N96-10037

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

[1] Bhaskar Bose. DDD - A Transformation System for Digital Design Derivation. Technical Report 331, Computer Science Dept. Indiana University, May 1991.

[2] Paul S. Miner. Verification of fault-tolerant clock synchronization systems. Technical Paper 3349, NASA, Langley Research Center, Hampton, VA, November 1993.

[3] Paul S. Miner, Shyamsundar Pullela, and Steven D. Johnson. Interaction of formal design systems in the development of a fault-tolerant clock synchronization circuit. In *Proceedings 13th Symposium on Reliable Distributed Systems*, pages 128–137, Dana Point, CA, October 1994.

[4] Sam Owre, John Rushby, Natarajan Shankar, and Friedrich von Henke. Formal verification for fault-tolerant architectures: Prolegomena to the design of PVS. *IEEE Transactions on Software Engineering*, 21(2):107–125, February 1995.

[5] Wilfredo Torres-Pomales. An optimized implementation of a fault-tolerant clock synchronization circuit. Technical Memorandum 109176, NASA, Langley Research Center, Hampton, VA, February 1995.

Formal Development of a Fault-Tolerant Clock Synchronization Circuit

Paul S. Miner
May 12, 1995

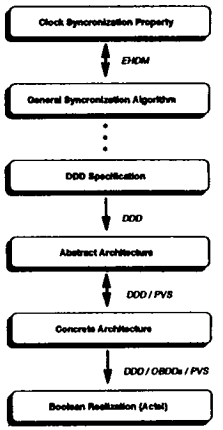
Outline

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

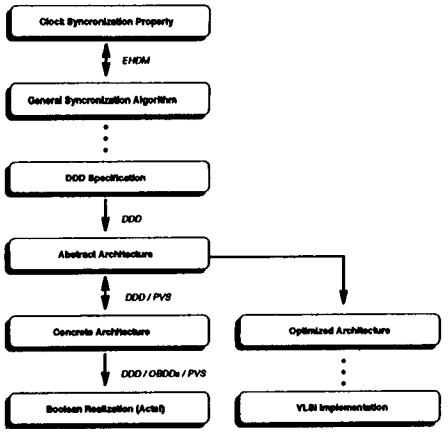
Prior Verification

- 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—Old



Design Hierarchy—New



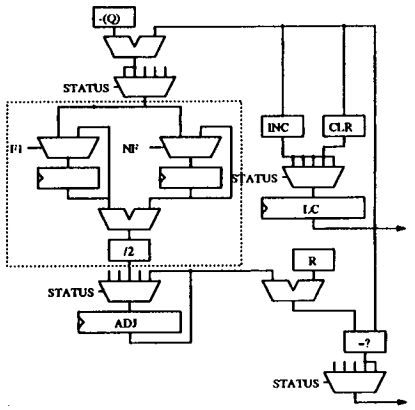
5

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

6

Intermediate Stage of Previous Derivation



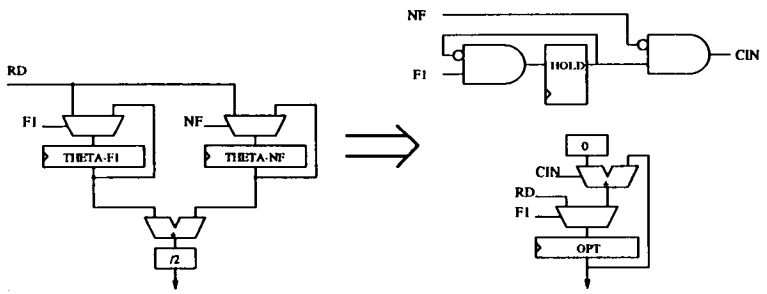
7

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

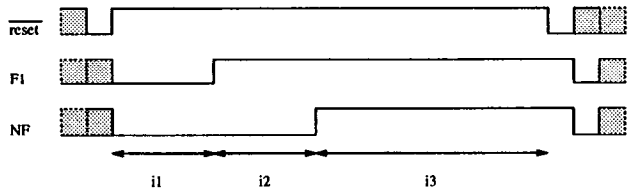
8

Torres-Pomales' Optimization



9

Signal Assumptions Justifying Optimization



Signal RD is the output of a counter.

10

Verification of Optimized Circuit

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

11

Streams in PVS

DECLARATIONS

```
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))
```

12

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 <=> FORALL n: nth(X, n) = nth(Y, n)

END Stream_adt
```


Defining Streams

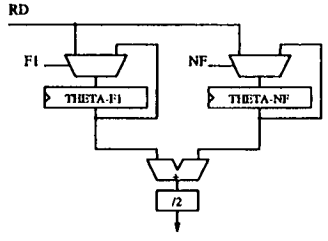
```
Stream_corec[alpha, beta: TYPE]: THEORY
  BEGIN
  IMPORTING Stream_adt[beta]

  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
```


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]]]

  Bisimulation: TYPE =
    {R | FORALL X, Y: R(X, Y) => hd(X) = hd(Y) & R(tl(X), tl(Y))}

  co_induct: THEOREM (EXISTS (R: Bisimulation): R(X, Y)) => X = Y

  END Stream_coinduct
```

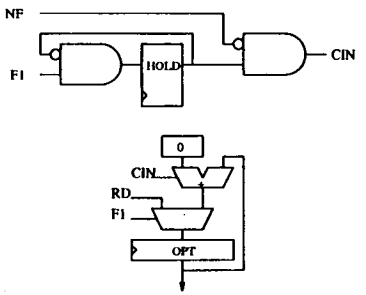

Stream Equations for Original Sub-Circuit

$$\texttt{THETA-F1} = \texttt{cs}(i, \texttt{MUX(F1, RD, THETA-F1)})$$
$$\texttt{THETA-NF} = \texttt{cs}(i, \texttt{MUX(NF, RD, THETA-NF)})$$
$$\texttt{CFN} = \left\lfloor \frac{\texttt{THETA-F1} + \texttt{THETA-NF}}{2} \right\rfloor$$



Stream Equations for Optimized Sub-Circuit

```
HOLD = cs(false, F1 & ¬HOLD)
 CIN = HOLD & ¬NF
 OPT = cs(i,MUX(F1, RD, INC(OPT,CIN)))
```

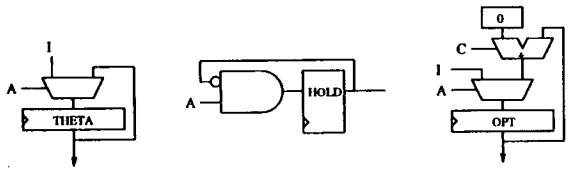


PVS Definitions for Circuit Verification

```
A,B,C,R: VAR Stream[bool]
a,b,c,r: VAR bool
I,J,K: VAR Stream[int]
i,j,k: VAR int

THETA(A,I,i): Stream[int]     %defined using corec

CFN(A,B,I,i,j): Stream[int]
        = DIV2(THETA(A,I,i) + THETA(B,I,j))

HOLD(A,a): Stream[bool]       %defined using corec

CIN(A,B): Stream[bool] = A AND NOT B

OPT(A,C,I,i): Stream[int]     %defined using corec
```


Recursive Stream Definitions

```
THETA(A,I,i) = cs(i,MUX(A,I,THETA(A,I,i)))
  HOLD(A,a) = cs(a, A & ¬HOLD(A,a))
OPT(A,C,I,i) = cs(i, MUX(A,I,INC(OPT(A,C,I,i),C) ))
```



Type Declarations for Assumptions on Input Signals

```
S(R): TYPE =
  {A |Invariant(IF R
                  THEN NOT tl(A)
                  ELSE A ⇒ tl(A)
                  ENDIF)}

C(R): TYPE =
  {I| Invariant(NOT R ⇒ EQ(tl(I),INC(I)))}
```

Correctness Theorem

```
Optimize_correct: THEOREM

∀ R,(RD : C(R)),(F1 : S(R)| ¬hd(F1)),
   (NF : S(R)|Invariant(NF ⇒ F1)),(i : int):

  CFN(F1,NF,RD,i,i) = OPT(F1,CIN(HOLD(F1,false),NF),RD,i)
```


Proof of Optimize_correct by co-induction

Define Bisimulation B as:

$\{(X,Y)|$
$\quad \exists\ R,(\mathtt{RD} : C(R)),(\mathtt{F1} : S(R)),(\mathtt{NF} : \{A : S(R)|A \Rightarrow \mathtt{F1}\}),(i : int)$
$\quad\quad (j : int|\mathtt{hd(F1)} \wedge \neg(\mathtt{hd(NF)}) \Rightarrow\ \mathtt{hd(RD)} = j + 1),$
$\quad\quad\quad (b : bool|\mathtt{hd(F1)} \wedge \neg(\mathtt{hd(NF)}) \Rightarrow\ b = \mathtt{odd?}(i + j)) :$
$\quad\quad \mathtt{X} = \mathtt{CFN(F1,NF,RD,}i,j\mathtt{)}\ \&$
$\quad\quad \mathtt{Y} = \mathtt{OPT(F1,CIN(HOLD(F1,}b\mathtt{),NF),RD,}\lfloor(i + j)/2\rfloor)\}$

Proof—B is a Bisimulation

Heads: For any $(X,Y) \in B$, $\mathtt{hd}(X) = \mathtt{hd}(Y) = \lfloor(i + j)/2\rfloor$.

Tails: For any $(X,Y) \in B$, show $(\mathtt{tl}(X),\mathtt{tl}(Y)) \in B$.

```
tl(CFN(F1,NF,RD,i,j))
      = CFN(tl(F1),tl(NF),tl(RD),
            IF hd(F1) THEN i ELSE hd(RD) ENDIF,
            IF hd(NF) THEN j ELSE hd(RD) ENDIF)

tl(OPT(F1,CIN(HOLD(F1,b),NF),RD,⌊(i + j)/2⌋)))
    = OPT(tl(F1),
          CIN(HOLD(tl(F1), (hd(F1)∧ ¬ b))),tl(NF)),
          tl(RD),
          IF hd(F1) THEN (⌊(i + j)/2⌋) + [b ∧ ¬hd(NF)]
          ELSE hd(RD)
          ENDIF)
```


Concluding Remarks

- Proof by co-induction effective technique for verifiying 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)