CS 2461
Lab- Week 5

Today….

- Review Design of Finite State Machines – example

- Bitwise operations in C – complete the exercise from lecture
  course webpage – Exercises-Week5 download sept27.c

Finite State Machines

- The behavior of sequential circuits can be expressed using finite state machines (FSMs).
  - FSMs consist of a set of nodes that hold the states of the machine and a set of arcs that connect the states.
  - FSM represented as a graph

▪ Elements of FSM:

  - Finite Number of states
  - Finite number of inputs and Finite number of outputs
  - A specification of the state transitions
  - A specification (Boolean function) of the outputs
  - Outputs are associated with a node/state

FSM Design Process

- The first step is to model the behavior of the machine
  - Based on problem statement
  - Identify what the inputs are
  - Identify the outputs
  - Determine what needs to be stored to capture the “state” of the machine

- Represented as a graph – finite state diagram
  - Nodes: States – a state stores summary of events (until current time)
  - Edges: Transition from current state to next state
    - Based on input and current state and computed by combinational logic
  - Outputs: value of outputs at each state
  - State: the state of a system is a “snapshot” of all relevant elements at an instant in time.
    - Example : vending machine should remember total money received,….

Designing and implementing a FSM

1. Understand the problem statement and determine inputs/outputs
2. Identify states and draw the state diagram
   - Encode each state in binary using N bits
   - State diagram will show transitions from state to state based on value of inputs
3. Next, derive the truth table (from state diagram)
   - "inputs" in truth table are N current state variables and the inputs
   - These N state variables will need to be stored in N flips flops,
   - Label the N state variables $S_{N-1}\,S_{N-2}\ldots S_1 S_0$
   - "outputs" are the values of the state variables in the next state and the output at each state -- common notation is S' but confusion with complement operator, so let's use S*
4. From truth table, implement combinational circuit (boolean function) for each of the next state values & outputs
   - State variables are stored in your N storage elements

FSM Design: Example

- design the finite state diagram for a sequential circuit that generates an output Z=1 whenever the input (binary) string it has read thus far has an odd number of 0's and an odd number of 1's.
  - For example, if the input string is 010010 (4 zeros and 2 ones) then output Z=0. If the input string is 1101 (1 zeros and 3 ones) then the output is Z=1.

- Assume: at each clock cycle the machine reads one bit (a 0 or 1)
  - Eg. If overall input = 0101 then after 2 cycles it would have read 01 (and output=1),

  after 3 cycles it has read 010 (and output =0), etc.

Question: What property in a state ?

- The first 'property' to note is that for any binary string, the string has some X number of 1's and some Y number of 0's.
  - For string 01100 we have $X=2$ and $Y=3$
- The question is now defined in terms of the properties of X and Y – if X and Y are both odd then output Z=1
- For any binary string how many cases can you have in terms of the evenness/oddness of X and Y ?
- Four cases = Four states:

1. X *even and* Y *even*
2. X *even and* Y *odd*
3. X *odd and* Y *even*
4. X *odd and* Y *odd*

State transitions

- you have 4 cases for any binary string of any length; next, what happens if a string that has been processed (i.e., read) thus far falls under case 3 (i.e., X is odd and Y is even) and then the machine reads a 1
- = String (thus far) now has X is even and Y is even = Case 1
- you have 4 cases for any binary string of any length; next, what happens if a string that has been processed (i.e., read) thus far falls under case 3 (i.e., X is odd and Y is even) and then the machine reads a 0
- = X is odd and Y is odd = Case 4

Complete the transitions

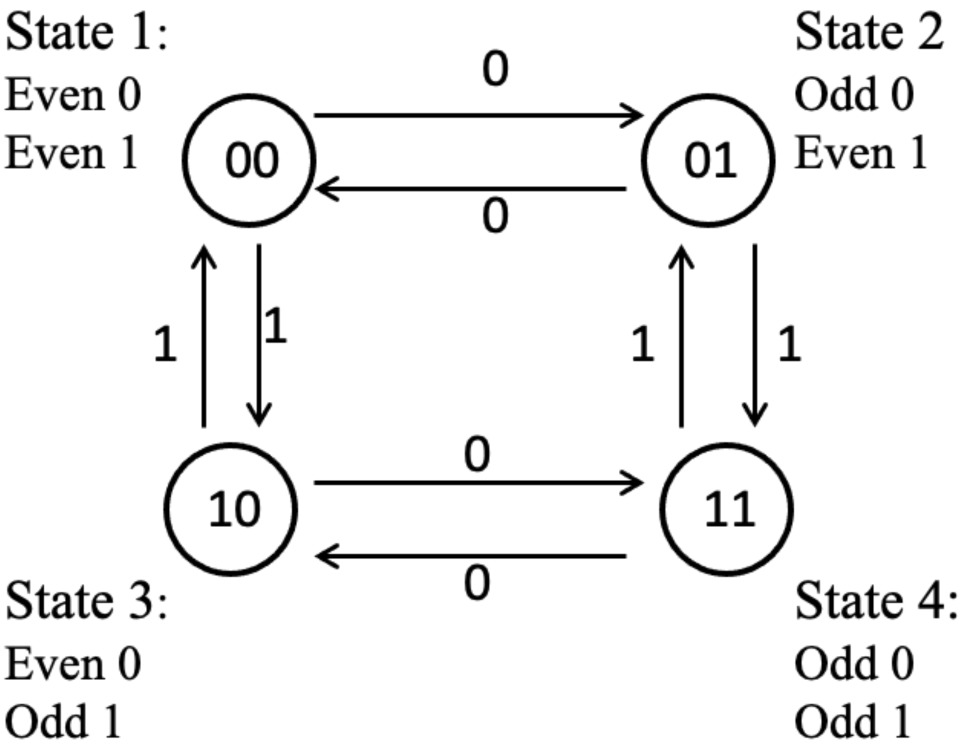
$X =$ no. of 1's in string
$Y =$ no. of 0's in string

- Case1 (State1): X even and Y even
  - Read 0 go to ?
  - Read 1 go to ?
- Case 2 (State 2): X even and Y odd
  - Read 0 go to ?
  - Read 1 go to ?
- Case 3 (State 3): X odd and Y even
  - Read 0 go to ?
  - Read 1 go to ?
- Case 4 (State 4): X odd and Y odd
  - Read 0 go to ?
  - Read 1 go to ?

States:

1. *X even and Y even*
2. *X even and Y odd*
3. *X odd and Y even*
4. *X odd and Y odd*

Draw the finite state diagram



Truth table

- How many state variables (storage bits) ?
- 2 bits – S_1 S_0
  - State1 = 00 State2 = 01 State3 = 10 State4 = 11

- If FSM is in State1 (00) and input=0 then next state is 01 and current output in State00 is Z=0
- In=0 S_1 =0 S_0 =0 then S_1* =0 S_0*=1 Z=0

- If FSM is in State 4 (11) and input=1 then next state is 01 and current output in State11 is Z=1
- In=1 S_1 =1 S_0 =1 then S_1* =0 S_0*=1 Z=1

Complete the Truth Table

In	S1	S0	S1*	S0*	Z
0	0	0	0	1	0
0	0	1	0	0	0
0	1	0	1	1	0
0	1	1	1	0	1
1	0	0	1	0	0
1	0	1	1	1	0
1	1	0	1	1	0
1	1	1	0	1	1

Questions on Finite State Machines ?

Bitwise operations in C

Go to course website
Download Exercises Week 5 – Sept27.c (under C and Data Rep)
(do not compile and run it yet!)

C data types and operators

- Unsigned and signed int – C allows casting
- Bitwise operations:
  - & (and)
  - | (or)
  - ~ (complement)
  - ^ (XOR)
  - Right shift >>   arithmetic shift = MSB is replicated rightwards shift
  - Left shift <<   shift in 0's into the LSBs
    - What is left shift one position ? What is the value of (x <<1) ?
- Logical operators: arguments are treated as binary (True or false)
  - && (logical And)
  - || (logical OR)
  - ! (logical NOT)
  - Key takeaway: if x is a non-zero integer then x is True

Time to test your C …

- Download/open Exercises-Week5 (a C file called sept27.c) from webpage
- Do NOT run the C code….Go through code and answer the questions without running the code
  - Reading code (without running it) is a very important skill
- Use the following values as inputs (this info included in the comments in C file):
  - `zz = abcd0123` (hex representation of a 32 bit number)
  - `a=4, b=7, n=2` and different values of c for `c>0, c<0,` and `c=0`
- 1. First answer the questions
- 2. Next: Compile and run your code & compare your answers with the run-time results
- 3. Can you explain what is going on (if your answer did not match)

Notation: a prefix of 0x indicates Hex representation
0x25 is the integer $2*16^1 + 5*16^0 = 37$
0xFF is the integer with 1's in last 8 bits, i.e., decimal value 255

Describing the function

- CallMeNext

```
int CallMeNext(int x){
   int t;
   t= (1 <<x);      what is 1 << 1 ? What is 1 << 2 ?
   t= t+1;          what is 1 << x ?
   return (t);
 }
```

Describing the function

- CallMeLast

```
int CallMeLast(int x, int y){
  int temp;
  temp = ~y;    computes complement of y (invert all bits)
  temp = temp +1;       what is temp= (NOT y)+1 ?
  temp = temp + x;
  return(temp);
}
```

Describing the function

- whoamI $0 \leq n \leq 3$

```
int whoamI(int x, int n){
  int rs;
   int y = n << 3;
```
what is y when n=2 ? What happens when you shift an integer left 3 places ?
```
   int xs = 0xFF << y;
```
what is oxFF left shifted 16 places
```
   rs=  xs & x;
```
what bits (bytes) are you masking?
```
   /* return(rs); */
    return ((rs >> y)& 0xFF);
}
```

```
Example: x= 0x abcd1234 and n=2
What is y = 2 <<3 = 2*8 = 16
What is xs = 0xFF << 16 = 0x 00FF0000
What is rs = 00FF0000 & abcd1234 = ?
```

Describing the function

- WhatamI

```
int whatamI(int A) {
    int X,Y;
    X = (A ^ (-A)) >> 31;
    Y = (X & 0x1);
    return(Y);
}
```

suppose Z= (A ^ (-A)) then what is the least significant bit of X in terms of Z?

when is Y=1 ? When is Y=0 ?

think of the binary representation of integers A and –A

if A=12 in binary = 0000………1010 and 0x 0000 000C in Hex

What is –A = -12 in binary ?

then what is the MSB of (A ^ (-A))