Academy of Sciences of the Czech Republic
Institute of Information Theory and Automation

department of signal processing

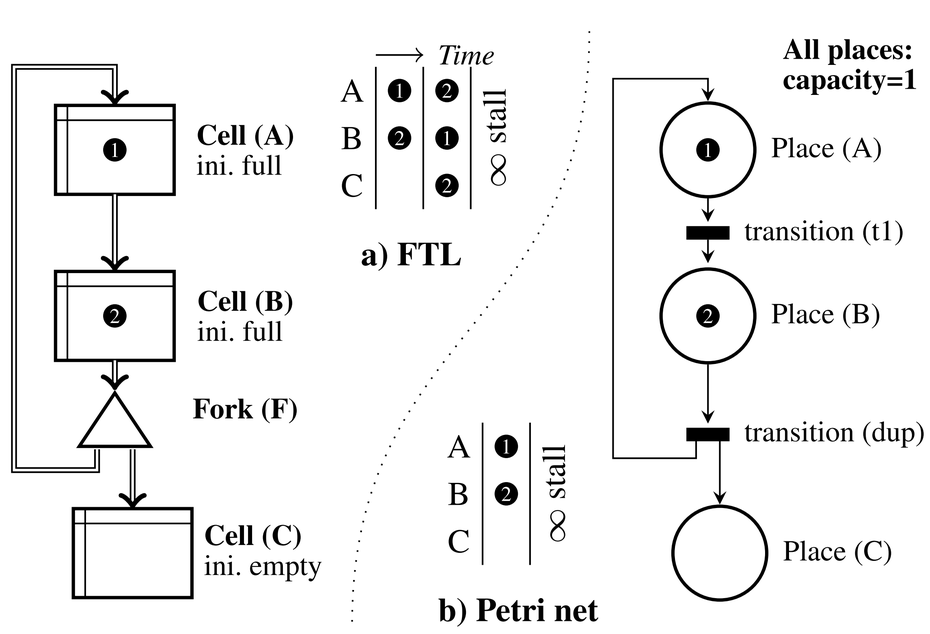
The *Flow-Transfer Level* Modelling: Data-Driven Circuits in the Clock-Synchronous Domain

Jaroslav Sýkora <*sykora@utia.cz*>

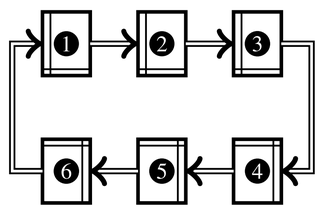
signal processing

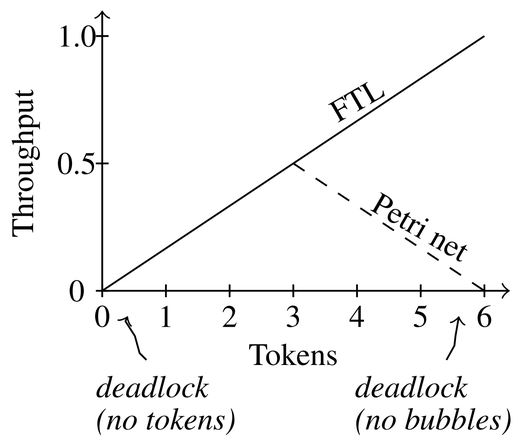
1. Motivation & Comparison to a Data-Flow Based on Petri Nets

- Modelling and synthesis of fine-grained data-driven circuits in the clock-synchronous hardware (FPGA).
- Full pipeline throughput as in manual RTL approach while enabling a higher-level data-oriented hardware programming style.



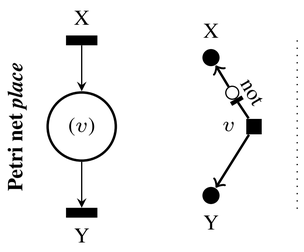
- *below:* Dataflow systems based on finite capacity Petri nets have their throughput bounded by the amount of *bubbles* and *tokens* in pipeline.
- *left:* Our approach keeps the data-driven paradigm but overcomes the need for equalizing tokens and bubbles.



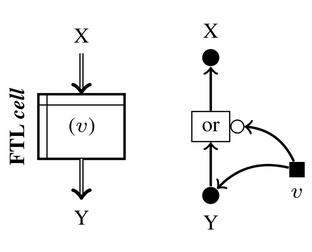


2. Activation Graphs

Petri net:

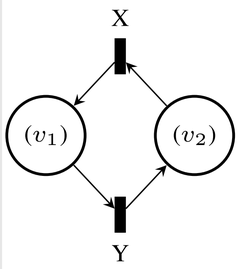


FTL:

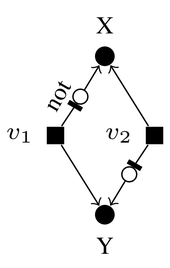


- Activation graphs compute the firing conditions (signals) for data-flow operators.
  - The behaviour of flow operators is defined by activation sub-graphs extended with data-paths.
- Cycles in activation graphs represent cyclic dependencies in *promises* of transferring *some* data.
  - Cycles are allowed as long as they do not create true data dependency loop.
  - Our tool recursively eliminates cycles in activation graphs.
  - The resulting acyclic act. graph is the basis for VHDL code generation.

Example: *Activation graph* from a Petri net:



a) flow graph



b) activation graph

v_1	v_2	X	Y	
0	0	0	0	
0	1	1	0	
1	0	0	1	
1	1	0	0	← !

c) dynamic states in the activation graph

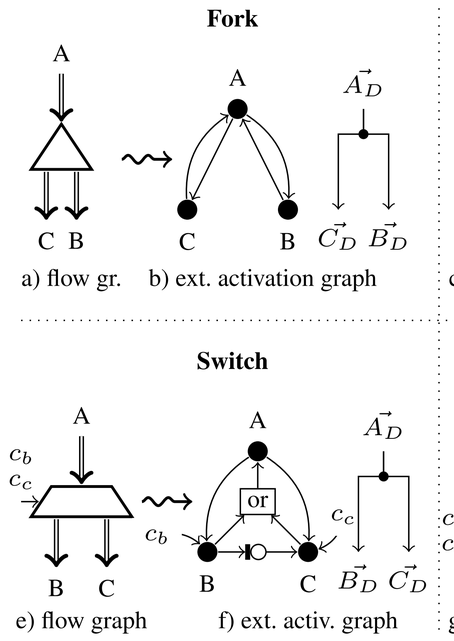
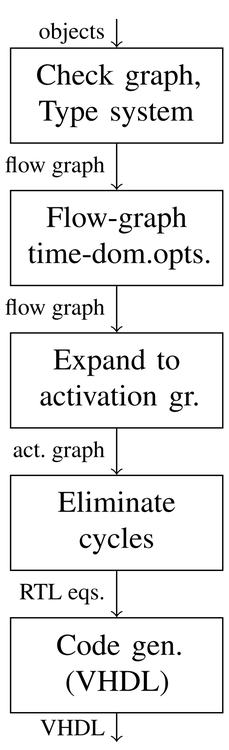
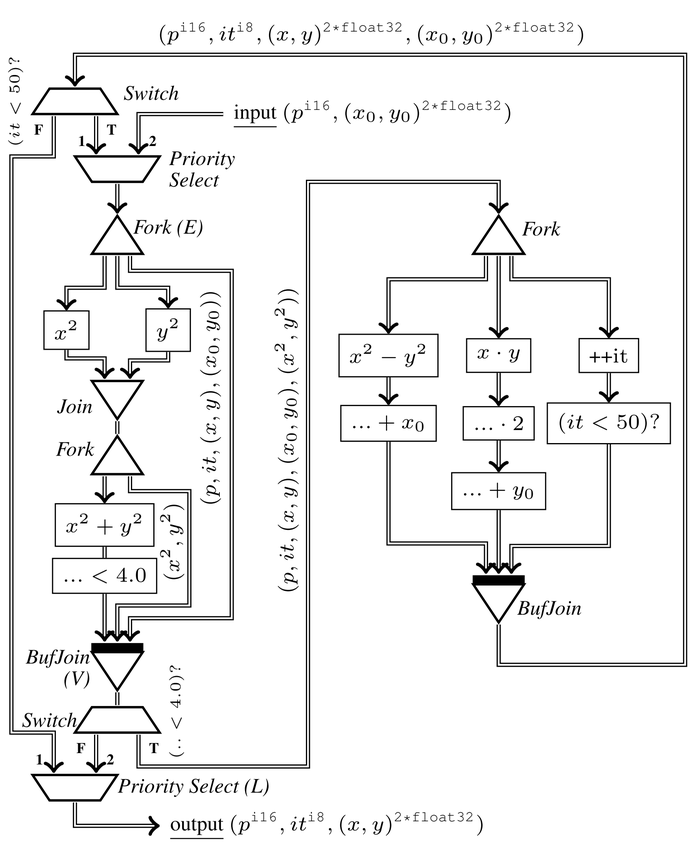


Figure : Some of the primitive *flow operators* (a, c, e, g) and their associated activation graphs, extended with the data-path.

3. Implementation & Evaluation



- *left:* Our tool expands input flow graph into global activation graph, resolves loops in the graph, and generates synthesisable VHDL code.
- *right:* Example 1: Mandelbrot Set pipeline flow graph.
  - Compute operations in hardware use 32-bit pipelined floating-point (FP) cores from the Xilinx CoreGen library.
  - Pipeline interlocks are generated by our tool.
  - Comparison to a design created in Xilinx System Generator.
- Example 2: FP vector unit.
  - implements vector addition, vector summation, minimum, maximum, index of the minima, index of the maxima.
  - Comparison to an optimized manually created VHDL code.



- Synthesis in Xilinx XST 12.4 in Spartan 6 FPGA.
- SG=Xilinx System Generator

Config.	Model	Freq	Slices	FF	LUT	LUTRAM
Example 1: Mandelbrot Set Pipeline						
A4M3	FTL	50M	916	1849	2692	270
	SG	50M	945	1588	2818	248
A4M4	FTL	105M	923	1847	2732	284
	SG	95M	1019	1592	2863	277
Example 2: FP Vector Unit						
A4	FTL	160M	265	384	607	48
	manual	150M	281	491	687	48

Conclusions:

- FPGA synthesis results of the tool-generated code are on par with manual and SG approaches.
- The data-driven paradigm in FTL increases the design abstraction level.