Formal Development of a Fault-Tolerant Clock Synchronization Circuit

Paul S. Miner
May 12, 1995

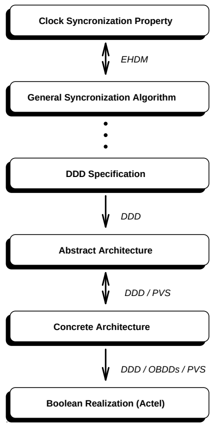
Outline

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

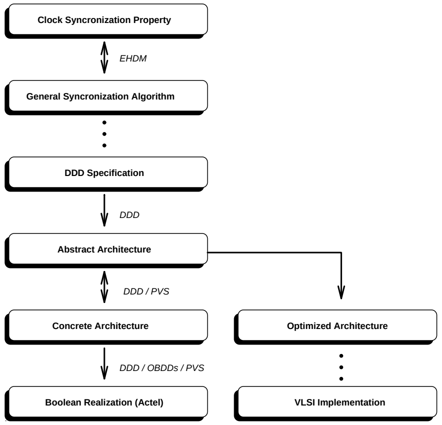
Prior Verification

- 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



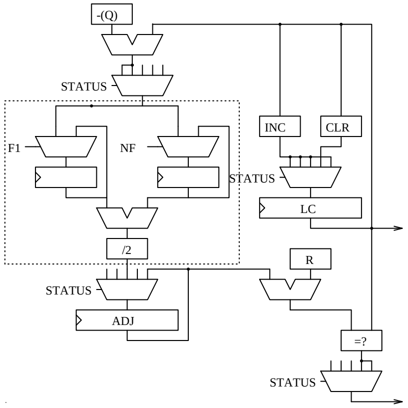
Design Hierarchy—New



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

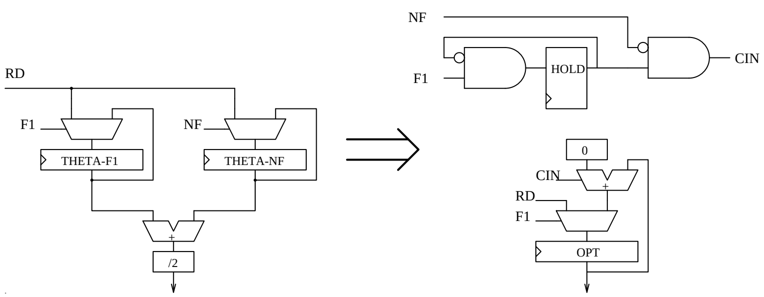
Intermediate Stage of Previous Derivation



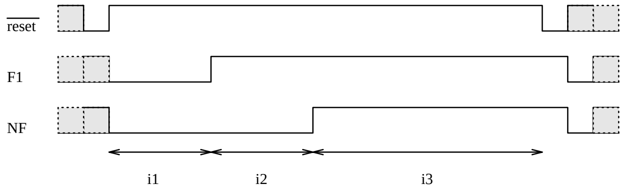
Intermediate Stage

- 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



Signal Assumptions Justifying Optimization



Signal `RD` is the output of a counter.

Verification of Optimized Circuit

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

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))
```

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

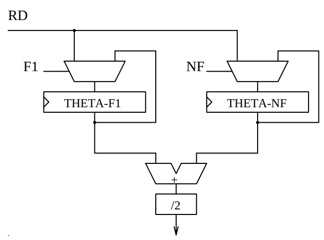
```
Stream_corec[alpha, beta: TYPE]: THEORY
  BEGIN
  IMPORTING Stream_adt[beta]

  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
```

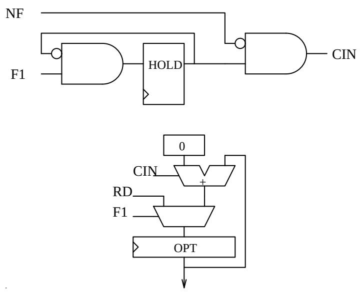

Stream Equations for Original Sub-Circuit

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



Stream Equations for Optimized Sub-Circuit

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

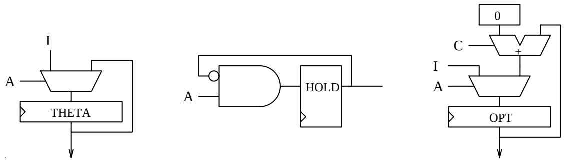


PVS Definitions for Circuit Verification

```
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

$\forall\ R, (\texttt{RD} : C(R)), (\texttt{F1} : S(R)|\ \neg\texttt{hd(F1)}),$
$\quad (\texttt{NF} : S(R)|\texttt{Invariant}(\texttt{NF} \Rightarrow \texttt{F1})), (i : int)$:

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:

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

Proof—B is a Bisimulation

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

Tails: For any $(X,Y) \in B$, show (tl(X),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)