Principles of Reversible Computation

Robert Glück
DIKU, University of Copenhagen, Denmark

joint work with
Holger Bock Axelsen, Stéphane Burignat
Alex De Vos, Torben Mogensen
Michael Kirkedal Thomsen, Tetsuo Yokoyama
(Denmark, Belgium, Japan)

Computation is Physical

- **Computation is a physical process.**
  - paper and pencil
  - electronic device
  - biological system (such as the brain)
- **Any computation is subject to physical laws.**
  - Physical laws govern all information processing, *no matter how* it is accomplished.
- **Every logical state of a computing device must correspond to a physical state.**

Thermodynamical Minumum for Irreversible Computation

He then computed the thermodynamical minimum of energy per elementary act of information from the formula $kT \log_e N$ ergs, where k is Boltzmann's constant (1.4×10^{-16} ergs per degree), T is the temperature in absolute units, and N is the number of alternatives. For a binary act $N = 2$, and taking the temperature to be about 300 degrees absolute, he obtained 3×10^{-14} ergs for the thermodynamical minimum.

von Neumann 1949
in: Burks 1966, p.66

(1 erg = 10^{-7} joule)

Information Loss has a Physical Cost

- **Physical states cannot be destroyed,** so any bits that are **discarded logically turn into entropy.**
  - Consequence of basic thermodynam[...]
  - This irreversibility directly contribute[...] and therefore power consumption.

10^{-21} joules per erased bit at room temperature

Mismatch

- Our **best models of computation are irreversible**.
  - Carries over to actual implementations in computers.
- Good news: **Computation can be organized reversibly**.
  - Information destruction is *not* intrinsic to the concept of a computation process.

Reversible Computation (RC)

- Study of computing models deterministic in both the forward & backward directions.
- Adopts a physical principle of reality into a computing model without information loss.
- Sometimes RC a necessity, but also interesting in its own right – regardless of its physical motivation.

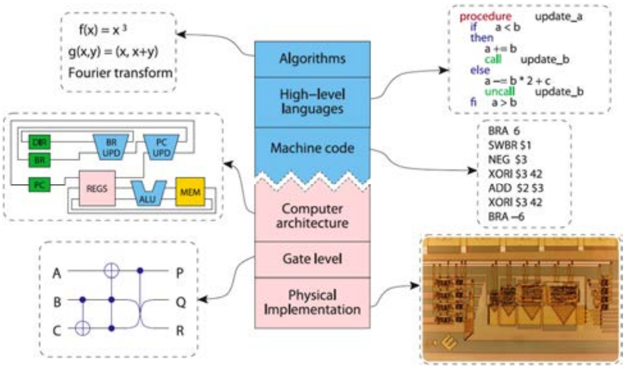
Reversible Languages

Languages:	fwd →	bwd ←	
• Logic languages	N	N	Prolog, ...
• Mainstream	D	N	C, ...
• Reversible languages	D	D	Janus, ...

State transitions:

- N ... non-deterministic
- D ... deterministic

Reversible Computing Systems

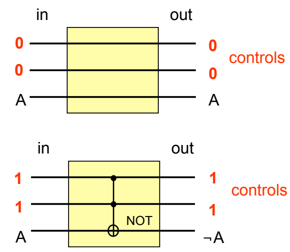


Reversible Logic Gates

Two essential properties:

1. The number of input lines is equal to the number of output lines (written $n \times n$).
2. Its Boolean function $B^n \rightarrow B^n$ is bijective.

Toffoli Gate [Toffoli '80]

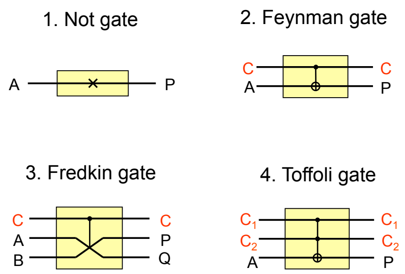


input			output		
C_1	C_2	A	C_1	C_2	P
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	1
1	1	1	1	1	0

Controlled-Controlled Not gate

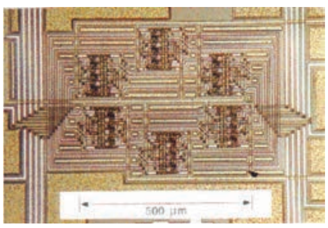
Universal: any rev. logic can be built by Toffoli gates

A Set of Reversible Logic Gates



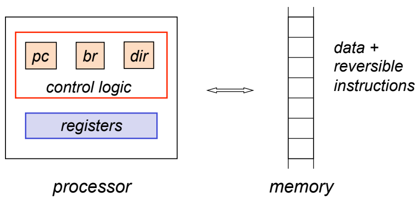
Realization of Reversible 4-Bit Adder

The circuit has *no power supply* and *no ground busbar*.
All energy provided to output pins comes from input pins.
Reversible gates used: Fredkin, Feynman, Toffoli.



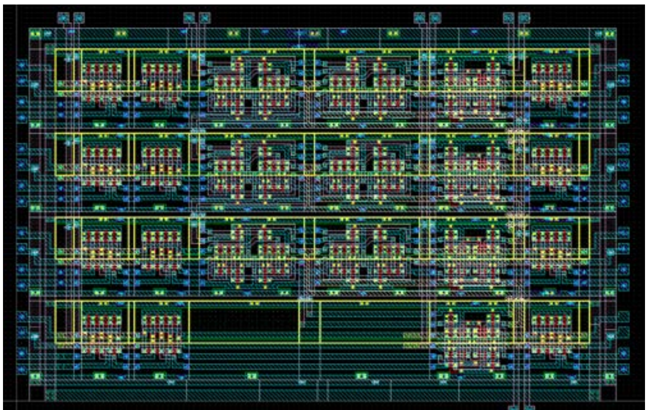
Dual-line pass-transistor logic [DeVos et al.]

Reversible Abstract Machine



Reversible von Neumann architecture

Shown reversible [AxelsenGlückYokoyama '07]



Fully Reversible Arithmetic-Logic Unit (ALU) [Thomsen et al.'10]

Gent's dual-line pass-transistors CMOS technology
256 transistors (0.35 μm), 117x77 μm size w/bonding pads

Computability

For all irreversible computing devices:

- Conventional (= irreversible) languages: equivalent to Turing-machine. [Turing 1936]

For all reversible computing devices:

- Reversible programming languages: equivalent to reversible Turing-machine. [Bennett 1974]
  = all computable <u>injective</u> functions

Two Main Steps – Example

1. **Specification**: **Injectivization** of non-injective **function**.

Fibonacci classic:	injectivize →	Fibonacci-pair injective:
fib(*n*)		[*fib*(*n*), *fib*(*n*+1)]

Recall: *fib*(1) = 1
fib(2) = 1

2. **Implementation**: in a **reversible programming language**.
   Algorithm design, programming, optimization, compilation.

[GlückKawabe'03]

Janus: a Reversible Language

- Imperative programming language
- Global store, no local store
- Scalar and array types, integer values
- Structured control-flow operators (IF, LOOP)
- Access to inverse semantics (UNCALL)
- Simple recursive procedures

[LutzDerby'82, YokoyamaGlück'07, YokoyamaAxelsenGlück'08]

Janus Example: Fibonacci-Pairs

```
n x1 x2
```

global store (integer variables)
initially zero

```
procedure fib
  if n=0 then x1 += 1
              x2 += 1
         else n -= 1
              call fib
              x1 += x2
              x1 <=> x2
  fi x1=x2
```

reversible procedure

test (n=0)

swap values of x1, x2

assertion (x1=x2)

Forward & Backward Computation

```
procedure main_fwd
  n += 4
  call fib
```

procedure forward

```
procedure main_bwd
  x1 += 5
  x2 += 8
  uncall fib
```

procedure backward

```
procedure fib
```

one procedure for both directions
⇒ reuse, code sharing

uncall: invokes inverse semantics

[ReillyFederighi65,LutzDerby82]

Program Inversion: Inverse Program

`n x1 x2` global store

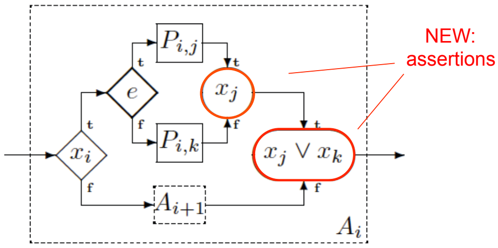
```
procedure fib
  if n=0 then x1+=1
              x2+=1
         else n-=1
              call fib
              x1+=x2
              x1<=>x2
  fi x1=x2
```

test (n=0); assertion (x1=x2)

⇒ fib^{-1} ?

Program Inversion $\mathcal{I}$

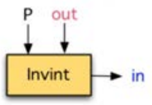
A Reversible Flowchart Diagram



[YokoyamaAxelsenGlück'08]

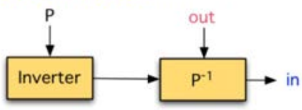
Two Approaches to Inversion of Program P in ⇒ out

- Inverse interpreter [McCarthy'56 ...]:



Applies to reversible & irreversible languages

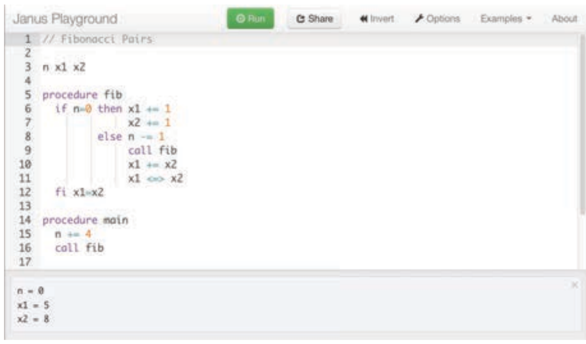
- Program inverter [Dijkstra'78, Gries'81...]:



- Related by Futamura projections [AbramovGlück'98]:

Janus Playground

http://topps.diku.dk/pirc/



Conclusion: Reversible Computing

Embraces the Theoretical, Practical, Technical, Applied.

- Computability, complexity, automata
- Abstract machines & computing models
- Languages design & implementation
  - Compilation, optimization, analysis, type systems
  - Program inversion, specialization, composition
  - Resource management: heap, parallelism
- Algorithms design
- Technical aspects
  - e.g. computer architectures
- Applied computing
  - e.g. robotics, rollback simulation, software engineering

References

- Axelsen H.B., Glück R., What do reversible programs compute? Hofmann M. (ed.), Foundations of Software Science and Computation Structures. LNCS 6604, 42-56, 2011.
- Axelsen H.B., Glück R., Yokoyama T., Reversible machine code and its abstract processor architecture. Diekert V., et al. (eds.), Computer Science - Theory and Applications. LNCS 4649, 56-69, 2007.
- De Vos A. et al., Designing garbage-free reversible implementations of the integer cosine transform. ACM Journal on Emerging Technologies, 11(2): article 11, 2014.
- Glück R., Kawabe M., A program inverter for a functional language with equality and constructors. Ohori A. (ed.), Programming Languages and Systems. LNCS 2895, 246-264, 2003.
- Glück R., Yokoyama T. (eds.), Reversible Computation. Proceedings. LNCS 7581, 2013.
- Thomsen M.K., Glück R., Axelsen H.B., Reversible arithmetic logic unit for quantum arithmetic. Journal of Physics A: Mathematical and Theoretical, 43(38): 382002, 2010.
- Yokoyama T., Axelsen H.B., Glück R., Towards a reversible functional language. De Vos A., Wille R. (eds.), Reversible Computation. LNCS 7165, 14-29, 2012.
- Yokoyama T., Glück R., A reversible programming language and its invertible self-interpreter. Partial Evaluation and Program Manipulation. 144-153, ACM 2007.

… and related references therein.