

Verification of Asynchronous Circuits

Mark B. Josephs

Centre for Concurrent Systems and Very Large Scale Integration
School of Computing, Information Systems and Mathematics
South Bank University, London

Overview

- Motivation
- Modelling languages
- Verification of model within test harness
- Verification that one model can be substituted for another
- Conclusion

Motivation

ITRS, 2001:

“Communications-centric design... As it becomes impossible to move signals across a large die within one clock cycle or in a power-effective manner, or to run control and dataflow processes at the same clock rate, the likely result is a shift to an asynchronous design style...”

Modelling languages

C.A.R. Hoare, Communicating Sequential Processes, CACM, 1978:

“This paper suggests that input and output are basic primitives of programming and that parallel composition of communicating sequential processes is a fundamental program structuring method. When combined with a development of Dijkstra's guarded command, these concepts are surprisingly versatile.”

The concepts of CSP have been widely used in asynchronous design flows since the mid 1980's

- A.J. Martin's group (Caltech)
- C.H. van Berkel's group (Philips Research)
- E. Brunvand (then at CMU) and R.F. Sproull
- S.B. Furber's group (Manchester)

Hoare made the important point that synchronization of $c!e$ with $c?x$ should have the same effect as $x = e$

Verification of model within test harness

SPIN — verification tool for concurrent systems

- useful for analyzing the logical consistency of asynchronous systems
- normally applied to distributed software and communication protocols
- uses CSP-based modelling language
PROMELA = PROcess MEta LAnguage

“New York, March 6, 2002 ... The Association for Computing Machinery (ACM) has recognized Dr. Gerard Holzmann for his contribution to a widely used software package called SPIN that quickly detects defects in networked computers, making them more reliable. Dr. Holzmann, director of research at Bell Labs, a division of Lucent Technologies developed SPIN to assure dependability in complex telephone switching systems, which have stringent reliability requirements. But as computers became more powerful, this verification tool has found applications in everything from Internet protocol software and railway signaling systems to distributed control software essential to nuclear power plants and spacecraft. Dr. Holzmann will receive the ACM Software System Award, which carries a \$10,000 prize.”

```
#define N 5 /* N-1 stage pipeline*/
#define M 2 /* test harness injects M data items
            and recirculates data upon output */
#define QSZ 0 /* synchronized communication */

chan data[N] = [QSZ] of { byte };

...

init {
    atomic {
        int i = 1;
        do :: i<N -> run stage(i); i++
           :: i==N -> break
        od;
        run test()
    }
}
```

```
proctype stage(int j) {
  byte x;
  do :: data[j-1]?x -> data[j]!x
  od
}
proctype test() {
  byte m = 0;
  do :: m<M -> data[0]!m; m++
      :: m==M -> break
  od;
  do :: data[N-1]?m -> data[0]!m
  od
}
```

SPIN can check this model of pipeline in test harness is free from deadlock

```
#define QSZ 1 /* delay-insensitive communication
              */
```

```
mtype = { t }; /* acknowledgement token */
```

```
chan ack[N] = [QSZ] of { mtype };
```

```
proctype stage(int j) {
```

```
    byte x;
```

```
    do :: data[j-1]?x -> assert( nfull(ack[j-1]) );
```

```
        ack[j-1]!t;
```

```
        assert( nfull(data[j]) );
```

```
        data[j]!x;
```

```
        ack[j]?t
```

```
    od
```

```
}
```

```

proctype test() {
  byte m = 0;
  do :: m < M -> assert( nfull(data[0]) );
    data[0]!m; ack[0]?t; m++
  :: m == M -> break
od;
do :: data[N-1]?m -> progress:
  assert( nfull(ack[N-1]) );
  ack[N-1]!t;
  assert( nfull(data[0]) );
  data[0]!m; ack[0]?t
od
}

```

SPIN can check this less abstract model is free from deadlock, transmission interference and divergence

Verification that one model can be substituted for another

Two approaches:

- Re-check (with SPIN) against the original set of test harnesses
- Verify (with another tool) that an appropriate formal relationship holds between the two models
 - FDR2 from Formal Systems and the Edinburgh Concurrency Workbench are tools worthwhile investigating for this purpose

Conclusion

- CSP-based languages are popular in the asynchronous circuit design research community
 - models often input to silicon compiler that generates handshaking circuits
- Formal verification of models is rarely done
- Verification tools developed by the formal methods research community may be applicable
 - state-explosion is always going to be a problem