EHDL

Easy Hardware Description Language

COMS 4115: Programming Languages and Translators, Fall 2011

Paolo Mantovani (pm2613)
Mashooq Muhaimen (mm3858)
Neil Deshpande (nad2135)
Kaushik Kaul (kk2746)

Overview & Motivation

- Why VHDL
  - Language in itself very verbose and low level
  - Becomes very complex as the complexity of digital system increases
  - Well understood problem domain

- Goals of EHDL
  - Simple C like syntax, so flat learning curve
  - Succinct and straightforward, will help in increasing productivity of hardware engineers
  - Easy to grasp imperative style of coding

Tutorial

- Start off – Nothing different from the ordinary
  - Open your favorite editor
  - Start off with the function main().
  - Write EHDL code within this function. May also create your own functions.
  - Save the file with '.ehdl' extension
  - Call the Ehdl compiler on the target file
- Data types: *int, array*
- Operations
  - Arithmetic Operations
  - Logical Operations
  - Binary Operations
  - Unary Operations

Tutorial

adder.ehdl

```
int(32) c main (int(32) a, int(32) b ) {
                        c = a + b;
        }
```

./ehdl -o adder.vhd adder.ehdl

Four_to_one_mux.ehdl

```
int(8) z main (int(8) a, int(8) b,
    int(8) c, int(8) d, int(2) sel ) {
    switch ( sel ) {

            case 0: z = a;
            case 1: z = b;
            case 2: z = c;
            default:z = d;


            }
    }
```

./ehdl -o adder.vhd adder.ehdl

POS

```
(int(1) sum, int(1) carry) fulladder(int(1) a, int(1) b, int(1)
carryin){

        sum = a ^ b ^ carryin;
        carry = (a && b) ^ (carryin && (a ^ b));

}


(int(4) s, int(1) overflow) main(int(4) a, int(4) b, int(1) carryin) {

        int(1) sum[4];
        int(1) carry[4];

        (sum[0], carry[0]) = fulladder(a(0),b(0),carryin);
        POS(1);
        (sum[1], carry[1]) = fulladder(a(1),b(1),carry[0]);
        POS(1);
        (sum[2], carry[2]) = fulladder(a(2),b(2),carry[1]);
        POS(1);
        (sum[3], carry[3]) = fulladder(a(3),b(3),carry[2]);
        POS(1);

        s(3) = sum[3];
        s(2) = sum[2];
        s(1) = sum[1];
        s(0) = sum[0];
        overflow = carry[3];
}
```

While Loop

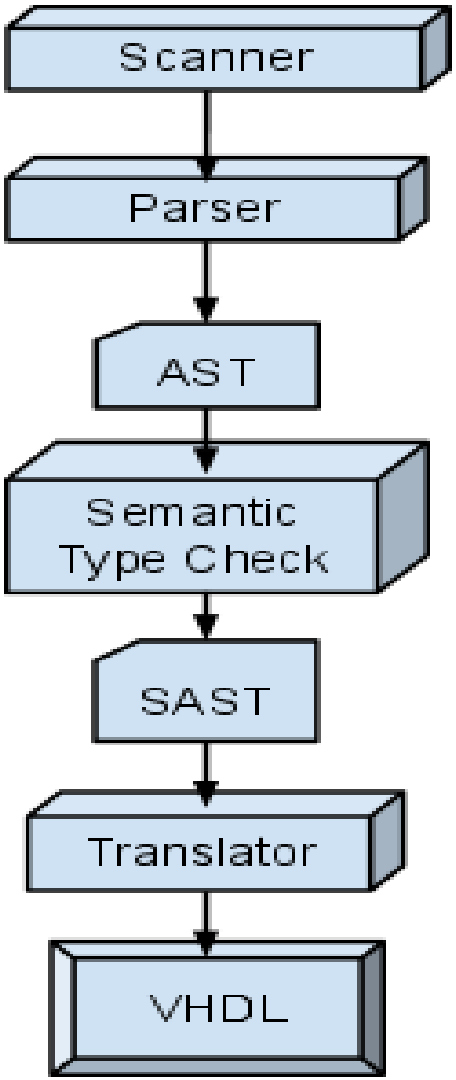
```
/* gcd */
    Int(8) c main(int(8) a, int(8) b){

            while (a != b) {
                if (a > b) {
                        a = a - b;
                }
                else{
                        b = b - a;
                }

            POS(1);
            }

    POS(a==b);
    c =a ;

    }
```

```
/* primes */
(int(32) primes=2) main (int(32) m) {
                int(1) a[200];
                int(1) sig;
                int(32) n = 2;
                int(32) k = 2;

                while (n <= m) {
                   if ((a[n] == 0) && (k <= m)) {
                                if (k == n) {
                                            primes = n;
                                } else {
                                            a[k] = 1;
                                }
                                k = k + n;
                }else {
                                n = n + 1;
                                k = n + 1;
                }
            }
}
```

Trafficlight.ehdl

```
const int(2) HG = 0;
const int(2) HY = 1;
const int(2) FG = 2;
const int(2) FY = 3;
const int(8) YDuration = 2;
const int(8) FDuration = 3;


(int(1) hwGreen, int(1) hwYellow, int(1) farmGreen, int(1)
farmYellow)
main ( int(1) car ) {

        int(2) state;
        int(8) yCntr;
        int(8) fCntr;

        state = HG;
        while (1) {
            switch ( state ) {
               case HG:
                        hwGreen = 1; hwYellow = 0;
                        farmGreen = 0; farmYellow = 0;
                        if ( car == 1 ) {
                            state = HY;
                            yCntr = 1;

                        }
              case HY:
                         hwGreen = 0; hwYellow = 1;
                        farmGreen = 0;farmYellow = 0;

                        yCntr = yCntr + 1;
                        if ( yCntr == YDuration ) {
                            state = FG;
                            fCntr = 1;
                        }
case FG:
                        hwGreen = 0;hwYellow = 0;
                        farmGreen = 1;farmYellow = 0;

                        fCntr = fCntr + 1;
                        if ((car == 0) || ( fCntr == FDuration )) {
                                    state = FY;
                                    yCntr = 1;
                        }


case FY:
                        hwGreen = 0; hwYellow = 0;
                        farmGreen = 0; farmYellow = 1;

                        yCntr = yCntr + 1;
                        if ( yCntr == YDuration ) {
                                    state = HG;
                        }

                    }
            }

}
```

Compiler Architecture



Lessons Learned

- Team-oriented development : complementary strengths
- Interface-oriented design: Some instances where other teams members had to wait
- Version control systems: SVN was a good productivity tool but we could have used more branches to cut the wait times
- Test suite : Helped uncover a ton of bugs
- Writing tests : Helped improve understanding of semantics
- Code coverage : Again, helped catch bugs by forcing us to devise new test cases
- Eclipse: is cranky

More lessons learnt

- Same syntax – wildly different semantics
- List.fold_left()
- Ocaml has for loops !!!