



Formal Verification on GCD Unit

Abhinandan Majumdar
MS. Computer Engineering
Columbia University



Outline

- VIS – A Model Checking Tool
- GCD Algorithm
- RTL Design
- Properties Tested
- Experience
- Conclusion



VIS – A Model Checking Tool

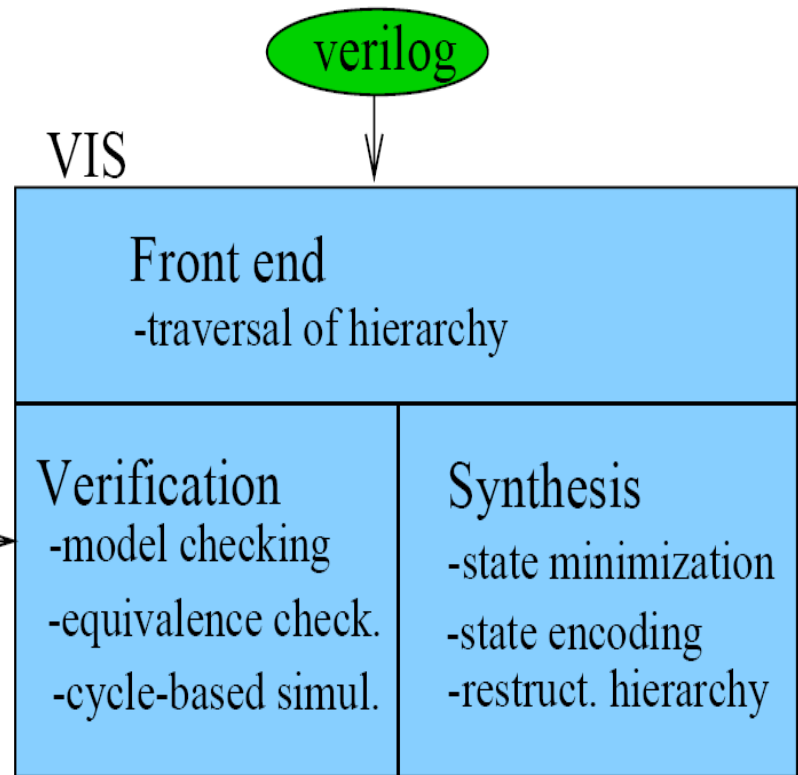
- VIS stands for Verification Interaction with synthesis.
- A verification and synthesis system for finite-state hardware systems.
- Developed at Berkeley and Boulder.

VIS Architecture

Consists of three parts

- Front End
- Verification
 - Optimizes various logical parts of the design
- Synthesis

CTL →





VIS – Front end

- A front-end to read and traverse a hierarchical system.
- System must be described in BLIF-MV.
- High level language like Verilog can be generate to BLIF-MV file using compilers like vl2mv.



VIS - Verification

- Performs Exhaustive search of the state space of the concurrent system to determine truth of specification.
- Properties are given in CTL (Computation Tree Logics) form.
- Also performs fairness constraints check.
- Also performs language emptiness check.

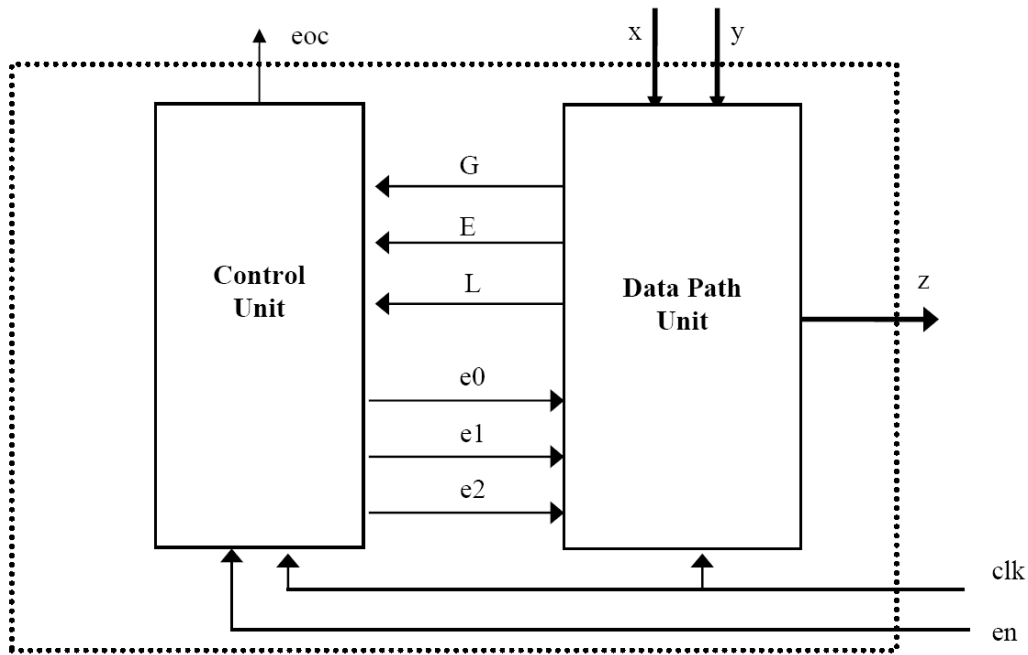


GCD Euclid's Algorithm

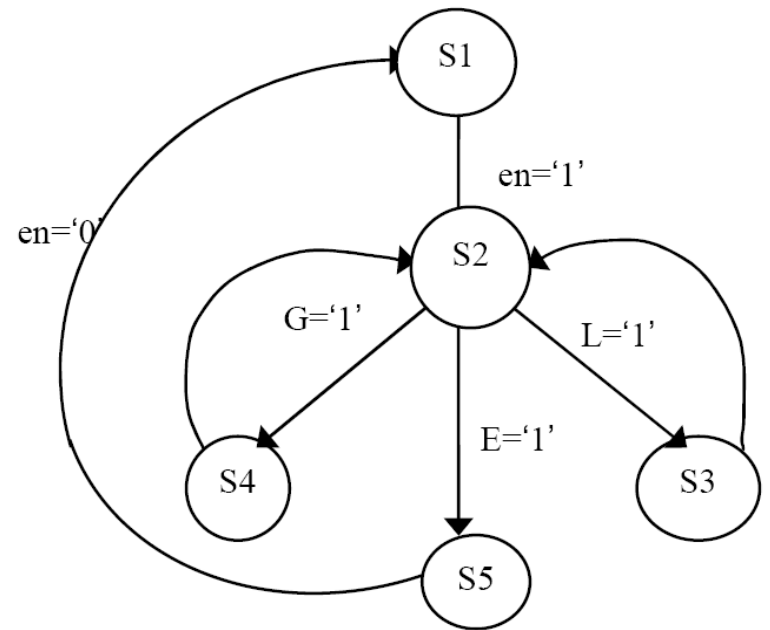
- GCD stands for Greatest Common Divisor.
- Algorithm

```
Input x, y;
while ( x ≠ y ) do
  if ( x > y )
    then x := x - y;
    else y := y - x;
  endif;
endwhile;
z := x;
end;
```

Design

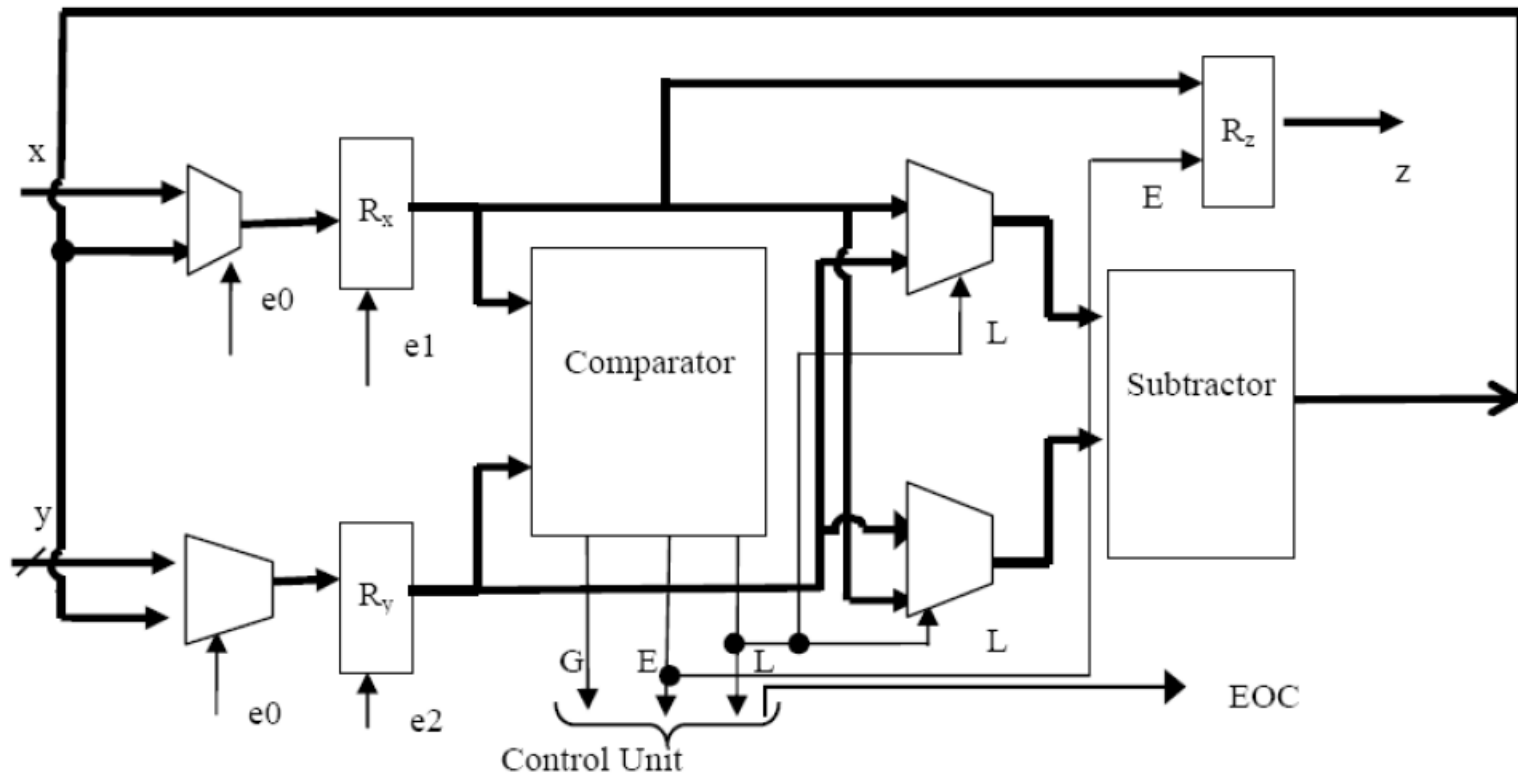


Block Diagram



Control Unit

Datapath Unit





Properties Tested

- The iteration of GCD Algorithm should eventually converge for almost all input possibilities (start=1 \rightarrow AF(eoc = 1)).
- For a particular input case where one of the input is zero, the system must fail (start=1 \rightarrow AF(eoc = 1)).
- After comparison, only one of the enable input of RegX and RegY should be 1 (start = 1 \rightarrow AG(!(e1=1) ^ (e2=1))).



Results Obtained

- Tested for both the original design and one of the sample example provided with VIS.
- Property 1 & 3 passed for both the designs.
- Property 2 holds true which we expect to FAIL.
- Example Code has additional support to take care of this particular input case.



Experience

Good

- Gave a hand on experience on working upon Model Checking Tool.
- Gained a thorough insight on how the properties are written and how the suspecting failure cases are detected, diagnosed and rectified.

Bad

- Huge time was spent on converting the VHDL code to ACCEPTABLE Verilog Code by VIS.
- VIS doesn't accept all possible Verilog language constructs, hence a portion of time was spent on converting the code so as to yield expected simulation results.



Conclusion

- We could formally verify the failure cases which might exist in a FAULTY design of GCD.
- We know for a particular case when the algorithm would fail, but couldn't prove it.
- We might need to add additional hardware support to overcome the particular input case.



References

- VIS User's Manual by Tiziano Villa, Gitanjali Swamy, and Thomas Shiple.
- http://en.wikipedia.org/wiki/Euclidean_algorithm