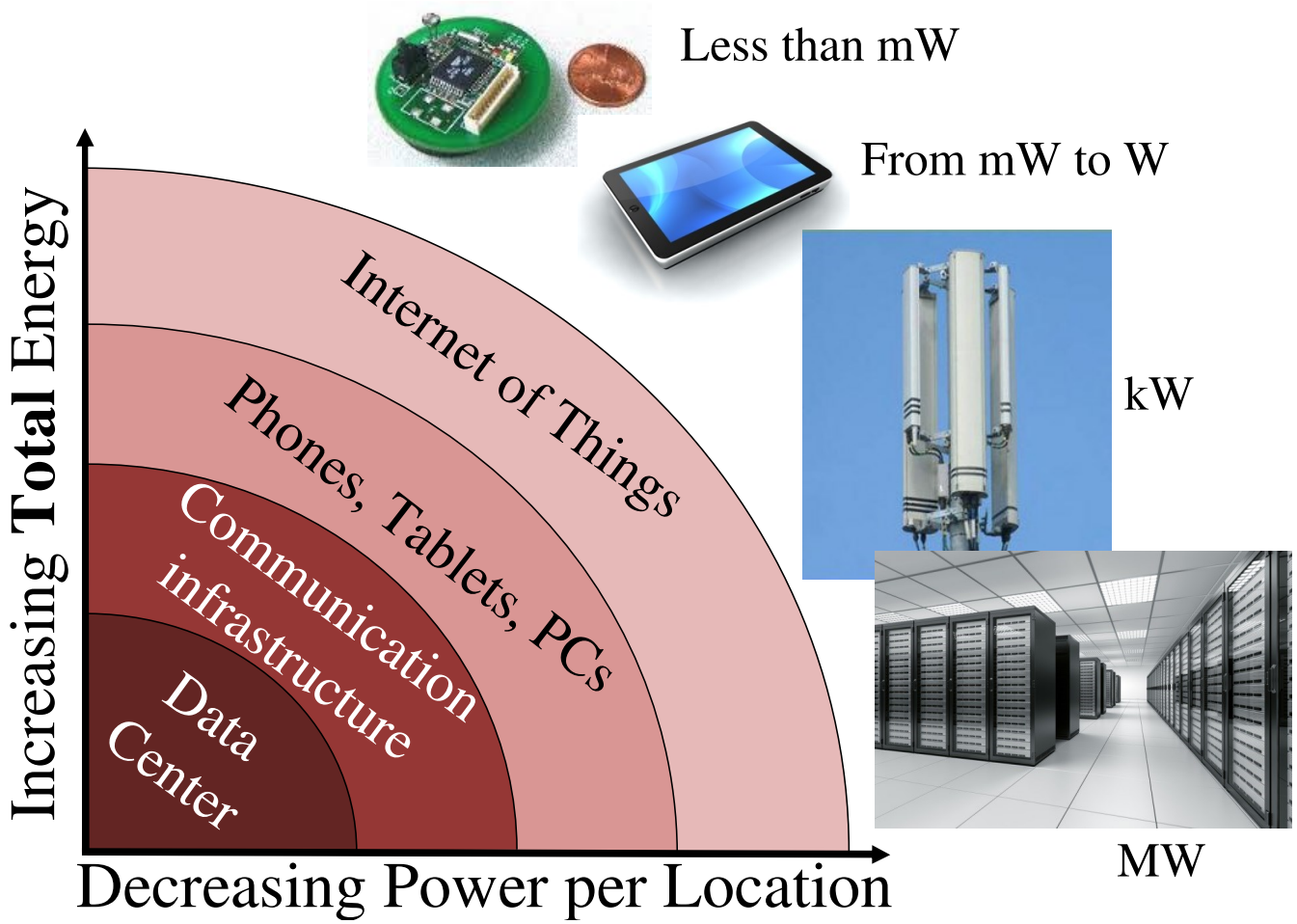
High-Level Synthesis of Event-based Systems

Jean Simatic, Rodrigo Possamai Bastos, Laurent Fesquet

June 7, 2016

Power challenge of the Internet of Things

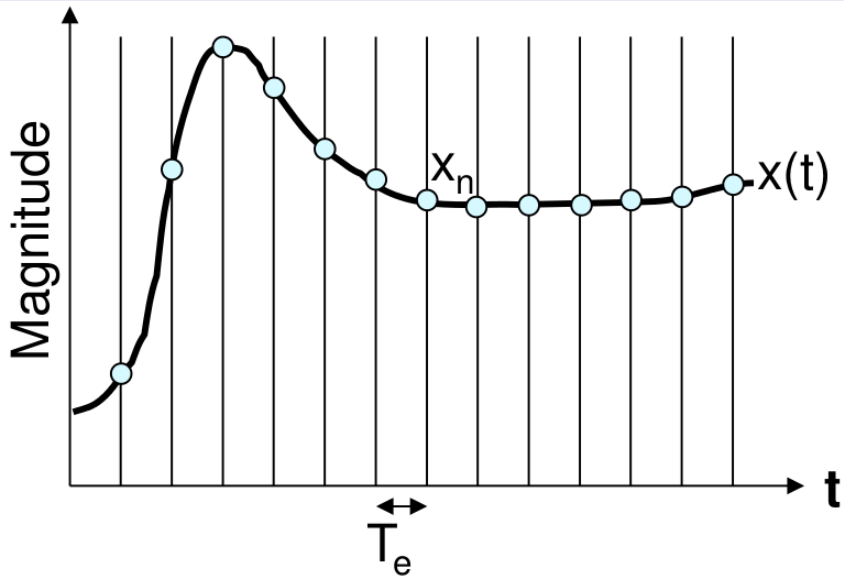


Motivation

Reduce power usage of signal processing chains thanks to event-based techniques

Technique 1: Level-crossing sampling

Uniform sampling

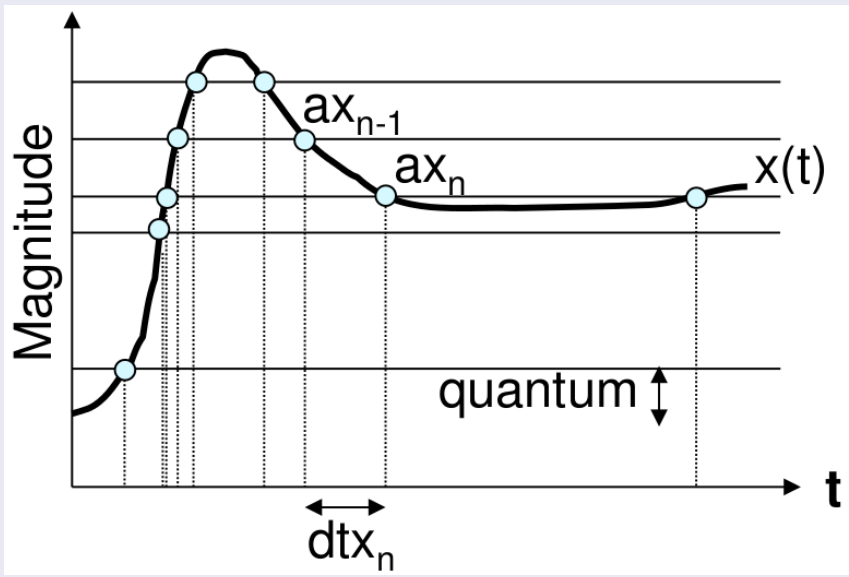


Reconstructibility under Shannon-Nyquist theorem:

$$f_{sample} \geq 2 * f_{max}$$

- Easy to formalize and process
- Some useless samples

Level-crossing sampling



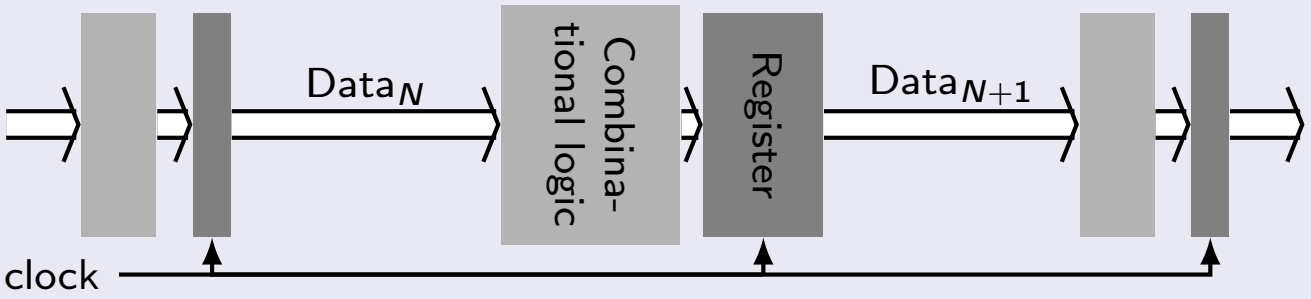
Reconstructibility under Beutler theorem:

$$E(f_{sample}) \geq 2 * f_{max}$$

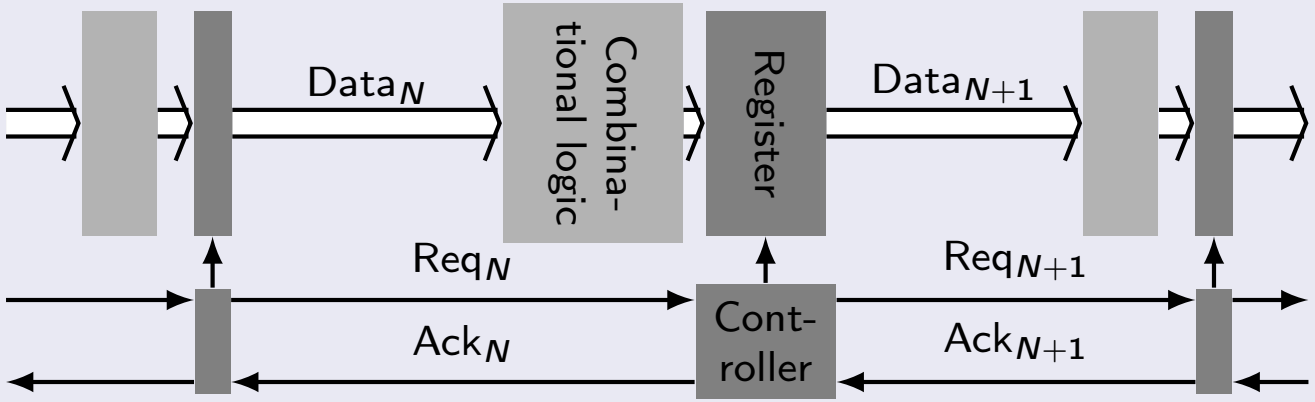
- Less samples
- More complex processing

Technique 2: Asynchronous circuits

Synchronous pipeline



Asynchronous pipeline (bundled-data)



- Local handshakes between components
- Naturally stalls in absence of new data

The design challenge

- Event-based sampling and processing are promising but little-known
- Tailor-made designs for specific applications

High Level Synthesis (HLS)

- Algorithmic-level design
- Automated circuit synthesis

Our HLS flow

- From the application: Algorithm description and signal information
- To the circuit: Event-driven ADC and DSP

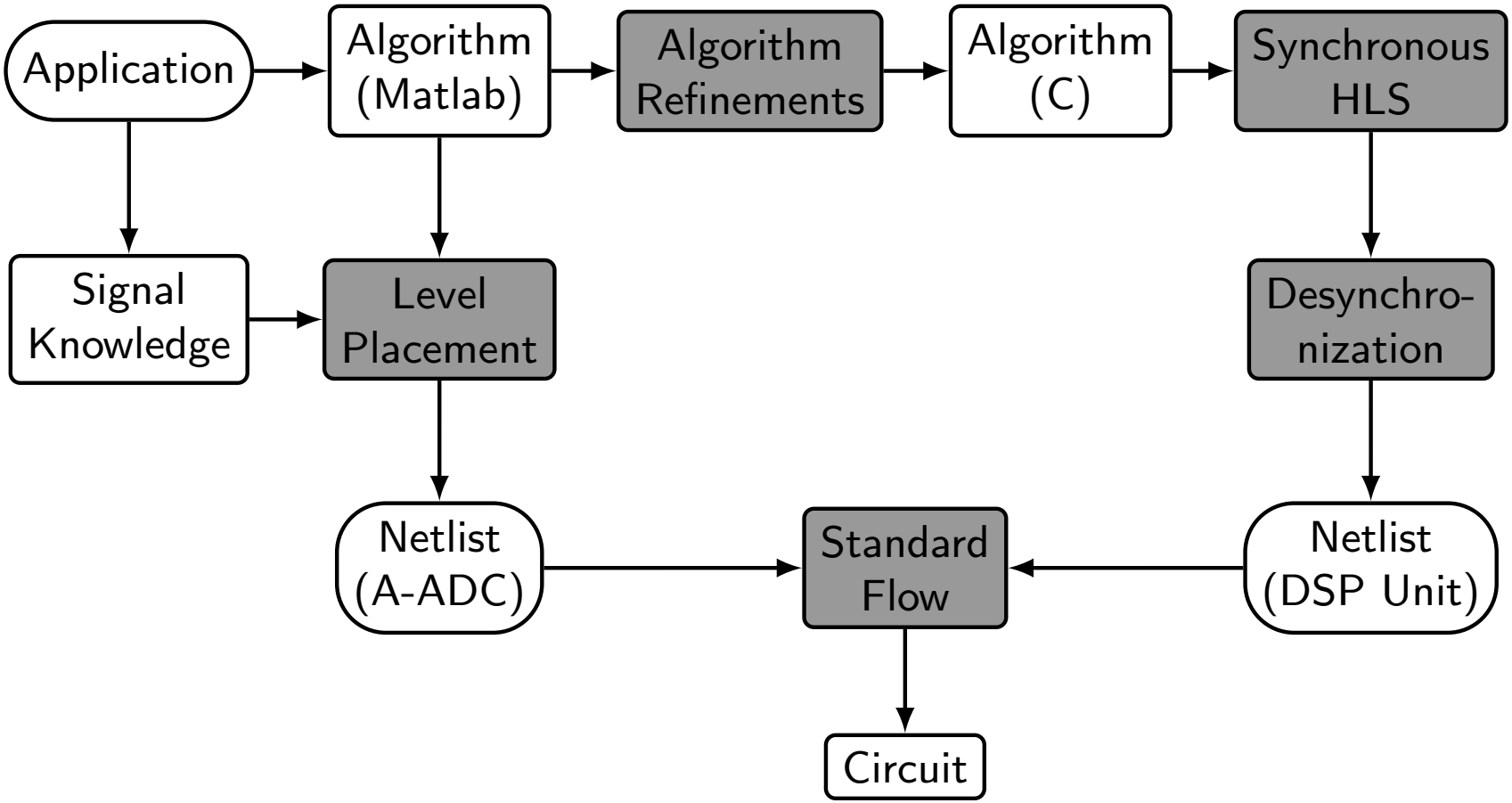
Outline

1. Proposed flow: from application specifications to chips
2. Difficulty 1: Choosing the sampling
   - Level-crossing sampling
   - Which levels?
3. Difficulty 2: Asynchronous High-Level Synthesis
   - Our proposition
   - Comparison with existing flows
   - Desynchonization method
4. Results

Outline

1. Proposed flow: from application specifications to chips
2. Difficulty 1: Choosing the sampling
   - Level-crossing sampling
   - Which levels?
3. Difficulty 2: Asynchronous High-Level Synthesis
   - Our proposition
   - Comparison with existing flows
   - Desynchonization method
4. Results

Design flow

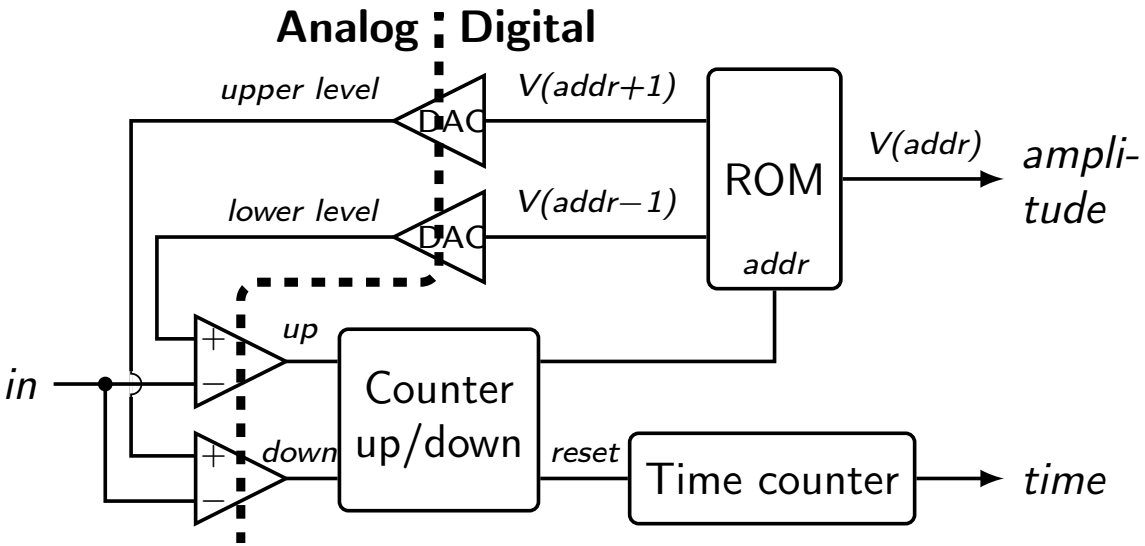

Application
Algorithm (Matlab)
Algorithm Refinements
Algorithm (C)
Synchronous HLS
Signal Knowledge
Level Placement
Desynchro-nization
Netlist (A-ADC)
Standard Flow
Netlist (DSP Unit)
Circuit

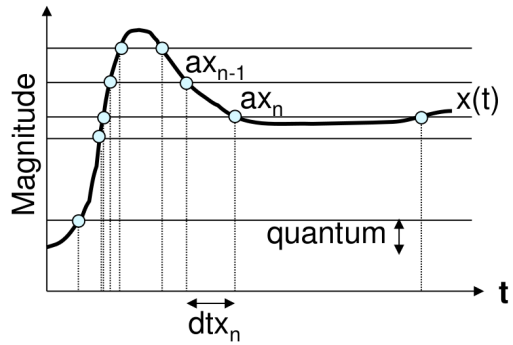
Outline

1. Proposed flow: from application specifications to chips
2. Difficulty 1: Choosing the sampling
   - Level-crossing sampling
   - Which levels?
3. Difficulty 2: Asynchronous High-Level Synthesis
   - Our proposition
   - Comparison with existing flows
   - Desynchonization method
4. Results

Level-crossing sampling

- Many possible non-uniform-sampling (level-crossing, peak, slope, send-on-delta . . .)
- For practical reasons, we choose the level-crossing sampling scheme and Allier's tracking ADC





Which levels?

Previous works

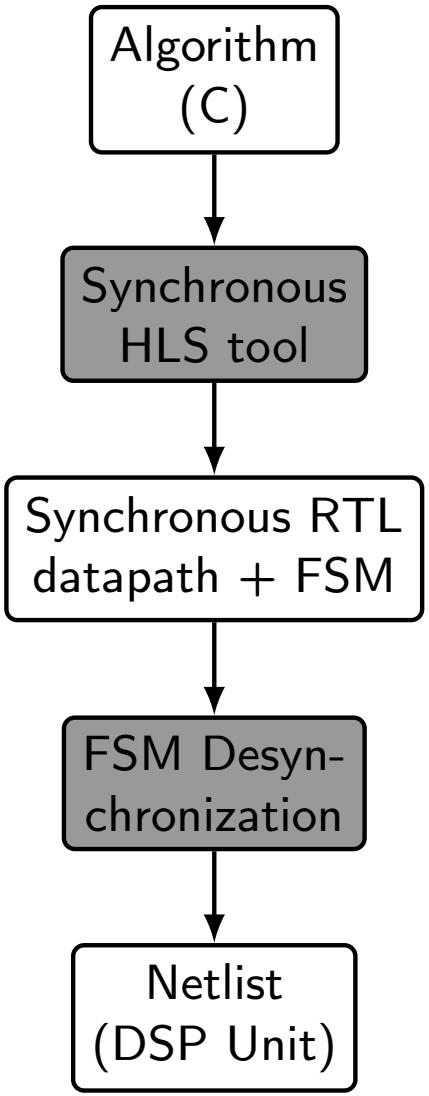
- Machine learning methods for classifiers (Arslan *et al*, Le Pelleter)
- Asymptotic convergence toward optimal levels (Guan 2008)

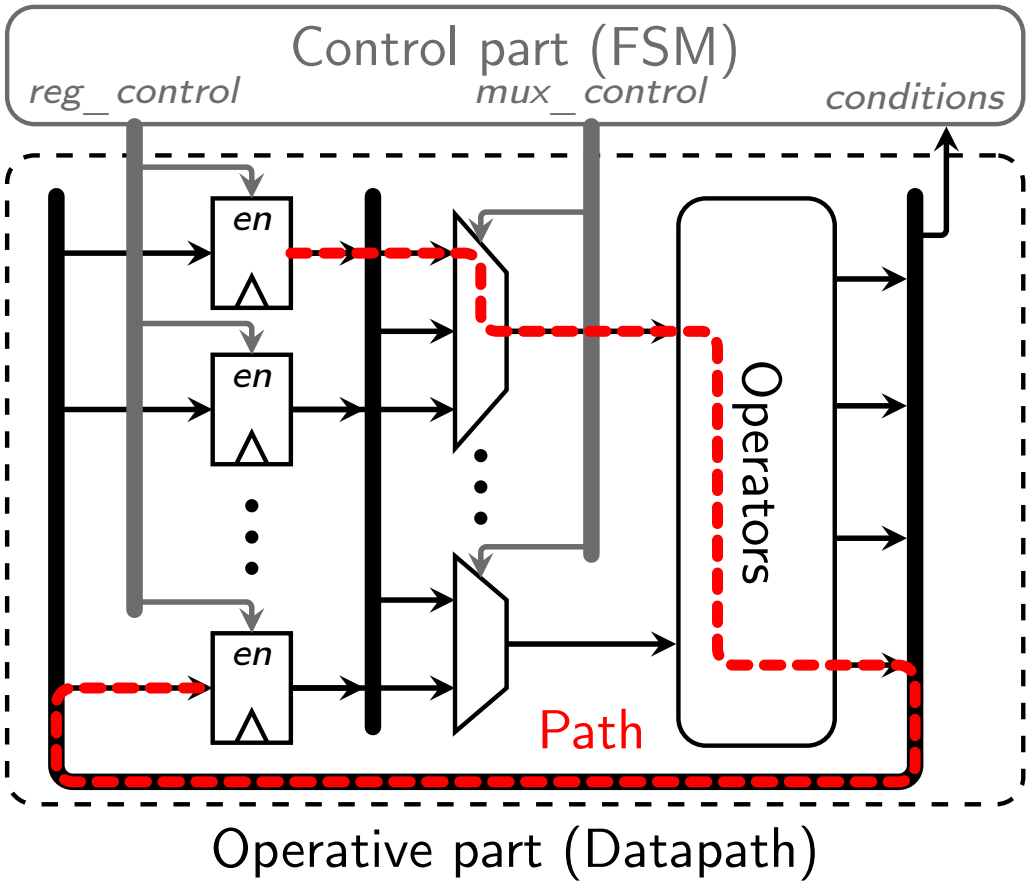
- Need knowledge about the signal
- Probably no general framework: classes of applications
- Probably approximate solution (NP-complete problem)

Outline

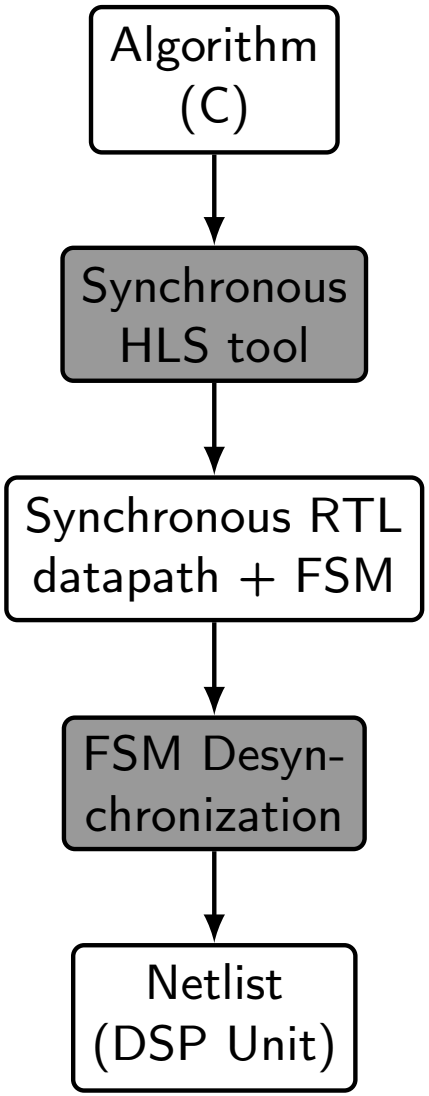
1. Proposed flow: from application specifications to chips
2. Difficulty 1: Choosing the sampling
   - Level-crossing sampling
   - Which levels?
3. Difficulty 2: Asynchronous High-Level Synthesis
   - Our proposition
   - Comparison with existing flows
   - Desynchonization method
4. Results

Synchronous HLS + Desynchronization





Synchronous HLS + Desynchronization



Why use a synchronous-dedicated tool?

- More available tools (academic and commercial)
- State-of the art design space exploration

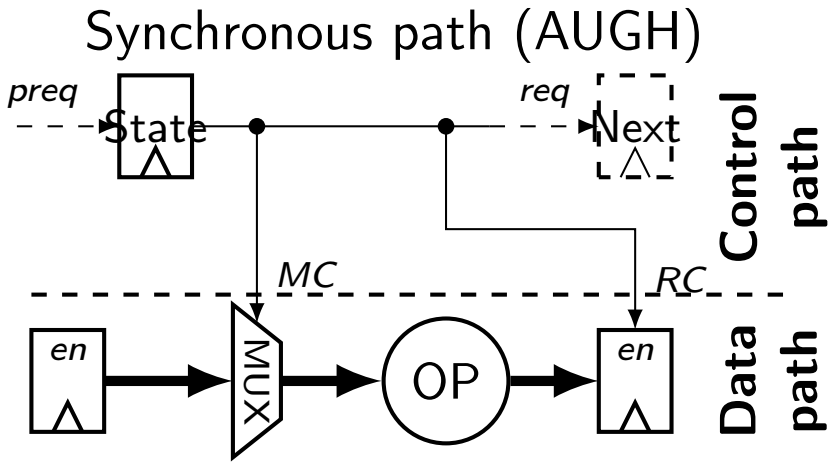
AUGH

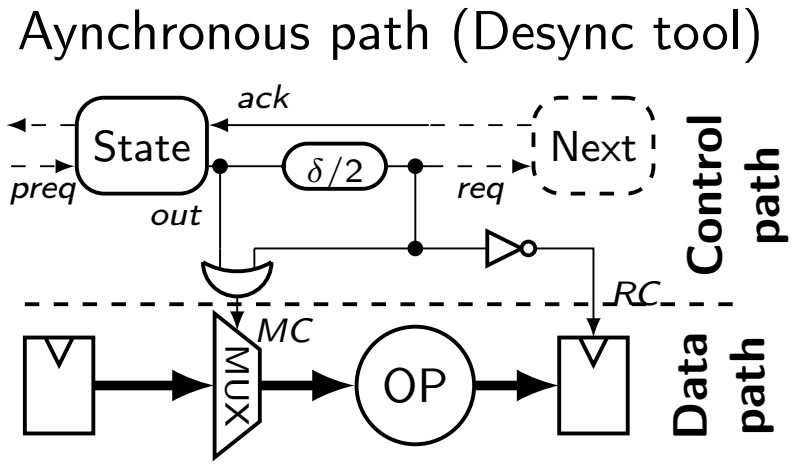
- Basics: loop unrolling, several memory models
- Recent features: multicycle paths, branch probabilities and loop iteration annotation
- Open source and readable generated code

Comparison with existing flows

- Desynchronization (Cortadella *et al*)
- Flows using specialized languages:
  - Syntax directed translation: Balsa, Haste
  - Compilation optimization: Code-to-code optimization, CHP
- From generic algorithmic language (C-like):
  - Venkataramani *et al*: Map IR from CASH to micropipeline constructs
  - Garcia *et al*: Synchronous-like datapath + centralized locally-clocked FSM.

Desynchonization method





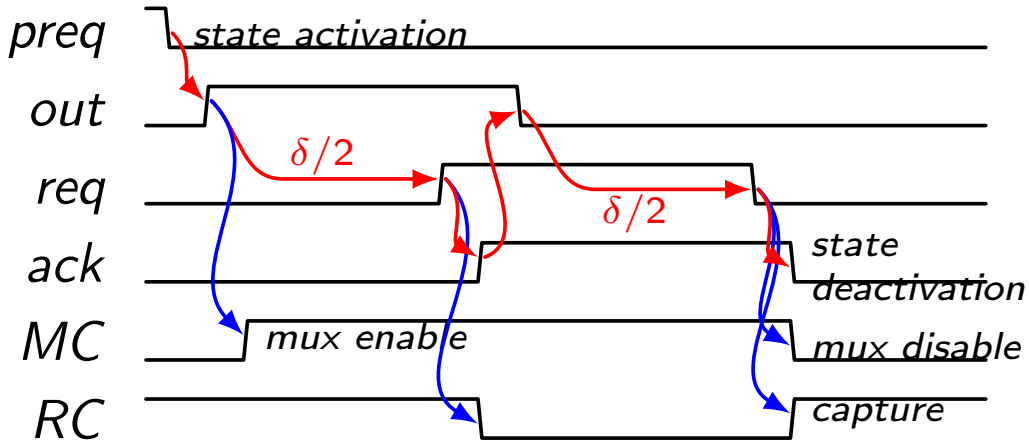


Figure: Stage chronogram.

Controller (Distributed)

- Speed insensitive
- Late-forward
  David cells (Hollaar 82)
  Burst mode ctrl (Yun 96)
- Early-acknowledgement

Outline

1. Proposed flow: from application specifications to chips
2. Difficulty 1: Choosing the sampling
   - Level-crossing sampling
   - Which levels?
3. Difficulty 2: Asynchronous High-Level Synthesis
   - Our proposition
   - Comparison with existing flows
   - Desynchonization method
4. Results

Test cases

	Gratest common divisor	Interpolated FIR (Aeschlimann *et al*)
C code lines	20	72
FSM states	8	15
Gates count	~ 600	~ 6000

Desycnhronization effects

Synthesis in 40nm TSMC technology (asynchronous cells by Dolphin Integration)

Area

Small overhead

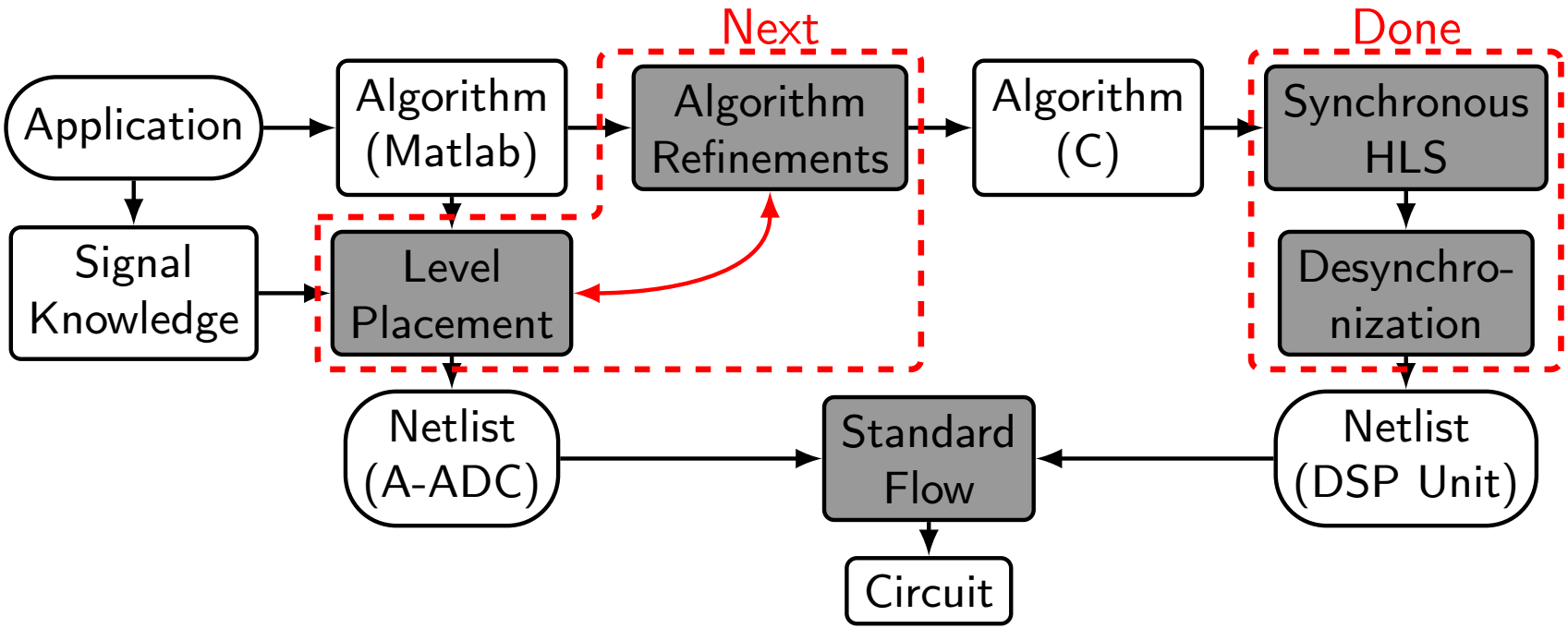
- FSM area alone is 2x to 5x bigger
- Overhead on the entire design is 12% and 5%
- The datapath size may decrease (no more enables)

Computation speed

Computation speed increases by 25% with desynchronization

- Under no timing constraints, the ripple carry adders are critical.
- Asynchronous FSMs can go faster on non-critical states.

Conclusion



Conclusion

- Complete event-driven flow from the application to the circuit
- Desynchronization: Low area overhead for significant speed gain

Perspectives

- Comparison with synchronous and asynchronous manual designs.
- Framework definition for choosing the levels.

Thank you

Outline

5 Cell area

Cell area

	Greatest common divisor								
	FSM			Datapath			Total		
Cell area (μm^2)	Sync	Async	Δ	Sync	Async	Δ	Sync	Async	Δ
Combinational	15.1	68.2	×4.5	388.0	310.9	−20%	403.1	379.1	−6%
Registers	26.3	15.4	−42%	316.2	316.1	0	342.5	331.5	−3%
Total	41.4	131.2	×3.2	704.1	627.0	−11%	745.5	758.2	**+2%**

	Interpolated FIR filter								
	FSM			Datapath			Total		
Cell area (μm^2)	Sync	Async	Δ	Sync	Async	Δ	Sync	Async	Δ
Combinational	30.6	287.0	×9.4	3550	1620	−64%	3592	1907	−47%
Registers	62.6	30.7	−51%	3698	4241	+15%	3751	4272	+14%
Total	93.2	401.0	×4.3	7238	5861	−19%	7342	6262	**−15%**