Behavioral Specification of a Circuit Using SyncCharts: a Case Study

C. ANDRÉ – M-A. PERALDI-FRATI
Introduction (1)

• Digital Circuit Design
  – Abstract specifications
  – State-Based Specifications

• Graphical Representation
  – User-friendliness
  – Must be mathematically defined

State Transition Graphs
Introduction (2)

State Transition Graphs
- well-founded
- but a flat model

Concurrency + preemption

SyncCharts

Expression of the expected behavior

Validation: Test Model-checking

Circuit optimization
Plan

• Introduction
• Illustrative Example
• Modular Specification with SyncCharts
• Validation
• Circuit Optimizations
• Conclusion
Encoding/Decoding

0++--00-0+0

encoding

decoding

Transmitter

Receiver

Bin

Bout

010000010

010000010
Specification

encoding: \{0,1\}^* \rightarrow \{-U, 0, +U\}^*

decoding: \{-U, 0, +U\}^* \rightarrow \{0,1\}^*

Requirement 1

\forall n \geq 1: \left| \sum_{k=1}^{k=n} u_k \right| \leq U

Requirement 2

\forall n \geq 4: \neg \left( (u_n = 0) \land (u_{n-1} = 0) \land (u_{n-2} = 0) \land (u_{n-3} = 0) \right)
In what follows: \(-U \leftrightarrow n; \quad 0 \leftrightarrow z; \quad +U \leftrightarrow p\)

**encoding:** \(\{0,1\}^* \rightarrow \{n,z,p\}^*\)

**decoding:** \(\{n,z,p\}^* \rightarrow \{0,1\}^*\)
Encoding

Standard encoding:

\[ 0 \rightarrow z \quad 1 \rightarrow \{n,p\} \text{ alternately} \]

4 successive 0:

\[ 0000 \rightarrow P z z V \]

- Parity
- Violation
**Example**

This system is not causal (depends on the future)
Making it causal

Delayed output: \( u'[k] = u[k-3] \)

Detection of 4 « 0 » in a row

FourZeros

Standard Mealy machine
Classical Design

[Zahnd 1987]

A bright design …

• Well-structured
• Easy to interpret

But …

• Try to find it!
• \(d\) and \(f\) are not independent
• Implicit shift register
SyncCharts-based Design

• **Decompose** into interacting agents
  – Detector (4 consecutive 0’s)
  – Parity manager
  – Sequence generator
  – Output manager

• **Test** on scenarios (Esterel Studio)

• **Prove** safety properties

• **Circuit optimization** (if HW controller)
Transmitter

ENCODER

@DETECTOR[..] @

@PARITY @

EMITTER

@NONZERO @

signal B3, Alternance, Violation, FourZeros, PlusOrMinus

where @DETECTOR[signal Bin/Bit0, B3/Bit3]
Detector

@DETECTOR[signal Bin/Bit0, B3/Bit3]

D0@DELAY[..] @

D1@DELAY[..] @

D2@DELAY[..] @

loop
  present Bit0 or Bit1 or Bit2 or Bit3 else emit FourZeros
  end present
  each tick

signal Bit1, Bit2

where
D0@DELAY[signal Bit0/Ev, Bit1/DEv]
D1@DELAY[signal Bit1/Ev, Bit2/DEv]
D2@DELAY[signal Bit2/Ev, Bit3/DEv]
Emitter

```
Emitter

EMITTER

NORMAL

loop
  present B3 then
  emit Alternance
  else
  emit Zero
  end present
end each tick

EXCEPTION

#FourZeros

present Even then
  emit Zero
else
  emit Alternance
end present; pause;
emit Zero; pause;
emit Zero; pause;
emit Violation; pause
```
Safety properties (1)

Property:
At each instant : one and only one signal out of \{n,z,p\} is emitted

Observer: An esterel module

loop

present (n and not z and not p) 
or (not n and z and not p) 
or (not n and not z and p)
else
emit non_exclusive
end present

each tick
Safety properties (2)

Requirement 1: \[ \forall n \geq 1: \left| \sum_{k=1}^{k=n} u_k \right| \leq U \]

Observer: A flat syncChart = a FSM

Observer:

\[
\begin{array}{c}
\text{OBSERVER\_R1} \\
\begin{array}{c}
\text{TooNegative} \\
\text{p} \\
\end{array} \\
\text{U} \\
\text{p} \\
\text{p} \\
\text{TooPositive} \\
\end{array}
\]
Safety properties (3)

Requirement 2:

\[ \forall n \geq 4 : \neg \left( (u_n = z) \land (u_{n-1} = z) \land (u_{n-2} = z) \land (u_{n-3} = z) \right) \]

Observer: A syncChart with preemption

\[ \text{OBSEVER\_R2} \]

\[ \begin{array}{c}
\# z \\
z \\
z \\
z \\
\text{TooManyZ} \\
p \text{ or } n
\end{array} \]
Safety properties (4)

Encoder/Decoder:

Beyond the sixth instant:
**Bout is identical to Bin**, upto a 6 instant delay.

Observer:
Performance

High-level behavioral description (e.g., encoder)

syncChart

esterel program

Structural translation

Esterel’s compiler

blif

SIS

105 states

optimized circuit

35 states

As efficient as the hand-coded solution
Conclusion (1)

• State-transition graphs
  – Understandable even by non-specialists
  – Flat model → **in-the-small**

• **SyncCharts =**
  – State-transition graphs +
    • Concurrency
    • Hierarchy
    • Preemption
  – Can include Esterel code
Conclusion (2)

« SyncCharts » is a Synchronous Formalism

- mathematical semantics
- safe code generation (ESTEREL)
- interactive simulation (XES)
- model-checking (XEVE)
- link to SIS

{ formal

\{ validation

efficient circuit