lo}} + y_{\mathrm{hi}}) \cdot 2^{\frac{n}{2}} \\
-\ & (x_{\mathrm{lo}} \times y_{\mathrm{lo}}) \cdot 2^{\frac{n}{2}} \\
-\ & (x_{\mathrm{hi}} \times y_{\mathrm{hi}}) \cdot 2^{\frac{n}{2}} \\
+\ & (x_{\mathrm{lo}} \times y_{\mathrm{lo}})
\end{aligned}
$$

```
karatsuba ::

  Unsigned n → Unsigned m → Unsigned (n + m)
karatsuba
  | SNat :: SNat s ← SNat @(Max n m)



 where
  xlo , ylo :: Unsigned (Low s)
  xhi , yhi :: Unsigned (High s)
  (xhi , xlo) = bitCoerce (resize x)
  (yhi , ylo) = bitCoerce (resize y)

  xs , ys :: Unsigned (High s + 1)
  xs = resize xlo + resize xhi
  ys = resize ylo + resize yhi
```

```
type Low  n = n `Div` 2
type High n = n - n `Div` 2
```

Multiply x and y:

$$
\begin{aligned}
& (x_{\mathrm{hi}} \times y_{\mathrm{hi}}) \cdot 2^{n} \\
+\ & (x_{\mathrm{lo}} + x_{\mathrm{hi}}) \times (y_{\mathrm{lo}} + y_{\mathrm{hi}}) \cdot 2^{\frac{n}{2}} \\
-\ & (x_{\mathrm{lo}} \times y_{\mathrm{lo}}) \cdot 2^{\frac{n}{2}} \\
-\ & (x_{\mathrm{hi}} \times y_{\mathrm{hi}}) \cdot 2^{\frac{n}{2}} \\
+\ & (x_{\mathrm{lo}} \times y_{\mathrm{lo}})
\end{aligned}
$$

```
karatsuba ::

  Unsigned n → Unsigned m → Unsigned (n + m)
karatsuba
  | SNat :: SNat s ← SNat @(Max n m)
```

```
type Low  n = n `Div` 2
type High n = n - n `Div` 2
```

```
 where
  x_lo , y_lo :: Unsigned (Low s)
  x_hi , y_hi :: Unsigned (High s)
  (x_hi , x_lo) = bitCoerce (resize x)
  (y_hi , y_lo) = bitCoerce (resize y)

  x_s , y_s :: Unsigned (High s + 1)
  x_s = resize x_lo + resize x_hi
  y_s = resize y_lo + resize y_hi

  z_0 , z_1 , z_2 :: Unsigned (n + m)
  z_0 = resize (karatsuba x_lo y_lo)
  z_1 = resize (karatsuba x_s y_s) - z_0 - z_2
  z_2 = resize (karatsuba x_hi y_hi)
```

Multiply x and y:

$$
\begin{aligned}
& (x_{\mathrm{hi}} \times y_{\mathrm{hi}}) \cdot 2^{n} \\
+\ & (x_{\mathrm{lo}} + x_{\mathrm{hi}}) \times (y_{\mathrm{lo}} + y_{\mathrm{hi}}) \cdot 2^{\frac{n}{2}} \\
-\ & (x_{\mathrm{lo}} \times y_{\mathrm{lo}}) \cdot 2^{\frac{n}{2}} \\
-\ & (x_{\mathrm{hi}} \times y_{\mathrm{hi}}) \cdot 2^{\frac{n}{2}} \\
+\ & (x_{\mathrm{lo}} \times y_{\mathrm{lo}})
\end{aligned}
$$

```
karatsuba ::

  Unsigned n → Unsigned m → Unsigned (n + m)
karatsuba
  | SNat :: SNat s ← SNat @(Max n m)
  =

                  z2 ≪ SNat @(2 * Low s) + z1 ≪ SNat @(Low s) + z0
 where
  xlo , ylo :: Unsigned (Low s)
  xhi , yhi :: Unsigned (High s)
  (xhi , xlo) = bitCoerce (resize x)
  (yhi , ylo) = bitCoerce (resize y)

  xs , ys :: Unsigned (High s + 1)
  xs = resize xlo + resize xhi
  ys = resize ylo + resize yhi

  z0 , z1 , z2 :: Unsigned (n + m)
  z0 = resize (karatsuba xlo ylo)
  z1 = resize (karatsuba xs ys) - z0 - z2
  z2 = resize (karatsuba xhi yhi)
```

```
type Low  n = n `Div` 2
type High n = n - n `Div` 2
```

Multiply x and y:

$$
\begin{aligned}
& (x_{\mathrm{hi}} \times y_{\mathrm{hi}}) \cdot 2^{n} \\
+\ & (x_{\mathrm{lo}} + x_{\mathrm{hi}}) \times (y_{\mathrm{lo}} + y_{\mathrm{hi}}) \cdot 2^{\frac{n}{2}} \\
-\ & (x_{\mathrm{lo}} \times y_{\mathrm{lo}}) \cdot 2^{\frac{n}{2}} \\
-\ & (x_{\mathrm{hi}} \times y_{\mathrm{hi}}) \cdot 2^{\frac{n}{2}} \\
+\ & (x_{\mathrm{lo}} \times y_{\mathrm{lo}})
\end{aligned}
$$

```
karatsuba ::
  ∀ k n m. (KnownNat n, KnownNat m) ⇒ SNat k →
  Unsigned n → Unsigned m → Unsigned (n + m)
karatsuba k@SNat x y
  | SNat :: SNat s ← SNat @(Max n m)
  =
```

$$z_2 \ll \texttt{SNat @(2 * Low s)} + z_1 \ll \texttt{SNat @(Low s)} + z_0$$

```
 where
  x_lo , y_lo :: Unsigned (Low s)
  x_hi , y_hi :: Unsigned (High s)
  (x_hi , x_lo) = bitCoerce (resize x)
  (y_hi , y_lo) = bitCoerce (resize y)

  x_s , y_s :: Unsigned (High s + 1)
  x_s = resize x_lo + resize x_hi
  y_s = resize y_lo + resize y_hi

  z_0 , z_1 , z_2 :: Unsigned (n + m)
  z_0 = resize (karatsuba k x_lo y_lo)
  z_1 = resize (karatsuba k x_s y_s) - z_0 - z_2
  z_2 = resize (karatsuba k x_hi y_hi)
```

```
type Low  n = n `Div` 2
type High n = n - n `Div` 2
```

Multiply x and y:

$$\begin{aligned}
&(x_{\text{hi}} \times y_{\text{hi}}) \cdot 2^{n} \\
+\ &(x_{\text{lo}} + x_{\text{hi}}) \times (y_{\text{lo}} + y_{\text{hi}}) \cdot 2^{\frac{n}{2}} \\
-\ &(x_{\text{lo}} \times y_{\text{lo}}) \cdot 2^{\frac{n}{2}} \\
-\ &(x_{\text{hi}} \times y_{\text{hi}}) \cdot 2^{\frac{n}{2}} \\
+\ &(x_{\text{lo}} \times y_{\text{lo}})
\end{aligned}$$

```
karatsuba ::
  ∀ k n m. (KnownNat n, KnownNat m) ⇒ SNat k →
  Unsigned n → Unsigned m → Unsigned (n + m)
karatsuba k@SNat x y
  | SNat :: SNat s ← SNat @(Max n m)
  = case compareSNat (SNat @(n + m)) k of
      SNatLE → resize x * resize y
      SNatGT → z2 ≪ SNat @(2 * Low s) + z1 ≪ SNat @(Low s) + z0
 where
  xlo , ylo :: Unsigned (Low s)
  xhi , yhi :: Unsigned (High s)
  (xhi , xlo) = bitCoerce (resize x)
  (yhi , ylo) = bitCoerce (resize y)

  xs , ys :: Unsigned (High s + 1)
  xs = resize xlo + resize xhi
  ys = resize ylo + resize yhi

  z0 , z1 , z2 :: Unsigned (n + m)
  z0 = resize (karatsuba k xlo ylo)
  z1 = resize (karatsuba k xs ys) - z0 - z2
  z2 = resize (karatsuba k xhi yhi)
```

```
type Low  n = n `Div` 2
type High n = n - n `Div` 2
```

Multiply x and y:

$$
\begin{aligned}
& (x_{\mathrm{hi}} \times y_{\mathrm{hi}}) \cdot 2^{n} \\
+\ & (x_{\mathrm{lo}} + x_{\mathrm{hi}}) \times (y_{\mathrm{lo}} + y_{\mathrm{hi}}) \cdot 2^{\frac{n}{2}} \\
-\ & (x_{\mathrm{lo}} \times y_{\mathrm{lo}}) \cdot 2^{\frac{n}{2}} \\
-\ & (x_{\mathrm{hi}} \times y_{\mathrm{hi}}) \cdot 2^{\frac{n}{2}} \\
+\ & (x_{\mathrm{lo}} \times y_{\mathrm{lo}})
\end{aligned}
$$

```
karatsuba ::
  ∀ k n m. (KnownNat n, KnownNat m) ⇒ SNat k →
  Unsigned n → Unsigned m → Unsigned (n + m)
karatsuba k@SNat x y
  | SNat :: SNat s ← SNat @(Max n m)
  = case compareSNat (SNat @(n + m)) k of
      SNatLE → resize x * resize y
      SNatGT → z2 ≪ SNat @(2 * Low s) + z1 ≪ SNat @(Low s) + z0
 where
  xlo , ylo :: Unsigned (Low s)
  xhi , yhi :: Unsigned (High s)
  (xhi , xlo) = bitCoerce (resize x)
  (yhi , ylo) = bitCoerce (resize y)

  xs , ys :: Unsigned (High s + 1)
  xs = resize xlo + resize xhi
  ys = resize ylo + resize yhi

  z0 , z1 , z2 :: Unsigned (n + m)
  z0 = resize (karatsuba k xlo ylo)
  z1 = resize (karatsuba k xs  ys ) - z0 - z2
  z2 = resize (karatsuba k xhi yhi)
```

```
type Low  n = n `Div` 2
type High n = n - n `Div` 2
```

n	(*) @(Unsigned n) *[# gates]*	karatsuba @8 @n @n *[# gates]*
8	**569**	993
16	**2536**	4118
32	**10437**	14898
64	**41924**	49220
128	166252	**156578**
256	661993	**487583**
512	2638547	**1496300**

$\mathcal{O}(n^{1.59})$

```
karatsuba ::
  ∀ k n m. (KnownNat n, KnownNat m) ⇒ SNat k →
  Unsigned n → Unsigned m → Unsigned (n + m)
karatsuba k@SNat x y
  | SNat :: SNat s ← SNat @(Max n m)
  = case compareSNat (SNat @(n + m)) k of
      SNatLE → resize x * resize y
      SNatGT → z2 ≪ SNat @(2 * Low s) + z1 ≪ SNat @(Low s) + z0
 where
  xlo , ylo :: Unsigned (Low s)
  xhi , yhi :: Unsigned (High s)
  (xhi , xlo) = bitCoerce (resize x)
  (yhi , ylo) = bitCoerce (resize y)

  xs , ys :: Unsigned (High s + 1)
  xs = resize xlo + resize xhi
  ys = resize ylo + resize yhi

  z0 , z1 , z2 :: Unsigned (n + m)
  z0 = resize (karatsuba k xlo ylo)
  z1 = resize (karatsuba k xs ys) - z0 - z2
  z2 = resize (karatsuba k xhi yhi)
```

```
type Low  n = n `Div` 2
type High n = n - n `Div` 2
```

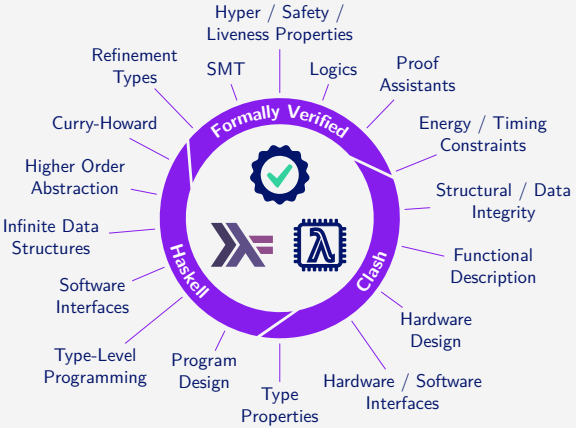
n	(*) @(Unsigned n) *[# gates]*	karatsuba @8 @n @n *[# gates]*
8	**569**	993
16	**2536**	4118
32	**10437**	14898
64	**41924**	49220
128	166252	**156578**
256	661993	**487583**
512	2638547	**1496300**

ROCQ ✓

$\mathcal{O}(n^{1.59})$

Clash Formal

Ecosystem formally verifiable IT



Proof Assistants

Agda ROCQ LEAN

SMT / Refinement Types

Liquid Haskell

Specification Frameworks

Sail Sail RISC-V CHERI

https://clash-formal.org