Modern hardware development with *Clash*

Jan Kuper

jan.kuper@qbaylogic.com

WoTS, Utrecht, 27-30 sept 2022

QBayLogic.

Overview

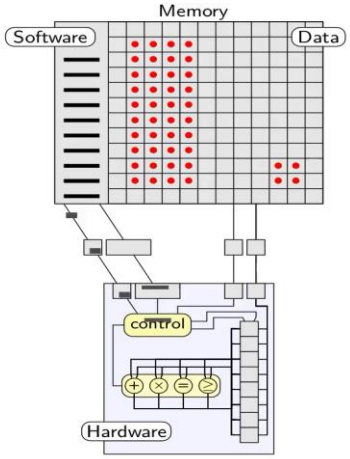
- Power of FPGAs
- Programming methodologies
- Open source: *Clash*
- Example: IIR
- Processor design
- Features

Power of FPGAs

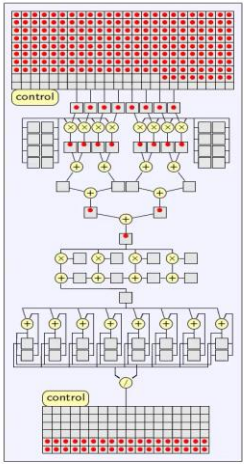
- When software is not fast enough
- High throughput
- Low latency
- Low energy consumption

- Accelerators
- Dedicated processors
- Network switches, communication

CPU vs FPGA



Normal processor
Software controls computation process



FPGA
Computation integrated in hardware

Programmability

HLS

- Well known languages (C/C++, Matlab, Simulink)
- Different style of thinking
- Black box translation
- No control over performance

Specification

Definition

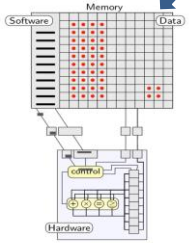
Model

VHDL / Verilog

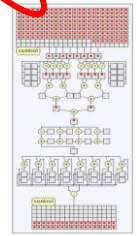
- Poor abstraction mechanisms
- Cumbersome simulation
- Error prone
- Time consuming, expensive

Clash

- One language
- Model based
- Strong abstractions
- Simulation
- Control over design
- Generates VHDL/Verilog



Normal processor


FPGA

Example: IIR-filter

Medical application; Requirements (a.o.):

- 250 sensors, 0.5M samples/second
- Floating Point
- FPGA: 300MHz
- Number of arithmetical operators minimal

HLS failed ...

Example: IIR-filter

Medical application; Requirements (a.o.):

- 250 sensors, 0.5M samples/second
- Floating Point
- FPGA: 300MHz
- Number of arithmetical operators minimal

HLS failed ...

Results

	Number of operators	Pipeline stages
Multiplier	1	8
Adder	1	11

Taps IIR	Cycles
6	49
10	61
20	78

Freq: 550MHz

IIR: Formal model

IIR: Formal model

IIR: Formal model

$$y_n = \frac{1}{a_0}\left(\sum_{i=0}^{N} b_i x_{n-i} - \sum_{j=1}^{M} a_j y_{n-j}\right)$$

$$= c\left(0 \oplus b_{0\cdots N} \mathbin{\hat{*}} x_{n\cdots n-N} - 0 \oplus a_{0\cdots M-1} \mathbin{\hat{*}} y_{n-1\cdots n-M}\right)$$

- Word-for-word translation
- Haskell = Math
- Executable

```
yA n  | n < 0      = 0
      | otherwise = c * ( foldl (+) 0 (zipWith (*) (b&[0..nn])   (x&[n,n-1..n-nn]))
                        - foldl (+) 0 (zipWith (*) (a&[0..mm-1]) (yA&[n-1,n-2..n-mm])))
                      )
```

IIR: Formal transformations

- Parameter accumulation
- Recursor

➔ Architecture

IIR: Architecture

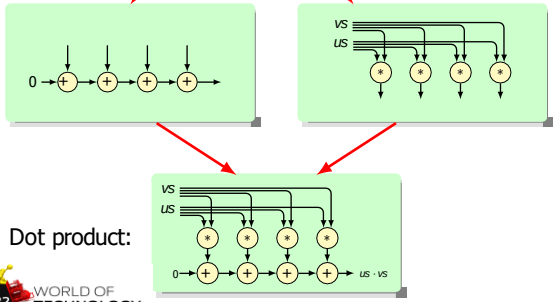
```
yC (xs,ys) x = ( (xs',ys') , y )
      where
        y    = c * (bs·xs - as·ys)
        xs' = x +>> xs
        ys' = y +>> ys
```

```
us · vs = foldl (+) 0 (zipWith (*) us vs)
```

IIR: Architecture

```
yC (xs,ys) x = ( (xs',ys') , y )
      where
        y    = c * (bs·xs - as·ys)
        xs'  = x +>> xs
        ys'  = y +>> ys
```

```
us · vs = foldl (+) 0 (zipWith (*) us vs)
```

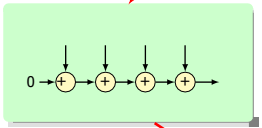


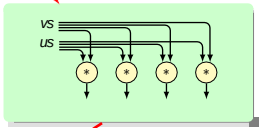
Dot product:

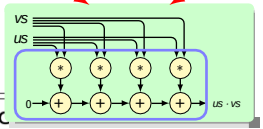
IIR: Architecture

```
yC (xs,ys) x = ( (xs',ys') , y )
      where
        y   = c * (bs·xs - as·ys)
        xs' = x +>> xs
        ys' = y +>> ys
```

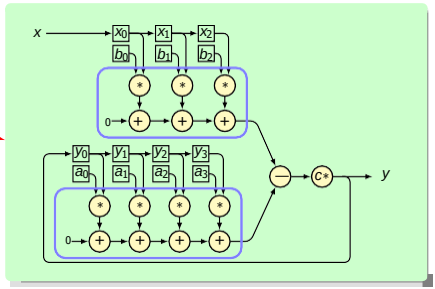
```
us · vs = foldl (+) 0 (zipWith (*) us vs)
```





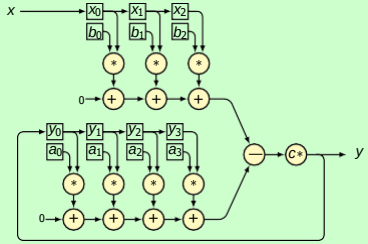


Dot product:



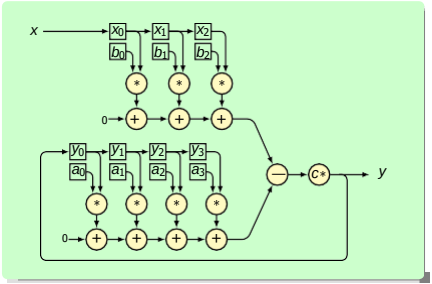
Optimisations needed

IIR: Linearisation

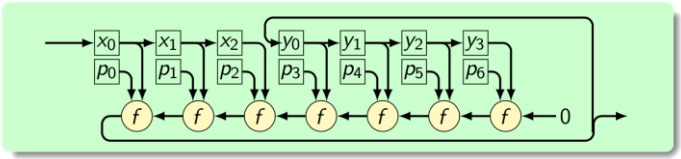


$$y = c\,(bs \cdot xs - as \cdot ys)$$

IIR: Linearisation



$$
\begin{aligned}
y &= c\,(bs \cdot xs \;-\; as \cdot ys) \\
&= c\,((bs \mathbin{+\!\!+} -as) \cdot (xs \mathbin{+\!\!+} ys)) \\
&= (c\,(bs \mathbin{+\!\!+} -as)) \cdot (xs \mathbin{+\!\!+} ys) \\
&= ps \cdot xys \\
&= \mathbf{foldl}\ (+)\ 0\ (\mathbf{zipWith}\ (*)\ ps\ xys) \\
&= \mathbf{foldl}\ ((+)\ a\ (*))\ 0\ pxys \\
&= \mathbf{foldl}\ f\ 0\ pxys \\
&= \mathbf{foldr}\ f\ 0\ pxys
\end{aligned}
$$



Some further design steps

- Sequentialisation
- Pipelining
- State machine
- …

➔All within the same language

➔All provably correct

➔Model driven design

Processor design

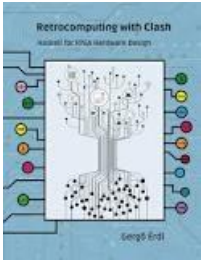
- Embedded languages
- Formal semantics of instruction set
- Automatic generation of bit representations

Again:
➔One language
➔Provably correct

Tool support

- Type checking
  - Numbers, functions, etc
  - Synchronisation
  - Clock domain crossings
- Interactive simulation (REPL), cycle accurate
- Random test generation
- Minimal debugging case

Book



Dr. Gergö Érdi: *Retrocomputing with Clash – Haskell for FPGA Hardware Design*, https://gergo.erdi.hu/retroclash/, December 2021

Hal 9: Development Club / IOT Paviljoen

Thank you