Formal Development of a Clock Synchronization Circuit

Paul S. Miner

This talk presents the latest stage in a formal development of a fault-tolerant clock synchronization circuit. The development spans from a high level specification of the required properties to a circuit realizing the core function of the system.

An abstract description of an algorithm has been verified to satisfy the high-level properties using the mechanical verification system EHDM [2]. This abstract description is recast as a behavioral specification input to the Digital Design Derivation system (DDD) developed at Indiana University [1]. DDD provides a formal design algebra for developing correct digital hardware. Using DDD as the principle design environment, a core circuit implementing the clock synchronization algorithm was developed [3]. The design process consisted of standard DDD transformations augmented with an ad hoc refinement justified using the Prototype Verification System (PVS) from SRI International [4].

Subsequent to the above development, Wilfredo Torres-Pomales discovered an area-efficient realization of the same function [5]. Establishing correctness of this optimization requires reasoning in arithmetic, so a general verification is outside the domain of both DDD transformations and model-checking techniques.

DDD represents digital hardware by systems of mutually recursive stream equations. A collection of PVS theories was developed to aid in reasoning about DDD-style streams. These theories include a combinator for defining streams that satisfy stream equations, and a means for proving stream equivalence by exhibiting a stream bisimulation.

DDD was used to isolate the sub-system involved in Torres-Pomales’ optimization. The equivalence between the original design and the optimized verified was verified in PVS by exhibiting a suitable bisimulation. The verification depended upon type constraints on the input streams and made extensive use of the PVS type system. The dependent types in PVS provided a useful mechanism for defining an appropriate bisimulation.

References


Formal Development of a Fault-Tolerant Clock Synchronization Circuit

Paul S. Miner
May 12, 1995

Outline

- Summary of Prior work
- Description of Torres-Pomales' Optimization
- Verification of Optimization
  - Definition of Streams in PVS
  - Proof by Co-Induction

Prior Verification

- Developed verified design of clock synchronization circuit using a combination of formal techniques.
  - Mechanized Proof System (EHDM, PVS)
  - Digital Design Derivation
  - OBDD-based tautology checking
Design Hierarchy—New

Informal Description of Algorithm

- Welch & Lynch Algorithm
- System of \( N \) clocks designed to tolerate \( F \) arbitrary faults
- Completely connected network
- Each Clock periodically
  - Gathers estimates of readings of all other clocks in the system
  - Discards the \( F \) largest and \( F \) smallest readings
  - Sets self to mid-point of the range of the remaining readings

Intermediate Stage of Previous Derivation

Intermediate Stage

- Circuit implements core function of algorithm
  - Network interconnect in different partition of design
- Independent of number of clocks in the system
- This stage was reached via a combination of standard DDD transformations and an ad hoc refinement verified using PVS
Torres-Pomales’ Optimization

Verification of Optimized Circuit

- Reasoning about Stream Equations using PVS
  - Definition of Streams in PVS
  - Proof by Co-Induction
- Verification Using PVS Streams Package

Signal Assumptions Justifying Optimization

Signal RD is the output of a counter.

Streams in PVS

```
Stream_adt[alpha: TYPE]: THEORY
BEGIN
Stream: TYPE
a: VAR alpha
S, X, Y: VAR Stream
cs?: [Stream -> boolean]
cs: [alpha, Stream -> Stream]
hd: [Stream -> alpha]
tl: [Stream -> Stream]
nth(S:Stream,n:nat):alpha = hd(iterate(tl,n)(S))
```
Streams in PVS

AXIOMS

Stream_inclusive: AXIOM cs?(S)
Stream_cs_eta: AXIOM cs(hd(S), tl(S)) = S
Stream_hd_cs: AXIOM hd(cs(a, S)) = a
Stream_tl_cs: AXIOM tl(cs(a, S)) = S
Stream_eq: AXIOM X = Y <\Rightarrow> FORALL n: nth(X, n) = nth(Y, n)
END Stream_adt

Proof by Co-Induction

Stream_coinduct[alpha: TYPE]: THEORY
BEGIN
IMPORTING Stream_adt
X, Y: VAR Stream[alpha]
R: VAR PRED[[Stream[alpha], Stream[alpha]]]
Bисimulation: TYPE = {R | FORALL X, Y: R(X, Y) \Rightarrow hd(X) = hd(Y) & R(tl(X), tl(Y))}
co_induct: THEOREM (EXISTS (R: Bисimulation): R(X, Y)) \Rightarrow X = Y
END Stream_coinduct

Defining Streams

Stream_corec[alpha, beta: TYPE]: THEORY
BEGIN
IMPORTING Stream_adt[alpha]
f: VAR [alpha -> beta]
g: VAR [alpha -> alpha]
a: VAR alpha
corec(f, g, a): Stream[beta]
corec_def: AXIOM corec(f, g, a) = cs(f(a), corec(f, g, g(a))
[...]
END Stream_corec

Stream Equations for Original Sub-Circuit

\[
\begin{align*}
\text{THETA-F1} & = \text{cs}(i, \text{MUX}(F1, RD, \text{THETA-F1})) \\
\text{THETA-NF} & = \text{cs}(i, \text{MUX}(NF, RD, \text{THETA-NF})) \\
\text{CFN} & = \frac{\text{THETA-F1} + \text{THETA-NF}}{2}
\end{align*}
\]
Stream Equations for Optimized Sub-Circuit

\[
\text{HOLD} = \text{cs}(\text{false}, F_1 \& \neg \text{HOLD}) \\
\text{CIN} = \text{HOLD} \& \neg H \\
\text{OPT} = \text{cs}(i, \text{MUX}(F_1, RD, \text{INC}(\text{OPT,CIN})))
\]

Recursive Stream Definitions

\[
\text{THETA}(A, I, i) = \text{cs}(i, \text{MUX}(A, I, \text{THETA}(A, I, i))) \\
\text{HOLD}(A, a) = \text{cs}(a, A \& \neg \text{HOLD}(A, a)) \\
\text{OPT}(A, C, I, i) = \text{cs}(i, \text{MUX}(A, I, \text{INC}(\text{OPT}(A, C, I, i), C)))
\]

PVS Definitions for Circuit Verification

\[
A, B, C, R: \text{VAR Stream[bool]} \\
A, B, C, R: \text{VAR Stream[bool]} \\
I, J, K: \text{VAR Stream[int]} \\
i, j, k: \text{VAR int}
\]

\[
\text{THETA}(A, I, i): \text{Stream[int]} \quad \%\text{defined using corec} \\
\text{CFN}(A, B, I, i, j): \text{Stream[int]} \\
\quad = \text{DIV2}(\text{THETA}(A, I, i) + \text{THETA}(B, I, j)) \\
\text{HOLD}(A, a): \text{Stream[bool]} \quad \%\text{defined using corec} \\
\text{CIN}(A, B): \text{Stream[bool]} = A \& \text{NOT B} \\
\text{OPT}(A, C, I, i): \text{Stream[int]} \quad \%\text{defined using corec}
\]

Type Declarations for Assumptions on Input Signals

\[
S(R): \text{TYPE =} \\
\{A \ | \\text{Invariant(IF } R \text{ THEN NOT t1(A) ELSE A } \Rightarrow t1(A) \\text{ ENDIF)}\}
\]

\[
C(R): \text{TYPE =} \\
\{I \ | \\text{Invariant(NOT R } \Rightarrow \text{EQ(t1(I),INC(I)))}\}
\]
Correctness Theorem

\textbf{Optimize\_correct: THEOREM}

\(\forall R, (\text{RD} : C(R)), (\text{F1} : S(R)] \rightarrow \text{hd}(\text{F1}),
\text{INVARIANT(NF} \Rightarrow \text{F1}), (i : \text{int}) :\)

\(\text{CFN(F1,NF,RD},i,i) = \text{OPT(F1,CIN(HOLD(F1,false),NF),RD},i)\)

---

\textbf{Proof of Optimize\_correct by co-induction}

\textbf{Define Bisimulation \(B\) as:}

\[\{(X,Y)\} \]

\[\exists R, (\text{RD} : C(R)), (\text{F1} : S(R)), (\text{NF} : \{A : S(R)\mid A \Rightarrow F1\}), (i : \text{int})\]

\[j : \text{int} \mid \text{hd}(\text{F1}) \land \neg(\text{hd}(\text{NF})) \Rightarrow \text{hd}(\text{RD}) = j + 1,\]

\[(b : \text{bool}\mid \text{hd}(\text{F1}) \land \neg(\text{hd}(\text{NF})) \Rightarrow b = \text{odd}(i+j):\]

\[X = \text{CFN(F1,NF,RD},i,j) \&\]

\[Y = \text{OPT(F1,CIN(HOLD(F1,b),NF),RD},[(i+j)/2])\]

---

\textbf{Proof—\(B\) is a Bisimulation}

\textbf{Heads: For any \((X,Y) \in B, \text{hd}(X) = \text{hd}(Y) = [(i+j)/2].\)}

\textbf{Tails: For any \((X,Y) \in B, \text{show } (t1(X), t1(Y)) \in B.\)}

\(t1(\text{CFN(F1,NF,RD},i,j))\)

\[= \text{CFN(t1(F1),t1(NF),t1(RD),}\]

\[\quad \text{IF } \text{hd}(\text{F1}) \text{ THEN } i \text{ ELSE } \text{hd}(\text{RD}) \text{ ENDIF,}\]

\[\text{IF } \text{hd}(\text{NF}) \text{ THEN } j \text{ ELSE } \text{hd}(\text{RD}) \text{ ENDIF}\]

\(t1(\text{OPT(F1,CIN(HOLD(F1,b),NF),RD},[(i+j)/2]))\)

\[= \text{OPT(t1(F1),}\]

\[\quad \text{CIN(HOLD(t1(F1), (hd(F1) \land \neg b)),t1(NF)),}\]

\[\quad \text{t1(RD),}\]

\[\quad \text{IF } \text{hd}(\text{F1}) \text{ THEN }([(i+j)/2] \land b \land \neg(\text{hd}(\text{NF}))\]

\[\quad \text{ELSE } \text{hd}(\text{RD})\]

\[\text{ENDIF})\]

---

\textbf{Concluding Remarks}

- Proof by co-induction effective technique for verifying circuit refinements.
- Possible to exploit circuit context to complete proof
- Developed general Stream library for PVS 2
- Torres-Pomales' optimization verified in PVS using proof by co-induction
- PVS dependent type mechanism useful
- Design implemented in VLSI (hand layout)