UNI-corn

A Java-like hardware description language

Agenda

Introduction (Gael)

Language Features (Dan)

Compiler Architecture (Lalo/Adiza)

Project Plan (Gael)

Testing (Maryam)

Lessons Learned (All)

Demo (Lalo)

Introduction

Team Members

gael

project manager

lalo

sys. architect

maryam

tester

dan

lang. guru

adiza

wildcard

Background

What: A simple hardware description language (HDL)

Why:

- Great HDL languages out there
- Syntax unfamiliar for CS students starting in Java/C++
- UNI-corn has Java-like syntax

Um...why the name?

- Only one data type - binary strings

Language Features

Building Blocks

Buses

```
id = (0 | 1)*b ⇒ e.g. a = 101b;
```

Gates

```
and, or, xor, not, nand, nor, xnor,
```

Modules

```
modID( ε | in1<N>...inM<N>) {
     ε | expr1;...;exprK;
     out: ε | expr1;...;exprK;
}
```

Registers

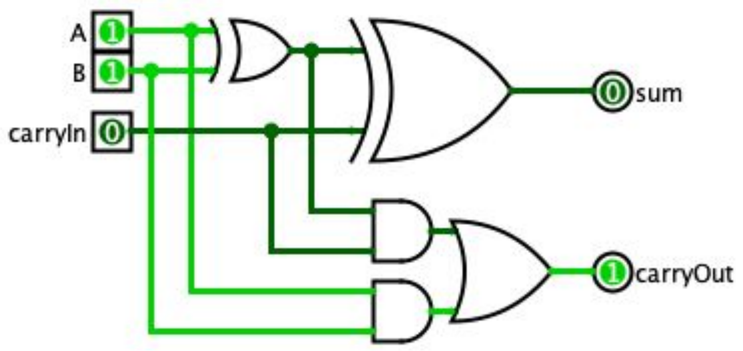
```
id := bus *initial bus*;
```

Loops

```
for i in N { expr0;...;exprK; };
```

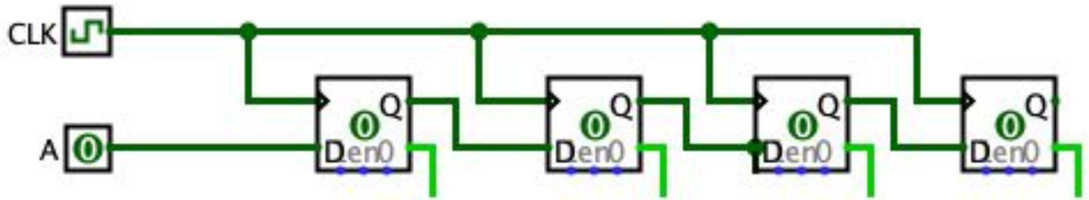
Combinational Logic

```
fullAdder(a<1>, b<1>, cin<1>) {

      sum = (a xor b) xor cin;
      cout = (sum and cin) or (a and b);

      out: sum<1>, cout<1>;

}

main() {

      a = 1b;
      b = 1b;
      c = 0b;

      print s : fullAdder(a,b,c)[0];
      out:;

} 
```



Sequential Logic

```
shift4Reg(a<4>) {

      b1 := a *0000b*;
      b2 := b1 *0000b*;
      b3 := b2 *0000b*;
      b4 := b3 *0000b*;

      out: b1<4>, b2<4>, b3<4>, b<4>;

}

main() {

      a = 1000b;
      print s : shift4Reg(a);
      out:;

} 
```



Bringing It All Together

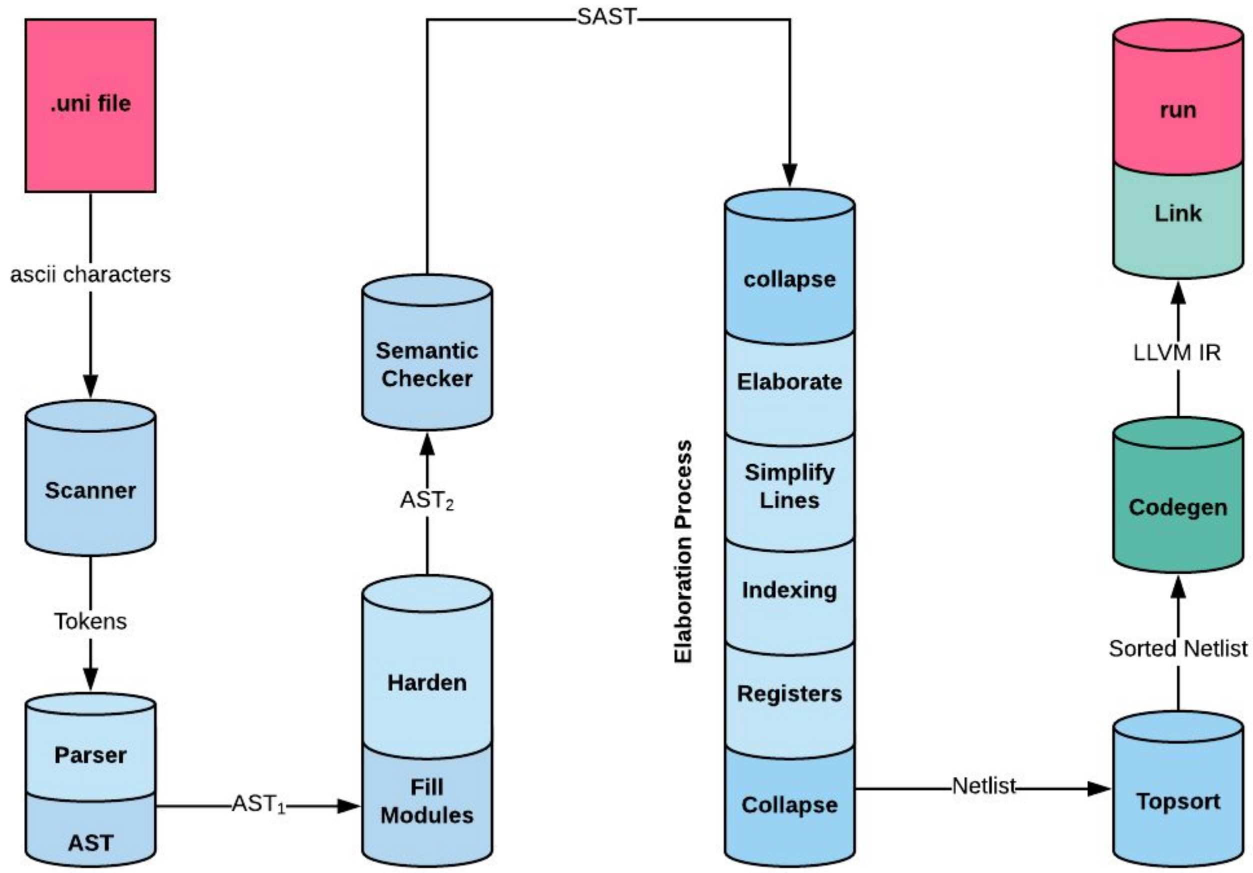
```
main() {

     a = 1011010b;
     b = 0011101b;
     m = modA(a,b)[sum];

     print m: m;

     out:;

}
```

```
fA(a, b, cIn) {

     axb = a xor b;
     sum = axb xor carryIn;
     carry = (axb and cIn) or
          (a and b);

     out: sum, carry;

}
```

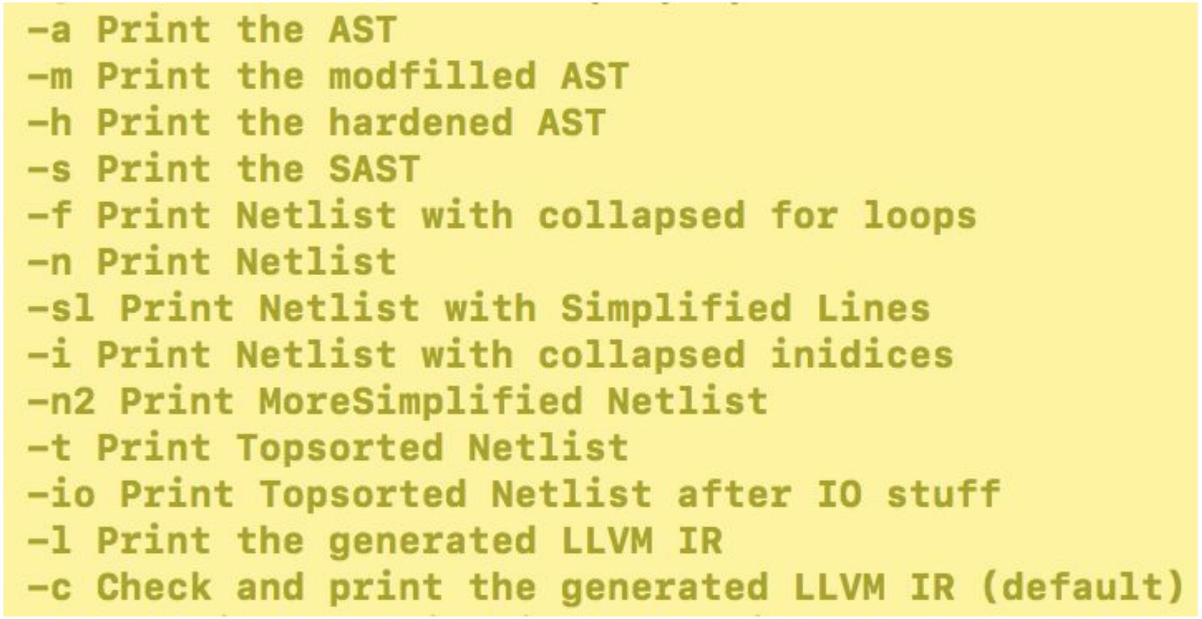
```
modA(a<n>,b<n>){

     c[0] = 0b;
     for (i from 0 to n-1) {
          sum[i] = fA(a[i],
          b[i],c[i])[sum];
          c[i+1] = fA(a[i],
          b[i],c[i])[carry];
     };

     out:sum<n>;

}
```

Compiler Architecture

Overview


.uni file
ascii characters
Scanner
Tokens
Parser
AST
AST1
Fill Modules
Harden
AST2
Semantic Checker
SAST
collapse
Elaborate
Simplify Lines
Indexing
Registers
Collapse
Elaboration Process
Netlist
Topsort
Sorted Netlist
Codegen
LLVM IR
Link
run

Flags



```
-a Print the AST
-m Print the modfilled AST
-h Print the hardened AST
-s Print the SAST
-f Print Netlist with collapsed for loops
-n Print Netlist
-sl Print Netlist with Simplified Lines
-i Print Netlist with collapsed inidices
-n2 Print MoreSimplified Netlist
-t Print Topsorted Netlist
-io Print Topsorted Netlist after IO stuff
-l Print the generated LLVM IR
-c Check and print the generated LLVM IR (default)
```

Fancy / Highlights from Compiler

Generics and loops

```
main(b) {

    modA(101b);
    out:c;

modA(a<n>){
    for(i to 4){
        b[i] = a[i];
    };
    out:;

}
```

C-linking

```
extern b_0;
extern c_0;

int main() {
    tick();
    b_0 = c_0;
    tick();
}
```

Features To Come:

-Multi-file compilation

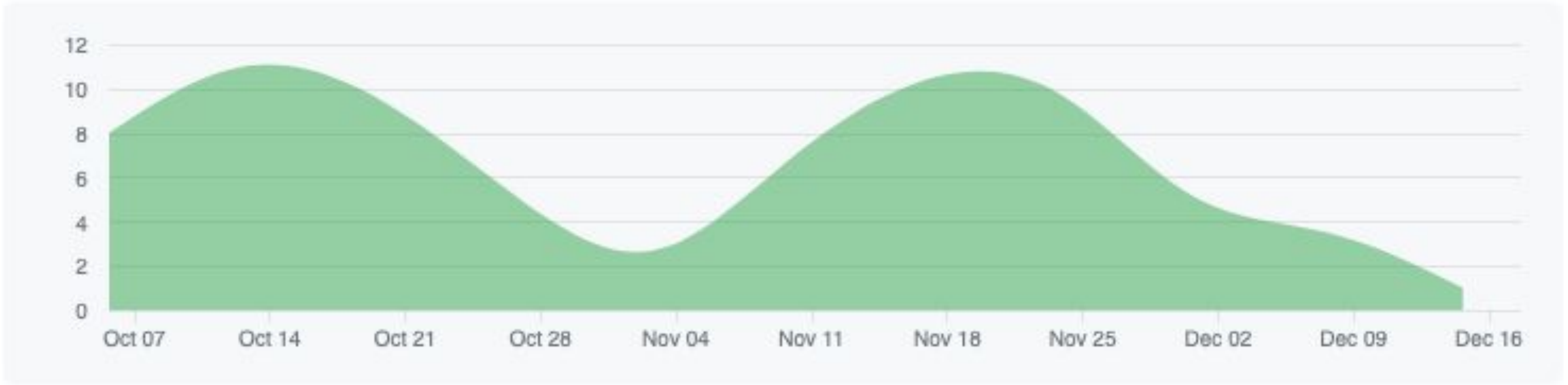
Project Plan

Timelines and Owners

DELIVERABLE	LEAD	CONTRIBUTOR(S)	COLLABORATORS	FEATURES	DEADLINE
KEY MILSETONES					
Proposal	**Gael**	Rest	N/A	N/A	Sept. 19
LRM	**Dan, Maryam**	Rest	N/A	N/A	Oct. 15
Hello World	**Lalo**	Gael, Maryam	N/A	N/A	Nov. 14
COMPILER					
Scanner.mll	**Gael**	Adiza, Dan	N/A	- syntax error checking	Oct 01
Modfill.ml	**Lalo**	Maryam	Gael	- basic modules - mutuall rec. loops	Oct 27
Semant.ml	**Lalo**	N/A	N/A	- variable declaration (scope) - type matching	Oct 29
Elaborate.ml	**Lalo**	Maryam	Gael		Dec 03
Topsort.ml	**Lalo**	Gael	Dan	- topologically sorted gates	Dec 10
Codegen.ml	**Lalo, Maryam**	Gael	N/A	Conjunction with above features deadlines	Conjunction with above feature deadlines
Test Suite	**Maryam**	Gael	Lalo	- break stuff (see plan)	Same as above
SUBMIT COMPILER					Dec 19
FINAL REPORT					
Final Report	**Gael**	Dan	Adiza		Dec 03, 10, 12
Final Presentation	**Gael**	N/A	N/A		Dec 10, 12
Demo	**Lalo**	Maryam	N/A		Dec 19

Many details excluded here, included in Final Report

Commit History Highlights



Lalo: 84 | Gael: 47 | Maryam: 42 | Dan: 12 | Adiza: 12

Testing

Plan and Strategy

- Scanner & Parser (Pretty Print)
- Testing the pipeline process
- Unit Testing
- Errors in Complicated Program
- Integration Testing
- Automated Testing

Unit Testing Strategy (per feature)

./testCases

```
./comments            ./indexing              ./registers
./creatingBuses       ./keywords              ./programs
./EOFTerminators      ./Main
./evaluatingGates     ./overloading
./gatePrecedence      ./printFunc
```

Results and Learnings

- Importance of Unit Testing

- Neigh!

- Double Negation

Lessons Learned

Lessons Learned

Gael: Being strategic about workflow from the start is key

Adiza: I learned about software development in a team setting.

Maryam: Time is not your friend in this class. Plan your every move! Start early! Use all the available resources to you

Lalo: Complexity breeds chaos. Work incrementally.

Dan: Teamwork and good communication are intangible yet valuable skills that can greatly help the development process