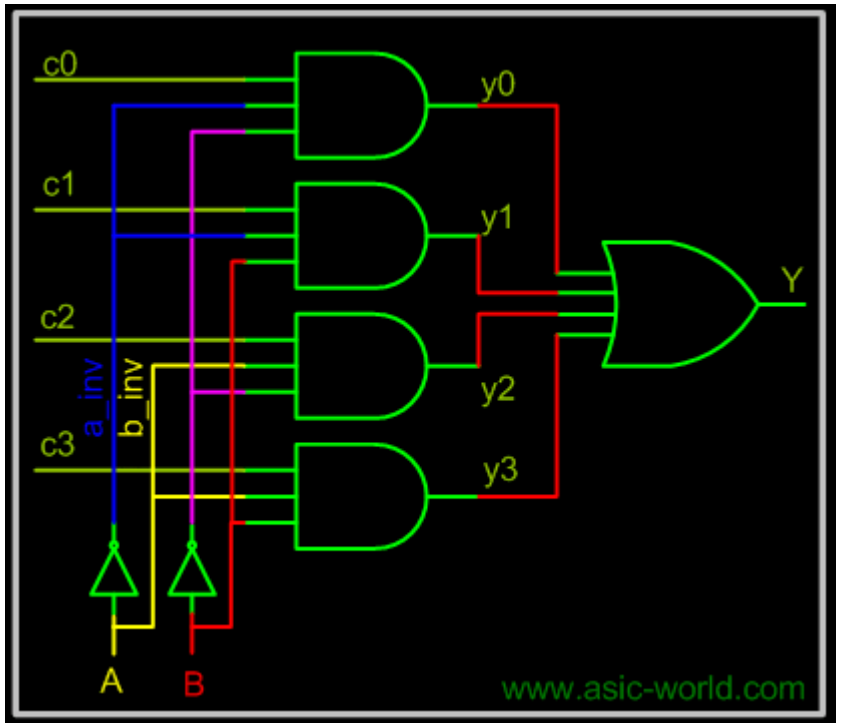
iClicker Attendance

Please click on A if you are here:

A. I am here today.

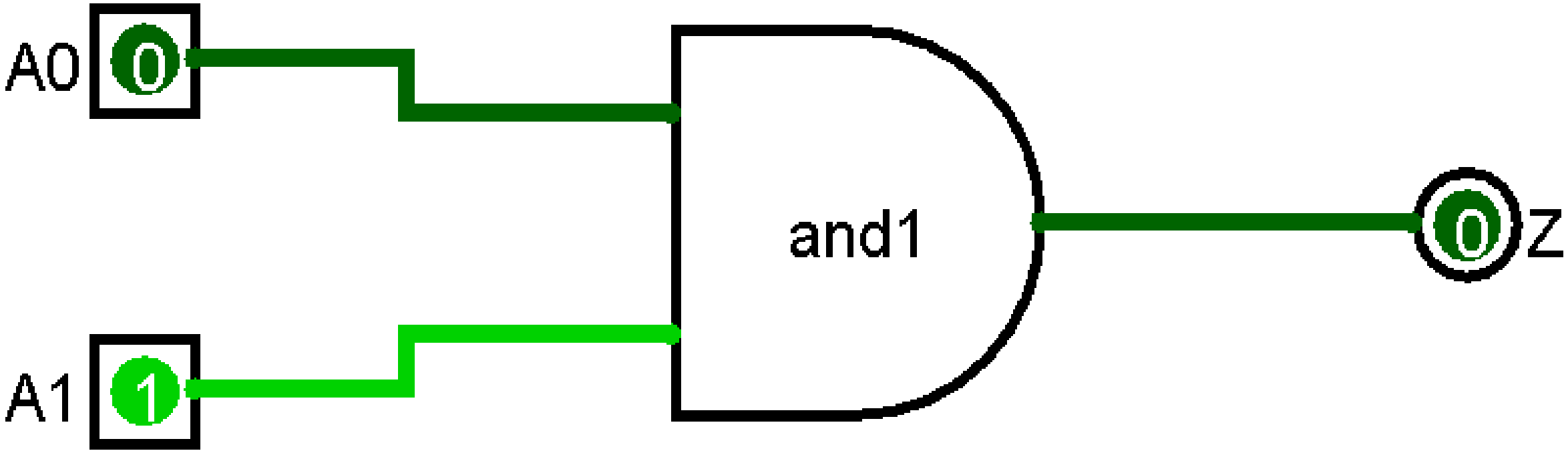


Verilog
(Project lecture)

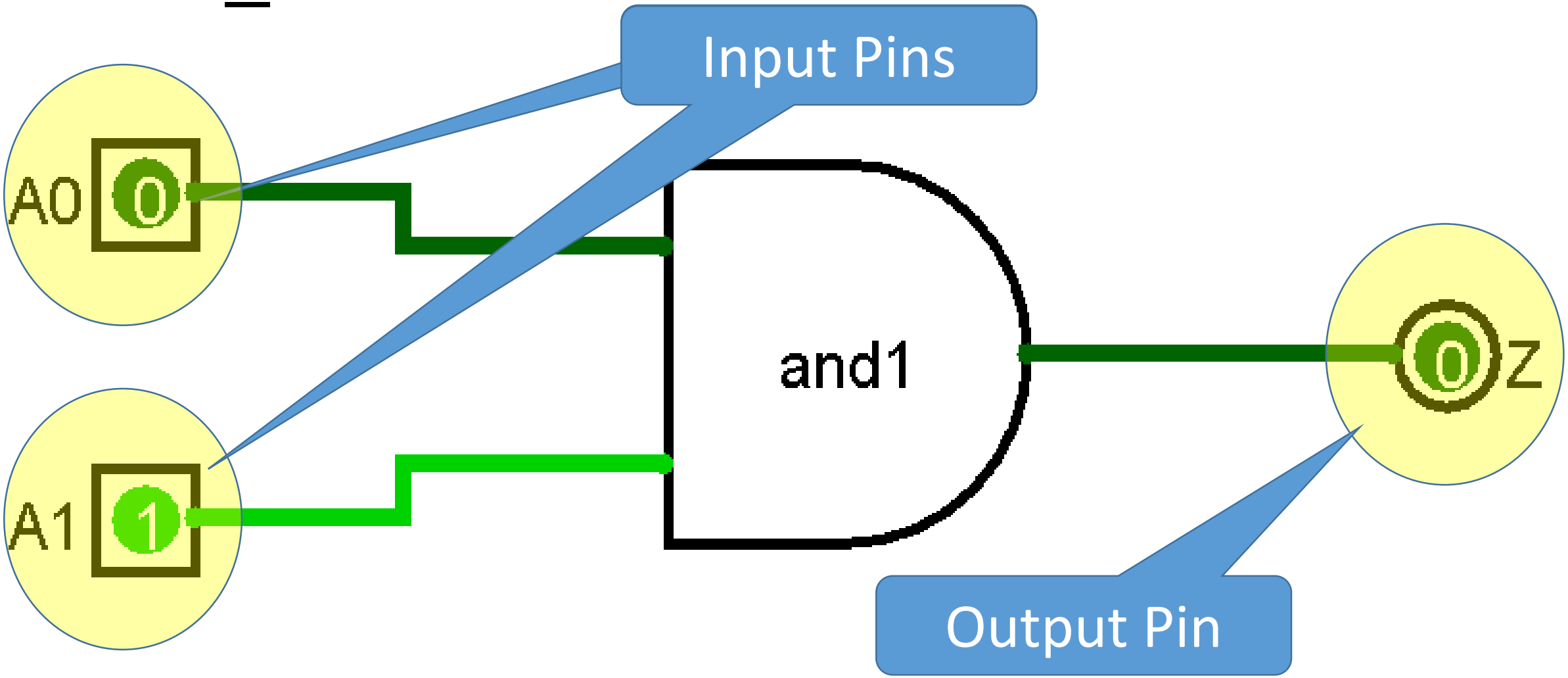
Verilog

- Hardware Description Language (HDL)
- Used to describe electronic circuits
- Text based (so computers can read), but has the same information as a circuit schematic
- For our project, we are using a simplified subset of Verilog
- For our purposes, everything in Verilog is pins, nets, and gates
  - we will expand the concept of “gates” in the final installment

AND_2 Schematic



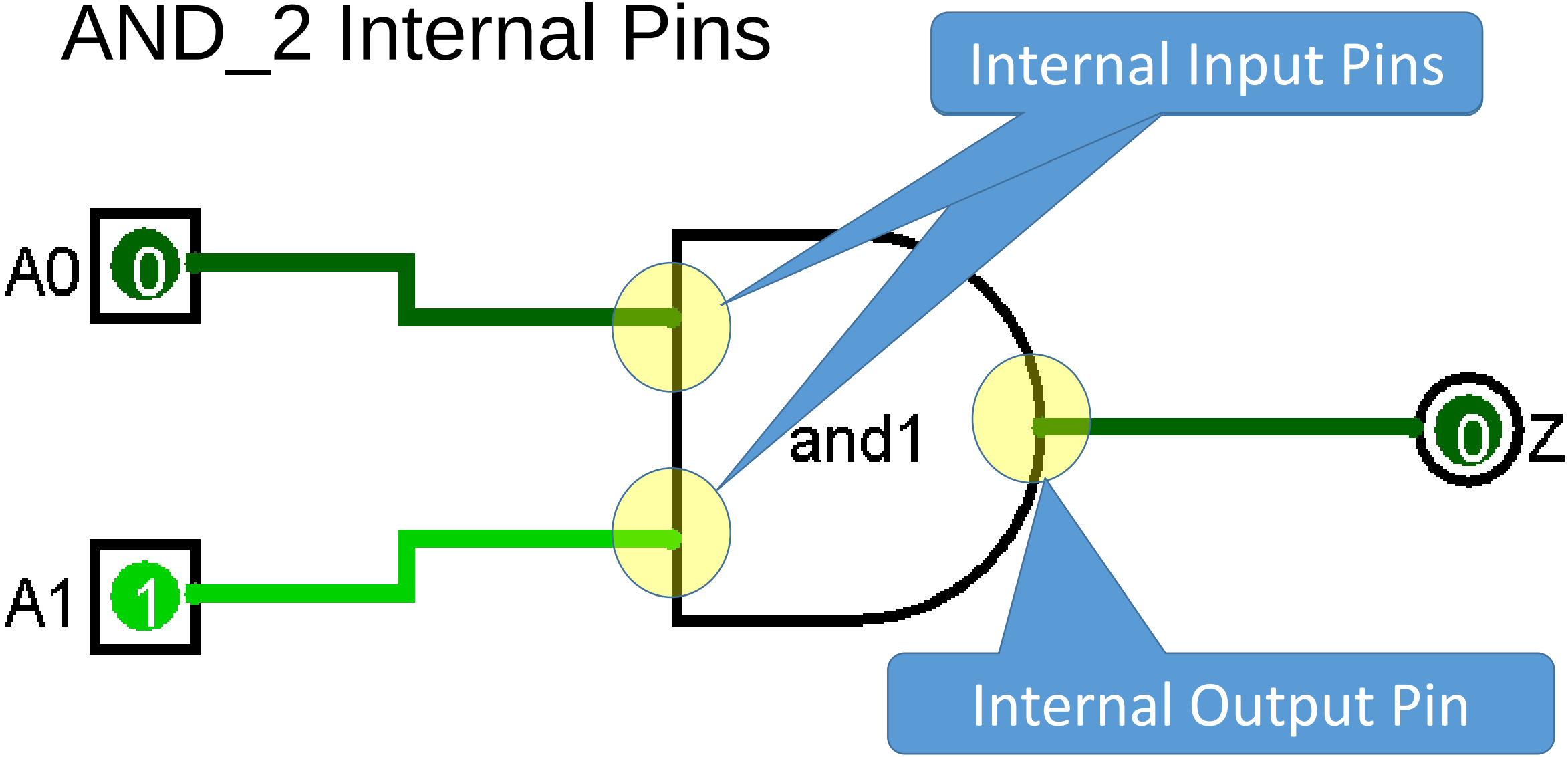
AND_2 Module Pins



Module Pins

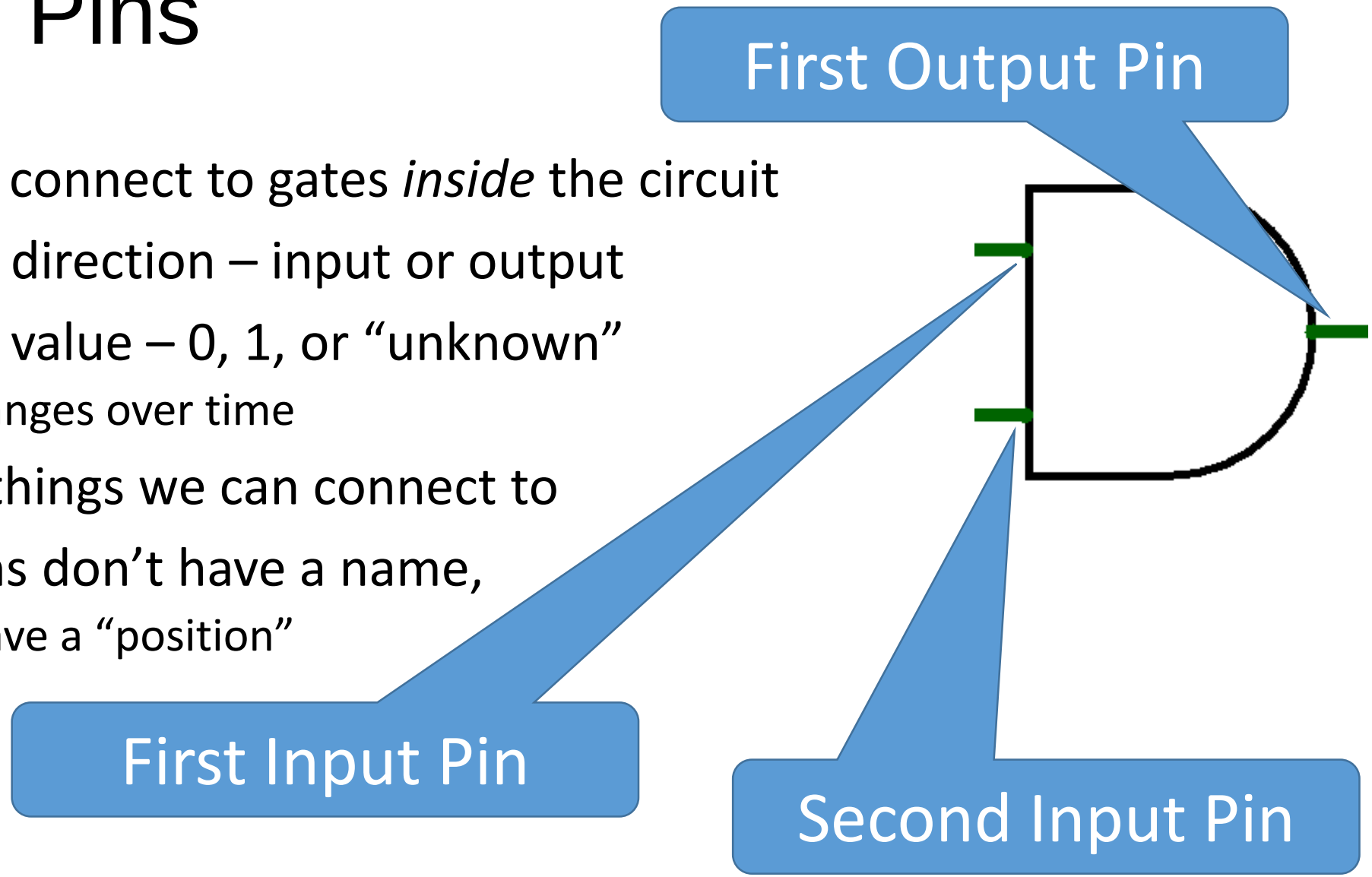
- Module Pins are the connection to the outside world
- Pins have a name, e.g. A, B, SEL, Z
- Pins have a direction – input or output
- Pins have a value – 1 or 0 or “unknown”
  - values may change over time
- Physically, things we can connect to

AND_2 Internal Pins

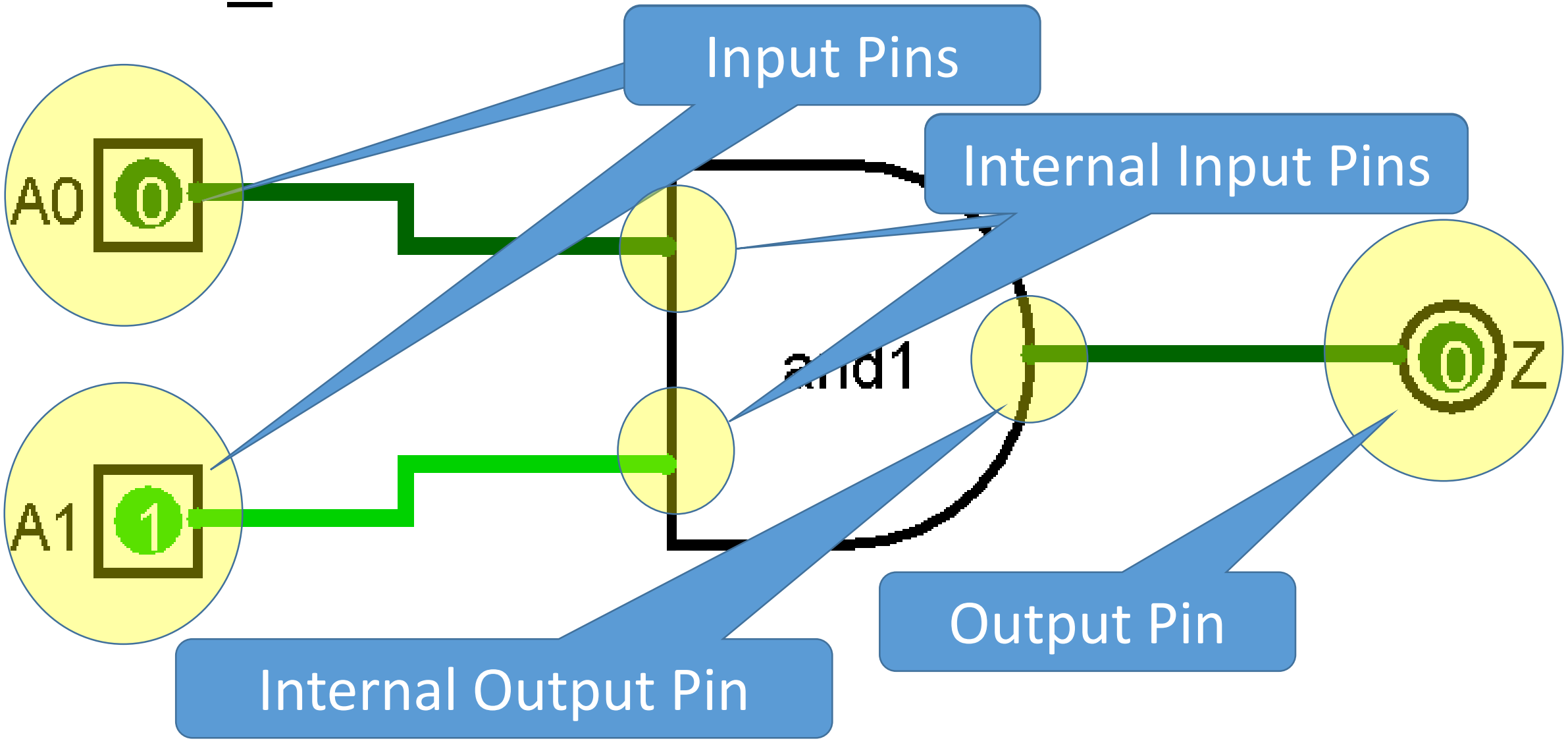


Internal Pins

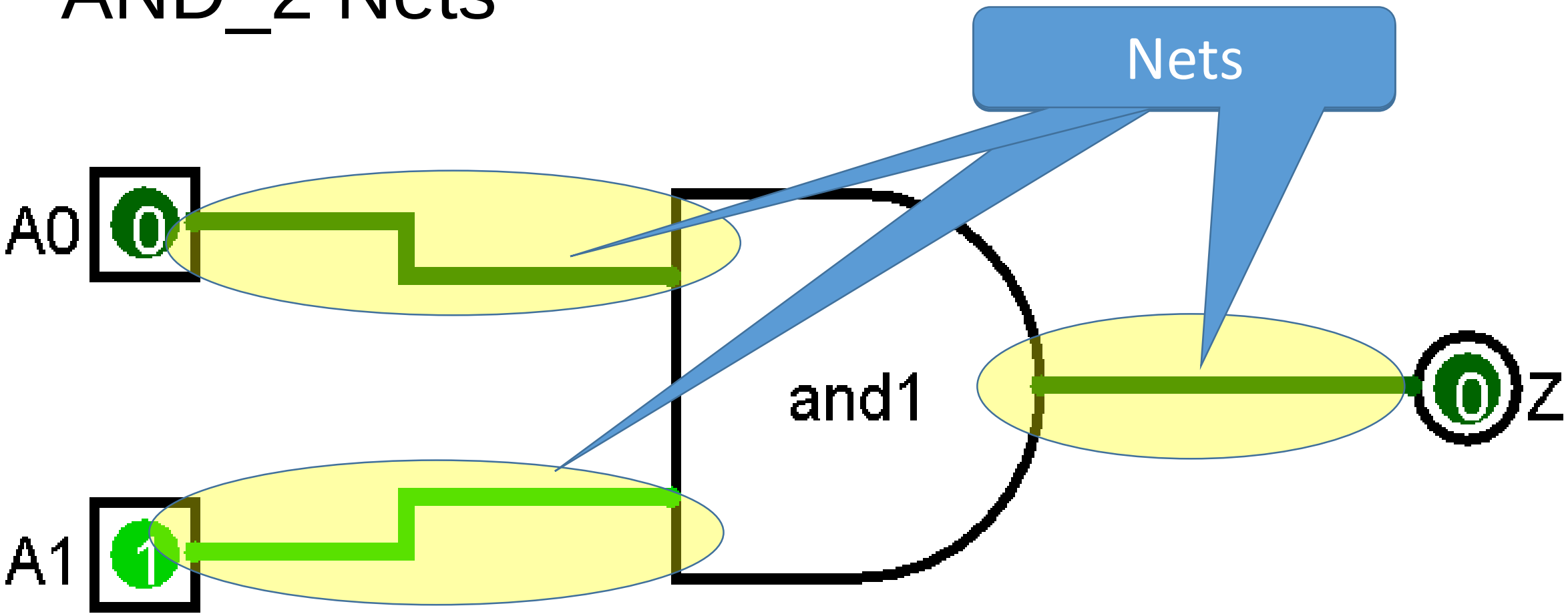
- Allow us to connect to gates *inside* the circuit
- Pins have a direction – input or output
- Pins have a value – 0, 1, or "unknown"
  - Value changes over time
- Physically, things we can connect to
- Internal pins don't have a name,
  - But do have a "position"



AND_2 All Pins

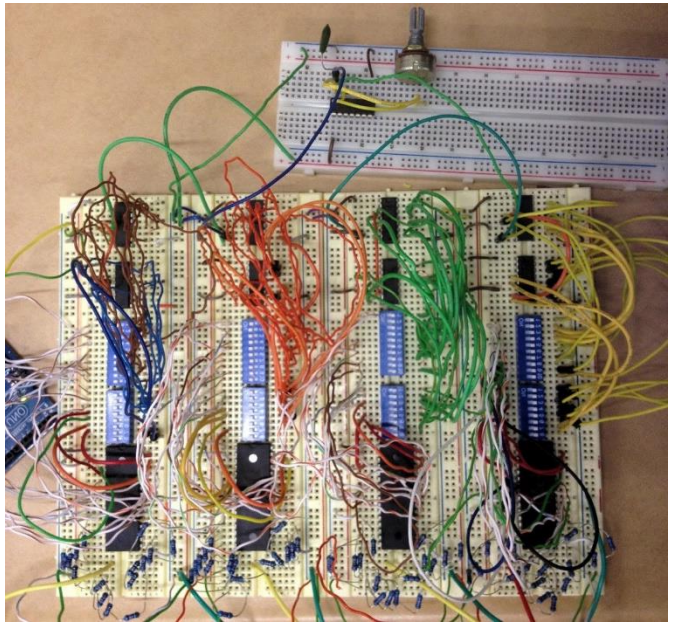


AND_2 Nets

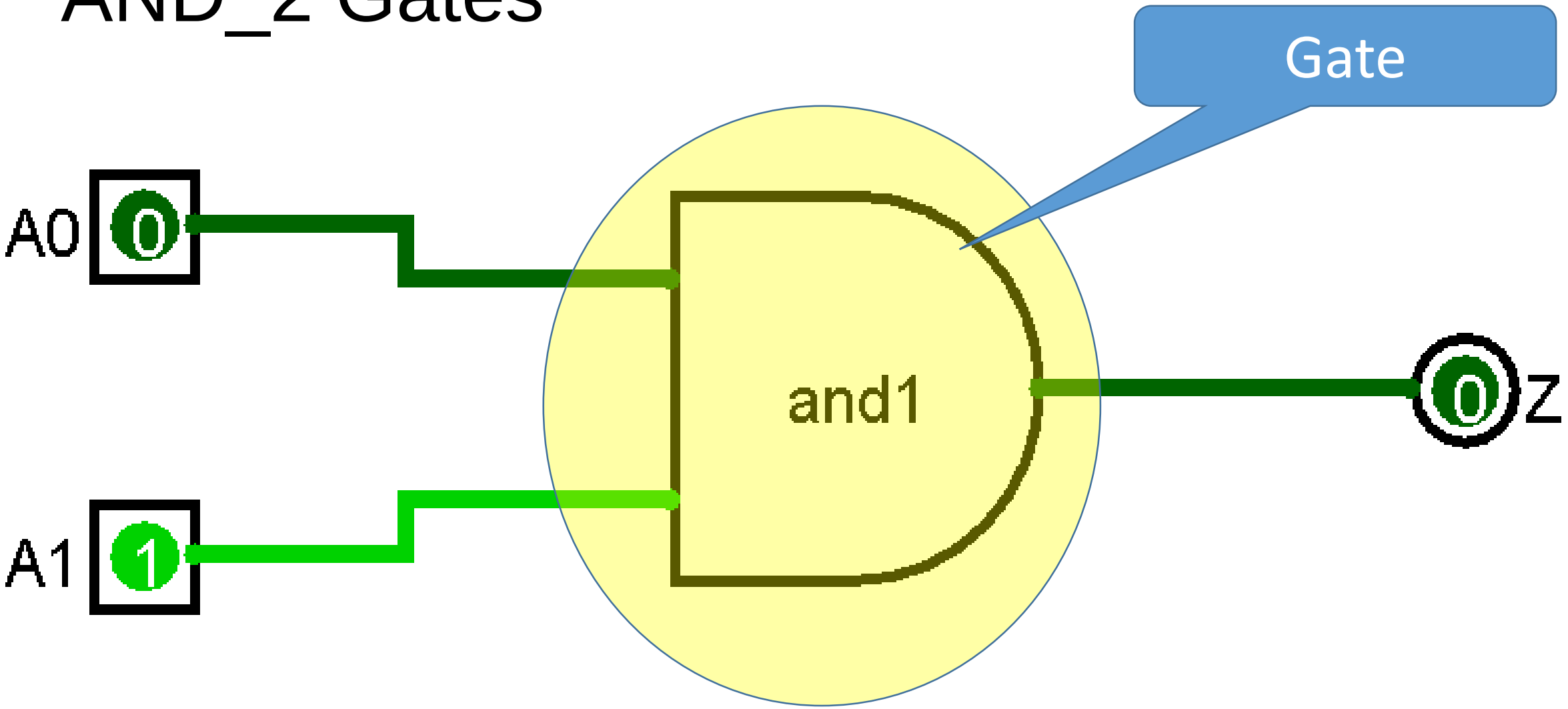


Nets

- Connect pins to each other (like wires)
- Have a net name (may match a pin name)
- Have a net value 0(dark green), 1 (light green) or unknown (blue)
- Have exactly one source pin
  - The value of the net is the value of the source
- Have one or more sink pins
  - Where the net value is consumed



AND_2 Gates

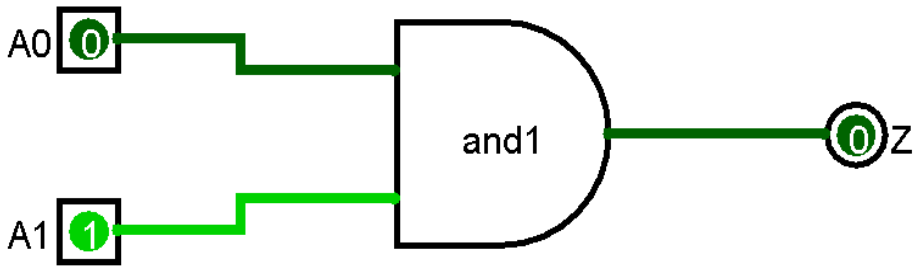


Gates

- Gates have a gate type: inv, and, or, xor
  - Each with a specific behavior
  - Note: Gates are really circuits of transistors, but we think of them as abstract gates.

- Gates have an instance name
- Gates have exactly one internal output pin
- Gates have one or more internal input pins
- Gate Pin Order Convention: OUTPUT first, then all inputs

AND_2 Schematic & Verilog




```
// example Verilog file
module AND_2 (A0,A1,Z);
        input A0,A1 ;
        output Z;
        and1 and(Z,A0,A1);
        endmodule
```

Verilog Comment

- Like C, everything from // to the end of the line
- Used to help human readers understand what is going on
- Ignored by Verilog “compilers” or readers

Verilog Module Statement

module *name*(*pinlist*);

- module keyword required
- *name* – Name of the circuit
- *pinlist* – Comma separated list of module pins (visible externally)
  - Note – direction not specified in the module statement
- Semicolon (;) required

module AND_2 (A0,A1,Z);

Verilog Net Declaration Statements

decl_type netlist;

- *decl_type* – Either input, output, or wire
  - input: Nets connected to module input pins
    - Net name is the same as the module pin name – so gives direction of input to pin
  - output: Nets connected to module output pins
    - Net name is the same as the module pin name – so gives direction of output to pin
  - wire: internal nets (not connected to a module pin)
- *netlist* – comma separated list of net names

```
input A0,A1 ;
output Z;
// No internal nets in AND_2, so no wire statement
```

Verilog Gate Instance Statements

instance_name gate_type(*pinlist*);

- *gate_type*: one of inv, and, or, or xor
- *instance_name*: name of this instance of the gate
- *pinlist* – nets connected to the gate in gate pin order
  - output, then inputs

```
and1 and(Z,A0,A1);
```

Project Installment 2

- Given a circuit description in a Verilog file that consists of a single gate, and given values for the input module pins, determine the value of the output module pin.

Verilog Compiler Infrastructure Provided

- I have written a "Verilog compiler" for use in this project.
- The Verilog compiler reads a Verilog file, and keeps track of all the information in that file.
- The Verilog compiler provides functions you can call to get information about the Verilog in the file.
- To use this, you must add: #include "verilog.h"
- To use this, you must compile with a command like:

  gcc –g –Wall –o simCircuit simCircuit.c verilog.c

Top Level functions

- int openModule(char * moduleName)
  - e.g. : int mn=moduleName(argv[1]);
  - The moduleName paratemeter is a string
  - Reads the file *moduleName*.v in the current directory
  - Exits if it finds errors (error checking is not complete)
  - Keeps information about the Verilog file in memory
  - returns a module number (mn in future parameters)

- void freeModule(int mn)
  - e.g. : freeModule(mn);
  - Gives back all the memory used to keep module information
  - Subsequent calls using the same module number are invalid or incorrect

View of Saved Information (pins)

Module Table

Module Number	Name	#ModPins	#Pins	#Nets	#Instances
1	AND_2	3	6	3	1
…					

Module Pin Xref

PinIndex	PinNum
0	0
1	1
2	2
3	3
4	4
5	5

Pin Table

PinNum	Name	Dir	Val	Inst	Net
0	A0	I	?	-	0
1	A1	I	?	-	1
2	Z	O	?	-	2
3	and1.Z	0	?	0	2
…					

Net Table

Instance Table

View of Saved Information (nets)

Module Table

Module Number	Name	#ModPins	#Pins	#Nets	#Instances
1	AND_2	3	6	3	1
…					

Module Net Xref

NetIndex	NetNum
0	0
1	1
2	2

Net Table

NetNum	Name	Val	Src	#sinks
0	A0	?	0	1
1	A1	?	1	1
2	Z	?	3	1
…				

Net Sink Xrefs

SinkIndex	PinNum
0	4

SinkIndex	PinNum
0	5

SinkIndex	PinNum
0	2

Pin Table

View of Saved Information (instances)

Module Table

Module Number	Name	#ModPins	#Pins	#Nets	#Instances
1	AND_2	3	6	3	1
…					

Module Inst Xref

InstIndex	InstNum
0	0
..	…

Instance Table

InstNum	Name	Module	#pins
0	and1	AND	3
…			

Ints Pin Xrefs

PinIndex	PinNum	NetNum
0	3	2
1	4	0
2	5	1

Pin Table

Net Table

Conventions for Lists

- When there are lists of things, there is a function to return the number of elements in that list… e.g.

```
for(int pi=0; pi<moduleNumModulePins(mn); pi++) {
```

returns the number of module pins for the module

- Each pin in the module has it's own "pin number" (pn)
- We need to use a separate function to get the pin number:

```
int pn=modulePin(mn,pi)
```

where pi is an integer index such that 0<=pi<numPins

Module Functions

- char * moduleName(int mn)
  - e.g. : printf("Module name is %s\n",moduleName(mn));
  - Returns the module name string for the mn module
- int moduleNumModulePins(int mn)
  - e.g. : int numModulePins=moduleNumModulePins(mn);
  - Returns the number of module pins for the mn module
- int moduleNumPins(int mn)
  - e.g. : int numPins=moduleNumPins(int mn);
  - Returns the total number of pins (module + internal) for the mn module
  - Note that module pins always come first, so numModulePins<=numPins
- int modulePin(int mn,int pi)
  - e.g. : int pn=modulePin(mn,pi);
  - Returns the pin number for the pi^{th} pin in module mn's list of pins

Module Functions (Continued)

- int moduleNumNets(int mn)
  - e.g. int numNets=moduleNumNets(mn);
  - Returns the number of nets in module mn
- int moduleNet(int mn,int ni)
  - e.g. int nn=moduleNet(mn,ni);
  - Returns the net number for the nith net in module mn
- int moduleNumInstances(int mn)
  - e.g. int numInst=moduleNumInstances(mn);
  - Returns the number of instances in module mn
- int moduleInstance(int mn,int ii)
  - e.g. int in=moduleInstance(mn,ii);
  - Returns the instance number of the iith instance in module mn

Pin Functions

- char * pinName(int pn)
  - e.g. printf("Pin %d has name %s\n",pn,pinName(pn));
  - Returns the string pin name for pin pn
- char pinDirection(int pn)
  - e.g. if ('I'==pinDirection(pn)) printf("input pin\n");
  - Returns the pin direction of pin pn ('I' - input or 'O' - ouput)
- int pinInstance(int pn)
  - e.g. int in=pinInstance(pn);
  - Returns the instance number for internal pins, returns -1 for module pins
- int pinNet(int pn)
  - e.g. int nn=pinNet(pn);
  - Returns the net index of the net connected to pin pn

Pin Functions (Continued)

- int setPinValue(int pn,int value)
  - e.g. setPinValue(pn,1);
  - Returns 1 if the pin value is set, zero if the pin was already at that value (aborts if pin was set to a different value)
- int pinValue(int pn)
  - e.g. if (pinValue(pn)==1) printf("pin is on\n");
  - Returns the pin value (-1 if not set.)

Net Functions

- char * netName(int nn)
  - e.g. printf("Net %s\n",netName(nn);
  - returns string net name for net nn
- int netSource(int nn)
  - e.g. int srcPinNum=netSource(nn);
  - returns the source pin number for net nn
- int netNumSinks(int nn)
  - e.g. int numSinks=netNumSinks(nn);
  - returns the number of sink pins for net nn
- int netSink(int nn,int si)
  - e.g. int sinkPinNum=netSink(nn,si);
  - returns the sink pin number for the si^{th} sink pin on net nn

Net Functions (continued)

- int setNetValue(int nn)
  - e.g. if (setNetValue(nn)) printf("Net value set to source value");
  - Returns 1 if source pin value is known and net value is unknown (-1)
  - Returns 0 if source pin value is unkown or source pin value matches net value
  - Aborts if source pin value is known and net value is known, but different
  - Note – does not propagate net value to sink pins
- int netValue(int nn)
  - e.g. if (netValue(nn)==0) printf("Net is off\n");
  - Returns the net value (-1 if the net value is not set)

Instance Functions

- char * instanceName(int in)
  - e.g. printf("Instance %s\n",instanceName(in));
  - Returns string name for instance in
- char * instanceModule(int in)
  - e.g. if (0==strcmp(instanceModule(in),"and")) printf("This is an and gate\n");
    - Note… strcmp returns 0 if arguments match
  - Returns string name of the gate for instance in

Instance Functions (continued)

- int instanceNumPins(int in)
  - e.g. int numInstPins=instanceNumPins(in);
  - returns the number of instance pins for this instance
- int instancePin(int in,int pi)
  - e.g. int pn=instancePin(in,pi);
  - Returns the pin number of the pith instance pin for instance in
- int instanceNet(int in,int pi)
  - e.g. int nn=instnaceNet(in,pi);
  - Returns the net number connected to the pith instance pin for instance in

Print Module Information

- void printModuleIO(int mn)
  - e.g. printModuleIO(mn);
  - Prints module pins and values on a single line
- void printModuleState(int mn)
  - e.g. printModuleState(mn)
  - Prints complete information about a module
    - All pins – pin number, name, direction, instance number, net number, and value
    - All nets – net number, name, source pin and value, sink pins and values
    - All instances – instance number, name, and number of connections
      - All instance connections – pin number and net number
  - Use for debug!